LEANDRO AUGUSTO DA SILVA
SARAJANE MARQUES PERES
CLODIS BOSCARIOLI

Introdução à MINERAÇÃO DE DADOS

Com aplicações em R

SÉRIE SBC
SOCIEDADE BRASILEIRA DE COMPUTAÇÃO

ELSEVIER

Introdução à Mineração de Dados

Com Aplicações em R

Leandro Augusto da Silva
Sarajane Marques Peres
Clodis Boscarioli

ELSEVIER

Copidesque: Christiane Simyss
Revisão: Gabriel Pereira
Editoração Eletrônica: Estúdio Castellani
Epub: SBNigri Artes e Textos Ltda.

Elsevier Editora Ltda.
Conhecimento sem Fronteiras
Rua Sete de Setembro, 111 – 16º andar
20050-006 – Centro – Rio de Janeiro – RJ – Brasil

Rua Quintana, 753 – 8º andar
04569-011 – Brooklin – São Paulo – SP – Brasil

Serviço de Atendimento ao Cliente
0800-0265340
atendimento1@elsevier.com

ISBN: 978-85-352-8446-1
ISBN (versão digital): 978-85-352-8447-8

Consulte nosso catálogo completo, os últimos lançamentos e os serviços exclusivos no site www.elsevier.com.br

NOTA

Muito zelo e técnica foram empregados na edição desta obra. No entanto, podem ocorrer erros de digitação,impressão ou dúvida conceitual. Em qualquer das hipóteses, solicitamos a comunicação ao nosso serviço de Atendimento ao Cliente para que possamos esclarecer ou encaminhar a questão.

Para todos os efeitos legais, nem a editora, nem os autores, nem os editores, nem os tradutores, nem os revisores ou colaboradores, assumem qualquer responsabilidade por qualquer efeito danoso e/ou malefício a pessoas ou propriedades envolvendo responsabilidade, negligência etc. de produtos, ou advindos de qualquer uso ou emprego de quaisquer métodos, produtos, instruções ou ideias contidos no material aqui publicado.

A Editora

CIP-BRASIL. CATALOGAÇÃO-NA-FONTE
SINDICATO NACIONAL DOS EDITORES DE LIVROS, RJ

S581i Silva, Leandro Augusto da
Introdução à mineração de dados: com aplicações em R / Leandro Augusto da Silva, Sarajane Marques Peres, Clodis Boscarioli. – 1. ed. – Rio de Janeiro: Elsevier, 2016.
il. ; 27 cm.

ISBN 978-85-352-8446-1

1. Mineração de dados (Computação). 2. Banco de dados. I. Peres, Sarajane M. II. Boscarioli, Clodis. III. Título.

16-32565

CDD: 005.74
CDU: 004.65

PREFÁCIO

Chegamos à conclusão de que, desde a época em que cursávamos nossos doutorados, e antes mesmo de pensarmos em lecionar Mineração de Dados (MD), já refletíamos individualmente sobre o que deveria ser um curso nessa área. Quando começamos a conversar sobre isso, veio a inspiração da criação deste livro-texto como material didático, escrito para apoiar aulas de Mineração de Dados em cursos da área da Computação ou de outras áreas interessadas em análise de dados. Construímos o projeto de um livro a partir da nossa vivência em pesquisa, mas, principalmente, tentamos introduzir no texto o aprendizado que obtivemos durante nossa experiência ao lecionar disciplinas de MD.

Pensando nos professores que, como nós, a princípio, possam ter dificuldades na definição do escopo e organização de conteúdos de um curso de MD e pensando no conhecimento, habilidades e competências que um aluno deve adquirir em um curso dessa natureza, decidimos reunir material teórico e prático usando R, com base na condução de um processo de mineração de dados. Este conteúdo, resultante de nossas reflexões, é basal para que atividades de MD possam ser depois incorporadas como práticas de trabalho e é apresentado neste livro a partir de exemplos didáticos, de concepção simples, de fácil assimilação e que possam ser reproduzidos durante o tempo de uma aula – pelo menos, essa foi nossa intenção.

Entendemos que não é viável abordar todos os aspectos de uma área de conhecimento em um único livro e, portanto, escolhemos o que julgamos ser os principais conceitos, métodos e algoritmos necessários a um curso de Introdução à MD com duração de um semestre da graduação, muito embora sua organização possa nortear a construção de disciplinas de pós-graduação (*lato* ou *stricto sensu*).

O livro está organizado em uma sequência lógica de conhecimentos a serem assimilados pelo leitor. O Capítulo 1 apresenta a Mineração de Dados como área, conceituando-a em linhas gerais e introduzindo suas principais tarefas: análise preditiva, análise de agrupamento e descobertas de associações. A leitura deste capítulo é importante para situar o leitor em conceitos fundamentais à compreensão do restante do livro e para conhecer o cenário sobre o qual são inspirados os exemplos de aplicação que ilustram os demais capítulos.

O Capítulo 2 discorre brevemente sobre estatística descritiva e visualizações gráficas, além de práticas de pré-processamento de dados, essenciais num processo de mineração. Para finalizar o capítulo, a preparação de dados textuais para mineração de dados é abordada como forma de exemplificar quão específico pode ser o tratamento de dados de natureza diferentes – escolhemos o tipo texto em razão de muitos dos dados não estruturados gerados atualmente estarem nesse formato. A tônica desse capítulo é, portanto, a compreensão de conjuntos de dados por meio de medidas estatísticas e por recursos visuais e sua preparação para a mineração de dados.

O Capítulo 3 traz conceitos e algoritmos da tarefa de predição categórica (a classificação) e da tarefa de predição numérica (a regressão). Depois, são apresentadas algumas medidas de avaliação para modelos preditivos e também algumas técnicas para treinamento, validação e teste de algoritmos. A tarefa de análise de agrupamento é assunto do Capítulo 4, que aborda diferentes estratégias de agrupamento e fornece subsídios para validar

resultados a partir de índices externos e internos. O Capítulo 5 apresenta a tarefa de descoberta de regras de associação em seus conceitos principais, notação e algoritmos.

Tentamos produzir um livro de fácil leitura, didático e convidativo à prática, seja pelos exemplos e implementações em R, seja pelos exercícios de fixação de conteúdo disponíveis em cada capítulo. Esperamos ter obtido êxito nessa nossa estratégia e que o leitor, a partir do conhecimento introdutório aqui apresentado, possa agregar valor à sua formação e despertar o interesse por essa promissora área.

Leandro Augusto da Silva
Sarajane Marques Peres
Clodis Boscarioli

SUMÁRIO

4.1. A tarefa de agrupamento

4.1.1. Agrupamento hierárquico – AGNES e DIANA

4.1.1.1. Exemplo didático para AGNES

4.1.1.2. Exemplo prático do AGNES em R

4.1.2. Agrupamento por partição – k-médias

4.1.2.1. Exemplo didático para o k-médias

4.1.2.2. Exemplo prático para o k-médias em R

4.1.3. Agrupamento por densidade – DBSCAN

4.1.3.1. Exemplo de didático para o DBSCAN

4.1.3.2. Exemplo prático para o DBSCAN em R

4.1.4. Mapas Auto-organizáveis

4.1.4.1. Exemplo didático para Mapas Auto-Organizáveis

4.1.4.2. Exemplo prático para Mapas Auto-Organizáveis em R

4.2. Avaliação de modelos para análise de agrupamento

4.2.1. Índices baseados em critérios externos

4.2.2. Índices baseados em critérios internos

4.2.3. Exemplo didático para avaliação de modelos para análise de agrupamento

4.2.4. Exemplo prático para avaliação de modelos para análise de agrupamento em R

4.3. Leituras adicionais

4.4. Exercícios

Capítulo 5 - Regras de associação

5.1. A tarefa de descoberta de regras de associação

5.1.1. Conceitos básicos

5.1.2. Itens frequentes e Propriedade Apriori

5.1.3. Descoberta de regras de associação e medidas de avaliação

5.1.4. Algoritmo Apriori

5.1.5. Algoritmo FP-Growth

5.2. Exemplo didático para descoberta de regras de associação

5.3. Exemplo prático para descoberta de regras de associação em R

5.4. Leituras adicionais

NOTAS

ÍNDICE

CAPÍTULO 1

INTRODUÇÃO

Este livro é sobre descoberta. Nele, o conceito de descoberta assume dois sentidos. Primeiro, o romântico, no qual a descoberta é vista como um fenômeno emocionante e prazeroso. E é esse fenômeno que se espera provocar durante sua leitura. Segundo, o sentido técnico, no qual a descoberta continua sendo igualmente emocionante e prazerosa, mas passa também a ser o resultado de um criterioso estudo sobre dados. A partir do estudo e da **mineração dos dados**, a descoberta acontece, e então novo conhecimento é produzido, contribuindo para a melhoria de produtos, sistemas, processos, negócios etc. Mineração de dados é, portanto, o meio pelo qual o fenômeno da descoberta se manifesta e será o assunto tratado nas páginas que se seguem.

Neste capítulo introdutório é apresentado, inicialmente, um contexto no qual a mineração de dados se faz presente, ainda que de forma imperceptível à primeira vista. É importante analisá-lo em todos os seus meandros antes de avançar com a leitura. Ele se constitui como um exemplo no qual um agente não se deu conta de que tinha em mãos uma mina de conhecimento escondido em meio a muitos dados e que, na realidade, precisava ser minerada para que pudesse ser usada na tomada de decisão que o levaria ao sucesso.

Contudo, minerar dados para descobrir conhecimento não é uma tarefa trivial. É preciso conhecer os dados, o processo de análise e descoberta, as tarefas e técnicas de mineração e as ferramentas matemáticas e

computacionais que se aplicam nesse contexto. Portanto, a descoberta é um processo. Ainda, é preciso conhecer o ambiente em que os dados são produzidos e que tipo de conhecimento esse ambiente necessita e espera receber. Enfim, minerar dados exige conhecimento técnico, tempo e dedicação, e uma boa ideia para começar é a leitura do restante deste capítulo.

1.1. A importância dos dados na tomada de decisão

A história de um restaurante ...

"Creio que muitos como eu enxergam a vida como um evento dicotômico. Quando crianças, queremos ser adultos e, quando adultos, temos saudades da infância. Esse anseio é bastante intensificado em nossa época de adolescência, quando vivemos uma transição. Entendemo-nos adultos, e nossos pais nos veem como crianças. Espero que muitos concordem comigo, pois penso ser uma impressão comum na vida de todos nós.

Tenho muitas boas lembranças da minha infância e da minha adolescência, vividas em uma importante cidade do interior do estado de São Paulo. Tive a oportunidade de morar em um bairro com muitas crianças da mesma idade. Brincadeiras na rua eram frequentes. Todo fim de tarde, a criançada se reunia para brincar.

Aquele momento da minha vida era muito interessante. As brincadeiras eram, em sua vasta maioria, coletivas. A única brincadeira individual da qual me recordo era a montagem de quebra-cabeças. A tecnologia naquele período era extremamente precária face ao que temos disponível nos dias de hoje, e, portanto, divertíamo-nos com brincadeiras e brinquedos bem mais simples dos que hoje embalam a infância da criançada.

Lembro perfeitamente dos primeiros contatos com as tecnologias antes vistas apenas nos filmes. Por exemplo, lembro-me da perplexidade da criançada quando fotografadas por uma Polaroid. Isso aconteceu na década de 1990, período em que o avanço tecnológico de câmeras fotográficas era a "grande coqueluche" dos aficionados nesse tipo de *gadget*. Seu grande atrativo era a revelação imediata do retrato tirado. Ao mesmo tempo, o tamanho dos retratos era diferente dos demais tipos de câmeras e, então, era difícil encontrar álbuns para o armazenamento, além do alto custo para tirar uma foto e a perda de qualidade com o passar do tempo.

Ainda na década de 1990, lembro também da chegada do videocassete. A vizinhança toda se reuniu para assistir o seu primeiro filme em um aparelho de videocassete de duas cabeças. O grande charme desses aparelhos era a possibilidade da gravação de programas da TV. E rapidamente chegaram as filmadoras e o *videogame*. Com exceção da câmera fotográfica, os videoscassetes e filmadoras evoluíram em um ritmo alucinante. O *videogame*, por outro lado, evoluía com passos mais lentos, porém, cada lançamento trazia mudanças significativas, a exemplo do Atari em relação ao Sega Mega Drive. Quem conhece sabe muito bem sobre o que estou falando. Evidente que, nesta época, os computadores pessoais ou PCs (do inglês, *personal computer*) começavam a aparecer. Mas eram privilégios de pouquíssimas pessoas.

Os leitores podem estar questionando algumas datas e menções a lançamentos, mas isso se deve ao fato de que a chegada da tecnologia para as cidades do interior era sempre mais lenta. Havia um tipo de hierarquia que determinava que as novidades chegassem às capitais e depois seguissem para o interior. Ainda hoje acontece desse jeito, mas em um intervalo bem menor.

Adicionalmente, devo dizer que eu morava em um bairro de classe média, com uma vizinhança formada principalmente por comerciantes e profissionais liberais. Por isso, por exemplo, o primeiro computador que conheci foi na casa de um colega de escola cujo pai era advogado. Ou seja, a profissão dos pais proporcionava o contato com as tecnologias, muitas vezes mais rápido para algumas que para outras crianças.

Durante o período de minha adolescência, meu pai era um dos sócios de um tradicional restaurante da cidade. Note que ele era mais um comerciante do bairro. No restaurante, era servido de tudo – o cardápio era extremamente generoso. E isso era um atrativo, pois as famílias podiam, em um único lugar, pedir a tradicional pizza para as crianças e apreciar os deliciosos aperitivos também feitos por lá. Muitas famílias adoravam passar horas no restaurante,

conversando e se divertindo, enquanto a comida era preparada sob demanda para ser servida fresquinha.

Naquela época, meu pai achava importante que os adolescentes começassem a trabalhar desde cedo, até por influência dos meus avós, que migraram do campo para a cidade, e isso era muito comum também nas famílias dos meus colegas. Eu achava normal até o momento em que me dei conta de que era necessário ajudá-lo também nos fins de semana, afinal de contas, eu era um adolescente.

A minha função principal no restaurante era a de “ficar no caixa”. Talvez isso tenha influenciado fortemente a minha paixão pelos números. Ainda bem que não tinha a função de “ficar na choperia”. Já imaginaram?

Até este momento, o comércio da família caminhava muito bem, com exceção do meu mau humor por ter de trabalhar nos fins de semana – mas que fique registrado que meu trabalho era sempre muito bem realizado. E foi no início da década de 1990, período marcado com o chamado Plano Collor, que o comércio, de forma geral, começou a mudar.

Para os que não lembram ou não viveram aquela época, naquele período, a economia do país sofria com elevadas taxas de juros. A inflação era altíssima. E o tal Plano Collor vinha para controlar a inflação. As medidas adotadas para esse controle mudaram abruptamente os hábitos dos brasileiros. E a principal mudança foi, sem dúvida, o corte de gastos com os chamados supérfluos, por exemplo, almoços e jantares em restaurantes. As famílias que ainda mantiveram o hábito de se alimentar fora de casa já não tinham a paciência de esperar pela comida, pois a espera gerava mais gastos com bebidas e aperitivos. Foi nesta época que o número de restaurantes *self-services* começou a crescer. E este novo modelo de negócio para os restaurantes passou a ser a salvação para quem ainda tentava sobreviver desse tipo de comércio.

Enfim, o restaurante da minha família começou a enfrentar uma crise. E, mesmo que não gostasse de trabalhar no restaurante, eu fazia tudo muito

benfeito: em épocas de crise, fazer um bom trabalho faz toda a diferença. Para meu pai, meu papel de caixa era apenas organizar os pedidos dos clientes, anotados em pedaços de papel padronizados, dentro de um tabuleiro com o número das mesas; somar os gastos de cada mesa anotados nos pedidos, usando máquina registradora de quatro operações matemáticas; conferir o pagamento; e dar o troco. Contudo, para mim, o papel exercido no caixa poderia ir além daquilo que meu pai esperava.

Como minhas atividades de caixa do restaurante só terminavam depois que o restaurante passasse pela limpeza para a próxima jornada de trabalho, eu aproveitava o tempo de espera para fazer algumas análises, como: quantidade de bebidas vendidas de cada uma das marcas; tipos de entrada mais, ou menos, solicitadas com determinados tipos de prato do cardápio; valores gastos em pedidos que tinham características específicas, como o horário ou o tipo de cliente (famílias, grupo de amigos). A atividade no caixa me permitia ser um bom observador. Eu conhecia muito bem as preferências de cardápio da maioria dos clientes e, consequentemente, o quanto cada família gastava em média e as formas de pagamento usuais, até os hábitos de onde cada um gostava de se sentar e o dia da semana que costumava frequentar o restaurante. Vale notar que a clientela era formada por frequentadores assíduos. Além disso, por conhecer tão bem a rotina do restaurante, eu sabia dos riscos que algum tipo de insatisfação poderia causar e quais medidas de contorno eram úteis empregar em cada tipo de situação. Por exemplo, se não tinha o prato do gosto do cliente, eu saberia sugerir outro prato que o mesmo cliente escolhera em outra ocasião ou, então, sugerir outro prato com preço ou acompanhamentos similares ao que ele costumava escolher. O importante era não perder o cliente, como acontecia em algumas ocasiões, principalmente na crise em que vivíamos.

No entanto, como fator extra advindo da crise, o preparo do prato, o diferencial do restaurante, passou a atuar como fator negativo, pois os hábitos dos clientes mudaram. Em vez de consumirem dois refrigerantes enquanto

aguardavam o preparo da comida, por exemplo, cada membro da família passou a consumir apenas um, e, sendo assim, era importante que a comida ficasse pronta o quanto antes.

A solução que meu pai e seu sócio propuseram para enfrentar a crise foi transformar um tradicional restaurante de comidas feitas na hora em um restaurante com o sistema *self-service*. Eu achava a pior ideia possível, pois era convicto de que certamente havia dois pratos no tradicional cardápio do restaurante presentes na minha preferência e na da maioria dos frequentadores: a Feijoada e o Filé à Parmegiana. Afinal, trabalhando no restaurante, eu tinha o privilégio de fazer ótimas refeições.

As análises que eu, sistematicamente, fazia ao final de cada expediente de trabalho me permitiam afirmar que o Filé à Parmegiana era o prato mais vendido do restaurante, desde que não chovesse ou fizesse frio, pois, nesse caso, a Feijoada era imbatível. Portanto, a solução para mim era óbvia: reduzir o cardápio para um único prato, o Parmegiana. Essa mudança afetaria todo o processo de negócios da empresa, mas principalmente, o tempo de preparo do prato, o que era a maior queixa dos clientes. Mas, e a Feijoada? Estamos falando de uma cidade com temperatura média anual de 23°C. Entenderam?

Enfim, como na época eu era um adolescente, minha sugestão para tomada de decisão, embora respaldada em estatísticas que resumiam os dados, observações que geravam informação e esforços analíticos para descobrir conhecimento naquele mundo analógico, não foi suficiente para convencer os donos do negócio. Infelizmente, a estratégia seguida pela minha família não deu certo, e o restaurante fechou as portas.

A pergunta que ficou em minha cabeça foi: “Será que minha solução teria evitado a falência?” Impossível saber. Entretanto, hoje fico muito feliz em lembrar como, de maneira informal, eu era capaz, assim como todos nós somos, de modelar e resolver tarefas de mineração de dados, incluindo previsão de séries referente às vendas mensais dos pratos e bebidas;

modelagem do comportamento de um cliente generoso; sugestões de combinações entre as opções do cardápio ou de acompanhamento – afinal, quem não sabe que, quando se escolhe um Filé à Parmegiana, espera-se receber também arroz e batatas fritas?

Vale lembrar que a informatização naquela época era pouco disseminada e cara. O preço de um computador pessoal era altíssimo, e todo o comércio varejista usava papéis e máquinas registradoras. Até mesmo o comércio atacadista usava esses parcos recursos. Logo, os dados eram acumulados em montanhosas pilhas de papéis, e toda a análise que apresentei era feita em papel e todo o conhecimento gerado estava restrito apenas à minha memória e à minha capacidade de processamento cerebral.

Então, caro leitor, eu lhe pergunto: “Hoje, a tomada de decisão seria diferente? O dono do restaurante teria se enveredado por um caminho diferente se tivesse recebido orientação de um especialista em mineração de dados? E qual teria sido a orientação desse analista se ele tivesse acesso automatizado a todos os dados gerados no dia a dia do restaurante? E se ele ainda pudesse contar com dados provenientes de outras fontes, como tendências gerais de mercado, economia local e redes sociais?”

1.2. Dados, informação e conhecimento

Embora sejam palavras que aparentemente se confundem como sinônimos, **dado**, **informação** e **conhecimento** têm definições diferentes quando se trata do contexto de Mineração de Dados.

O **dado** é um fato, um valor documentado ou um resultado de medição. Quando um sentido semântico ou um significado é atribuído aos dados, gera-se **informação**. Quando estes significados se tornam familiares, ou seja, quando um agente os aprende, este se torna consciente e capaz de tomar decisões a partir deles, e surge o **conhecimento**.

Um semáforo de trânsito pode ser usado como exemplo para discutir essas diferenças conceituais. Um semáforo constitui-se por um sistema de sinalização que usa três cores: verde, amarelo e vermelho. Essas cores são fatos, ou dados, e podem assumir significados diferentes a depender do contexto. Quando, por meio de leis de trânsito, convencionou-se significado a essas cores, ou seja, verde permitindo seguir, amarelo requerendo atenção e vermelho exigindo parar, os dados referentes às cores foram transformados em informação. O conhecimento surge quando tais convenções são assimiladas e aplicadas por pedestres e motoristas. Note que, no caso do semáforo de trânsito, os significados (informação) das cores (dados) são aprendidos (conhecimento), e não há necessidade de repetir todo o processo de aquisição de conhecimento a cada vez que nos deparamos com um semáforo.

Entretanto, diferentes convenções podem ser assumidas em uma mesma situação, de forma que dados diferentes possam levar à geração do mesmo tipo de conhecimento, ou os mesmos dados possam levar à geração de conhecimento diferente. No caso do semáforo de trânsito, a existência de luz (e não as cores) composta com suas posições pode ser assumida como dados. A informação é gerada a partir de uma convenção diferente (superior, central, inferior) para as ações esperadas (parar, ter atenção, seguir). A assimilação

desta convenção, embora não oficial em nossa sociedade, também pode ser assumida como um conhecimento válido em um contexto ou condição diferentes do padrão. Ainda, um sistema de lâmpadas coloridas pode ser usado em um contexto diferente para indicar risco de incêndio, sendo que a cor vermelha indicaria alto risco, a laranja, risco médio, e a verde, baixo risco de ocorrência. Situações similares a essas são comuns em Mineração de Dados, e isso será perceptível no decorrer da leitura deste livro.

1.3. Tipos de dados

Dados se constituem como a matéria-prima para que processos de mineração ocorram. Essa matéria-prima pode se manifestar, basicamente, de duas formas:[1] estruturada e não estruturada. A forma como os dados estão disponíveis para a realização da mineração é importante para determinar o tipo de tarefa de mineração que é possível resolver, o tipo de conhecimento factível de ser descoberto e o tipo de técnica de mineração aplicável. Além disso, a quantidade e a qualidade dos dados disponíveis também são determinantes para o sucesso da mineração de dados.

As restrições ou limitações que os dados impõem ao processo de mineração ficarão mais claras no decorrer do estudo do conteúdo deste livro. Por enquanto, é importante entender as formas como os dados podem ser encontrados.

1.3.1. Dados estruturados

Tipicamente, uma base de dados usada em sistemas informatizados convencionais é organizada de forma que se tenham dados armazenados em estruturas tabulares, em que as linhas armazenam uma ocorrência de um evento caracterizado por um conjunto de colunas que representam características que descrevem um exemplar (instância) daquele evento.

Na maioria dos casos, os dados estruturados são resultantes de processos de geração de dados inerentes a sistemas transacionais (Seção 1.5) ou resultantes de observações e processos de medição. Esses dados geralmente são armazenados em um conjunto de tabelas relacionadas entre si. Exemplos de processos de geração de dados são:

- Um sistema bancário que armazena informações sobre seus clientes, sobre suas respectivas contas bancárias e sobre as transações executadas sobre tais contas.

- O trabalho de funcionários de um departamento de recursos humanos de uma empresa que observam o comportamento de candidatos a uma vaga de emprego, fazendo anotações sobre suas atitudes durante a realização de atividades de um processo de seleção.
- Uma empresa que deseja lançar um produto novo e, para isso, realiza pesquisas sobre sua aceitação com um conjunto específico de pessoas que representa o público-alvo.
- Uma corrida de Fórmula 1 na qual se medem os tempos que cada piloto gasta para chegar em diferentes trechos de um circuito.

Considere uma base de dados na qual são armazenados dados sobre candidatos a vagas de emprego em um restaurante. Nela são armazenados dados que descrevem a unidade observacional: o candidato. Uma forma de estruturar os dados referentes a cada candidato é apresentada na Figura 1-1, em que cada uma de todas as *n* observações possui um identificador (ID), seis atributos descritivos e um atributo de rótulo.

	atributos						
ID	*descritivos*						*rótulo*
01	João	masculino	viúvo	65	garçom	12.000,00	SIM
02	Maria	feminino	solteiro	32	cozinheiro	4.200,00	SIM
...	...	...	...	...	...	...	...
n	Pedro	masculino	casado	18	entregador	1.650,00	NÃO

Figura 1-1: Base de dados de observações documentadas a partir de um processo seletivo

À unidade observacional e seus descritores é possível e útil associar significado, dando a ela o caráter informacional. Uma representação resumida para essa associação é:

Candidato = {Matrícula, Nome, Sexo, Estado civil, Idade, Especialidade, Pretensão salarial, Experiência}

e a base de dados pode então ser semanticamente representada pela Figura 1-2.

ID	*atributos*						
	descritivos						*rótulo*
Matricula	Nome	Sexo	Estado civil	Idade	Especialidade	Pretensão salarial	Experiência
01	João	masculino	viúvo	65	garçom	R$ 12.000,00	SIM
02	Maria	feminino	solteiro	32	cozinheiro	R$ 4.200,00	SIM
...	...	...	...	...	...	...	...
n	Pedro	masculino	casado	18	entregador	R$ 1.650,00	NÃO

Figura 1-2: Base de dados sobre candidatos com associação de significado

1.3.2. Dados não estruturados

Muitos dos dados disponíveis para análise e extração de informação e conhecimento estão apresentados de forma não estruturada, a exemplo de textos, imagens, vídeos e sons.

Uma coleção de textos (ou documentos) compõe uma base textual, que pode ser usada como entrada em um processo de mineração de dados. Um exemplo de dados não estruturados com dois exemplares (dois dados, dois documentos) em uma base textual é mostrado na Figura 1-3. Os textos versam sobre postagens extraídas da rede social de um restaurante, que podem ser analisadas em um processo de mineração de dados (por exemplo, ser classificadas em polaridades como positiva ou negativa; ou agrupadas por similaridade de assuntos).

```
Ambiente agradável e tranquilo. Comida e música com qualidade. Adoramos o Filé
à Parmegiana.

O Filé à Parmegiana da cidade. Ambiente agradável e qualidade no atendimento.

O Filé à Parmegiana com fritas é uma delícia.
```

Figura 1-3: Exemplos de textos (dados não estruturados)

Os dados presentes em uma imagem se referem a valores relacionados com um sistema de cores e associados a uma estrutura matricial no arquivo

da imagem. A informação se torna presente a partir das relações de vizinhança e disposição das cores que, a partir de uma interpretação, manifestam-se em formas e texturas que representam conceitos abstratos ou concretos do mundo real. Um vídeo, de forma simplificada, pode ser visto como uma sequência de imagens que, em associação à dimensão “tempo”, manifestam também conceitos do mundo real. De forma similar, um arquivo de som é uma composição de ondas que se manifestam em uma informação audível.

Para fins de mineração de dados, dados não estruturados precisam passar por uma etapa de pré-processamento, de forma que uma representação adequada lhes seja produzida. Procedimentos de pré-processamento para textos serão abordados no Capítulo 2, e, para os demais tipos de dados não estruturados, há indicação de leituras na Seção 1.8.

1.3.3. Convenções

Existem algumas convenções aplicadas a bases de dados, usadas no processo de mineração de dados, sendo a mais comum o uso da nomenclatura “conjunto de dados” (do inglês, *dataset*). Ainda, normalmente os próprios dados são usados para a construção de um modelo que os explique sob algum aspecto, caracterizando o processo de descoberta de conhecimento e permitindo também a denominação dessas bases de dados como “bases de conhecimento”.

Outra convenção muito comum se refere à padronização sobre como fazer referência a um evento, ora denominado também como “dado”, e seus atributos descritivos dentro de um conjunto de dados. Por essa convenção, cada linha de um conjunto de dados se refere a uma unidade observacional também chamada de exemplar, objeto, instância ou, formalmente, $\mathbf{x}_{i}$. Um conjunto de dados $\mathbf{X}$ com n exemplares pode ser denotado como $\mathbf{X} = \{\mathbf{x}_{1}, \mathbf{x}_{2}, ..., \mathbf{x}_{i}, ..., \mathbf{x}_{n}\}$. Os dados de um conjunto de dados são descritos por d características (as colunas da representação tabular) ou atributos, os quais

serão representados por x_{ij}. Sendo assim, o *i-ésimo* exemplar do conjunto de dados (ou um dado) deve ser denotado por: $\mathbf{x}\rightarrow_i = \{x_{i1}, x_{i2}, ..., x_{ij}, ..., x_{id}\}$, com $i = \{1,...,n\}$.

Os atributos x_{ij}, que representam os exemplares de um conjunto de dados, são chamados descritivos ou regulares. Os valores que tais atributos assumem são usados pelos algoritmos de mineração de dados. No entanto, há um segundo tipo de atributo, denominado "rótulo" (do inglês, *label*), que tem um significado especial, uma vez que assume um papel específico em processos de mineração de dados que envolvam tarefas de predição (Capítulo 3). Se for esse o caso em estudo, o *i-ésimo* exemplar do conjunto de dados deve ser denotado por $\mathbf{x}\rightarrow_i = \{x_{i1}, x_{i2}, ..., x_{ij}, ..., x_{id}, y_i\}$ ou $(\mathbf{x}\rightarrow_i, y_i)$. Nos exemplos das Figuras 1-1 e 1-2, o atributo "Experiência" assume a função de rotular cada um dos dados, ou seja, classificar cada um dos candidatos como experiente (em algum nível) ou não. É comum também a existência de um atributo identificador (como é o caso do atributo "ID" nos exemplos citados); contudo, esse atributo deve ser usado apenas como um índice de busca e controle nas rotinas computacionais desenvolvidas para implementação dos algoritmos de pré-processamento, mineração e pós-processamento de dados. A base de dados seguindo as convenções aqui definidas está apresentada na Figura 1-4.

		x_{ij}							y_i
		ID	*i*1	*i*2	*i*3	*i*4	*i*5	*i*6	
		Matrícula	Nome	Sexo	Estado civil	Idade	Especialidade	Pretensão salarial	Experiência
$\vec{x}_i$	1	01	João	masculino	viúvo	65	garçom	R$ 12.000,00	SIM
	2	02	Maria	feminino	solteiro	32	cozinheiro	R$ 4.200,00	SIM
	...	...	...	...	...	...	...	...	...
	n	n	Pedro	masculino	casado	18	entregador	R$ 1.650,00	NÃO

Figura 1-4: Base de dados sobre candidatos com convenções de nomenclatura

1.4. Descoberta de conhecimento em bases de dados e a mineração de dados

"*Encontrar o que era desconhecido, o que estava escondido*" pode ser entendido como o ato de descobrir. Considerando que as bases de dados são geralmente volumosas e que o conhecimento pode estar implícito, faz-se necessário um trabalho de busca detalhado – o que, metaforicamente, diz-se ser um trabalho de "mineração" – associado a um processo analítico, sistemático e, até onde possível, automatizado.

É nesse cenário que, finalmente, a Mineração de Dados (do inglês, *Data Mining*) é definida. De forma simplificada, a mineração de dados pode ser definida como um processo automático ou semiautomático de explorar analiticamente grandes bases de dados, com a finalidade de descobrir padrões relevantes que ocorrem nos dados e que sejam importantes para embasar a assimilação de informação importante, suportando a geração de conhecimento. Trata-se de uma subárea de conhecimento geralmente estudada em cursos de Computação ou Engenharia e que faz uso de conceitos provenientes de Inteligência Artificial, Aprendizado de Máquina, Estatística e Banco de Dados.

Na realidade, iniciativas de estudo para sistematizar a descoberta de padrões a partir de bases de dados é uma tarefa bastante antiga. Já nos anos 1960, estatísticos se esforçavam na área de Análise de Dados a partir do uso de procedimentos indutivos.[2] Com o passar dos anos e com o aumento no volume de dados, heterogeneidade de formato e necessidade de alta disponibilidade, a área de análise evolui para metodologias exploratórias chamadas, então, de "*Data Fishing*" ou "*Data Dredging*", até que, nos anos 1990, surgiu o termo "*Data Mining*" (Han *et al.*, 2011). Não se sabe exatamente quando o termo surgiu e quem o cunhou, mas a comunidade de pesquisadores de Banco de Dados, havia algum tempo já interessada no tema,

passou a adotá-lo para nominar tal análise exploratória de dados. No mesmo período, Piatetsky- -Shapiro (1989) cunhou o termo *Knowledge Discovery in Databases* – KDD, ou em português “Descoberta de Conhecimento em Bases de Dados ”, para o primeiro *workshop* na temática de análise de dados.

Os termos KDD e *Data Mining* não podem ser entendidos como sinônimos, embora estejam altamente relacionados. Outros termos muito populares se confundem com o termo Mineração de Dados, como ETL (*Extraction, Transform and Loading*), cubos de um *Data Warehouse*, OLAP (*Online Analytical Processing*) e Estatística Descritiva. De fato, hoje, Mineração de Dados é entendida como uma etapa do processo de descoberta de conhecimento em bases de dados, que por sua vez, pode englobar os demais termos.

1.4.1. Visão geral do processo de descoberta de conhecimento

A descoberta de conhecimento em bases de dados tem como objetivo encontrar padrões intrínsecos aos dados nela contidos, apresentando-os de forma a facilitar sua assimilação como conhecimento. Como já discutido, tal descoberta está associada a um processo analítico, sistemático e, até onde possível, automatizado. Este processo denomina-se, para efeitos de simplificação, “processo de KDD ”.

Na Figura 1-5, as fases que constituem o processo de KDD são ilustradas. O processo KDD inicia com a organização das bases de dados que contêm os dados da área de interesse, a partir dos quais se deseja descobrir algum tipo de conhecimento útil. Esses dados passam por uma série de procedimentos de pré-processamento que inclui, mas não se limita a (Capítulo 2): organização em um repositório único (por exemplo, em um *Data Warehouse*); tratamento para eliminação de instâncias repetidas e ou valores discrepantes; procedimentos de seleção de dados que permitem que se escolha apenas os exemplares e/ou os atributos relevantes para a mineração de dados; normalizações para que os valores dos atributos estejam em uma mesma

escala e sejam considerados com a mesma relevância pelos algoritmos de análise. Com os dados pré-processados, inicia-se a tarefa de mineração de dados propriamente dita, envolvendo a resolução de tarefas como predição, agrupamento ou associação. Por fim, os resultados obtidos são validados, avaliados e formatados em gráficos, tabelas e relatórios estruturados. Esse processo pode ser iterativo e interativo, ou seja, cada fase pode ser executada mais de uma vez, na sequência usual ou fora dela, a depender do conjunto de dados original e/ou de decisões tomadas pelo analista (especialista no domínio dos dados).

Figura 1-5: O processo de KDD

1.4.2. Mineração de dados

Mineração de dados é definida em termos de esforços para descoberta de padrões em bases de dados. A partir dos padrões descobertos, têm-se condições de gerar conhecimento útil para um processo de tomada de decisão. Trata-se, portanto, da aplicação de técnicas, implementadas por meio de algoritmos computacionais, capazes de receber, como entrada, um conjunto de fatos ocorridos no mundo real e devolver, como saída, um padrão de comportamento, o qual pode ser expresso, por exemplo, como uma regra de associação, uma função de mapeamento ou a modelagem de um perfil.

Para efeitos de ilustração, considere a história de motivação narrada no início deste capítulo. Como bom observador e pessoa proativa, o “funcionário” (narrador da história) não se limitava a somar os valores anotados nos papéis que descreviam o consumo de cada mesa. Ele ia além e

observava o que era consumido pelos clientes. A partir de muitas observações em sua base de dados (os papéis nos quais os pedidos eram anotados), o "funcionário" extraiu um padrão de comportamento que pode ser entendido como uma regra de associação, nesse caso, expressando uma correlação:

" (...) **quanto maior** o tempo de espera pelo preparo de um prato, **maior** o consumo de bebidas."

Ainda, como observado na história do restaurante, há uma regra de associação que é praticamente senso comum em nossa sociedade:

" (...) **se** filé à parmegiana é servido, **então** arroz branco e batatas fritas também são servidos."

Extrapolando um pouco as observações realizadas pelo "funcionário", um algoritmo de mineração de dados é capaz, por exemplo, de gerar uma função que, dadas as condições atuais dos recursos humanos do restaurante, como número de garçons presentes e humor dos clientes, gere o valor aproximado das gorjetas que os clientes oferecerão naquele dia (isso, é claro, se o sistema de gorjetas não for predefinido como porcentagem do valor consumido no restaurante). Para gerar essa função que mapeia as condições no restaurante para o valor das gorjetas, o algoritmo de mineração precisaria analisar o contexto histórico do cenário, ou seja, o que ocorreu no passado em relação às variáveis disponíveis.

Esses são alguns exemplos de como descobrir padrões e prover conhecimento para suportar um processo decisório. A depender do tipo de dado à disposição e do tipo de conhecimento demandado, a mineração de dados oferece diferentes soluções e possibilidades. Assim, a área de mineração de dados é comumente dividida em tarefas, as quais ajudam a entender como situar um problema real junto aos diferentes algoritmos de análise de dados disponíveis e também que tipo de padrão, e consequentemente, que tipo de conhecimento é possível descobrir.

1.4.3. Tarefas de mineração de dados

Na literatura especializada são encontradas diferentes taxonomias para caracterizar as tarefas de mineração de dados. Fayyad *et al.* (1996) apresentam uma taxonomia em dois níveis. No primeiro nível, as tarefas de mineração de dados são divididas em **preditivas** e **descritivas**. Tarefas preditivas usam os valores dos atributos descritivos para predizer valores futuros ou desconhecidos de outros atributos de interesse. Já as tarefas descritivas têm o objetivo de encontrar padrões que descrevem os dados de maneira que o ser humano possa interpretar. No segundo nível, as tarefas preditivas e descritivas são especializadas. No conjunto de tarefas preditivas, os autores inserem classificação e regressão. Já no conjunto de tarefas descritivas, as especializações agrupamento, sumarização, modelagem de dependências e detecção de desvios são incluídas pelos autores.

Han, Kamber e Pei (2011) também seguem o primeiro nível da taxonomia já apresentada; porém, o segundo nível da taxonomia seguida por esses autores difere levemente da taxonomia de Fayyad *et al*. (1996). Para Han, Kamber e Pei (2011), as tarefas de mineração de dados no segundo nível se dividem em: classificação e regressão; mineração de padrões frequentes, associações e correlações (que correspondem a um dos objetivos da "modelagem de dependências" na primeira taxonomia); análise de grupos (equivalente a "agrupamento"); e análise de *outliers* (similar à detecção de desvios). Ainda, no segundo nível da taxonomia, para o caso de tarefas descritivas, Rokach e Maimon (2008) enumeram duas tarefas adicionais: resumo linguístico e visualização. Interessante notar que esses autores, na realidade, apresentam uma taxonomia em três níveis para "paradigmas de mineração de dados", em que um nível ainda mais alto é definido, dividido em paradigmas de verificação e de descoberta. No primeiro, tarefas de teste de hipótese, análise de variância e teste de *goodness-of-fit* são incluídas. Abaixo do paradigma de descoberta, desenvolvem-se os dois níveis de taxonomia já discutidos.

Este livro traz uma visão mais abrangente sobre as tarefas de mineração de dados, enumerando as três grandes tarefas, ilustradas graficamente na Figura 1-6: predição, agrupamento de dados e associação (ou descoberta de regras de associação), que podem assumir mais de uma variação e dar origem a subtarefas. Por exemplo, detecção de desvios pode ser resolvida a partir de uma análise de agrupamento de dados.

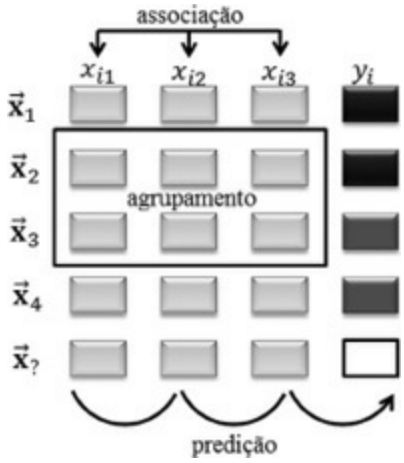


Figura 1-6: Tarefas de mineração de dados

Tarefas de **predição** consistem na análise de um conjunto de dados nos quais estão presentes os dados, descritos por atributos, e seus rótulos associados. O objetivo nessa tarefa é descobrir um modelo capaz de mapear corretamente cada um dos dados $\mathbf{x}_{\rightarrow 1}$, $\mathbf{x}_{\rightarrow 2}$, $\mathbf{x}_{\rightarrow 3}$ e $\mathbf{x}_{\rightarrow 4}$ aos seus rótulos (y) (Figura 1-6). Esse objetivo é alcançado por meio de técnicas chamadas supervisionadas, ou seja, capazes de encontrar o modelo de mapeamento a partir de procedimentos que associam um dado a um rótulo e corrigem tal associação quando ela não corresponde ao rótulo esperado (aquele associado ao dado no conjunto de dados). A análise preditiva pode ser dividida em duas subtarefas: análise preditiva categórica, também chamada de tarefa de classificação; e análise preditiva numérica, também chamada de tarefa de regressão. A primeira subtarefa se manifesta quando os rótulos associados aos dados pertencem a um conjunto discreto e finito de categorias. Já a

segunda se faz presente quando os rótulos associados aos dados são numéricos e pertencentes a um conjunto de valores contínuos.

Para exemplificar situações nas quais a resolução de tarefas de predição pode ser útil, considere o contexto do restaurante. Imagine que o dono do restaurante desejasse oferecer um serviço de harmonização aos seus clientes, ou seja, quisesse **classificar** os pratos ($\mathbf{x}\rightarrow_i$) servidos no restaurante em relação ao tipo de vinho (y) que deveria ser servido como acompanhamento. Tendo a descrição de cada um dos pratos (x_{i1}, x_{i2} e x_{i3} para cada $\mathbf{x}\rightarrow_i$) em termos de ingredientes (temperos principalmente), determinado tipo de vinho (branco seco, branco meio seco, tinto seco ou tinto meio seco) deve ser associado.

Para aprender o modelo que associa um tipo de vinho a determinado prato, o dono do restaurante convidaria um *sommelier* para ajudá-lo. O *sommelier* iria ao restaurante e, a cada prato, diante dos ingredientes que os compõe, associaria o tipo de vinho correto; e aquele "funcionário", como um bom aprendiz (ou um bom algoritmo), observaria e entenderia o comportamento da função de mapeamento (tipo de prato para tipo de vinho). Suponha que alguns meses depois, o cozinheiro do restaurante inserisse um novo prato ($\mathbf{x}\rightarrow_?$) no cardápio. Diante dessa situação, para associar o novo prato ao vinho correto (o y desconhecido representado pelo retângulo em branco na Figura 1-6), o "funcionário" analisaria os ingredientes usados e aplicaria a função de mapeamento que aprendeu, concluindo o tipo de vinho associado ao prato.

Para exemplificar a subtarefa de **regressão**, considere que o dono do restaurante deseja saber a relação entre o número de clientes que frequenta o estabelecimento (x) e seu faturamento mensal (y). A partir daí, ele poderia descobrir, por exemplo, qual seria seu faturamento se o número de clientes fosse o dobro, ou ainda, qual o impacto no faturamento com a redução do número de clientes em 10%. Assim, a empresa pode modelar uma tarefa de regressão e induzir a função capaz de fazer o mapeamento entre uma

perspectiva de quantidade de clientes ($\boldsymbol{x}_?$) e o faturamento mensal relacionado (o *y* desconhecido representado pelo retângulo em branco na Figura 1-6).

A tarefa de **agrupamento de dados** consiste na análise de conjuntos de dados em que estão presentes apenas as descrições dos dados. Não há, nesse caso, a necessidade do uso da informação sobre qualquer tipo de rotulação dos dados. O objetivo na resolução dessa tarefa é descobrir relações entre os dados por meio de suas similaridades e fornecer, como resposta, a indicação de quais dados são similares entre si, oferecendo um modelo de agrupamento ou perfis para grupos de dados. Os algoritmos aplicados na resolução dessa tarefa executam procedimentos que organizam os dados em grupos, de forma que a similaridade entre os dados de um grupo seja máxima (ou seja, devem ser colocados em um grupo os mais similares entre si) e a similaridade entre dados colocados em grupos diferentes seja mínima (ou seja, devem ser separados em grupos diferentes os dados não similares entre si).

Como exemplo para a tarefa de agrupamento, considere um restaurante que possui vários ambientes. É interessante que clientes com características similares sejam direcionados ao mesmo ambiente, enquanto clientes com características diferentes sejam direcionados a ambientes distintos. Por exemplo, clientes jovens, que preferem ambientes mais movimentados e dinâmicos, devem ser colocados no grupo de clientes que irão para o ambiente no qual há, por exemplo, música eletrônica e uma pequena pista de dança. Já clientes com suas famílias e com crianças devem ir para o grupo direcionado para o ambiente mais próximo ao playground. Embora, nesse exemplo, a tomada de decisão muito provavelmente não será feita por um software (a menos que, em um cenário futurista, haja um funcionário recepcionista robô – o que seria muito interessante); o funcionário que tomará a decisão está analisando as características de cada cliente e os agrupando em ambientes adequados ao seu perfil. Observando a Figura 1-6, pode-se associar cada cliente a um dado $\mathbf{x}_{\rightarrow i}$ e, seguindo o esquema gráfico da figura, descobrir que os clientes $\mathbf{x}_{\rightarrow 2}$ e $\mathbf{x}_{\rightarrow 3}$ são mais similares entre si que em

relação aos clientes $\mathbf{x}_{\rightarrow 1}$ e $\mathbf{x}_{\rightarrow 4}$. Note que o agrupamento realizado sobre os dados independe de qualquer informação de rótulo.

Finalmente, a **tarefa de associação** é definida como a busca por ocorrências frequentes e simultâneas entre elementos de um contexto. Os algoritmos que resolvem essa tarefa analisam conjuntos de dados que representam eventos ou transações ($\mathbf{x}_{\rightarrow 1}$, $\mathbf{x}_{\rightarrow 2}$, $\mathbf{x}_{\rightarrow 3}$ e $\mathbf{x}_{\rightarrow 4}$), procurando por itens (x_{i1}, x_{i2} e x_{i3}) (Figura 1-6) frequentemente envolvidos nos mesmos eventos ou que apresentam algum tipo de correlação em seus comportamentos em tais eventos.

Nesse tipo de tarefa, é comum a descoberta de padrões triviais, mas o que se espera, no entanto, é que padrões inesperados sejam revelados. No contexto do restaurante, considere que cada prato seja um evento e que os itens que aparecem nos eventos sejam os ingredientes. Uma análise sobre a base de dados (as receitas de cada prato) pode revelar que "*sempre que cebola é usada no preparo de um prato, alho também é usado*". Esse é um exemplo de uma descoberta sobre uma relação óbvia entre dois itens, a cebola e o alho. Porém, regras como "*pratos nos quais se usa queijo brie também se usa geleia de pimenta*" ou "*quanto menor o teor de álcool na cerveja usada em molhos, menor a necessidade do uso de açúcar*" podem representar conhecimento novo, inesperado e interessante, a depender, é claro, do nível de conhecimento sobre gastronomia de quem recebe a resposta da resolução da tarefa de associação.

Cada uma dessas tarefas é discutida em capítulos particulares neste livro. Em cada capítulo, são apresentadas informações mais específicas, como modelagem formal e algoritmos que comumente se aplicam na resolução de cada uma.

1.5. Inteligência de negócios e tomada de decisão estratégica

Relacionados com o contexto de mineração de dados e o processo de descoberta de conhecimento em bases de dados estão os conceitos relacionados com a Inteligência de Negócios, também muito conhecida como *Business Intelligence* (BI), e referentes à tomada de decisão estratégica.

Em uma tradução livre da definição estabelecida pelo Grupo Gartner, tem-se que o termo Inteligência de Negócios pode ser visto como "*um termo guarda-chuva, que inclui aplicações, infraestrutura e ferramentas e as melhores práticas que capacitam o acesso e a análise da informação, com o intuito de melhorar e otimizar decisões de desempenho*". Em outras palavras, pode-se dizer que Inteligência de Negócios é o nome que se atribui a iniciativas para analisar informações complexas de um contexto organizacional e de apresentar o resultado de tal análise de maneira sintetizada e simples, que facilite a interpretação pelo(s) gestor(es).

Um exemplo didático sobre o que se entende por apresentação sintetizada e simples de uma informação importante pode ser visto na Figura 1-7, na qual o tamanho dos quadrados representa a variável "quantidade de vendas em um restaurante", e o posicionamento deles na estrutura matricial indica o relacionamento de uma representação de "tipo de bebida" (alcoólica e não alcoólica) com "dia da semana" (fim de semana ou dia de semana), e finalmente os tons de cinza (escuro e claro) representam resultados de uma análise de dados que indicam se o consumo está de acordo (escuro) ou abaixo do esperado (claro).

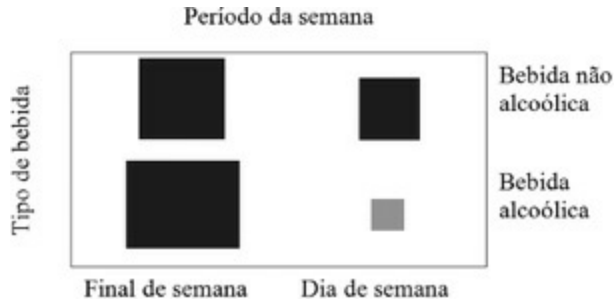


Figura 1-7: Apresentação sintetizada e simples para informações de vendas de tipos de bebidas por período da semana

O resultado da análise mostrada na Figura 1-7 pode vir da execução de um processo de KDD, implementado tanto por meio da resolução de tarefas de mineração de dados quanto pelo uso de operações analíticas (OLAP) sobre um *Data Warehouse*.

Embora o exemplo descrito seja bastante simples, ele traz informação num formato apropriado para ser usado em um processo de tomada de decisão estratégica. Um gestor de um restaurante tem, a partir da visão dada pela Figura 1-7, condições de estabelecer ações diferenciadas e principalmente adequadas para cada tipo de situação.

1.5.1. OLTP *versus* OLAP

No contexto dos sistemas computacionais em uma organização, principalmente as que pretendem adotar processos voltados para a área de Inteligência de Negócios, é importante que se tenha uma visão clara e diferenciada sobre duas classes importantes de sistemas: OLTP e OLAP.

- OLTP (*Online Transaction Processing*): trata-se de uma classe de sistemas amplamente presente em todos os tipos de organizações que possuem processos de negócios automatizados. Sistemas pertencentes a essa classe suportam registro e controle de todas as transações rotineiras que dão condições ao funcionamento de uma organização. Exemplos de sistemas OLTP: sistemas de gerenciamento de recursos humanos, de contabilidade, de controle de chão de fábrica, de operações bancárias,

de controle de cartões de créditos, de gerenciamento de passagens aéreas etc. São sistemas que operam sobre bases de dados transacionais, comumente gerenciadas por sistemas gerenciadores de banco de dados relacionais.

- OLAP (*Online Analytical Processing*): trata-se de uma classe de sistemas menos presente nas organizações, mas com taxa de uso em franca expansão. Sistemas pertencentes a essa classe suportam o armazenamento, gerenciamento e análise de dados históricos, geralmente derivados de diferentes fontes de dados, incluindo os sistemas OLTP. São sistemas direcionados para gestores que precisam de informações resumidas e de alto nível sobre os negócios de uma organização. Tais sistemas operam sobre bases de dados históricas e multidimensionais, comumente gerenciadas por sistemas gerenciadores de banco de dados relacionais integrados com ferramentas específicas para carga e análise de dados multidimensionais.

Ambas as classes de sistemas são importantes para as organizações, porém, devem receber tratamento adequado quando usadas como fonte de dados ou fonte de conhecimento no processo que suporta uma tomada de decisão estratégica.

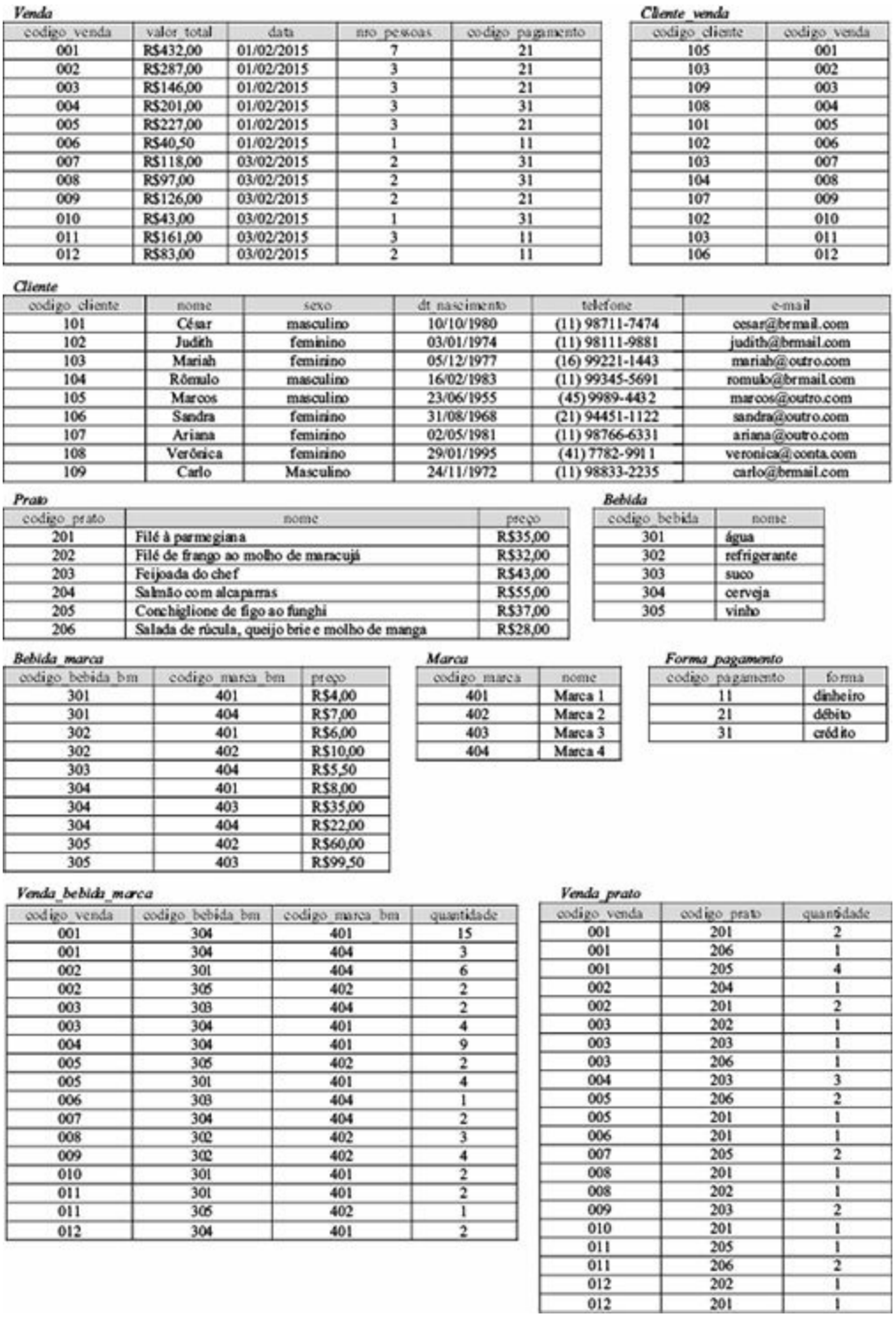
1.5.2. Dados transacionais *versus* Dados históricos

Na diferenciação entre sistemas OLTP e OLAP, há dois conceitos que influenciam diretamente o entendimento do propósito de cada classe de sistemas: **dados transacionais** e **dados históricos**. A fim de entendê-los, considere um sistema de gerenciamento do restaurante (OLTP) que opera sobre uma base de dados relacional, capaz de armazenar dados sobre as transações rotineiras (**dados transacionais**) que ocorrem no contexto automatizado pelo sistema. Um **modelo relacional** parcial para tal base de dados é mostrado na Figura 1-8.

Cliente =(@codigo_cliente, nome, sexo, data_nascimento, telefone, email)
Prato = (@codigo_prato, nome, preço)
Bebida = (@codigo_bebida, nome)
Marca = (@codigo_marca, nome)
Bebida_marca = (@#codigo_bebida_bm, @#codigo_marca_bm, preço)
Forma_pagamento = (@codigo_pagamento, forma)
Venda = (@codigo_venda, valor_total, data, nro_pessoas, #codigo_pagamento)
Venda_prato = (@#codigo_venda, @#codigo_prato, quantidade)
Venda_bebida_marca = (@#codigo_venda, @#codigo_bebida_bm, @#codigo_marca_bm, quantidade)
Cliente_venda = (@#codigo_cliente, @#codigo_venda)

Figura 1-8: Modelo relacional parcial de uma base de dados de um sistema OLTP para gerenciamento de um restaurante, sendo que @ representa o atributo “chave primária”, e # o atributo “chave estrangeira”

O modelo expressa o relacionamento entre “clientes, vendas, produtos e formas de pagamento” que ocorrem no restaurante. A regra de negócio que gera os dados a serem armazenados nessa base de dados diz que é necessário armazenar o cliente que realiza o pagamento da venda (já cadastrado no sistema), a forma de pagamento (dentre as também já cadastradas no sistema), o número de pessoas que consumiram produtos nessa venda, quais produtos foram consumidos e o valor total da venda. Note que, nos dados que descrevem o cliente, há um e-mail de contato. Caso o cliente troque de *e-mail* ou decida que outro deve ser usado, o valor do atributo deverá ser atualizado no sistema. Ou seja, os dados transacionais armazenados na base de dados relacional dos sistemas da classe OLTP normalmente são referentes ao estado atual das entidades do mundo real modeladas no sistema. Instâncias de relações correspondentes ao modelo em discussão são mostradas na Figura 1-9.



Venda

codigo_venda	valor_total	data	nro_pessoas	codigo_pagamento
001	R$432,00	01/02/2015	7	21
002	R$287,00	01/02/2015	3	21
003	R$146,00	01/02/2015	3	21
004	R$201,00	01/02/2015	3	31
005	R$227,00	01/02/2015	3	21
006	R$40,50	01/02/2015	1	11
007	R$118,00	03/02/2015	2	31
008	R$97,00	03/02/2015	2	31
009	R$126,00	03/02/2015	2	21
010	R$43,00	03/02/2015	1	31
011	R$161,00	03/02/2015	3	11
012	R$83,00	03/02/2015	2	11

Cliente_venda

codigo_cliente	codigo_venda
105	001
103	002
109	003
108	004
101	005
102	006
103	007
104	008
107	009
102	010
103	011
106	012

Cliente

codigo_cliente	nome	sexo	dt_nascimento	telefone	e-mail
101	César	masculino	10/10/1980	(11) 98711-7474	cesar@bmail.com
102	Judith	feminino	03/01/1974	(11) 98111-9881	judith@bmail.com
103	Mariah	feminino	05/12/1977	(16) 99221-1443	mariah@outro.com
104	Rômulo	masculino	16/02/1983	(11) 99345-5691	romulo@bmail.com
105	Marcos	masculino	23/06/1955	(45) 9989-4432	marcos@outro.com
106	Sandra	feminino	31/08/1968	(21) 94451-1122	sandra@outro.com
107	Ariana	feminino	02/05/1981	(11) 98766-6331	ariana@outro.com
108	Verônica	feminino	29/01/1995	(41) 7782-9911	veronica@conta.com
109	Carlo	Masculino	24/11/1972	(11) 98833-2235	carlo@bmail.com

Prato

codigo_prato	nome	preço
201	Filé à parmegiana	R$35,00
202	Filé de frango ao molho de maracujá	R$32,00
203	Feijoada do chef	R$43,00
204	Salmão com alcaparras	R$55,00
205	Conchiglione de figo ao funghi	R$37,00
206	Salada de rúcula, queijo brie e molho de manga	R$28,00

Bebida

codigo_bebida	nome
301	água
302	refrigerante
303	suco
304	cerveja
305	vinho

Bebida_marca

codigo_bebida_bm	codigo_marca_bm	preço
301	401	R$4,00
301	404	R$7,00
302	401	R$6,00
302	402	R$10,00
303	404	R$5,50
304	401	R$8,00
304	403	R$35,00
304	404	R$22,00
305	402	R$60,00
305	403	R$99,50

Marca

codigo_marca	nome
401	Marca 1
402	Marca 2
403	Marca 3
404	Marca 4

Forma_pagamento

codigo_pagamento	forma
11	dinheiro
21	débito
31	crédito

Venda_bebida_marca

codigo_venda	codigo_bebida_bm	codigo_marca_bm	quantidade
001	304	401	15
001	304	404	3
002	301	404	6
002	305	402	2
003	303	404	2
003	304	401	4
004	304	401	9
005	305	402	2
005	301	401	4
006	303	404	1
007	304	404	2
008	302	402	3
009	302	402	4
010	301	401	2
011	301	401	2
011	305	402	1
012	304	401	2

Venda_prato

codigo_venda	codigo_prato	quantidade
001	201	2
001	206	1
001	205	4
002	204	1
002	201	2
003	202	1
003	203	1
003	206	1
004	203	3
005	206	2
005	201	1
006	201	1
007	205	2
008	201	1
008	202	1
009	203	2
010	201	1
011	205	1
011	206	2
012	202	1
012	201	1

Figura 1-9: Exemplo de instâncias para as relações da base de dados referente à Figura 1-8

Já para o contexto de um sistema da classe OLAP, é esperado que os dados estejam organizados sob uma perspectiva histórica e multidimensional. Então, um modelo de dados diferente deve ser considerado a fim de atender a essa expectativa de organização. Um modelo comumente usado para modelagem histórica e multidimensional de dados é o **modelo estrela**, em que uma relação central (tabela fato) armazena dados sobre os fatos que se quer analisar e relações que orbitam essa relação central armazenam dados que oferecem informação sob diferentes dimensões em relação a um fato (tabelas dimensão). Na Figura 1-10, é apresentado um modelo estrela simplificado, que pode ser usado para armazenar dados históricos derivados dos dados transacionais do modelo relacional apresentado na Figura 1-8. Instâncias para as tabelas do modelo estrela são mostradas na Figura 1-11. Note que a tabela fato está instanciada com a menor granularidade de valores presentes nas tabelas dimensão associadas.

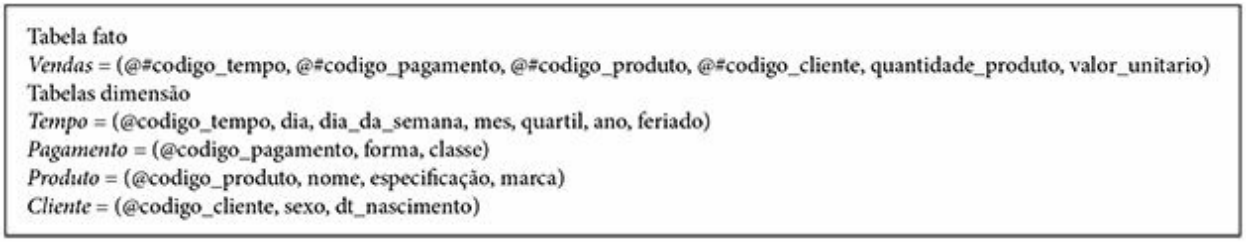


Tabela fato
Vendas = (@#codigo_tempo, @#codigo_pagamento, @#codigo_produto, @#codigo_cliente, quantidade_produto, valor_unitario)
Tabelas dimensão
Tempo = (@codigo_tempo, dia, dia_da_semana, mes, quartil, ano, feriado)
Pagamento = (@codigo_pagamento, forma, classe)
Produto = (@codigo_produto, nome, especificação, marca)
Cliente = (@codigo_cliente, sexo, dt_nascimento)

Figura 1-10: Modelo estrela simplificado de uma base de dados de um sistema OLAP para suporte à decisão em processos de negócios de um restaurante

1.5.3. Extração de dados *versus* Análise de dados

No contexto de estudo deste livro, é importante comentar sobre a atividade de extração de dados, comumente realizada em sistemas da classe OLTP, a qual se dá na forma de consultas implementadas na linguagem de consultas SQL (*Structured Query Language*, ou linguagem de consulta estruturada), linguagem-padrão de consulta declarativa para banco de dados relacional. Essa extração de dados, embora eficiente para os propósitos de um sistema da classe OLTP, não se configura como tarefa de análise de dados,

muito embora, a depender de como os resultados da extração sejam usados, informações importantes possam ser acessadas.

Considere, por exemplo, que o dono do restaurante deseja fazer uma promoção de cervejas. Para isso, requisita a informação de quais marcas de cervejas vêm sendo consumidas em uma venda com quantidade acima de cinco unidades. A extração de dados requerida é possível a partir da seguinte instrução SQL:

```
SELECT M.nome, VBM.quantidade
FROM Bebida B, Marca M, Venda_bebida_marca VBM
WHERE B.codigo_bebida = VBM.codigo_bebida_bm AND
M.codigo_marca = VBM.codigo_marca_bm AND B.nome =
‘cerveja’ AND VBM.quantidade >= 5;
```

Esta consulta resultará em uma lista com a marca das cervejas e as respectivas quantidades vendidas em cada venda, que atendem ao predicado da cláusula WHERE. Essa instrução em SQL foi feita para responder a uma requisição simples, que não exige uma análise mais detalhada dos perfis de consumo dos clientes do restaurante, por exemplo. Portanto, essa simples consulta não apresenta evidências de que poderá haver aumento dos lucros no restaurante a partir de uma promoção de cervejas. Consultas em SQL mais sofisticadas, inclusive fazendo uso de operações agregadas, poderiam também ser implementadas, mas ainda estariam apenas satisfazendo a necessidade de extração de dados que algum usuário requisitou.

Tempo (dimensão)

codigo_tempo	dia	dia_da_semana	mes	quartil	ano	feriado
001	01	domingo	fevereiro	1	2015	Não
002	03	terça	fevereiro	1	2015	Não

Pagamento (dimensão)

codigo_pagamento	forma	classe
001	dinheiro	à vista
002	débito	cartão
003	crédito	cartão

Cliente (dimensão)

codigo_cliente	sexo	dt_nascimento
001	masculino	10/10/1980
002	feminino	03/01/1974
003	feminino	05/12/1977
004	masculino	16/02/1983
005	masculino	23/06/1955
006	feminino	31/08/1968
007	feminino	02/05/1981
008	feminino	29/01/1995
009	masculino	24/11/1972

Produto (dimensão)

codigo_produto	nome	especificacao	marca
001	Filé à parmegiana	prato	fabricação própria
002	Filé de frango ao molho de maracujá	prato	fabricação própria
003	Feijoada do chef	prato	fabricação própria
004	Salmão com alcaparras	prato	fabricação própria
005	Conchiglione de figo ao funghi	prato	fabricação própria
006	Salada de rúcula, queijo brie e molho de manga	prato	fabricação própria
007	água	bebida não alcoólica	marca 1
008	água	bebida não alcoólica	marca 4
009	refrigerante	bebida não alcoólica	marca 1
010	refrigerante	bebida não alcoólica	marca 2
011	suco	bebida não alcoólica	marca 4
012	cerveja	bebida alcoólica	marca 1
013	cerveja	bebida alcoólica	marca 3
014	cerveja	bebida alcoólica	marca 4
015	vinho	bebida alcoólica	marca 2
016	vinho	bebida alcoólica	marca 3

Vendas (fato)

codigo_tempo	codigo_pagamento	codigo_produto	codigo_cliente	quantidade_produto	valor_unitario
001	001	001	002	1	R$35,00
001	001	011	002	1	R$5,50
001	002	001	001	1	R$35,00
001	002	006	001	2	R$28,00
001	002	015	001	2	R$60,00
001	002	007	001	4	R$4,00
001	002	001	003	2	R$35,00
001	002	004	003	1	R$55,00
001	002	008	003	6	R$7,00
001	002	015	003	2	R$60,00
001	002	001	005	2	R$35,00
001	002	005	005	4	R$37,00
001	002	006	005	1	R$28,00
001	002	012	005	15	R$8,00
001	002	014	005	3	R$22,00
001	002	002	009	1	R$32,00
001	002	003	009	1	R$43,00
001	002	006	009	1	R$28,00
001	002	011	009	2	R$5,50
001	002	012	009	4	R$8,00
001	003	003	008	2	R$43,00
001	003	012	008	9	R$8,00
002	001	005	003	2	R$37,00
002	001	014	003	2	R$22,00
002	001	001	006	1	R$35,00
002	001	002	006	1	R$32,00
002	001	012	006	2	R$8,00
002	002	003	007	2	R$43,00
002	002	010	007	4	R$10,00
002	003	001	002	1	R$35,00
002	003	007	002	2	R$4,00
002	003	005	003	1	R$37,00
002	003	006	003	2	R$28,00
002	003	007	003	2	R$4,00
002	003	015	003	1	R$60,00
002	003	001	004	1	R$35,00
002	003	002	004	1	R$32,00
002	003	010	004	3	R$10,00

Figura 1-11: Exemplo de instâncias para as relações da base de dados referente à Figura 1-10

O uso do modelo de dados estrela oferece meios de, a partir de consultas SQL ou com o uso de outros recursos, extrair informação mais refinada,

atendendo à requisição, por exemplo, de extrair os perfis de consumo de clientes ou perfil de consumo por sazonalidade. Uma forma de alcançar esse objetivo e extrair dados que permitam a criação de informação, como mostrado na Figura 1-7, é a construção de "cubos" – estruturas de dados, que permitem organizar os dados de forma a proporcionar diversas visões multidimensionais dos mesmos. A Figura 1-12 mostra a representação gráfica de um cubo, e sua representação tabular segue apresentada na Figura 1-13. Note que as informações mostradas neste cubo são, na realidade, a realização de agregações das dimensões *Produto, Forma de pagamento* e *Tempo.*

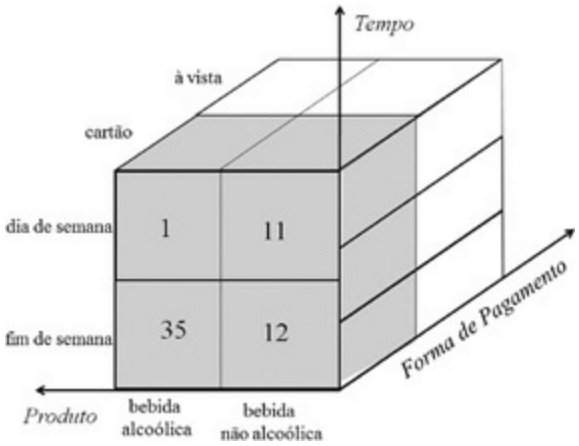


Figura 1-12: Representação gráfica de um cubo de três dimensões (Produto, Forma de Pagamento, Tempo)

Cubo – Vendas

Produto	*Pagamento*	*Tempo*	*Qtd-vendas*
bebida alcoólica	cartão	dia de semana	1
		fim de semana	35
	à vista	dia de semana	4
		fim de semana	0
bebida não alcoólica	cartão	dia de semana	11
		fim de semana	12
	à vista	dia de semana	0
		fim de semana	1

Figura 1-13: Representação tabular das informações presentes no cubo da Figura 1-12

1.6. Considerações finais

Mineração de Dados, como já deve estar claro ao leitor, é uma área que existe porque também existe uma grande quantidade de dados disponível para ser explorada. É importante perceber que as evoluções teórica e prática (aplicações) de mineração de dados vêm gradativamente ocorrendo, uma vez que a cada ano há um significativo aumento no volume e na complexidade de dados disponível para análise.

Com a evolução tecnológica marcada tanto pelo aumento da capacidade de armazenamento quanto pelo seu barateamento, pela popularização de *gadgets* (por exemplo, *smartphones* e *tablets*) e pela massificação do uso da internet, muito influenciada inclusive pelas redes sociais, evoluiu também a capacidade de geração de dados, não apenas os do tipo estruturados, mas também os não estruturados (multimídia). Dados como imagem e vídeo, além das mensagens de texto trocadas nas redes sociais, provocaram mudanças de paradigmas no armazenamento e gerenciamento de dados, muito devido ao seu grande volume, mas, sobretudo, devido ao seu formato.

Há, portanto, motivos para crer que a tomada de decisão em qualquer que seja o domínio de negócio (empresarial, educacional, político, social etc) deve ser feita com base em dados (fatos) disponíveis nos mais variados formatos e armazenados nas mais diferentes mídias. Contudo, embora a informatização seja praticamente pervasiva no gerenciamento de processos transacionais, não o é ainda, infelizmente, presente em todo o gerenciamento de processos analíticos. E parte desse problema está justamente na complexidade da automatização de processos desse tipo, que exige profissionais altamente qualificados (o que significa alto custo) e quantidade e qualidade de dados (o que também representa alto custo).

Neste ponto, o leitor deve estar se perguntando: “Alto custo em relação aos dados? Mas o custo do armazenamento dos dados não está decaindo com o tempo?” A resposta é: sim, o custo de armazenamento está decaindo, porém

o custo do gerenciamento do grande volume de dados ainda é muito alto devido ao conhecimento especializado necessário para que tal gerenciamento seja feito de forma adequada.

De maneira geral, existe um conjunto de teorias, métodos, processos, arquiteturas e tecnologias para suportar o gerenciamento de dados e a extração de conhecimento a partir dos dados, como comentado na Seção 1.5. Embora muito se tenha evoluído nesta área,[3] soluções para suportar a Inteligência de Negócios ainda são caras. Contudo, a necessidade de tal suporte é tão veemente que os investimentos na área estão em franca expansão.

Uma forma interessante de acompanhar como a evolução das tecnologias relacionadas com a mineração de dados está ocorrendo e sendo vista pela indústria e pelos consumidores é acompanhar as pesquisas anualmente feitas pela Gartner[4] que originam os gráficos Hype Cycle .[5] Esses gráficos relacionam o nível de maturidade de uma tecnologia com a visibilidade alcançada até o momento de realização da pesquisa. Nessa relação, as tecnologias são posicionadas em cinco grandes partições do espaço:[6] a primeira, originalmente chamada de gatilho tecnológico (*technology trigger*), abrange tecnologias que estão passando pelas primeiras provas de conceito e já recebem uma atenção significativa da mídia especializada; a segunda partição, chamada de pico de grandes expectativas (*peak of inflated expectations*), mostra tecnologias que estão recebendo atenção de esforços de pesquisa e desenvolvimento, com alguns casos de falha e outros de sucesso e, nesse caso, recebendo já atenção de parte da indústria; na próxima partição, vale de desilusão (*trough of disillusionment*), os investimentos na tecnologia tornam-se mais tímidos, visto que a visibilidade da tecnologia começa a decair; na partição encosta do esclarecimento (*slope of enlightenment*), a utilidade da tecnologia e o quão benéfica ela pode ser começam a ficar mais frequentes e mais bem entendidas pela comunidade, além disso, novas versões (mais maduras) dos produtos derivados da tecnologia aparecem, e os

investimentos começam a voltar; finalmente, na última partição, chamada platô de produtividade (*plateau of productivity*), a aplicabilidade e relevância da tecnologia se tornam claras e confiáveis.

A fim de mostrar como as tecnologias relacionadas com mineração de dados vêm se posicionando nas pesquisas da Gartner, os gráficos da Figura 1-14, baseados no próprio gráfico Hype Cycle , apresentam um resumo histórico de cinco anos de informações advindas dos gráficos apresentados nos anos 2010,[7] 2011,[8] 2012,[9] 2013[10] e 2014.[11] A informação apresentada na síntese da Figura 1-14(a,b) representa a evolução de alguns tópicos relacionados, direta ou indiretamente, com a área de Mineração de Dados, no que diz respeito aos seus níveis de maturidade e visibilidade. Oito tópicos foram escolhidos, dentre os citados em cada um dos gráficos individuais: ciência de dados, análise de conteúdo, análise preditiva, análise prescritiva, análise social, análise de texto, análise e busca de vídeo, e processamento extremo de transações e *Big Data*. Interessante destacar que, em gráficos Hype Cycle anteriores ao ano de 2010, esses tópicos eram praticamente inexistentes. Alguns tópicos são citados em mais de um dos gráficos Hype Cycle originais e, por isso, aparecem mais de uma vez no resumo histórico aqui apresentado. Nesses casos, é possível observar como se deu a evolução de cada tópico durante alguns anos.

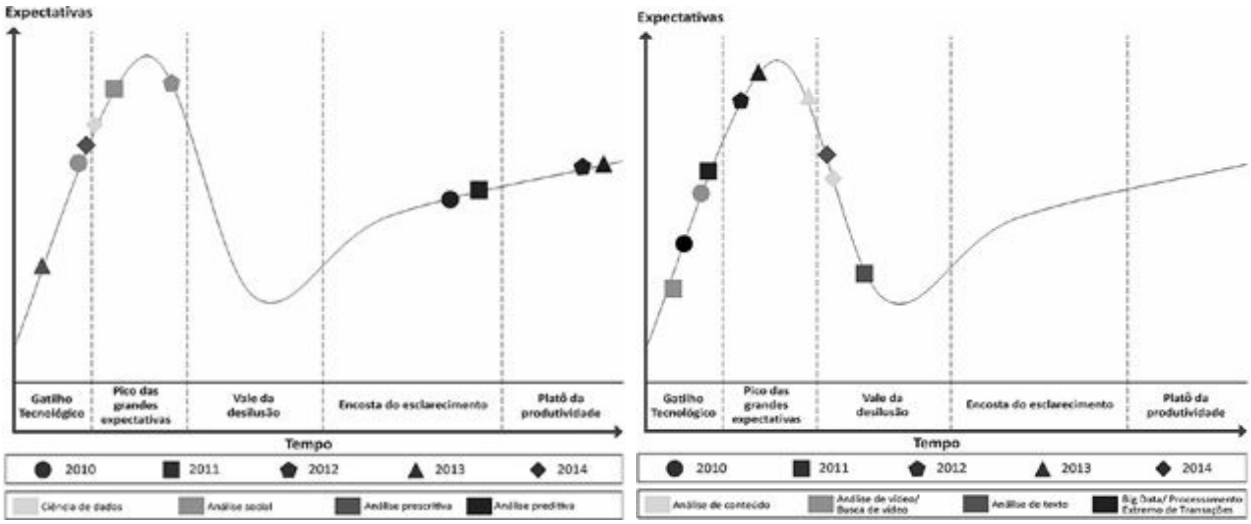


Figura 1-14: Perspectiva histórica de tecnologias relacionadas com Mineração de Dados na visão das pesquisas da Gartner

Há algumas considerações sobre a informação apresentada nesse resumo histórico especialmente importantes para o leitor interessado em aprender sobre mineração de dados. Ao olhar para as regiões "encosta do esclarecimento" e "platô da produtividade", constata-se a presença do tópico análise preditiva durante quatro anos (2010 a 2013), o que significa que se trata de uma área do conhecimento já, felizmente, madura e visível o suficiente para suportar a geração de aplicações industriais. No entanto, isso não quer dizer que o tópico não tenha ainda de ser estudado e aprimorado.

Big Data é um tópico que, hoje, "está na moda". Confirmando esse fato, a informação provinda das pesquisas da Gartner posicionam este tópico, de 2010 a 2013, na região de "entusiasmo de desenvolvimento." Entretanto, há, em 2014, uma tendência não tão entusiástica, uma vez que o assunto perde já um pouco de sua visibilidade e passa a ser um tópico que recebe a atenção de uma parcela menor de organizações de pesquisa ou desenvolvimento. Contudo, é esperado que, embora a atenção seja um pouco menor, ela deva ser mais especializada e que se deva trabalhar para que o tópico alcance um nível de maturidade mais alto.

Os tópicos ciência de dados e análise prescritiva aparecem nos gráficos, nesses termos,[12] apenas a partir de 2013 e 2014, respectivamente. Especialmente sobre análise prescritiva, é importante informar que se trata de uma terceira forma de análise que poderia ser inserida na taxonomia, apresentada na Seção 1.4.3, no mesmo nível da análise descritiva e preditiva. Trata-se de um termo definido no início dos anos 2000 e que diz respeito à tarefa de analisar não apenas o que e quando vai acontecer no futuro, mas também analisar por que irá acontecer.

Outros três tópicos encontram-se na região do "vale das desilusões": análise de conteúdo, análise social e análise de textos, mostrando que tais assuntos estão evoluindo de forma similar ao tópico *Big Data*. Uma exceção

nesse resumo histórico é o tópico análise de vídeo, ainda na primeira divisão do gráfico, tratando-se de um assunto com pouca maturidade e para o qual os interesses ainda estão se manifestando, porém, com tendência a receber mais atenção nos próximos anos.

1.7. Organização do livro

O público-alvo deste livro são **leitores iniciantes** na área de Mineração de Dados. De fato, o texto aqui apresentado foi construído com o objetivo de oferecer um ambiente de aprendizado por meio do qual um estudante, ou um altivo curioso, pudesse iniciar seus estudos já analisando como a teoria de mineração de dados se aplica em problemas reais, já praticando o uso desses conceitos para fixação. O livro está organizado em cinco capítulos, incluindo esta introdução e um apêndice – o qual versa sobre conceitos básicos e estratégias necessárias para a reprodução dos códigos em R apresentados nos demais capítulos. Os capítulos são, sempre que possível, organizados seguindo a mesma estrutura: primeiramente, os conceitos teóricos são apresentados, a seguir, um exemplo didático é usado para ilustrar a aplicação dos conceitos e, na sequência, uma implementação em R é apresentada. Por fim, leituras adicionais e exercícios são sugeridos. Cada uma das tarefas de mineração de dados tratadas neste livro é apresentada em um capítulo diferente, de forma que o leitor interessado em apenas uma delas pode se dirigir ao respectivo capítulo.

R é um ambiente de software livre para implementação de funções estatísticas e para criação de gráficos (R Core Team , 2015), o que o torna muito útil para ser aplicado em análise de dados. Existe uma série de pacotes disponíveis para uso nesse ambiente, entre eles, os que permitem a implementação e teste de algoritmos usados na resolução das tarefas de mineração de dados.

O primeiro capítulo deste livro tem o objetivo de contextualizar o leitor sobre em que consiste o tópico “mineração de dados”, onde, como e por quê a mineração de dados pode ser útil e, em linhas gerais, quais são os objetivos das principais tarefas da área: análise preditiva, análise de agrupamento e descobertas de associações. Além disso, o capítulo traz uma revisão sobre conceitos que margeiam a área de Mineração de Dados, importantes para

diferenciá-la de áreas correlatas. A leitura deste capítulo é importante para que as expectativas corretas sobre o restante do livro sejam formadas e também para que o leitor conheça a situação hipotética – um restaurante – sobre a qual são inspirados todos os exemplos usados no decorrer do livro.

Medidas estatísticas e visualizações gráficas constituem o assunto que inicia o Capítulo 2. Um leitor, já conhecedor dos conceitos básicos de estatística e também acostumado ao uso de gráficos para exploração visual de dados, poderia seguir diretamente para a segunda parte do capítulo, que traz algumas práticas usuais na fase de pré-processamento de dados. E, finalizando o capítulo, discute-se como se dá a preparação de dados textuais quando objeto da mineração de dados.

O Capítulo 3 apresenta quatro algoritmos para a resolução da tarefa de predição categórica – a classificação – e dois algoritmos para a resolução da tarefa de predição numérica – a regressão. Nessa apresentação, são discutidas diversas definições relacionadas com ambas as formas de predição e como cada um dos algoritmos pode ser modelado para solucioná-las adequadamente. Ainda são introduzidas algumas medidas de avaliação para modelos preditivos. São tratadas medidas referentes a taxas de erro e aquelas derivadas da matriz de confusão. Enfim, com a análise preditiva resolvida a partir de algoritmos de aprendizado indutivo, algumas técnicas para treinamento, validação e teste de algoritmos são discutidas.

Para a resolução da tarefa de análise de agrupamento são apresentadas quatro estratégias no Capítulo 4: agrupamento hierárquico, agrupamento por partição, agrupamento por densidade e agrupamento via uma estratégia conexionista. Para fornecer ao leitor o conhecimento sobre como avaliar os resultados de uma análise de agrupamento, alguns índices externos e internos são também discutidos.

O Capítulo 5 é dedicado ao contexto de resolução da tarefa de descoberta de regras de associação. Nesse momento da leitura do livro, o leitor poderá sentir uma mudança nas nomenclaturas usadas ou até mesmo na notação. Isso

ocorre porque a forma de lidar com um conjunto de dados é diferente na tarefa de descoberta de regras de associação, quando comparada à forma usada nas tarefas de análise preditiva e análise de agrupamento. Solicita-se ao leitor que permita alguma liberdade no uso da notação, sem prejuízo para a interpretação e corretude do conteúdo apresentado. Dois algoritmos muito conhecidos são discutidos neste capítulo.

1.8. Leituras adicionais

A discussão sobre definições para dados, informação e conhecimento vai muito além do conteúdo brevemente discutido neste capítulo. Para iniciar um estudo sobre esse assunto, é sugerida a leitura do texto de Setzer (Setzer, 2001 – Capítulo 11). Nele, o autor apresenta sua visão sobre os termos dados, informação e conhecimento, provendo definições não usuais, porém amplas, desses termos. Também, a indicação do livro "*Infoscience: Turning Information into Knowledge*" (Devlin, 1999) se faz interessante àqueles que querem se aprofundar na diferenciação desses conceitos e em seus benefícios para a gestão da informação. Uma visão mais gerencial de informação pode ser encontrada em Laudon e Laudon (2015).

Mineração de dados aplicada a dados não estruturados do tipo imagem, vídeo e som é muito popular hoje em dia. Além disso, a mineração de dados envolvendo todos esses tipos juntos, em uma análise multimídia, tem sido cada vez mais necessária para a descoberta de conhecimento no conteúdo gerado na Web e em outros meios. Esse assunto, em específico, não é tratado neste livro, e, para que o leitor inicie um estudo sobre ele, seguem as indicações de leituras referentes a pré-processamento desses dados e mineração: (Alcain e Oliveira, 2011), (Zhang e Zhang 2008) e (Gonzalez e Woods, 2011), (Prandoni e Vetterli, 2008) e (Fisher *et al*., 2000).

Lógica é, sem dúvida, a base das técnicas que permitem a Mineração de Dados. O raciocínio lógico pode assumir diferentes formas, e estudar essa área é fascinante. Uma infinidade de livros poderia ser indicada, e, sem desmerecimento da maioria, uma leitura bastante interessante sobre esse assunto é o livro "Introdução à Lógica", de Irving M. Copi (Copi, 1981).

Conceitos de *Data Warehouse* , OLAP e Inteligência de Negócios não podem ficar de fora do planejamento de estudo de quem quer se especializar na área. Ralph Kimball e William Harvey Inmon são considerados os precursores da teoria de *Data Warehouse*, diferindo em termos de

metodologia de projeto, sendo que o primeiro defende uma abordagem *bottom-up*, ou seja, a criação de vários *Data Marts* (mini *Data Warehouse*) orientados por assunto que, interligados, *a posteriori* gerariam o *Data Warehouse* corporativo (Kimball e Margy, 2002), e o segundo preconiza uma abordagem *top-down*, ou seja, parte-se do projeto de um único modelo corporativo (o *Data Warehouse*) para, então, chegar aos *Data Marts* departamentais (Inmon, 2005).

Para se aprofundar em Inteligência de Negócios, seus conceitos, componente e tecnologias, é recomendada a obra de Turban (2009).

1.9. Exercícios

As situações a seguir são exemplos nos quais algum tipo de análise de dados se faz útil, seja as definidas na área de Mineração de Dados, seja análise de naturezas diferentes, como as realizadas sobre bases de dados relacionais e multidimensionais. Faça uma reflexão analítica em cada caso e desenvolva propostas para análise de dados. Reflita sobre o processo completo de descoberta de conhecimento em base de dados e, para cada situação, identifique no que se constituiria cada uma das fases do processo. Especificamente para a fase de mineração de dados, identifique qual(is) tarefa(s) pode(m) ser empregada(s).

a) Uma faculdade decide criar uma página em uma rede social. Os participantes desejáveis nessa rede são professores, alunos e funcionários da própria instituição. As publicações desejáveis pela faculdade são as que devem, de alguma maneira, ser de algum interesse aos participantes. A rede social possibilita acesso aos metadados dos usuários (idade, cidade de origem etc.) e aos dados referentes às publicações realizadas (texto, horário, data etc.) e suas repercussões (comentários, aprovações – curtidas e compartilhamentos).

b) Uma empresa de cartões de crédito armazena todas as operações de compra dos clientes. Em cada transação, além do valor e quantidade de parcelas, a operadora ainda tem a informação do tipo de estabelecimento (alimentação, vestuário, entretenimento etc.). A operadora possui uma base de dados histórica referente às transações que cada cliente executou com seus cartões atuais e com cartões antigos atualmente inativados. O cliente de uma operadora de cartões de crédito tem seus cartões roubados. Os ladrões começam a realizar compras com os cartões em estabelecimentos comerciais de diferentes cidades, em curtos espaços de tempo.

c) Uma loja de comércio eletrônico armazena o histórico de navegação de cada usuário e um histórico de produtos pesquisados. A loja também registra todos os produtos comprados e a forma de pagamento usada em cada transação. A partir dessas informações, a loja deseja estabelecer alguma estratégia para incentivar o consumo de produtos à venda.

d) Alguns analistas do mercado financeiro possuem dados de diferentes naturezas que permitem conhecer o ambiente econômico e político de um país. A partir dessas informações, esses analistas têm condições de determinar valores para certos índices e taxas referentes à economia de um país.

e) Uma escola do ensino médio resolve organizar seus alunos a fim de descobrir quais estão capacitados para participar de maratonas de matemática, programação etc. Para isso, a escola usa as informações de notas que os alunos obtiveram nas diferentes avaliações e atividades de diferentes matérias, informações sobre preferências dos alunos por conteúdos, habilidades de resolução de problemas rápida e eficientemente, dados observados sobre os alunos em relação ao rendimento sob situações de pressão.

f) Um supermercado armazena as informações sobre os produtos, como preço, data de validade etc., e sobre todas as compras realizadas diariamente, com informações que identificam o caixa, o tempo gasto na contabilização das compras, os produtos comprados e a forma de pagamento. A partir dessas informações, o gerente do supermercado deseja melhorar os índices de venda do estabelecimento.

CAPÍTULO 2

Análise exploratória

A mineração de dados é, como já discutido na introdução deste livro, uma das fases do processo de descoberta de conhecimento, o qual envolve ainda, pelo menos, três outras fases importantes: coleta de dados , pré-processamento de dados e o pós-processamento dos resultados provenientes da mineração. Sob o ponto de vista da fase de mineração de dados, é preciso saber que não será possível obter bons resultados se dois pré-requisitos não forem atendidos: (a) o analista de dados precisa conhecer o contexto em que os dados estão inseridos e como eles ocorrem nesse contexto; (b) o analista de dados precisa executar procedimentos que tornem o conjunto de dados o mais adequado possível para a etapa de mineração de dados.

Sabe-se também que a mineração de dados se faz muito útil quando a quantidade de dados disponível é grande e representativa – motivo pelo qual a fase de coleta de dados e a tarefa de amostragem são muito importantes no processo de descoberta de conhecimento. Os autores Tamhane e Dunlop (1999) e Liu e Motoda (2001) trazem discussões sobre essa fase. Uma vez que a quantidade de dados disponível para análise extrapola a capacidade humana para uma investigação manual, torna-se imprescindível o uso de ferramentas que tenham a capacidade de mostrar diferentes aspectos dos dados, a fim de fornecer “pistas” sobre eles, de forma que seja possível fundamentar escolhas referentes a métodos de análise adotados para implementar as ações de mineração de dados. Uma das ferramentas mais

úteis nesse processo de exploração inicial dos dados é a estatística descritiva. Aspectos importantes dos conjuntos de dados podem ser obtidos por meio da aplicação de medidas de tendência central, dispersão e correlação, ou também pelo uso de recursos gráficos para visualização dessas e outras medidas.

Além disso, a amostra de dados disponível para análise pode conter uma série de imprecisões e desvios ou pode estar representada de maneira inadequada. Esses fatores influenciam negativamente qualquer tipo de análise de dados, e, portanto, estratégias de pré-processamento de dados que podem amenizar tais efeitos negativos se somam ao contexto de exploração de dados tratado neste capítulo. Vale destacar que conceitos de **estatística descritiva** são também úteis para o planejamento de estratégias de pré-processamento, pois suportam a verificação da presença de ruídos (atributos cujo conjunto de valores varia acima das extremidades dos quartis), a necessidade de transformação de valores (com uso da média e desvio-padrão) ou a utilidade da seleção de dados ou atributos (com uso de medidas de correlação), por exemplo. Nessa linha de raciocínio, merece destaque também a utilidade da análise de frequência na preparação de conjuntos de dados.

A informação adquirida na análise exploratória , sem dúvida, apoia a tomada de decisão sobre o tipo de tarefa de mineração de dados e sobre o algoritmo mais adequado a um contexto. Um breve exemplo para ilustrar essa afirmação está na identificação do tipo de valores que os atributos descritivos assumem (valores numéricos ou valores categóricos), pois esse pode ser um dos fatores em favor da escolha do algoritmo a ser aplicado.

Os conceitos tratados neste capítulo também são úteis para a execução da etapa de pós-processamento no processo de descoberta de conhecimento. Embora tal etapa não seja um tópico diretamente tratado neste livro, é interessante saber que medidas de tendência central, dispersão e outras são importantes para análises e comparação de resultados.

A fim de atender a necessidade exposta aqui, este capítulo é dedicado a apresentar alguns conceitos de estatística descritiva, com o objetivo de

proporcionar um ambiente para a realização da análise exploratória de dados. Além disso, são discutidos alguns procedimentos de pré-processamento de dados, incluindo a preparação de dados do tipo texto organizados em um *corpus*, para submissão a algoritmos de mineração de dados.

2.1. Conceitos de estatística descritiva

A estatística descritiva pode ser entendida como uma ferramenta capaz de descrever ou resumir dados, mostrando aspectos importantes do conjunto de dados, como o tipo de distribuição associada e os valores mais representativos do conjunto, e permitindo criar visualizações referentes a tais aspectos.

2.1.1. População, amostra e variáveis

Os conceitos de população , amostra e variáveis são brevemente comentados nesta seção. **População**, ou universo, é o nome que se dá a um conjunto de unidades (elementos ou exemplares/dados se aproximarmos a definição ao contexto de mineração de dados) que compartilham características comuns e sobre o qual é pretendido o desenvolvimento de um estudo. O conjunto de vendas realizadas durante o tempo de funcionamento de um restaurante é um exemplo de população (de vendas). A população pode ser finita ou infinita. Sendo finita, se pequena, um estudo pode envolver sua totalidade. Caso seja muito grande ou infinita, faz-se necessário estabelecer uma amostra, para que um estudo sobre ela se viabilize. Uma **amostra**, então, é um subconjunto da população (também chamado de fração ou parte) que deve ser estabelecido com cuidado, seguindo técnicas apropriadas, pois o objetivo de obter uma amostra é reduzir o tamanho da população sem que características essenciais associadas a ela sejam perdidas. Boas técnicas de amostragem geram amostras representativas e imparciais da população, ou seja, mantêm a proporcionalidade dos fenômenos que ocorrem na população e conferem chances iguais aos elementos da população de fazerem parte da amostra. Há diferentes formas de estabelecer uma amostra, que resultam em diferentes tipos de amostragem: aleatória, estratificada, por conglomerados, acidental, intencional, por quotas etc. Em Lohr (2010),

discussões aprofundadas sobre o problema de construção de amostras são apresentadas.

Na estatística descritiva, uma **variável** diz respeito a uma característica associada a elementos de uma população. Por exemplo, num restaurante, uma variável pode ser a temperatura em que um vinho deve ser servido (com valores como 5°C ou 12°C) ou o tipo de um prato presente no cardápio (com os seguintes valores, por exemplo: aperitivo, pasta, carne branca ou sobremesa quente). Fazendo uma analogia à representação de um dado usada neste livro, uma variável é equivalente a um atributo descritivo. As variáveis, no contexto da estatística descritiva, podem ser quantitativas ou qualitativas.

As **variáveis quantitativas** (também nomeadas no contexto de mineração de dados como variáveis numéricas) são aquelas que podem ser medidas em uma escala de valores numéricos, e se dividem em discretas (assumem valores dentro de um conjunto finito ou infinito contável) ou contínuas (assumem valores em uma escala contínua). As **variáveis qualitativas** (também nomeadas no contexto de mineração de dados como variáveis categóricas) são aquelas que representam uma classificação do elemento ao qual o valor da variável está associado, e se dividem em nominais (quando não há ordenação entre valores) ou ordinais (quando há uma ordenação entre os valores). Alguns exemplos de variáveis, considerando o contexto de um restaurante, e alguns valores que elas podem assumir estão organizados no Quadro 2-1.

Quadro 2-1: Exemplos de variáveis, considerando a área de estatística descritiva e o contexto do restaurante

Variáveis quantitativas
Discretas Quantidade de bebidas vendidas: 15, 6, 3, 0 Idade do cliente: 5, 10, 25, 32, 57 *Contínuas* Preço de um item do cardápio: 5,50, 35,00, 99,50 Peso de um corte de carne: 100g, 226g, 500g
Variáveis qualitativas

Nominais
Código da forma de pagamento: 11, 21, 31
Forma de pagamento: dinheiro, débito, crédito
Ordinais
Faixa etária dos clientes: 0 a 10 anos, 11 a 17 anos, 18 a 30 anos, 31 a 50 anos, mais de 50 anos
Mês de observação: janeiro, abril, novembro

Sobre as variáveis podem ser executadas operações a fim de relacioná-las ou transformá-las. Neste capítulo, são apresentadas algumas ferramentas (medidas e gráficos) que permitem manipular e analisar variáveis.

2.1.2. Medidas de posição e separatrizes

Medidas de posição , ou medidas de tendência central , permitem encontrar os valores que orientam a análise dos dados no que diz respeito à sua localização, ou como a distribuição associada aos valores se comporta no universo amostrado. As medidas de posição mais comuns no contexto de análise exploratória de dados são média aritmética, mediana e moda. Relacionadas com a mediana estão as medidas separatrizes percentis e quartis.

Para um conjunto de valores $\mathbf{v} = \{v_1, v_2, ..., v_n\}$, em que n é a quantidade de valores do conjunto, a **média** aritmética simples (*média*(**v**) ou $\bar{v}$) é a soma dos valores do conjunto **v**, dividida pela quantidade de valores presentes nesse conjunto. Formalmente, *média* (**v**) é definida como:

$$média(\mathbf{v}) = \frac{1}{n}\sum_{i=1}^{n} v_i$$

Equação 2-1

A **mediana** (*mediana*(**v**)) é o valor que divide a distribuição dos valores exatamente ao meio (podendo ser vista também como uma medida separatriz). Importante lembrar que tal valor não precisa estar presente no

conjunto **v**. Para o cálculo da mediana, todos os valores presentes no conjunto devem ser ordenados de forma crescente (formando o conjunto **v'**), e a mediana será, então, dada por:

$$mediana\,(\mathbf{v}) = \begin{cases} v'_i & \text{se } n \text{ é impar, } i = \dfrac{n+1}{2} \\ \dfrac{1}{2}(v'_i + v'_{i+1}) & \text{se } n \text{ é par, } \quad i = \dfrac{n}{2} \end{cases}$$

Equação 2-2

em que *i* é uma posição no conjunto de valores **v'**.

A **moda** , por sua vez, é o valor mais frequente em um conjunto de valores. Ela é a única medida de posição que pode assumir mais de um valor. Essa situação ocorre quando dois ou mais valores aparecem no conjunto de valores **v** com a mesma frequência, a máxima no conjunto. Assim, um conjunto de valores pode ser amodal (não possui moda), unimodal (possui uma moda), bimodal (possui duas modas) ou multimodal (possui diversas modas).

Para exemplificar o uso dessas medidas e como a comparação entre elas pode ser útil, considere os dados da Tabela 2-1, que representa a quantidade de vendas, durante um mês, dos nove itens presentes no cardápio promocional de um restaurante.

Tabela 2-1: Quantidade de vendas de nove itens do restaurante durante um mês

Item	712	068	002	065	103	809	111	601	044
Quantidade	29	30	32	65	65	65	25	25	90

Presumindo que se queira calcular as medidas de posição em relação à variável *Quantidade*, por causa da mediana, e apenas por causa dela, é preciso ordenar os dados da Tabela 2-1 considerando os valores dessa variável (Tabela 2-2).

Tabela 2-2: Quantidade de vendas ordenada

	1	2	3	4	5	6	7	8	9
Item	601	111	712	068	002	065	103	809	044
Quantidade	25	25	29	30	32	65	65	65	90

Iniciando a análise exploratória desses dados pelo cálculo da média, com o uso da Equação 2-1, tem-se: *média(Quantidade)* = $\frac{1}{9}$ (25 + 25 + 29 + 30 + 32 + 65 + 65 + 65 + 90) = 47,3. Para o cálculo da mediana (Equação 2-2), desde que haja uma quantidade ímpar de valores, tem-se $i = (n + 1)/2 = (9 + 1)/2 = 5$ e, portanto, *mediana*(*Quantidade*) = 32, ou seja, a mediana é o valor da variável *Quantidade* na quinta posição da ordenação. Enfim, a moda é o valor com maior quantidade de repetições. No caso da variável *Quantidade*, o valor 65 aparece três vezes, enquanto todos os demais valores aparecem uma ou duas vezes; portanto, 65 é a moda do conjunto. As medidas de posição calculadas para a variável *Quantidade* estão resumidas na Tabela 2-3.

Em termos práticos, essas medidas permitem algumas intepretações sobre o cardápio promocional: durante o mês de observação, a quantidade média de vendas de um item foi de 47,33 unidades; a mediana revela que, embora as vendas para o item 002 sejam em quantidade maior que as vendas para metade dos itens observados (é o quinto mais vendido dentro de nove itens), a venda desse item está abaixo da média de vendas do conjunto.

Tabela 2-3: Medidas de posição para a variável *Quantidade*

Média	Mediana	Moda
47,33	32	65

A comparação da média, mediana e moda pode revelar informações interessantes sobre a distribuição dos valores do conjunto em questão.

Quando essas medidas assumem o mesmo valor (ou valores com variações muito pequenas entre si) significa que o conjunto de valores de uma variável tem simetria, ou seja, é possível usar uma dessas medidas de posição para separar o conjunto de valores em duas partes com a mesma distribuição de frequência. Por outro lado, se os valores são diferentes, diz-se que a distribuição dos valores da variável é assimétrica (veja mais detalhes sobre esses conceitos na Seções 2.1.4 e 2.1.6). No exemplo da variável *Quantidade*, a média, a mediana e a moda assumem valores diferentes, o que indica que a distribuição da quantidade de vendas é assimétrica. Nesse caso, menos da metade dos itens vende mais que a média (quatro itens vendem mais e cinco itens vendem menos).

Os percentis e quartis são medidas separatrizes que dividem o conjunto de valores, ordenado de forma crescente, em partes tão iguais quanto possível. O percentil de ordem p determina os $p\%$ menores valores contidos em **v**', e a posição i, que delimita o percentil de ordem p em **v**', é dada por $i = \frac{p(n+1)}{100}$ (arredondado, se necessário). Isso significa que os $p\%$ menores valores em **v**' estão abaixo do valor da posição i. Então, considerando a variável *Quantidade* (Tabela 2-2), o percentil de ordem $p = 50$ indica os 50% menores valores no conjunto ordenado e estão abaixo do valor localizado na posição $i = \frac{50(9+1)}{100} = 5$. Assim, o valor 32 é o percentil de ordem 50 em **v**'.

Casos particulares de percentis definem os quartis. O primeiro quartil , ou Q1, é o percentil de ordem $p = 25$, ou seja, o valor de **v** que separa os 25% valores menores que v_i dos 75% valores maiores que v_i, sendo v_i o valor que ocupa a $\frac{25(n+1)}{100}$ posição em **v**'. O segundo quartil, ou Q2, é o percentil de ordem $p = 50$ e é equivalente à mediana do conjunto de valores. O terceiro quartil, ou Q3, é equivalente ao percentil de ordem $p = 75$ e é o valor que separa os 75% valores menores que v_i dos 25% valores maiores que v_i, sendo v_i o valor que ocupa a $\frac{75(n+1)}{100}$ posição em **v**'. Para o caso da variável *Quantidade* (Tabela 2-2), Q1, Q2 e Q3 são, respectivamente, $v_3 = 29$, $v_5 = 32$,

v_8 = 65. Em termos práticos, poderíamos dizer que o item 044 está acima do percentil de ordem 75 ou acima de Q3, indicando que pelo menos 75% dos itens observados vendem menos que ele, evidenciando sua boa aceitação por parte dos clientes.

Os quartis fornecem uma indicação de centro, dispersão e forma da distribuição dos dados. A combinação dos quartis com valores de mínimo e máximo do conjunto de valores formam um conjunto de cinco medidas, chamado de *resumo dos cinco números*. Esse conjunto de medidas pode ser visualizado em um diagrama de caixa ou *boxplot* (Seção 2.1.6).

2.1.3. Medidas de dispersão

As medidas de posição são úteis para apresentar uma sumarização dos dados. No entanto, não são capazes de descrever a variação ou dispersão do conjunto de valores. Para esse fim, aplicam-se as medidas de dispersão , capazes de descrever o quanto os valores de um conjunto estão próximos ou distantes de uma medida central, como a média. As medidas mais comuns de dispersão ou variância dos dados são a amplitude, variância, desvio-padrão e coeficiente de variação.

Para um conjunto de valores **v**, a medida da **amplitude** é a diferença entre o maior e o menor valor do conjunto, portanto: *amplitude* (**v**) = *max* (**v**) – *min* (**v**). Embora a medida da amplitude seja simples, sua interpretação precisa ser feita com cuidado, pois ela pode ser influenciada por valores extremos, conhecidos como *outliers* (Seção 2.2.1).

A **variância** é uma medida de dispersão definida como a média dos quadrados das diferenças entre cada valor do conjunto **v** e a média desse conjunto. Formalmente, tem-se:

$$\sigma^2(\mathbf{v}) = \frac{1}{n}\sum_{i=1}^{n}(v_i - media(\mathbf{v}))^2$$

Equação 2-3

O **desvio-padrão** ($\sigma(\mathbf{v})$) é a raiz quadrada da variância. O uso do desvio-padrão só faz sentido quando a média for usada como medida de posição (ou de centro). Ele é semelhante à medida de amplitude, com a diferença de que o cálculo do desvio-padrão usa todos os valores de um conjunto. O resultado dessa medida geralmente é usado para verificar a consistência de um fenômeno (um fenômeno é consistente quando o cálculo do desvio-padrão resulta em valores baixos).

Outra utilidade para o desvio-padrão é conhecida como *regra empírica* , ou *regra 68-95-99* ou ainda *regra dos 3-sigmas* . Essa regra se aplica a conjunto de valores com distribuição normal e afirma que aproximadamente 68% dos valores desse conjunto estão a menos de um desvio-padrão da sua média; 95% dos valores desse conjunto estão a menos de dois desvios-padrão da sua média; 99,7% dos valores desse conjunto estão a menos de três desvios-padrão da sua média.

Para ilustrar os conceitos de medidas de dispersão, considere novamente o conjunto de valores assumidos pela variável *Quantidade* (Tabela 2-2). Para calcular a medida de dispersão *amplitude*, é necessário encontrar os valores mínimo (*min*(*Quantidade*) = 25) e máximo (*max*(*Quantidade*) = 90), e, portanto, a *amplitude*(*Quantidade*) = 90 – 25 = 65.

Para o cálculo da variância e, consequentemente, do desvio-padrão, considere o exposto na Tabela 2-4. Nela tem-se como valores do conjunto **v**, os valores da variável *Quantidade*, a média de **v**, as diferenças entre os valores v_i e a média do conjunto **v** e o cálculo do quadrado dessas diferenças. Note que o sinal dos resultados dos cálculos das diferenças permite verificar o posicionamento do valor *v* em relação à média do conjunto **v**. Os resultados da última coluna da tabela são aplicados na Equação 2-3, resultando no valor da variância (representado ao final da última coluna). Consequentemente, ao extrair a raiz da variância, obtém-se o valor do desvio-padrão.

Por fim, a medida do **coeficiente de dispersão** é capaz de expressar a dispersão relativa dos valores e é útil para comparar a dispersão de dois ou

mais conjuntos de valores (distribuições diferentes). O coeficiente de dispersão, também conhecido como coeficiente de variação de Pearson, é dado por:

$$cv(\mathbf{v}) = \frac{\sigma(\mathbf{v})}{média(\mathbf{v})}$$

Equação 2-4

Tabela 2-4: Cálculo da variância e do desvio-padrão para a variável *Quantidade*

	v	**v_i – média(v)**	**$(v_i$ – média(v))^2**
v_1	25	-22,33	498,78
v_2	25	-22,33	498,78
v_3	29	-18,33	336,11
v_4	30	-17,33	300,44
v_5	32	-15,33	235,11
v_6	65	17,67	312,11
v_7	65	17,67	312,11
v_8	65	17,67	312,11
v_9	90	42,67	1820,44
soma(**v**):	426		4626

$$média(\mathbf{v}) = \frac{426}{9} = 47,33$$

$$\sigma^2(\mathbf{v}) = \frac{4626}{9} = 513,99$$

$$\sigma(\mathbf{v}) = \sqrt{513,99} = 22,67$$

Considere fazer uma comparação entre a quantidade de vendas de cada item do cardápio promocional e a quantidade de vendas de cada item do cardápio regular do restaurante (Tabela 2-5). Os coeficientes de variação de Pearson para cada variável são, respectivamente, 0,47 e 0,28, ou seja, no

primeiro caso, o desvio-padrão representa 47% da média das quantidades de vendas observadas no cardápio promocional, e, no segundo caso, 28% da média das quantidades de vendas observadas no cardápio regular. Note que a relação entre o desvio-padrão e a média das quantidades vendidas para o cardápio regular é menor, mostrando que há uma uniformidade maior das vendas nesse cardápio que no cardápio promocional.

Tabela 2-5: Quantidade de venda dos itens do cardápio regular

Item	255	079	861	224	441	531	089	106	549	237	554	112	058	066	001
Quantidade	15	15	18	20	23	18	20	13	13	13	11	11	11	10	10

2.1.4. Distribuição de frequência

Dados podem ser organizados de maneira a serem resumidos e visualizados por meio de gráficos ou tabelas. Uma organização possível é pela **distribuição de frequência**, que pode ser obtida distribuindo um conjunto de valores em faixas (intervalos ou *bins*) e fornecendo o número (ou porcentagem) de valores do conjunto que aparece em cada faixa (ou intervalo). Assim, os valores de um conjunto são resumidos de maneira alternativa ao uso de medidas que resultam em valores únicos (como média, desvio-padrão etc.), permitindo a construção de uma visualização compacta desse conjunto.

A distribuição dos dados pela frequência pode ser feita de modo relativo ou acumulado. A **frequência relativa** é a apresentação da frequência de valores que aparecem em cada uma das faixas, dividida pela frequência total de valores de um conjunto e, geralmente, é expressa em porcentagem. A frequência relativa é dada por:

$$frequência\ relativa\ (faixa) = \frac{(frequência\ da\ faixa)}{(frequência\ total)} * 100$$ **Equação 2-5**

A **frequência acumulada relativa** , por sua vez, é a frequência relativa acumulada a cada faixa. No cálculo da frequência relativa acumulada de uma faixa, consideram-se as frequências acumuladas das faixas anteriores.

A representação gráfica da informação sobre frequência é feita por meio de histogramas (Seção 2.1.6). Aqui é apresentado um exemplo de frequências organizadas em uma tabela. Considere novamente o conjunto de valores assumidos pela variável *Quantidade* na Tabela 2-2, e a distribuição de frequência de valores a partir de 10 faixas, como ilustrado na primeira coluna da Tabela 2-6. A quantidade de faixas foi definida apenas para fins de exemplificação e não segue nenhuma regra vinda do contexto do restaurante ou qualquer regra empírica, como a quantidade de faixas menor que um limiar ou a quantidade de classes estimada seguindo alguma lei (veja, na Seção 2.8, indicações de leitura sobre esse tópico).

Para estabelecer as frequências em cada faixa, verifica-se o número de vezes que os valores assumidos pela variável sob análise ocorrem dentro do intervalo que define a faixa. O resultado dessa contagem é a frequência absoluta. Esse resultado ainda pode ser apresentado de forma relativa (seguindo a Equação 2-5) ou na forma acumulada. Os três resultados estão apresentadas na Tabela 2-6. As frequências relativas podem também ser apresentadas de maneira acumulada.

Tabela 2-6: Variável *Quantidade* representada segundo a frequência de valores por faixas

Faixas	**Frequências**		**Frequências acumuladas**	
	absolutas	**relativas (%)**	**absolutas**	**relativas (%)**
1-10	0	0	0	0
11-20	0	0	0	0
21-30	4	44,44	4	44,44
31-40	1	11,11	5	55,56
41-50	0	0	5	55,56
51-60	0	0	5	55,56

61-70	3	33,33	8	88,88
71-80	0	0	8	88,88
81-90	1	11,11	9	100
91-100	0	0	9	100

Quando apresentados no formato tabular, os resultados referentes ao cálculo de frequências não são de tão fácil interpretação quanto na representação gráfica (histograma). Na forma gráfica, é possível verificar o formato da distribuição do conjunto de valores. Se as frequências forem ordenadas (em ordem decrescente), elas podem ser usadas como um importante indicador de qualidade, chamado de Diagrama de Pareto (Triola, 2008; Freund, 2006).

2.1.5. Análise de correlação

As medidas de posição e de dispersão se constituem como ferramentas de análise exploratória de um único conjunto de valores.[1] A análise de correlação , por outro lado, permite estudar a relação entre dois conjuntos de valores. Nesse tipo de análise, quantifica-se o quanto um conjunto de valores (ou uma variável) está relacionado com outro, no sentido de determinar a intensidade e a direção dessa relação. Em outras palavras, a correlação indica se, e com que intensidade, os valores de uma variável aumentam (ou diminuem) enquanto os valores da outra variável aumentam (ou diminuem).

A quantificação da correlação é feita por meio de um coeficiente, sendo que o mais conhecido é o coeficiente de correlação de Pearson (*r*). O cálculo de *r* para dois conjuntos de valores (ou variáveis) **v1** e **v2**, com *n* valores cada, é:

$$r_{v_1,v_2} = \frac{\sum_{i=1}^{n} (v_{1_i} - média(\mathbf{v}_1))\,(v_{2_i} - média(\mathbf{v}_2))}{\sqrt{\sum_{i=1}^{n} (v_{1_i} - média(\mathbf{v}_1))^2} * \sqrt{\sum_{i=1}^{n} (v_{2_i} - média(\mathbf{v}_2))^2}}$$

Equação 2-6

em que v_{1i} e v_{2i} são elementos dos conjuntos de valores $\mathbf{v}_1$ e $\mathbf{v}_2$. O resultado do coeficiente de correlação assumirá valores entre –1 e +1. O sinal do resultado indica a direção, se a correlação é positiva ou negativa, e o valor indica a intensidade da correlação. Valores de correlação acima de |0,70| indicam forte correlação, sendo que |·| indica a operação de módulo.

Como exemplo, imagine que se deseja comparar a quantidade de vendas do prato *feijoada* (*Quantidade* I) e da bebida *caipirinha* (*Quantidade* II), no decorrer dos meses de novembro a julho, que ocorrem conforme apresentado na Tabela 2-7.

Tabela 2-7: Quantidade de vendas em dois itens em um período

Mês	**nov**	**dez**	**jan**	**fev**	**mar**	**abr**	**mai**	**jun**	**jul**
Quantidade I (feijoada)	24	27	29	30	32	58	64	64	65
Quantidade II (caipirinha)	10	25	20	36	28	38	50	60	69

Para facilitar os cálculos, considere que *Quantidade* I será representada pela variável **v1**, e a *Quantidade* II, pela variável **v2**. Todos os cálculos necessários para encontrar o coeficiente de correlação de Pearson estão resumidos na Tabela 2-8. A correlação $r_{v}1,_{v}2$ é de 0,89 (uma correlação alta e positiva), indicando que o comportamento das vendas dos dois itens é similar em relação à tendência de aumento (ou queda) de vendas. Quando o prato *feijoada* tem mais procura no restaurante, a bebida *caipirinha* também tem o comportamento de venda alterado positivamente. Se cai a venda da *feijoada*, cai a venda da *caipirinha*.

Nesse exemplo, o uso da correlação permitiu interpretar a relação do comportamento dos valores em duas variáveis, fazendo uma análise exploratória de dados. No entanto, no contexto de análise de dados, a medida de correlação pode ainda ser útil na fase de pré-processamento de dados (Seção 2.2), mais especificamente para selecionar atributos interessantes para

serem submetidos a algoritmos de classificação ou de agrupamento. Trata-se, portanto, de uma ferramenta útil para a realização de redução de dimensionalidade, que contribui para diminuir a complexidade de um problema de análise de dados.

Tabela 2-8: Cálculo do coeficiente de correlação para as variáveis Quantidade I e Quantidade II

			A	**B**			
	v1	**v2**	**v1 – média(v1)**	**v2 – média(v2)**	**A*B**	**(v1 – média(v1))^2**	**(v2 – média(v2))^2**
	24	10	-19,67	-27,33	537,56	386,78	747,11
	27	25	–16,67	–12,33	205,56	277,78	152,11
	29	20	–14,67	–17,33	254,22	215,11	300,44
	30	36	–13,67	–1,33	18,22	186,78	1,78
	32	28	–11,67	–9,33	108,89	136,11	87,11
	58	38	14,33	0,67	9,56	205,44	0,44
	64	50	20,33	12,67	257,56	413,44	160,44
	64	60	20,33	22,67	460,89	413,44	513,78
	65	69	21,33	31,67	675,56	455,11	1002,78
somas	393	336				2690	2966
				aux1 = 2528		aux2 = 51,87	aux3 = 54,46
médias	43,67	37,33		$r_{v_1,v_2} = \frac{\text{aux1}}{\text{aux2} * \text{aux3}} = \frac{2528}{51{,}87 * 54{,}46} = 0{,}89$			

2.1.6. Representações gráficas

Representações gráficas auxiliam na visualização das características dos dados, o que é útil para os analistas, seja em relação a uma primeira ação de

estudo dos dados ou à necessidade de comparação de resultados obtidos com a resolução das tarefas de mineração de dados.

Diagrama de caixa ou ***Boxplot*** é uma maneira popular de visualizar a distribuição dos dados. Incorpora o *resumo dos cinco números*, obtidos a partir da combinação dos quartis com valores de mínimo e máximo de um conjunto de valores, da seguinte maneira (acompanhe a explicação com atenção à Figura 2-1):

- Os pontos finais da caixa são os Q1 e Q3. O tamanho da caixa é também chamado de faixa interquartil (ou *interquatile range*, IQR = Q3 – Q1).
- A mediana é marcada por uma linha dentro da caixa (Q2).
- Barreiras de *outlier*, os traços no final das linhas transversais, são dadas por: barreira inferior = Q1 – (1,5 * IQR); barreira superior = Q3 + (1,5 * IQR).
- Duas linhas nas transversais fora da caixa delimitadas pelas barreiras são chamadas bigodes.

Adicionalmente aos resultados e análises supracitadas, o uso da caixa ainda permite inferir conclusões sobre a forma da distribuição dos dados , como mostrado na Figura 2-2. A partir dessa figura, com diagramas genéricos, é possível concluir se o conjunto de dados está mais concentrado ou mais disperso, em que região se encontra a maioria dos dados e como eles se dispersam. Veja na figura os três tipos de distribuição: assimétrica negativa ou à esquerda, simétrica e assimétrica positiva ou à direita.

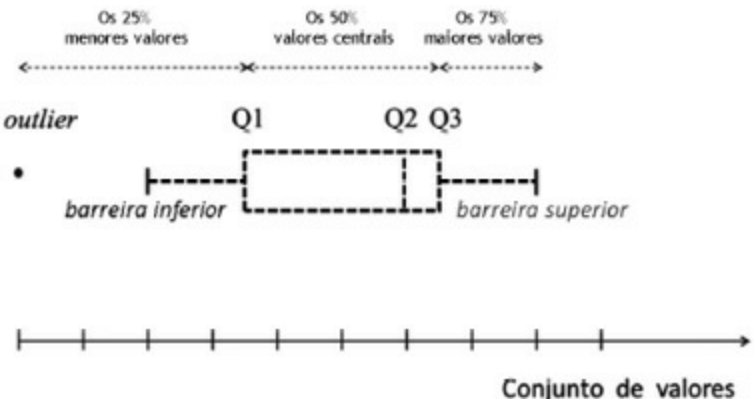


Figura 2-1: Exemplo de diagrama de caixa ou *boxplot*

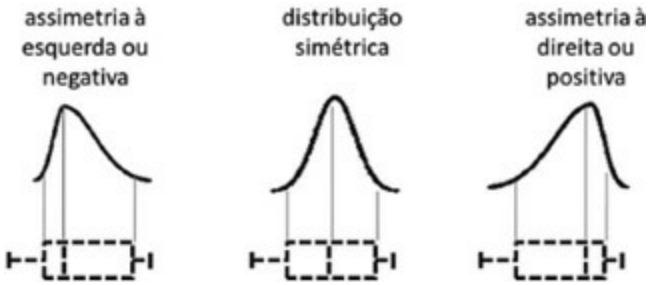


Figura 2-2: Exemplo de interpretação de caixa quanto à dispersão de dados

A análise do diagrama de caixas permite inferir conhecimento sobre conjunto de valores individualmente, incluindo a detecção da presença de *outliers*. Esse tipo de análise é útil em mineração de dados quando se deseja entender o comportamento dos valores assumidos por um atributo descritivo em um conjunto de dados, por exemplo.

Histograma é uma representação gráfica da distribuição de frequências, como pode ser observado na Figura 2-3 (histogramas referentes aos conjuntos de valores apresentados na Tabela 2-1, na Tabela 2-6 e na Tabela 2-7). Um histograma é construído alocando as faixas no eixo horizontal, e as frequências (absolutas ou relativas, acumuladas ou não), no eixo vertical. Para cada faixa, é atribuída uma barra, de forma que sua altura represente a frequência da faixa. Gráficos desse tipo são capazes de indicar o tipo de curva criada pela distribuição das frequências. Há vários tipos de curvas de frequência, como ilustrado na Figura 2-4.

A quantidade de faixas, segundo Triola (2008), implica o tipo de informação que pode ser extraída do histograma. Segundo o mesmo autor, geralmente a quantidade satisfatória é entre 5 e 20 faixas, mas, na prática, trabalha-se com um número aproximadamente igual à raiz quadrada do tamanho do conjunto de valores ou com o número de faixas apropriado já conhecido *a priori*.

As informações sobre frequências podem ainda ser apresentadas como um gráfico de setores ou pelo diagrama de Pareto. O **gráfico de setores** (ou gráfico de pizza) apresenta uma visualização diferente da disposição dos valores de frequência; porém, essa visualização não permite observar informação referente a curvas de frequência.

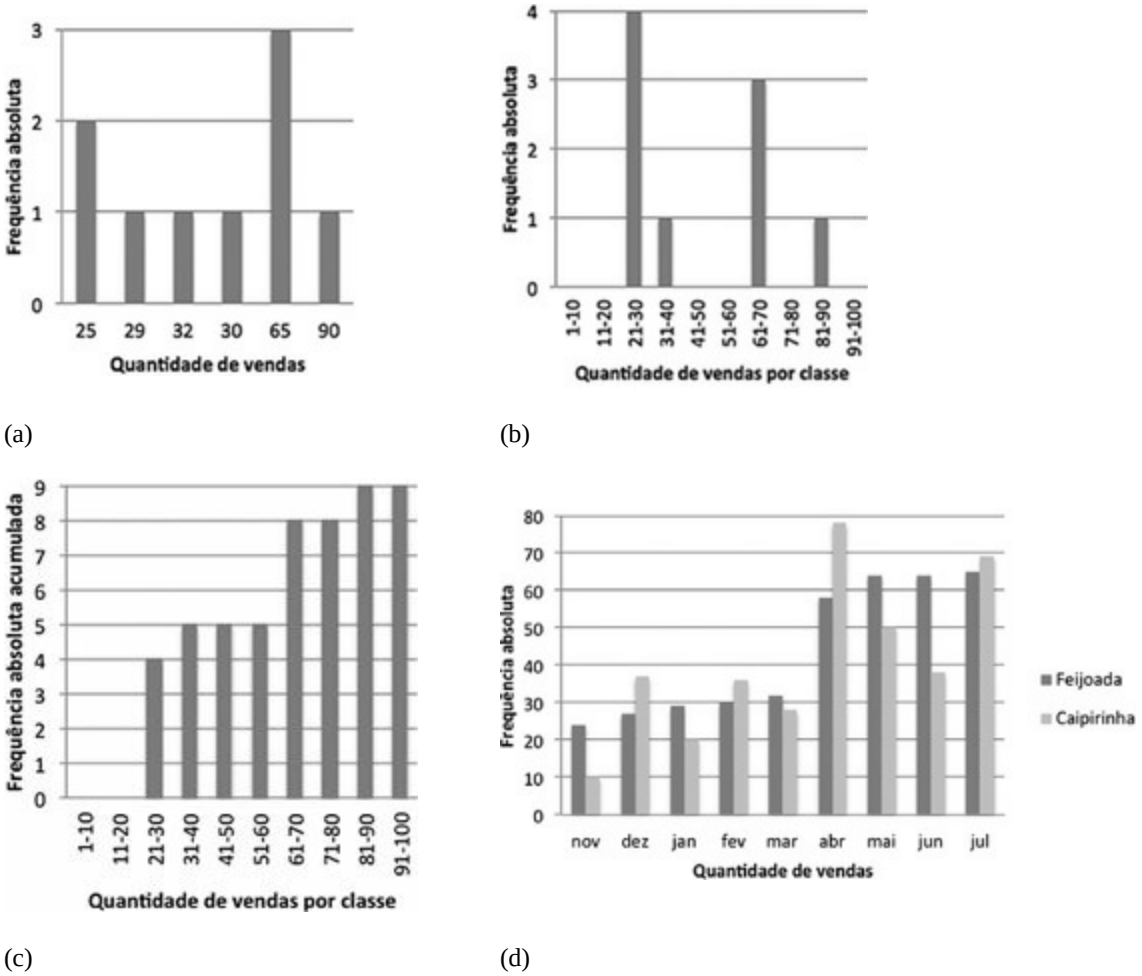


Figura 2-3: Exemplos de histrograma: (a) e (b) histograma de frequências absolutas; (c) histograma de

frequências absolutas acumuladas; (d) histograma de frequências absolutas para dois conjuntos de valores

Simétricas ou em forma de sino: indicam que valores equidistantes do valor modal têm a mesma frequência. Um exemplo é a curva normal.

Assimétricas: a cauda da curva é mais longa em um dos lados. Se a parte mais alongada fica à direita, a curva é dita desviada para a direita, ou de assimetria positiva; se ocorre o inverso, diz-se que a curva é desviada para a esquerda, ou de assimetria negativa.

Na curva em forma de J, ou em J invertido, o valor máximo ocorre em uma das extremidades.

Uma curva de frequência em forma de U tem os valores máximos nas extremidades.

Uma curva de frequência bimodal tem dois valores máximos.

Uma curva de frequência multimodal tem mais de dois valores máximos.

Figura 2-4: Exemplos de curvas de frequência interpretáveis a partir de um histograma

O **diagrama de Pareto** é uma variação do histograma a partir da ordenação das frequências das faixas, da maior para a menor, e é bastante usado como visualização de causas ou prioridades em resolução de problemas. Usualmente, uma curva de frequências acumuladas (relativas ou não) acompanha o gráfico, a fim de apoiar a interpretação da informação (veja indicações de leitura na Seção 2.8 sobre o Princípio de Pareto, para obter suporte para interpretar mais facilmente a informação que esse diagrama é capaz de fornecer). No diagrama de Pareto da Figura 2-5, percebe-se que vender 65 unidades de um item é a situação mais frequente no contexto do cardápio promocional. Essa frequência de venda para um item ocorre em pouco mais de 30% das medições de desempenho de vendas. No restante das medições, com exceção de um item que alcança 90 vendas, os itens têm desempenho de venda abaixo de 32 unidades vendidas (abaixo da média de vendas). Trata-se de uma situação na qual grandes quantidades de vendas são alcançadas pela minoria dos itens do cardápio.

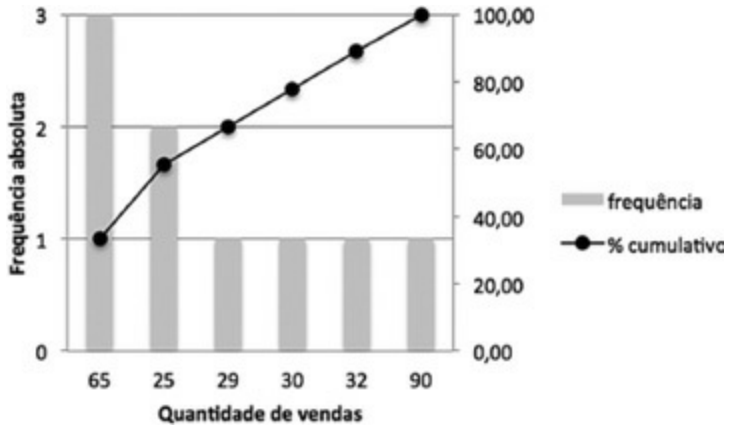


Figura 2-5: Diagrama de Pareto

O **diagrama de dispersão** , ou *scatter plot* , é um gráfico no qual os valores assumidos pelas variáveis podem ser representados simultaneamente, permitindo observar a relação existente entre elas, como mostrado na Figura 2-6. Na Figura 2-6(b), a relação entre os valores da quantidade de vendas de duas variáveis (feijoada e caipirinha) é evidenciada pelas curvas de tendência associadas a cada uma delas.

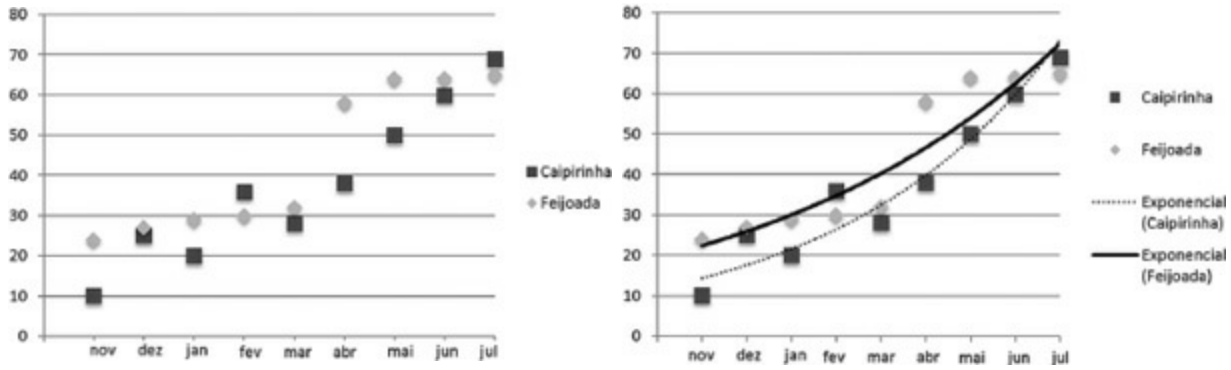


Figura 2-6: Gráfico de dispersão (a) e gráfico de dispersão com linhas de tendência (b)

2.2. Pré-processamento de dados

Como discutido no Capítulo 1, os dados submetidos à mineração podem ser provenientes de diferentes fontes: sistemas transacionais, observações de situações do mundo real ou processos de medição em geral. Não raramente, os dados provenientes de diferentes fontes ou, em alguns casos, mesmo se provenientes de uma mesma fonte, apresentam algumas falhas ou são organizados de forma inadequada para serem submetidos a alguns algoritmos. Tais condições geram dificuldades no processo de descoberta de conhecimento e frequentemente levam ao fracasso da análise pretendida. Portanto, é imprescindível executar procedimentos que pré-processem os dados, corrigindo inconsistências e, muitas vezes, agregando valor aos dados, de forma a dar condições ao processo de descoberta de conhecimento e chegar a resultados de qualidade. Problemas típicos associados à baixa qualidade de dados são ausência de valores, dados ruidosos, valores inconsistentes, atributos de naturezas distintas e redundância de dados. Muitas dessas características não são evidentes no conjunto de dados, mas podem ser reveladas por procedimentos da análise exploratória, discutidos na primeira parte deste capítulo.

Na presente seção, procedimentos de pré-processamento de dados são discutidos. Os procedimentos abordados se concentram nos seguintes objetivos: limpeza, integração e transformação de dados.

2.2.1. Limpeza de dados

Procedimentos para limpeza de dados (também conhecidos como *data cleaning*) têm o objetivo de amenizar dois importantes problemas advindos de processos imprecisos de aquisição de dados: a existência de valores ausentes (*missing values*) e a existência de valores ruidosos (*noise values*).

O problema de **ausência de valores** pode ser encontrado sob duas formas principais: quando os atributos de um conjunto de dados não apresentam valores para determinados exemplares; ou quando um conjunto de dados não possui valores para um atributo de interesse ou apresenta apenas valores agregados em relação àquele atributo. Esse problema pode acarretar dificuldades na produção de uma análise de qualidade ou pode inviabilizar um processo de análise se o algoritmo envolvido não puder lidar com a ausência de valores.

Como exemplo, considere a funcionalidade de um sistema para cadastro dos clientes de um restaurante ou para um levantamento de dados sobre a concorrência. No primeiro caso, é comum que alguns dos clientes, ao preencherem um formulário de cadastro, não informem sua ocupação profissional no campo do formulário em que esse dado é solicitado, por descuido, falta de interesse ou, ainda, por estarem desempregados. No segundo caso, os preços praticados pela concorrência podem não estar disponíveis para todas as classes de pratos pesquisados, pois um restaurante pode, por exemplo, não oferecer sobremesas ou pode não trabalhar com a venda de vinhos. Ainda, em um processo de medição automático, como a aferição da velocidade de um veículo por um sistema de radares, uma falha de software ou de hardware, uma interferência inesperada do ambiente, ou mesmo um problema no canal de transmissão dos dados, pode prejudicar o processo por alguns instantes, gerando a ausência da medição durante um intervalo de tempo. Em todos esses casos, independentemente das causas, a consequência é que o valor esperado para um atributo em relação a um exemplar do conjunto de dados está ausente.

As soluções para resolver o problema de valores ausentes são diversas, sendo as mais comuns:

- Remoção do exemplar em que ocorre a falta do valor: trata-se de uma solução bastante simples, porém não é efetiva, principalmente quando a quantidade total de exemplares no conjunto de dados é pequena ou

quando a ausência dos valores ocorre em um grande número de exemplares.

- Preenchimento manual dos valores: se houver a possibilidade de fazê-lo, ou seja, se o processo de aquisição de dados puder ser executado novamente ou se houver um especialista no contexto, é possível encontrar o valor correto para determinado atributo de determinado exemplar; entretanto, também se torna inviável quando a quantidade de valores ausentes é grande, e corre-se o risco de impor um viés aos dados, caso o preenchimento seja feito a partir de um especialista com conhecimento parcial do contexto dos dados.
- Preenchimento automático dos valores: o preenchimento automático pode ser feito a partir do estabelecimento de um valor constante (o mais frequente no conjunto de dados para o atributo em questão), um valor médio ou mediano (considerando os demais valores presentes no atributo em questão) ou com base no resultado de um método de análise preditiva (um modelo de regressão, por exemplo). Contudo, há de sempre se tomar o cuidado de não associar um viés aos dados.

Os **valores ruidosos** referem-se a modificações dos valores originais e que, portanto, consistem em erros de medidas ou em valores consideravelmente diferentes da maioria dos outros valores do conjunto de dados, os chamados *outliers*. Casos típicos de ruído são percebidos quando se conhece o domínio esperado para os valores em um atributo ou a distribuição esperada para os valores de um atributo, mas, no conjunto de dados, aparecem alguns valores fora desse domínio ou dessa distribuição.

Alguns exemplos de ruído são: valores que naturalmente deveriam ser positivos, mas, no conjunto de dados, aparecem como negativos, como seria o caso de observar um valor negativo para a agregação da quantidade de vendas no restaurante, num período de um mês; ou ainda, observar que o valor de vendas ultrapassa, com uma grande margem, o valor de vendas de

todos os anos anteriores. No primeiro caso, diz-se que o valor está errado por não fazer sentido no contexto dos dados; no segundo caso, tem-se um exemplo de *outlier*, que pode representar um valor errado ou uma mudança não esperada, porém real, do comportamento dos valores de um atributo, e, nesse caso, há a necessidade de identificar qual é a explicação que se adéqua à situação.

Há duas abordagens para o tratamento de dados ruidosos:

- Inspeção e correção manual: a análise exploratória pode dar condições ao analista de dados de perceber a existência de ruídos nos dados (por exemplo, com a análise de gráficos de caixas ou com a observação da forma da distribuição de frequências). Esses valores ruidosos, se em pouca quantidade, podem ser manualmente corrigidos. Se em quantidades maiores, sua influência na análise de dados pode ser amenizada com procedimentos de suavização que usam a criação de faixas de valores guiadas por medidas de posição, como média ou mediana e medidas separatrizes.
- Identificação e limpeza automática: existem algoritmos que podem ser usados para suavizar, ou anular, ruídos em conjuntos de dados. Algoritmos que resolvem problemas de predição e de agrupamento podem ser modelados para identificar e separar exemplares ruidosos de um conjunto de dados (um exemplo de algoritmo que pode ser usado para este fim é o DBSCAN, veja Capítulo 4), ou para mapear um valor ruidoso para uma classe ou grupo de dados ou para uma faixa de valores que o torne compatível com os valores esperados. A depender do tipo de dado sob análise, técnicas de tratamento de sinais podem ser aplicadas para separar valores ruidosos de não ruidosos.

2.2.2. Integração de dados

Procedimentos para **integração de dados** (também conhecida como *data integration*) consistem em realizar ações que permitam integrar, adequadamente, dados provenientes de diversas fontes de dados. Geralmente, quando os dados provêm de diferentes fontes, os valores assumidos por atributos não possuem o mesmo domínio ou não estão formatados sob o mesmo tipo de dado, ainda que digam respeito à mesma descrição de uma entidade do mundo real. As principais motivações para a aplicação de procedimentos de integração de dados são, portanto, a presença de valores inconsistentes e a presença de valores redundantes.

Os **valores inconsistentes** são observados quando, para um mesmo atributo, ou para atributos equivalentes, são encontrados valores com discrepâncias em termos de tipo ou de domínio. Por exemplo: se, no caso de cadastro de cliente, um cliente preenche o campo “data de nascimento” com dia, mês e ano escritos por meio de números, e outro usa o dia e o mês por extenso, ocorrerá uma inconsistência nesse atributo; se, numa pesquisa de satisfação do cliente, pede-se para atribuir um conceito de qualidade, e o cliente atribui uma nota numérica, enquanto outro atribui um conceito por meio de letra; ou ainda, aproveitando o exemplo de pesquisa de satisfação, dois clientes apresentam a mesma avaliação qualitativa de satisfação, mas com notas finais diferentes. A inconsistência pode também ser caracterizada por conflito de unidades, representações e escalas. Por exemplo, um guia turístico que classifica os restaurantes do mundo todo com “estrelas”, enquanto outro usa notas de 1 a 10, ou, ainda, quando o custo nos restaurantes é divulgado usando a moeda local de cada país.

Soluções típicas para o tratamento de inconsistências incluem a remoção do exemplar que apresenta o valor inconsistente, a correção manual e a análise dos esquemas (*schema*) das fontes de dados e do conjunto de dados elaborados para a construção de procedimentos de correção automática.

Ainda na integração de dados, outro problema que deve ser manipulado é a **redundância de dados** . Este problema ocorre principalmente devido a três

fatores: uso de nomenclaturas diferentes para atributos equivalentes, porém provenientes de diferentes fontes de dados; a prática de armazenar atributos do tipo derivado (cujos valores são calculados a partir de valores em outros atributos); a inserção de exemplares repetidos no conjunto de dados, por consequência de um erro de aquisição de dados. Nesses casos, é interessante realizar a redução do conjunto. Essa redução pode ocorrer tanto de forma horizontal (eliminação de exemplares) quanto na forma vertical (eliminação de atributos), e pode também ser vista como um procedimento de seleção, se, em vez do procedimento procurar o que deve ser eliminado, ele for preparado para escolher o que é útil à análise realizada. É importante enfatizar que a redução de dados deve manter, no conjunto de dados, a mesma capacidade analítica do conjunto original.

A redução de dados pode ser implementada por meio de agregações em cubos OLAP (Seção 1.5.3 do Capítulo 1), análises de correlação para redução de dimensionalidade (eliminação de atributos), aplicação de medidas de ganho de informação (Capítulo 3), métodos de invólucro (do inglês *wrapper*) (Kohavi e John, 1997) e técnicas de compressão de dados, como Análise de Componentes Principais (PCA, do inglês, *Principal Component Analysis*) (Jolliffe, 2002) ou Análise de Fator (Hair *et al*., 1995).

2.2.3. Transformação de dados

Os atributos descritivos em um conjunto de dados assumem valores que contribuem para a caracterização de um exemplar. Porém, essa caracterização pode assumir valores de diferentes naturezas. É comum, por exemplo, o uso de diferentes tipos de sensores para aquisição de dados sobre um fenômeno, e cada sensor faz medições específicas, armazenadas em grandezas diferentes. Ainda, uma base de dados transacional que armazena características de objetos ou fenômenos possui atributos com valores pertinentes a diferentes domínios.

O problema nesses contextos é que as diferentes medições e caracterizações, frequentemente, possuem domínios de naturezas e grandezas bastante distintas. É possível, por exemplo, que o cliente do restaurante seja caracterizado pelos atributos descritivos “idade” e “salário”. O primeiro terá um domínio com valores pertencentes ao intervalo [0, 120] (sendo bem otimista). Já o segundo terá um domínio com valores pertencentes ao intervalo [0, 100.000], por exemplo. Essa diferença de domínios em diferentes atributos para um mesmo exemplar gera dificuldades em algoritmos que usam todos esses valores para compor um único valor de comparação entre exemplares. Por exemplo, se o critério de comparação for a distância euclidiana, a depender dos salários dos clientes sendo comparados, suas idades não farão qualquer diferença nas comparações. É também um problema importante a adequabilidade do tipo dos dados com as estratégias usadas pelos algoritmos de mineração de dados. Há algoritmos adequados para a aplicação sobre valores categóricos, outros, para uso com valores numéricos.

Para amenizar os efeitos causados por situações como essas, é necessário aplicar procedimentos de **transformação de dados** (também conhecidos como *data transformation*). Esses procedimentos abrangem a **normalização de dados** e a **conversão de dados** .

As técnicas de normalização mais comuns são a *normalização min-max*, que altera os valores extremos do conjunto de valores sob normalização e organiza os demais valores dentro do novo intervalo de domínio; e a *normalização z-score*, por meio da qual os valores são reorganizados pela média e desvio-padrão do conjunto de valores original.

A *normalização min-max* é uma das formas mais simples de normalização aplicada em problemas de mineração de dados. Quando aplicada a cada atributo de um conjunto de dados, permite que todos eles sejam organizados em uma mesma escala de valores. Esse procedimento é formalizado como:

$$v' = \frac{v - min(\mathbf{v})}{max(\mathbf{v}) - min(\mathbf{v})} (max_{novo} - min_{novo}) + min_{novo}$$ **Equação 2-7**

em que *v* é um valor do conjunto de valores **v**, *v'* é o valor *v* normalizado, min_{novo} é o novo limite mínimo desejado para **v**, e max_{novo} é o novo limite máximo desejado para **v**. Note que o conjunto de valores **v** é o conjunto de valores presumido por um atributo descritivo do conjunto de dados.

A *normalização z-score* também permite que os valores sejam padronizados. No entanto, isso ocorre com base na distribuição dos valores de cada atributo. Com esta normalização, cada valor do atributo é determinado pela posição que ocupa em relação aos demais, em uma distribuição simétrica. Esse procedimento é formalizado como:

$$v' = \frac{v - média(\mathbf{v})}{\sigma(\mathbf{v})}$$ **Equação 2-8**

em que *v* é um valor do conjunto de valores **v**, e as operações de média e desvio-padrão são como definido nas Seções 2.1.2 e 2.1.3.

A conversão de dados pode ser necessária, pelo menos, sob duas formas: conversão de valores numéricos para categóricos e conversão de valores categóricos para numéricos. O primeiro caso, também conhecido como discretização , pode ser implementado por meio da criação de categorias. Já o segundo caso é mais bem nomeado como codificação. Trata-se de representar os dados categóricos usando números, sem, no entanto, permitir que os algoritmos que analisarão tais valores possam estabelecer relações entre eles quando elas não existem originalmente (por exemplo, relações de ordem, prioridade, importância ou quantidade). Uma técnica comum para essa codificação é usar cadeias de *bits*. Por exemplo, para representar tipos de pratos (aperitivo, massa, carne, sobremesa), pode ser usada uma cadeia de *bits* de tamanho quatro (1000, 0100, 0010, 0001), sendo que, comumente, cada *bit* deve ser considerado um atributo descritivo do conjunto de dados; o atributo original "tipos de pratos" se transforma em quatro novos atributos que podem ser vistos como *flags*, que associam cada exemplar do conjunto de dados a um dos quatro "tipos de pratos". No caso de atributos que assumem

apenas dois valores nominais possíveis (masculino e feminino, por exemplo), é adequado e suficiente o uso de apenas um atributo valorado com 0 ou 1, em vez de uma cadeia de dois *bits*.

2.3. Exemplo didático para pré-processamento de dados

Nesta seção, uma série de procedimentos de pré-processamento de dados é aplicada a um conjunto de dados advindo de um processo hipotético de entrevistas realizadas com *chefs* de restaurantes. O objetivo aqui é ilustrar algumas estratégias de pré-processamento citadas na seção anterior. Suponha uma pesquisa realizada com 11 *chefs*, na qual lhes foram feitas perguntas sobre dados pessoais, sobre seu trabalho e sobre o restaurante onde trabalham. Os seguintes atributos descritivos referentes ao contexto de trabalho de cada um dos *chefs* foram obtidos:

1) Idade: valores inteiros, entre 18 (maioridade) e 90.

2) Salário: valores reais entre 0 e 30.000 mensais.

3) Formação: nenhuma, fundamental, médio, superior, pós-graduação.

4) Carga horária de trabalho: valores reais entre 1 e 8 horas diárias.

5) Quantidade de pratos sob sua responsabilidade: valores inteiros entre 1 e 30.

6) Responsabilidade sobre a carta de vinhos: sim ou não.

7) Capacidade do restaurante: valores inteiros entre 4 a 500 (pessoas).

8) Localização do restaurante: aeroportuário, bairro, centro, portuário, rodoviário, zona rural.

9) Tipo do cardápio: argentino, (especializado em) carnes, espanhol, francês, internacional, italiano, mediterrâneo, mexicano, oriental, (especializado em) pizzas, português, regional.

10) Lucratividade do restaurante: valores reais entre –100.000 a 100.000 mensais.

11) Saldo mensal: positivo (+) ou negativo (–) (derivado da lucratividade do restaurante).

A partir das respostas recebidas, foi possível organizar um conjunto de dados em que cada *chef* é considerado um exemplar, as perguntas são os atributos descritivos, e as respostas recebidas são os valores atribuídos a cada atributo. As respostas foram organizadas em uma base de dados, como na Tabela 2-9. Um resumo referente a uma análise preliminar sobre a base de dados leva às informações ilustradas no Quadro 2-2.

Tabela 2-9: Base de dados CHEFS

ID	idade	salário	formação	carga horária	qtde pratos	vinhos	capacidade	localização	tipo cardápio	lucra
1	45	7.000,00	superior	8,0	15	não	100	portuário	mediterrâneo	30.00
2	48	30.000,00	pós-graduação	8,0	25	sim	100	portuário	espanhol	150.0
3	31	7.500,00	superior	8,0	10	não	70	centro	oriental	-15.0(
4	40	8.000,00	médio	6,0	15	não	80	aeroportuário	argentino	50.00
5	25	8.000,00	superior	6,0	18	não	300	rodoviário	carnes	70.00
6	75	??	superior	6,0	23	não	150	centro	italiano	50.00
7	43	15.000,00	pós-graduação	8,0	20	sim	100	bairro	francês	70.00
8	29	7.000,00	médio	8,0	13	sim	35	centro	argentino	30.00
9	50	8.500,00	superior	??	12	sim	50	bairro	mexicano	-2.00(
10	51	15.000,00	superior	6,0	16	não	1	bairro	internacional	10.00
11	39	3.000,00	médio	4,0	10	não	50	zona rural	regional	10.00

Quadro 2-2: Resumo da base de dados de CHEFS

Quantidade de exemplares: 11			
Valores ausentes	Sim	Valores ruidosos	Sim
Valores inconsistentes	Sim	Redundância de dados	Sim
Quantidade de atributos: 12			

• identificadores	1	descritivos	11
		• quantitativos discretos	3
		• quantitativos contínuos	3
		• qualitativos nominais	4
		• qualitativos ordinais	1

Observando o Quadro 2-2, verifica-se a necessidade de pré-processamento para correção de alguns problemas. Segundo esse quadro, há valores ausentes, inconsistentes, ruidosos e redundantes. Porém, o pré-processamento deve ser aplicado com cuidado, pois não se pode incorrer no erro de descaracterizar o contexto real. Assim, segue a discussão sobre cada um dos problemas apontados no Quadro 2-2:

- *Problema 1 – valores ausentes*: os exemplares com ID *6* e *9* apresentam valores ausentes, respectivamente, nos atributos “salário” e “carga horária”. A remoção desses exemplares não é uma boa solução, pois o conjunto de dados já é muito pequeno em relação à quantidade de exemplares. Note, ainda, que remover o exemplar *9* é remover um dos dois exemplares que apresentam lucratividade negativa, ou seja, corre-se o risco de se perder uma valiosa informação sobre o contexto em análise. Preencher os valores manualmente seria possível, a partir de um retorno do entrevistado, porém, nem sempre é viável. O preenchimento automático pode ser uma solução, usando medidas estatísticas. Um estudo sobre a distribuição dos valores em cada atributo deveria ser feito, a fim de analisar a adequabilidade do preenchimento por meio dessas medidas.
- *Problema 2 – valores ruidosos* : por meio de uma inspeção visual, percebe-se que o valor do atributo “salário”, para o exemplar com ID *2*, precisa ser analisado com mais cuidado, pois pode representar um

outlier. O valor 30.000,00, embora dentro do intervalo de valores esperado, está um pouco distante de todos os demais valores assumidos por esse atributo na base de dados. A análise deste caso pode influenciar o tratamento dado aos valores ausentes. Se esse valor realmente representar um *outlier*, uma decisão deverá ser tomada sobre considerá-lo um ruído e alterá-lo, deslocando-o para dentro da distribuição dos demais valores, ou mantê-lo, entendendo que, de fato, esse é o único *chef* na amostra que consegue chegar a esse salário. A depender da decisão tomada, o conjunto de valores para cálculo da medida estatística usada para preenchimento do valor ausente no atributo pode mudar.

- *Problema 3 – valores inconsistentes* : o valor registrado para a "lucratividade do restaurante" associado ao exemplar *2*, está fora do intervalo de valores esperado, segundo o esquema estabelecido para a base de dados (nesse momento, considera-se que o esquema da base de dados foi corretamente projetado). Era esperado um valor máximo de 100.000,00, e este exemplar possui o valor 150.000,00 para o atributo em questão. O mesmo ocorre para o atributo "capacidade" do restaurante no exemplar *10*.
- *Problema 4 – valores redundantes* : no caso da base de dados em questão, a redundância aparentemente se manifesta nos atributos "lucratividade do restaurante" e "saldo mensal". É possível derivar o segundo atributo a partir do primeiro, e isso pode representar redundância. Nesse sentido, o segundo atributo poderia ser suprimido. Entretanto, a simplicidade do atributo "saldo mensal" pode ser benéfica para a tomada de decisão dentro de estratégias algorítmicas; sendo assim, um analista de dados pode optar por manter esse atributo no conjunto de dados.

Diante das discussões apresentadas, é preciso tomar algumas decisões a fim de pré-processar os dados para construir um (ou dois) conjunto(s) de

dados que possa(m) ser processado(s) por algoritmos de mineração. Sendo assim, considere:

- *Problema 2 – valores ruidosos*: manter o valor 30.000,00 para salário, assumindo ser, de fato, o valor real e não um ruído.
- *Problema 1 – valores ausentes*: usar a mediana[2] dos valores de salário (8.000,00) para substituir o valor faltante[3] do exemplar *6* e usar a média dos valores de carga horária (6,8) para o valor faltante do exemplar *9*.
- *Problema 3 – valores inconsistentes:* usar o valor médio da distribuição de valores do atributo "lucratividade do restaurante" (30.300,00)[4] para corrigir o valor inconsistente no exemplar *2*. E usar a mediana (90) para corrigir o valor inconsistente no atributo "capacidade", exemplar *10*.
- *Problema 4 – valores redundantes*: excluir o atributo "saldo mensal".

A partir desse pré-processamento e já colocando os dados num formato de conjunto de dados, tem-se o conjunto mostrado na Tabela 2-10. No conjunto de dados, há 11 exemplares, 10 atributos descritivos (entre numéricos e categóricos), e não há valores ausentes, redundantes ou inconsistentes, restando apenas o *outlier*, considerado pertinente ao contexto sob análise.

Tabela 2-10: Conjunto de dados CHEFS

	$x_{i\,1}$	$x_{i\,2}$	$x_{i\,3}$	$x_{i\,4}$	$x_{i\,5}$	$x_{i\,6}$	$x_{i\,7}$	$x_{i\,8}$	$x_{i\,9}$	$x_{i\,10}$
$\mathbf{x}_{\rightarrow 1}$	45	7.000,00	superior	8,0	15	não	100	portuário	mediterrâneo	30.000,00
$\mathbf{x}_{\rightarrow 2}$	48	30.000,00	pós-graduação	8,0	25	sim	100	portuário	espanhol	30.300,00
$\mathbf{x}_{\rightarrow 3}$	31	7.500,00	superior	8,0	10	não	70	centro	oriental	–15.000,00
$\mathbf{x}_{\rightarrow 4}$	40	8.000,00	médio	6,0	15	não	80	aeroportuário	argentino	50.000,00
$\mathbf{x}_{\rightarrow 5}$	25	8.000,00	superior	6,0	18	não	300	rodoviário	carnes	70.000,00
$\mathbf{x}_{\rightarrow 6}$	75	8.000,00	superior	6,0	23	não	150	centro	italiano	50.000,00
$\mathbf{x}_{\rightarrow 7}$	43	15.000,00	pós-graduação	8,0	20	sim	100	bairro	francês	70.000,00

$\mathbf{x}_{\rightarrow 8}$	29	7.000,00	médio	8,0	13	sim	35	centro	argentino	30.000,00
$\mathbf{x}_{\rightarrow 9}$	50	8.500,00	superior	6,8	12	sim	50	bairro	mexicano	–2.000,00
$\mathbf{x}_{\rightarrow 10}$	51	15.000,00	superior	6,0	16	não	90	bairro	internacional	10.000,00
$\mathbf{x}_{\rightarrow 11}$	39	3.000,00	médio	4,0	10	não	50	zona rural	regional	10.000,00

Há, agora, o problema de trabalhar com atributos descritivos numéricos e categóricos. Alguns algoritmos só aceitam valores numéricos, e outros trabalham apenas com valores categóricos. Assim, a conversão de dados se faz necessária. A discretização será aplicada para criar um conjunto de dados apenas de valores categóricos, e a codificação será usada na criação de um conjunto de dados com valores numéricos. Para a discretização, a divisão de valores em categorias será adotada, e, por simplificação, apenas duas categorias serão usadas em todos os atributos (idealmente, um estudo minucioso deve ser feito para cada atributo, para estimar o número de categorias). Para codificação , os atributos serão transformados usando cadeias de *bits*. Os dois conjuntos de dados resultantes são apresentados, respectivamente, na Tabela 2-11 e na Tabela 2-13. Os intervalos aplicados na discretização dos atributos são listados na Tabela 2-12.

Tabela 2-11: Conjunto de dados CHEFS apenas com atributos categóricos

	$x_{i\,1}$	$x_{i\,2}$	$x_{i\,3}$	$x_{i\,4}$	$x_{i\,5}$	$x_{i\,6}$	$x_{i\,7}$	$x_{i\,8}$	$x_{i\,9}$	$x_{i\,10}$
$\mathbf{x}_{\rightarrow 1}$	jovem	baixo	superior	alta	baixa	não	pequena	portuário	mediterrâneo	positiva
$\mathbf{x}_{\rightarrow 2}$	jovem	alto	pós-graduação	alta	alta	sim	pequena	portuário	espanhol	positiva
$\mathbf{x}_{\rightarrow 3}$	jovem	baixo	superior	alta	baixa	não	pequena	centro	oriental	negativa
$\mathbf{x}_{\rightarrow 4}$	jovem	baixo	médio	baixa	baixa	não	pequena	aeroportuário	argentino	positiva
$\mathbf{x}_{\rightarrow 5}$	jovem	baixo	superior	baixa	alta	não	alta	rodoviário	carnes	positiva

$\mathbf{x}_{\mathbf{6}}$	sênior	baixo	superior	baixa	alta	não	alta	centro	italiano	positiva
$\mathbf{x}_{\mathbf{7}}$	jovem	alto	pós-graduação	alta	alta	sim	pequena	bairro	francês	positiva
$\mathbf{x}_{\mathbf{8}}$	jovem	baixo	médio	alta	baixa	sim	pequena	centro	argentino	positiva
$\mathbf{x}_{\mathbf{9}}$	sênior	baixo	superior	alta	baixa	sim	pequena	bairro	mexicano	negativa
$\mathbf{x}_{\mathbf{10}}$	sênior	alto	superior	baixa	alta	não	pequena	bairro	internacional	positiva
$\mathbf{x}_{\mathbf{11}}$	jovem	baixo	médio	baixa	baixa	não	pequena	zona rural	regional	positiva

Tabela 2-12: Classes e intervalos usados na discretização dos atributos do conjunto de dados CHEFS. Note que os intervalos devem abranger todo o domínio de valores da variável

Atributo	**Intervalos**	**Valor categórico**
x_{i1}	[18 – 49] [50 – 90]	jovem e sênior
x_{i2}	[0 – 10.000]]10.000 – 30.000]	baixo e alto
x_{i4}	[1 – 6]]6 – 8]	baixa e alta
x_{i5}	[1 – 15] [16 – 30]	baixa e alta
x_{i7}	[4 – 100] [101 – 500]	pequena e alta
x_{i10}	[–∞ – 0]]0 – +∞]	negativa e positiva

Tabela 2-13: Conjunto de dados de dados CHEFS apenas com atributos numéricos. Os valores para o atributo x_{i3} e as cadeias de *bits* foram dimensionados seguindo os domínios completos dos atributos

	$\boldsymbol{x}_{i\,\mathbf{1}}$	$\boldsymbol{x}_{i\,\mathbf{2}}$	$\boldsymbol{x}_{i\,\mathbf{3}}$	$\boldsymbol{x}_{i\,\mathbf{4}}$	$\boldsymbol{x}_{i\,\mathbf{5}}$	$\boldsymbol{x}_{i\,\mathbf{6}}$	$\boldsymbol{x}_{i\,\mathbf{7}}$	$\boldsymbol{x}_{i\,\mathbf{8}}$	$\boldsymbol{x}_{i\,\mathbf{9}}$	$\boldsymbol{x}_{i\,\mathbf{10}}$	$\boldsymbol{x}_{i\,\mathbf{11}}$	$\boldsymbol{x}_{i\,12}$	$\boldsymbol{x}_{i\,13}$	$\boldsymbol{x}_{i\,14}$	$\boldsymbol{x}_{i\,15}$	$\boldsymbol{x}_{i\,16}$	$\boldsymbol{x}_{i\,17}$	$\boldsymbol{x}_{i\,18}$	$\boldsymbol{x}_{i\,19}$	$\boldsymbol{x}_{i\,20}$	$\boldsymbol{x}_{i\,21}$	$\boldsymbol{x}_{i\,22}$	$\boldsymbol{x}_{i\,23}$	$\boldsymbol{x}_{i\,2}$
$\mathbf{x}_{\mathbf{1}}$	45	7.000,00	4	8	15	0	100	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0
$\mathbf{x}_{\mathbf{2}}$	48	30.000,00	5	8	25	1	100	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0
$\mathbf{x}_{\mathbf{3}}$	31	7.500,00	4	8	10	0	70	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0
$\mathbf{x}_{\mathbf{4}}$	40	8.000,00	3	6	15	0	80	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
$\mathbf{x}_{\mathbf{5}}$	25	8.000,00	4	6	18	0	300	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0
$\mathbf{x}_{\mathbf{6}}$	75	8.000,00	4	6	23	0	150	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0

$\mathbf{x}_{\rightarrow 7}$	43	15.000,00	5	8	20	1	100	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0
$\mathbf{x}_{\rightarrow 8}$	29	7.000,00	3	8	13	1	35	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0
$\mathbf{x}_{\rightarrow 9}$	50	8.500,00	4	6,8	12	1	50	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
$\mathbf{x}_{\rightarrow 10}$	51	15.000,00	4	6	16	0	90	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
$\mathbf{x}_{\rightarrow 11}$	39	3.000,00	3	4	10	0	50	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0

Enfim, o conjunto de dados com atributos numéricos permite ilustrar o papel da normalização de dados. A fim de compreender com clareza essa questão, analise as medidas de distância euclidiana entre o exemplar x_1 e os demais exemplares (veja como calculá-la no Capítulo 3), considerando apenas dois atributos descritivos, por simplificação.[5] Na Tabela 2-14 estão ilustradas tais medidas, considerando os valores originais e os valores normalizados (*normalização min-max*) dos dois atributos descritivos. As distâncias entre os exemplares foram calculadas usando apenas um atributo, $dist\_x_{i1}$ e $dist\_x_{i26}$, e usando os dois atributos, $dist\_x_{i1}\_x_{i26}$.

Observe que, no caso das distâncias calculadas sobre os valores originais, o atributo $x_{i,26}$ exerce total influência sobre os resultados obtidos. Os valores de distância, considerando apenas o atributo ($dist\_x_{i26}$) e considerando os dois atributos ($dist\_x_{i1\_}x_{i26}$), são praticamente iguais. Apenas no caso da distância entre o exemplar x_1 e o exemplar x_8 há influência do atributo x_{i1}. Isto ocorre porque, no cálculo da distância, a subtração do valor do atributo x_{i26} em x_1 e x_8 resultará no valor zero. Já no caso dos resultados obtidos a partir dos dados normalizados, a influência dos dois atributos descritivos pode ser observada. Assim, a normalização é capaz de minimizar a influência excessiva de um ou de vários atributos.

Tabela 2-14: Cálculos de distância euclidiana entre o exemplar x_1 e todos os demais. O resultado da normalização dos atributos $x_{i,}$ e x_{i26} é apresentado nas colunas $x_{i,1}n$ e $x_{i,26}n$

	x_{i1}	$x_{i\,26}$				x_{i1}n	x_{i26}n			
$\mathbf{x}\rightarrow_1$	45	30.000,00	$dist\_x_{i1}$	$dist\_x_{i26}$	$dist\_x_{i1}\_x_{i26}$	0,40	0,53	$dist_x_{i1}$n	$dist_x_{i26}$n	$dist_x_{i1}$n_
$dist(\mathbf{x}\rightarrow_1, \mathrm{x}\rightarrow_2)$	48	30.300,00	3	300	300	0,46	0,53	0,06	0,00	0,06
$dist(\mathbf{x}\rightarrow_1, \mathbf{x}\rightarrow_3)$	31	– 15.000,00	14	45000	45000	0,12	0	0,28	0,53	0,59
$dist(\mathbf{x}\rightarrow_1,$	40	50.000,00	5	20000	20000	0,3	0,76	0,1	0,23	0,25

$\mathbf{x}\rightarrow_4$)										
dist($\mathbf{x}\rightarrow_1$, $\mathbf{x}\rightarrow_5$)	25	70.000,00	20	40000	40000	0	1	0,4	0,47	0,61
dist($\mathbf{x}\rightarrow_1$, $\mathbf{x}\rightarrow_6$)	75	50.000,00	30	20000	20000	1	0,76	0,6	0,23	0,64
dist($\mathbf{x}\rightarrow_1$, $\mathbf{x}\rightarrow_7$)	43	70.000,00	2	40000	40000	0,36	1	0,04	0,47	0,47
dist($\mathbf{x}\rightarrow_1$, $\mathbf{x}\rightarrow_8$)	**29**	**30.000,00**	16	0	16	0,08	0,52	0,32	0	0,32
dist($\mathbf{x}\rightarrow_1$, $\mathbf{x}\rightarrow_9$)	50	–2.000,00	5	32000	32000	0,5	0,15	0,10	0,37	0,38
dist($\mathbf{x}\rightarrow_1$, $\mathbf{x}\rightarrow_{10}$)	51	10.000,00	6	20000	20000	0,52	0,29	0,12	0,23	0,26
dist($\mathbf{x}\rightarrow_1$, $\mathbf{x}\rightarrow_{11}$)	39	10.000,00	6	20000	20000	0,28	0,29	0,12	0,23	0,26

2.4. Exemplo prático para pré-processamento de dados em R

Na solução didática apresentada na seção anterior, os problemas em relação à base de dados CHEFS foram determinados a partir de uma inspeção manual. No entanto, para bases de dados do mundo real, esse tipo de inspeção é praticamente inviável. Então, a mesma análise será explorada nessa seção, porém, com uso do ambiente R. Adicionalmente, são também apresentados alguns recursos que auxiliam na identificação e correção de problemas em conjuntos de dados. Por fim, são apresentados os comandos e funções em R que permitem realizar a transformação dos dados considerando as duas situações anteriormente apresentadas, discretização e codificação de dados.

Em R, uma função `summary( )` auxilia na construção de resumos sobre os dados. Essa função recebe o conjunto de dados como parâmetro de entrada, e apresenta um resumo como resultado. Quando o atributo é numérico, o resumo é apresentado em termos dos valores: mínimo (`Min`), máximo (`Max`), média (`Mean`), mediana (`Median`) e os quartis (primeiro – `1`st e terceiro – `3`rd); em caso de atributo categórico, o resumo é apresentado em termos da frequência absoluta dos valores presentes no atributo; valores ausentes são apresentados pelo símbolo `NA's` (de *not available*), com o número de ocorrências desse tipo de valor.

Por exemplo, considerando `d` a variável que recebe o conjunto de dados original, ainda sem pré-processamento, do exemplo didático por meio da leitura de um arquivo .csv,[6] a aplicação da função de resumo e os resultados obtidos por ela são apresentados a seguir e no Quadro 2-3:

```
> d <- read.table("C:\\Users\\LivroDM\\Exploracao\\chefs.csv",
header=TRUE,sep=";")
> summary(d)
```

Quadro 2-3: Resultado apresentado pela função SUMMARY, aplicada ao conjunto de dados CHEFS

```
       ID              idade            salario          formacao        carga          qtd.pratos
 Min.   : 1.0   Min.   :25.00   Min.   : 3000   medio        :3   Min.   :4.0   Min.   :10.00
 1st Qu.: 3.5   1st Qu.:35.00   1st Qu.: 7125   pos-graduacao:2   1st Qu.:6.0   1st Qu.:12.50
 Median : 6.0   Median :43.00   Median : 8000   superior     :6   Median :7.0   Median :15.00
 Mean   : 6.0   Mean   :43.27   Mean   :10900                     Mean   :6.8   Mean   :16.09
 3rd Qu.: 8.5   3rd Qu.:49.00   3rd Qu.:13375                     3rd Qu.:8.0   3rd Qu.:19.00
 Max.   :11.0   Max.   :75.00   Max.   :30000                     Max.   :8.0   Max.   :25.00
                NA's   :    1   NA's   :    1

 vinhos     capacidade        localizacao        tipo.cardapio     lucratividade     saldo
 nao:7    Min.   :  1.00   aeroportuario:1   argentino     :2   Min.   :-15000   -:2
 sim:4    1st Qu.: 50.00   bairro        :3   carnes        :1   1st Qu.: 10000   +:9
          Median : 80.00   centro        :3   espanhol      :1   Median : 30000
          Mean   : 94.18   portuario     :2   frances       :1   Mean   : 41182
          3rd Qu.:100.00   rodoviario    :1   internacional:1   3rd Qu.: 60000
          Max.   :300.00   zona rural    :    italiano      :1   Max.   :150000
                                              (Other)       :4
```

Outro recurso interessante que auxilia na inspeção do conjunto de dados é a função `str( )`. Essa função apresenta a estrutura (*structure*) do conjunto de dados, e o resultado de sua execução é ilustrado a seguir:

```
> str(d)

data.frame': 11 obs. of 12 variables:
$ ID : int 1 2 3 4 5 6 7 8 9 10 ...
$ idade : int 45 48 31 40 25 75 43 29 50 51 ...
$ salario : num 7000 30000 7500 8000 8000 NA 15000 7000 8500 15000
...
$ formacao : Factor w/ 3 levels "medio","pos-graduacao",..: 5 2 5
3 5 5 4 3
5 5 ...
$ carga : num 8 8 8 6 6 6 8 8 NA 6 ...
$ qtd.pratos : int 15 25 10 15 18 23 20 13 12 16 ...
$ vinhos : Factor w/ 2 levels "nao","sim": 1 2 1 1 1 1 2 2 2 1 ...
$ capacidade : int 100 100 70 80 300 150 100 35 50 1 ...
$ localizacao : Factor w/ 6 levels "aeroportuario",..: 4 4 3 1 5 3
2 3 2 2 ...
$ tipo.cardapio: Factor w/ 10 levels "argentino","carnes",..: 7 3
9 1 2 6 4 1 8
5 ...
$ lucratividade: num 30000 150000 -15000 50000 70000 50000 70000
30000 -2000
10000 ...
$ saldo : Factor w/ 2 levels "-","+": 2 2 1 2 2 2 2 2 1 2 ...
```

O resultado da função `str( )` apresenta na primeira linha o tipo da variável que está recebendo o conjunto de dados (`data.frame`), a quantidade de exemplares (`obs.` de *observations*) e o número de atributos (`variables`). As linhas seguintes, iniciadas pelo identificador `$,` apresentam nome, tipo e amostragem de valores da variável (atributo). Note que, no R, o tipo `int` é designado para o tipo numérico discreto, e `num` para numérico contínuo. O tipo categórico é definido como `Factor`. Quando esse tipo de valor está presente no conjunto de dados, duas outras informações são apresentadas: o nível (`levels`), que representa a quantidade de valores existentes na variável na amostragem sob análise, alguns dos valores e uma conversão para números (apenas por conveniência da linguagem e não para fins de mineração de dados). O tipo `Factor` é, intrinsicamente, um categórico nominal. Para que um categórico ordinal seja tratado pelo R, é necessário fazer a conversão da variável com o uso da função `ordered( )`. Essa função recebe um `Factor`, a sequência dos valores possíveis em ordem de importância, e retorna um categórico ordinal. Como exemplo, considere a variável `formacao`. A impressão dessa variável como `Factor` tem o seguinte resultado:

```
> d$formacao
[1] superior pos-graduacao superior medio superior superior
[7] pos-graduacao medio superior superior medio
Levels: medio pos-graduacao superior
```

A seguir, a impressão da variável convertida para ordinal:

```
> ordered(d$formacao,c("medio","superior","pos-graduacao"))
[1] superior pos-graduacao superior medio superior superior
[7] pos-graduacao medio superior superior medio
Levels: medio < superior < pos-graduacao
```

Note que o descritivo `Levels` em nominal foi apresentado como um simples conjunto de valores e, após a conversão para ordinal, o descritivo

`Levels` é apresentado como uma lista ordenada de valores, com o sinal "<" para designar a hierarquia na lista.

Finalizada a apresentação dos recursos para a inspeção do conjunto de dados, os próximos procedimentos resolvem os *problemas 1 a 4*, elencados no exemplo didático.

A solução para o *Problema 2* não necessita de implementação, haja vista a tomada de decisão em manter o valor 30.000,00 para o salário no exemplar *2* do conjunto de dados *Chefs*. Mas é importante lembrar que uma alteração nesse valor influenciaria a solução para o *Problema 1*, e, por isso, a solução do *Problema 2* foi analisada primeiro. Nesse caso, a inspeção do resumo dos dados por meio da análise dos valores extremos (mínimo e máximo), assim como os quartis, indicou que o valor 30 mil era um o*utlier*. Perceba que um gráfico de caixa permite a visualização dessa situação (Figura 2-7).

```
> boxplot(d$salario)
```

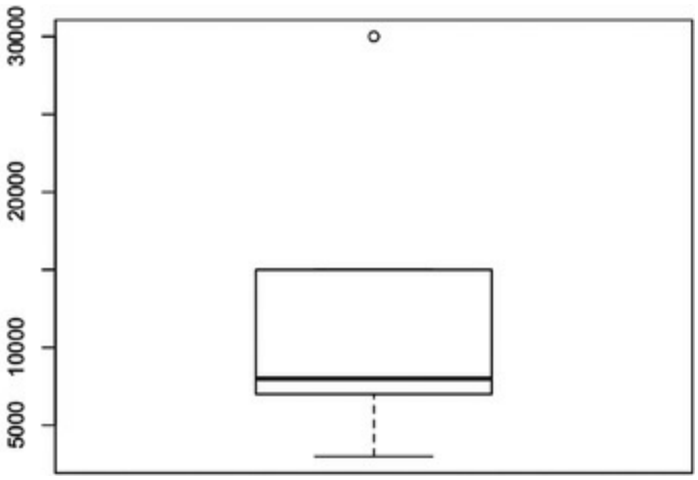


Figura 2-7: Gráfico de caixa para o atributo "salário" do conjunto de dados CHEFS

Para o preenchimento do valor ausente, *Problema 1*, considere que o valor desejado no preenchimento seja previamente calculado (a mediana – 8.000,00 – no caso do atributo "salário" e a média – 6,8 – no caso do atributo "carga horária"), como mostrado no resultado da função `summary( )`. Os preenchimentos são executados como segue:

```
> d$salario[is.na(d$salario)==TRUE]<-8000.00
> d$carga[is.na(d$carga)==TRUE]<-6.8
```

Para explicar as instruções, considere o primeiro caso – preenchimento da variável "salário": `d` é a variável que armazena o conjunto de dados, e `d$salario` indica a coluna referente ao atributo `salario`; o trecho entre colchetes retorna os índices das linhas (exemplares) nas quais a condição `is.na(d$salario)==TRUE` é satisfeita, sendo `is.na` uma consulta para verificar se o valor do atributo entre parênteses (`d$salario`) é faltante (`na` – *not available*); em caso positivo, o comando atribui (`<-`) o valor `8000.00` à variável (em que antes havia um `na`). Esse é um importante recurso da programação funcional e imperativa que permite a construção de códigos enxutos e eficientes, já que deixa para a própria linguagem de programação o trabalho de decidir a melhor forma de realizar uma varredura, ou *loop*, neste caso.

A solução para o *Problema 3* é ajustar o valor de um exemplar no atributo "lucratividade", usando a média dos valores existentes no atributo, desconsiderando o valor inconsistente, e ajustar um exemplar no atributo "capacidade" usando a mediana dos valores existentes no atributo, também desconsiderando o valor inconsistente. De maneira similar ao apresentado para o *Problema 3*, os seguintes comandos devem ser executados:

```
> d$lucratividade[d$lucratividade==150000.00] <- 30300.00[7]
> d$capacidade[d$capacidade==1] <- 90
```

No caso do *Problema 4*, a análise deve começar pelo esquema do conjunto de dados no qual se especifica que o atributo "saldo" é derivado de "lucratividade". Para uma análise quantitativa, pode-se também calcular a correlação de Pearson das duas variáveis. Uma correlação alta e positiva indica que uma das variáveis pode ser retirada da análise, a critério do analista dos dados. Para isso, a variável `saldo` deve ser convertida para

valores numéricos, da seguinte forma, convencionando-se valores `+1` e `-1` para os valores, `+` e `-`, respectivamente:

```
> saldo_numerico <- matrix(rep(0,11),nrow=11,ncol=1)
> saldo_numerico[d$saldo=="+"] <- + 1
> saldo_numerico[d$saldo=="-"] <- - 1
```

E, para a variável `lucratividade`, nesse caso, interessam para a correlação apenas os sinais dos valores. Isso pode ser obtido com o uso da função `sign( )`, que transforma os valores da variável para `+1` ou `-1`, de acordo com o sinal do valor por ela assumido.

```
> lucratividade <- sign(d$lucratividade)
```

Finalmente, a correlação de Pearson entre as duas variáveis pode ser calculada com a aplicação da função `cor( )`:

```
> cor(saldo_numerico,lucratividade)
[,1]
[1,] 1
```

Com os problemas do conjunto de dados corrigidos, a transformação dos valores usando comandos do R pode ser ilustrada. A solução para transformar variáveis do tipo numérico para o tipo categórico (discretização) se faz de maneira bastante semelhante à apresentada no *Problema 3*. A diferença é que, aqui, a condição colocada no `[]` utiliza outros operadores de comparação (`<=`, `<`, `>` e `>=`) e operadores lógicos, (AND `&`, OR `|`). Na Tabela 2-15, estão os procedimentos de conversão de todas as variáveis numéricas seguindo as faixas definidas na Tabela 2-12. O conjunto de dados resultante da conversão está ilustrado na Tabela 2-11.

Tabela 2-15: Comandos para discretização de atributos

Atributo	Procedimento para transformação de valores numéricos em categóricos em R
x_{i1}	`> d$idade[18<=d$idade & d$idade<=49] <- "jovem"` `> d$idade[50<=d$idade & d$idade<=90] <- "senior"`
x_{i2}	`> d$salario[0.00<=d$salario & d$salario<=10000.00] <- "baixo"` `> d$salario[10000.00<d$salario & d$salario<=30000.00] <- "alto"`
x_{i4}	`> d$carga[1<=d$carga & d$carga<=6] <- "baixa"` `> d$carga[6<d$carga & d$carga<=8] <- "alta"`
x_{i5}	`> d$qtd.pratos[1<=d$qtd.pratos & d$qtd.pratos<=15] <- "baixa"` `> d$qtd.pratos[16<=d$qtd.pratos & d$qtd.pratos<=30] <- "alta"`
x_{i7}	`> d$capacidade[4<=d$capacidade & d$capacidade<=100] <- "pequena"` `> d$capacidade[101<=d$capacidade & d$capacidade<=500] <- "alta"`
x_{i10}	`> d$lucratividade[d$lucratividade>0] <- "positiva"` `> d$lucratividade[d$lucratividade<=0] <- "negativa"`

Para a transformação de valores categóricos em numéricos (codificação), dois procedimentos devem ser usados: para valores categóricos ordinais, comandos de codificação serão aplicados; para valores categóricos nominais, cadeias de *bits* serão criadas. Em ambos os casos, como a transformação não se dá por meio de uma função do R, o procedimento adotado é semelhante ao apresentado na solução do *Problema 4*: no primeiro caso, a codificação gera um único atributo; no segundo caso, é criada uma matriz cujo número de colunas será dado pela quantidade de *bits* que se deseja organizar no conjunto de dados. Tal matriz será inicializada com valores zero, adequadamente substituídos por 1, de acordo com uma convenção pré-estabelecida.

Para o atributo categórico ordinal "formação", a seguinte sequência de comandos deve ser executada:

```
> formacao_codes <- matrix(rep(0,11),nrow=11,ncol=1)
> formacao_codes [d$formacao=="nenhuma"] <- 1
> formacao_codes [d$formacao="fundamental"] <- 2
> formacao_codes [d$formacao=="medio"] <- 3
> formacao_codes [d$formacao=="superior"] <- 4
> formacao_codes [d$formacao=="pos-graduacao"] <- 5
```

Para os atributos categórico nominais “vinhos”, “localização” e “tipo de cardápio”, a seguinte sequência de comandos deve ser executada:

```
> vinho_bits <- matrix(rep(0,22),nrow=11,ncol=1)
> vinho_bits [(d$vinhos=="sim"),1] <- 1

> localizacao_bits <- matrix(rep(0,66),nrow=11,ncol=6)
> localizacao_bits [(d$localizacao=="aeroportuario"),1] <- 1
> localizacao_bits [(d$localizacao=="bairro"),2] <- 1
> localizacao_bits [(d$localizacao=="centro"),3] <- 1
> localizacao_bits [(d$localizacao=="portuario"),4] <- 1
> localizacao_bits [(d$localizacao=="rodoviario"),5] <- 1
> localizacao_bits [(d$localizacao=="zona rural"),6] <- 1
> cardapio_bits <- matrix(rep(0,132),nrow=11,ncol=12)
> cardapio_bits[(d$tipo.cardapio=="argentino"),1] <- 1
> cardapio_bits[(d$tipo.cardapio=="carnes"),2] <- 1
> cardapio_bits[(d$tipo.cardapio=="espanhol"),3] <- 1
> cardapio_bits[(d$tipo.cardapio=="frances"),4] <- 1
> cardapio_bits[(d$tipo.cardapio=="internacional"),5] <- 1
> cardapio_bits[(d$tipo.cardapio=="italiano"),6] <- 1
> cardapio_bits[(d$tipo.cardapio=="mediterraneo"),7] <- 1
> cardapio_bits[(d$tipo.cardapio=="mexicano"),8] <- 1
> cardapio_bits[(d$tipo.cardapio=="oriental"),9] <- 1
> cardapio_bits[(d$tipo.cardapio=="pizza"),10] <- 1
> cardapio_bits[(d$tipo.cardapio=="portugues"),11] <- 1
> cardapio_bits[(d$tipo.cardapio=="regional"),12] <- 1
```

As transformações realizadas geraram novos atributos que precisam ser recompostos para a formação de um novo conjunto de dados. Segue um exemplo de como um novo conjunto pode ser composto. O conjunto de dados `d_novo` possui atributos de diversas naturezas. O analista de dados deve escolher os atributos que comporão o conjunto de dados de acordo com cada situação de análise.

```
> d_novo <- cbind(d$idade, d$salario, formacao_codes, d$carga,
d$qtd.pratos, vinho_bits, d$capacidade, localizacao_bits,
cardapio_bits, d$lucratividade)
```

O último procedimento ilustrado no exemplo didático refere-se à normalização dos atributos. No exemplo, considerou-se a normalização pelo método *min-max*. Como essa normalização não está disponível no ambiente em R, ela deve ser implementada da seguinte forma:

```
> min.max <- function(x,min,max){
x_norm <- (x-min(x)) / (max(x)-min(x)) * (max-min) + min
return(x_norm)
}
```

A função criada chama-se `min.max`. Ela tem como parâmetros o atributo (variável `x`), a ser normalizado, e os valores novos desejados, mínimo (`min`) e máximo (`max`). A função retorna o atributo normalizado. Por exemplo, a normalização do atributo "idade" (considerando o conjunto original, no qual os valores para idade são numéricos) com os valores mínimos e máximos, respectivamente sendo 0 e 1, é da seguinte forma (o resultado desta normalização está disponível na Tabela 2-14):

```
> idade_norm <- min.max(d$idade,0,1)
```

Por fim, caso haja interesse em mensurar a similaridade pelo uso de distâncias, assunto a ser tratado nos capítulos subsequentes, pode-se usar a função chamada `dist`, que tem como parâmetros a variável a ser comparada e a distância a ser usada (`method`). A distância entre os exemplares, considerando o atributo "idade" normalizado (`idade_norm`) é:

```
> dist(idade_norm,method="euclidean")
```

O resultado do cálculo das distâncias está disponível na Tabela 2-14.

2.5. Preparação de dados não estruturados – do tipo texto

Enquanto os dados estruturados estão em uma organização e representação adequadas para serem submetidos a análises exploratórias e para passarem por procedimentos de pré-processamento , os dados não estruturados necessitam de uma preparação prévia, que pode ser considerada um tipo de pré-processamento. Para ilustrar procedimentos de preparação, esta seção trata de dados do tipo texto.[8] Entretanto, mesmo passando por essa preparação, os dados do tipo texto, muito provavelmente, não se apresentarão totalmente adequados para serem submetidos a algumas das estratégias de análise apresentadas neste livro, uma vez que podem assumir uma representação numérica caracterizada por esparsidade. O tratamento de dados esparsos exige a aplicação de procedimentos de análises específicos, como apresentado por Francis Bach (2010).

Em bases de dados textuais, também conhecidas como *corpora* , cada exemplar é tratado como um “documento”. Cada documento em um *corpus* pode assumir diferentes características em relação a, por exemplo, tamanho do texto (sequência de caracteres), tipo de conteúdo (assunto que aborda), língua na qual é escrito (inglês, português, japonês, árabe etc.) ou tipo de linguagem adotada (formal, coloquial, poética, irônica etc.).

A transformação de um *corpus* em um conjunto de dados que possa ser submetido à procedimentos de análise consiste em um processo que gera uma representação capaz de descrever cada documento em termos de suas características. Uma das formas mais tradicionais de representar um documento, muito conhecida como representação *bag of words,* consiste em usar uma lista de ocorrências de palavras. A lista de todas as palavras que ocorrem em todos os documentos de um *corpus* pode ser nomeada de dicionário. Com base no dicionário e na frequência com que as palavras do

dicionário aparecem nos documentos e no *corpus*, será possível construir uma representação para o *corpus*. Tal representação pode ser, então, tratada como um conjunto de dados, seguindo a definição apresentada no Capítulo 1.

Formalmente, nesta seção, um *corpus* já representado como um conjunto de dados é definido por um conjunto de *n* documentos (***doc***), ou seja, $\Psi = \{\boldsymbol{doc}_1, \boldsymbol{doc}_2, ..., \boldsymbol{doc}_n\}$.[9] Cada um dos documentos, por sua vez, é definido como um conjunto de *d* termos (radicais, palavras ou conjunto de palavras), de forma que $\boldsymbol{doc}_i = (wte_{i1}, wte_{i2}, ..., wte_{ij}, ..., wte_{id}\}$ com $i = \{1, ..., n\}$, sendo que wte_{ij} assume os valores 1 ou 0, em uma representação binária, na qual o valor 1 significa que o termo está presente no documento, e o valor 0 significa que o termo não está presente no documento. Ou wte_{ij} pode assumir valores reais, em uma representação contínua, sendo que o valor assumido representa um peso que relaciona o termo com o documento, ou relaciona o termo com o documento e com o *corpus*, a depender do tipo de representação contínua escolhida. Seguindo a representação de conjunto de dados usada neste livro, cada documento é visto como um exemplar do conjunto de dados, e cada termo é visto como um atributo descritivo do documento.

Entretanto, originalmente, um *corpus* precisa ser preparado por meio da execução de alguns procedimentos, para que ruídos ou informações não discriminantes sejam retiradas dos seus documentos. Esse tratamento é importante para que sua representação na forma de um conjunto de dados seja adequada para a descoberta de conhecimento. Esta seção é dedicada a apresentar alguns desses procedimentos.

Exemplos típicos de aplicações que envolvem mineração de textos e que podem servir de motivação para o estudo desta seção são o reconhecimento de *spam* em caixas de mensagens, a análise de sentimento em redes sociais e o desenvolvimento de sistemas de recomendação. No primeiro caso, o *corpus* é formado por mensagens que se constituem como *spams* e também mensagens que não são *spams*. Então, se pelo menos parte desse *corpus* estiver rotulado, um classificador pode ser construído para tomar a decisão

sobre se novas mensagens são ou não *spams*. Processo semelhante pode ser realizado sobre um *corpus* formado por publicações presentes em redes sociais, e a análise de sentimento pode ser implementada por meio da resolução de tarefas de classificação. Um *corpus* formado por notícias, ou por descrições ou avaliações sobre produtos de uma empresa, pode dar suporte para a construção de sistemas de recomendação nos quais notícias (ou produtos) pertencentes a determinada classe podem ser recomendadas a um leitor (comprador) que acessou outra notícia (produto) também daquela classe. Alternativamente, outras tarefas de mineração de dados se aplicam nesses casos, como o agrupamento de dados e a descoberta de regras de associação.

2.5.1. Um pequeno *corpus*

Diferentemente de dados provenientes de um banco de dados (modelo relacional ou modelo estrela – Capítulo 1), um *corpus* é originalmente apresentado como um conjunto de documentos do tipo texto – cadeia de caracteres, ou seja, é um caso de uma base de dados com conteúdo não estruturado (veja mais detalhes na Seção 2.8). Um exemplo simplificado de um *corpus* com três documentos é apresentado no Quadro 2-4. Os textos nesse exemplo são referentes a três postagens em uma rede social, realizadas por clientes de um restaurante.

Quadro 2-4: Exemplo de um *corpus* com três documentos

doc1	`Ambiente agradável e tranquilo. Comida e música com qualidade. Adoramos o Filé à Parmegiana.`
doc2	`O Filé à Parmegiana da cidade. Ambiente agradável e qualidade no atendimento.`
doc3	`O Filé à Parmegiana com fritas é uma delícia.`

Uma possível aplicação de mineração de dados nesse contexto é a resolução da tarefa de agrupamento para descoberta de perfis de opiniões. A partir do conhecimento de características desses perfis, seria possível direcionar melhor as campanhas de publicidade lançadas pelo restaurante ou dar suporte a algum outro tipo de planejamento estratégico e gerencial. Mas, observando as postagens do Quadro 2-4, fica evidente que encontrar um padrão de opinião vai exigir que o algoritmo entenda linguagem natural, pois os padrões possíveis, a exemplo de "satisfação com o Filé à Parmegiana", estão indiretamente presentes em construções linguísticas que fazem uso de palavras, conceitos e construções gramaticas diferentes em cada postagem.

As características referentes ao estilo de escrita, quantidade de caracteres, uso de sinônimos etc. usadas em diferentes textos não seguem uma estrutura comum que permita diretamente medir (quantificar) similaridades entre documentos. Como o objetivo aqui não é estudar e automatizar a análise de estruturas gramaticais no documento (o que se tornaria uma tarefa específica para a área de Processamento de Linguagem Natural), é preciso preparar esses dados textuais de forma a adequá-los às técnicas de mineração de dados discutidas neste livro. Assim, é necessário, em linhas gerais, transformar esse *corpus* no conjunto de dados Ψ, como já definido.

2.5.2. Processo para geração do conjunto de dados Ψ

Para transformar o *corpus* original em uma representação que defina o conjunto de dados Ψ, uma série de procedimentos devem ser executados,[10] e a execução de tais procedimentos se constitui em fases do processo (Figura 2.8) que podem ser entendidas como um tipo de pré-processamento de dados.

Figura 2-8: Processo para geração do *conjunto de dados* Ψ

A primeira fase do processo é a **análise lexical** e tem como objetivo gerar uma lista de *tokens* ou a primeira lista de termos (ao final do processo, a lista de termos pode ser modificada e reduzida). Essa fase recebe os documentos do *corpus* original, os converte em uma sequência de caracteres e então procede com a geração dos *tokens*. A geração ocorre a partir da eliminação de caracteres de pontuação,[11] como ponto, vírgula, ponto-e-vírgula, ponto de exclamação, ponto de interrogação e outros caracteres que se julgarem necessários dentro do contexto em que o *corpus* se encontra (por exemplo, o caractere "hífen" pode ou não ser eliminado nesse processo). Além disso, é frequentemente útil fazer a eliminação dos dígitos, visto que é alta a probabilidade de eles não carregarem sentido semântico útil para representar o conteúdo de um documento, e também a eliminação de acentos, se análises de variações gramaticais não fizerem parte do objetivo em questão. Os *tokens* podem ser gerados a partir da definição de um (ou mais) separador(es), como o espaço em branco. Por fim, ainda nessa fase, como procedimento alternativo, a capitalização da letra pode ser alterada, de forma que todos os termos sejam transformados para letras maiúsculas ou minúsculas. A aplicação desses procedimentos sobre o *corpus* apresentado no Quadro 2-4 resulta na lista de *tokens* apresentada no Quadro 2-5.

Quadro 2-5: Lista de *tokens* obtida após a execução da análise lexical sobre o *corpus* do Quadro 2-4. O símbolo | está sendo usado nessa ilustração como um separador de *tokens*

doc1	`ambiente | agradável | e | tranquilo | comida | e | música | com | qualidade | adoramos | o | filé | à | parmegiana`
doc2	`o | filé | à | parmegiana | da | cidade | ambiente | agradável | e | qualidade | no | atendimento`
doc3	`o | filé | à | parmegiana | com | fritas | é | uma | delícia`

É importante ressaltar que cada problema de mineração de texto está dentro de determinado domínio ou área de aplicação, e, por isso, informações e conhecimento inerentes ao domínio ou área de aplicação devem ser sempre

considerados em todas as tomadas de decisão no processo de pré-processamento do *corpus*. Por exemplo, se no problema for importante considerar o endereço de uma página Web, o URL (*Uniform Resource Locator*), a eliminação de caracteres especiais vai prejudicar o processo de análise, e, nesse caso, é importante que seja feito um tratamento diferenciado, mantendo os caracteres especiais como “dois-pontos”, “barras” e “pontos” quando no contexto de URLs. O mesmo tipo de cuidado na tomada de decisão deve ser considerado na eliminação de dígitos. Se o *corpus* trata de discussões sobre preços praticados nos restaurantes, os dígitos podem permitir uma análise mais, ou menos, refinada sobre o conteúdo das discussões. Nesse caso, há um sentido semântico importante associado aos dígitos. A transformação da capitalização do texto, por ser alternativa, não deve comprometer o processo. Ela existe para tornar a representação homogênea. No entanto, pode alterar o significado de um termo, por exemplo, “Prato” pode ser um tipo de refeição, e “prato”, um utensílio doméstico. Nesse caso, caracteriza o problema chamado de ambiguidade, muito frequente, por exemplo, em análise de textos provenientes de interações em redes sociais. Para melhor lidar com tais problemas, faz-se útil estudar Gramáticas Regulares (Menezes, 2011) e soluções para desambiguação (Agirre e Edmonds, 2007).

A segunda fase do processo (Figura 2-8) trata da **eliminação de termos irrelevantes** para a análise de documentos. Sabe-se que um texto-padrão contém artigos definidos e indefinidos, preposições, pronomes, numerais, conjunções e advérbios. Palavras que pertencem a essas classes gramaticais são importantes na construção do discurso expresso por meio do texto, mas são frequentemente irrelevantes em um processo de mineração de texto no qual se quer descobrir padrões, pois não carregam sozinhas um sentido semântico e são, geralmente, muito frequentes em qualquer tipo de texto. Por isso, tais palavras devem fazer parte de uma lista de termos irrelevantes, também conhecida como *stopwords* , e devem ser removidas dos textos. A

remoção das *stopwords* pode reduzir o tamanho de um documento na taxa de 30% a 50% (Rijsbergen, 1979), o que já diminui a complexidade do espaço do conjunto de dados submetido à mineração. Além da redução da dimensionalidade do problema, o documento passa a ser uma lista de *tokens* formada por termos com maior representatividade no contexto do problema e, portanto, de maior poder discriminatório. Considerando uma lista de *stopwords* formada, em parte, por termos referentes às classes gramaticais antes citadas, a aplicação desse processo sobre o conteúdo do Quadro 2-5 resulta no conteúdo apresentado no Quadro 2-6.

Quadro 2-6: Lista de *tokens* obtida após a remoção de *stopwords* sobre o conteúdo do Quadro 2-5

doc1	`ambiente | agradável | tranquilo | comida | música | qualidade | adoramos | filé | parmegiana`
doc2	`filé | parmegiana | cidade | ambiente | agradável | qualidade | atendimento`
doc3	`filé | parmegiana | fritas | delícia`

É importante notar que, após a execução das duas primeiras fases do processo em discussão, os textos já assumem uma representação que se distancia da linguagem natural. Na representação assumida no Quadro 2-6, por exemplo, já é possível perceber que os textos são parecidos entre si por terem palavras em comum e não apresentarem mais as construções gramaticais complexas que exigem uma capacidade de interpretação de texto apurada para perceber a similaridade entre eles.

A lista de *stopwords*, embora classicamente inclua as classes gramaticas comentadas, pode ser modificada de acordo com o contexto dos conteúdos dos textos ou de acordo com a tarefa de mineração de tex- tos pretendida. Nesse sentido, pode ser interessante executar o processo de preparação do *corpus* de forma iterativa e interativa, modificando a lista de *stopwords* se for observado que existem palavras influenciando negativamente as análises. No sentido oposto, algumas classes gramaticais geralmente presentes na lista de

stopwords podem ser retiradas da lista se o objetivo for analisar estilo de escrita ou expressões de emoção.

Redução do termo ao seu radical é a terceira fase do processo descrito na Figura 2-8. Um termo no documento pode sofrer variações, como plural, gerúndio, verbos flexionados, aumentativo, diminutivo etc.; e, na análise do documento, termos com o mesmo radical (*stem*) não devem ser tratados como diferentes. Assim, com a redução do termo ao seu radical, processo também conhecido por *stemming*, os prefixos ou sufixos são eliminados, possibilitando a uniformização de termos. Essa fase também tem o efeito de reduzir o tamanho de um documento ou o tamanho do dicionário de termos associado a um *corpus*.

Na literatura, existem diferentes técnicas para encontrar o radical de um termo, como *table look-up*, em que se mantêm os radicais em uma tabela que representa um dicionário, ou *affix removal*, que remove os sufixos e prefixos de um termo. Ainda, outras formas de tratar o problema de diminuição das variações dos termos são: *SucessorVariety*, que considera os morfemas da língua, *n-gramas*, em que os termos são separados em sequências de *n tokens*. Para redução do termo ao seu radical, algoritmos vêm sendo propostos, sendo um dos mais usados o algoritmo de Porter (Porter, 1980). O algoritmo de Porter tem sido a base para a implementação de muitos softwares especializados (*stemmers*), porém, muitos deles apresentam várias imprecisões. Há, nesse contexto, uma iniciativa interessante: a *Snowball*, linguagem na qual *stemmers* podem ser definidos, e a partir da qual softwares *stemmers* eficientes podem ser gerados (em ANSI C ou Java).[12] Um algoritmo de *stemming* específico para a língua portuguesa já foi desenvolvido no projeto *Snowball*.[13] Na língua portuguesa, o problema de redução dos termos assume uma complexidade maior que em línguas como o inglês. Isso ocorre devido ao grande número de formas para o plural e para flexões verbais e as diferentes regras para tratamento de aumentativo, diminutivo, feminino, masculino.

Considerando o algoritmo de *stemming* para a língua portuguesa do projeto *Snowball* aplicado ao conteúdo do Quadro 2-6, obtém-se o conjunto de termos apresentados no Quadro 2-7.

Quadro 2-7: Aplicação do *stemmer* sobre os termos apresentado no Quadro 2-6

doc1	ambient | agrad | tranquil | com | músic | qualidad | ador | fil | parmegian
doc2	fil | parmegiana | cidad | ambient | agrad | qualidad | atend
doc3	fil | parmegian | frit | delíci

Então, nesse ponto do processo, é possível criar um dicionário de termos presentes no *corpus*, o qual será útil na próxima fase. O dicionário, em ordem alfabética, é mostrado no Quadro 2-8.

Quadro 2-8: Dicionário de termos obtido a partir do conteúdo do Quadro 2-7

termo 1	ador	termo 8	fil
termo 2	agrad	termo 9	frit
termo 3	ambient	termo 10	músic
termo 4	atend	termo 11	parmegian
termo 5	cidad	termo 12	qualidad
termo 6	com	termo 13	tranquil
termo 7	delíc		

Finalmente, realiza-se a geração do conjunto de dados Ψ por meio da **construção de uma representação vetorial** para os documentos (última fase do processo ilustrado na Figura 2.8). Trata-se, na realidade, de um mapeamento das relações entre termos, documentos e *corpus* para dados numéricos. Em linhas gerais, pode-se dizer que é a atribuição de pesos a cada termo presente no dicionário de termos.

Um tipo de representação simplificada é obtida atribuindo valores binários (ou pesos binários) que representam a presença ou ausência do termo em um documento. Nesse caso, cada documento do *corpus* original passa a ser um documento $\boldsymbol{doc}_i$ no conjunto de dados Ψ, representado por um vetor de 0s e 1s considerando todos os termos do dicionário de termos. Dessa

forma, tem-se que, para esse tipo de representação, $wte_{ij} \in \{0,1\}$, em que 0 significa ausência do termo *j* no documento *i* e 1 significa presença do termo *j* no documento *i*. Para esse tipo de representação, o conjunto de dados para o *corpus* (Quadro 2-4), considerando o dicionário de termos do Quadro 2-8, é mostrado na Tabela 2-16:

Tabela 2-16: Conjunto de dados Ψ considerando uma representação binária

	ador	**agrad**	**ambient**	**atend**	**cidad**	**com**	**delíc**	**fil**	**frit**	**músic**	**parmegian**	**qualidad**	**tranq**
Ψ	wte_1	wte_2	wte_3	wte_4	wte_5	wte_6	wte_7	wte_8	wte_9	wte_{10}	wte_{11}	wte_{12}	wte_{13}
doc_1	1	1	1	0	0	1	0	1	0	1	1	1	1
doc_2	0	1	1	1	1	0	0	1	0	0	1	1	0
doc_3	0	0	0	0	0	0	1	1	1	0	1	0	0

Note que o número de vezes que o termo aparece no documento não é considerado na representação binária. Esse tipo de quantificação, chamada de frequência do termo ou simplesmente *tf* (do inglês, *term of frequency*) pode também constituir-se como uma forma de representação, na qual números inteiros podem aparecer nas coordenadas dos vetores de documento. Devido à simplicidade do exemplo tratada aqui, a representação por frequência do termo será igual à representação binária.

Contudo, tanto a representação binária quanto a representação por frequência do termo por documento podem esconder relações existentes entre a presença das palavras no documento e no *corpus*, simplificando a representação do *corpus* como um todo. Por exemplo, se um termo **A** é muito frequente em alguns documentos, porém raro no *corpus*, ele deveria ter um peso maior na representação de tais documentos que um termo **B**, frequente nos documentos e também no *corpus*. Nesse caso, o termo **A** é uma característica que discrimina um subconjunto de documentos do conjunto total de documentos. Para obter uma medida que pondere os termos dos

documentos de forma representativa, é preciso relacionar a frequência dos termos no documento com a frequência dos termos no *corpus*.

Antes de apresentar uma medida que atenda à tal discussão, faz-se interessante analisar o quão significativo é um termo em um documento. Na Figura 2-9(a), representando conceitos discutidos por Hans Peter Luhn em 1958, é ilustrada a curva formada pela relação entre a frequência dos termos (*tf*) em um documento (eixo *y*) e a ordenação (descendente) dos termos, de acordo com suas respectivas frequências (eixo *x*). A curva exponencial formada por tal relação corresponde à Lei de Zipf (ou *Zipf's Law*, como é mais conhecida) (Zipf, 1949), que diz que "a *n-ésima* maior frequência de ocorrência de uma palavra (eixo *y*) em uma língua é inversamente proporcional a sua posição na ordenação descendente (eixo *x*)".

Segundo Rijsbergen (Rijsbergen, 1979), os estudos de Luhn concluíram que é possível estabelecer limites de cortes na curva exponencial, de forma a separar termos que não contribuem significantemente para discriminar o conteúdo de um documento (termos muito frequentes ou comuns e termos pouco frequentes ou raros). Ainda, a partir destes estudos, Luhn apresentou uma curva capaz de representar o que chamou de "poder de resolução de palavras significativas", ou seja, a capacidade de discriminação de conteúdo apresentada pelos termos. Essa curva alcança seu ápice nos termos de frequência e posição médios; e tende a zero nos pontos dos limites de corte. Ainda, a análise combinada da curva exponencial e da curva referente ao poder de resolução dos termos (Figura 2-9(a)) mostra que existem termos com frequências maiores (ponto **A**), que possuem poder de resolução mais baixo do que termos com frequências menores (ponto **B**).

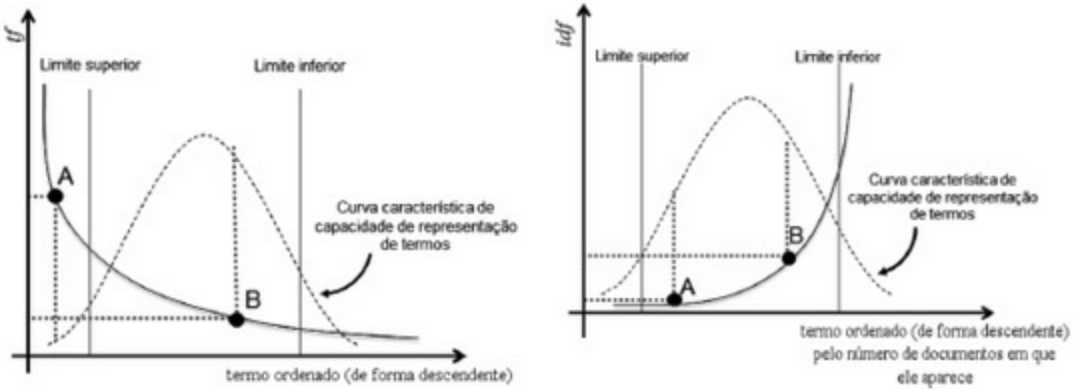
A fim de melhorar o relacionamento entre frequências e poder de resolução de termos, é interessante acrescentar a essa discussão o fato de que o documento em que um termo aparece pertence a um contexto – o *corpus*. A medida de *frequência inversa nos documentos* , ou simplesmente *idf* (do

inglês, *inverse document frequency*), pode fornecer essa relação da seguinte maneira:

$$idf\,(te) = \log\left(\frac{n}{nt}\right)$$

Equação 2-9

sendo *n* o número total de documentos no *corpus*, e *nt* o número de documentos em que o termo *te* aparece. Seguindo empiricamente um raciocínio similar ao da (Figura 2-9(a)), porém considerando a medida *idf*, obtém-se o gráfico da Figura 2-9(b). Observe que essa medida prioriza termos que aparecem em poucos documentos no *corpus*, indicando que existem termos com frequência invertida maior que possuem poder de resolução mais alto do que termos com frequência invertida menor (Figura 2-9(b)).



(a) Análise das frequências dos termos no documento (adaptado de Rijsbergen, 1979)

(b) Análise da frequência inversa nos documentos

Figura 2-9: Caracterização de frequências e representatividade de termos

Assim, diante dessas discussões, justifica-se o estabelecimento de uma medida que combine dois fatores importantes: a frequência do termo dentro de um documento, $tf(doc_i, te_j)$, e a frequência inversa dos termos nos documentos do *corpus* que o contém, $idf(te_j)$. Essa medida, conhecida como *tf-idf* , é capaz de estabelecer uma ponderação aos termos presentes em um

documento, considerando tanto o próprio documento quanto seu relacionamento com o *corpus*, e é dada por:

$$tf\text{-}idf\,(doc_i, te_j) = tf(doc_i, te_j) * idf\,(te_j) \qquad \textbf{Equação 2-10}$$

Se essa representação for considerada, o conjunto de dados Ψ será como ilustrado na Tabela 2-17:

Tabela 2-17: Conjunto de dados Ψ considerando uma representação por *tf-idf*

	ador	**agrad**	**ambient**	**atend**	**cidad**	**com**	**delíc**	**fil**	**frit**	**músic**	**parmegian**	**qualidad**	**tranq**
Ψ	wte_1	wte_2	wte_3	wte_4	wte_5	wte_6	wte_7	wte_8	wte_9	wte_{10}	wte_{11}	wte_{12}	wte_{13}
doc_1	1,58	0,58	0,58	0,00	0,00	1,58	0,00	0,00	0,00	1,58	0,00	0,58	1,58
doc_2	0,00	0,58	0,58	1,58	1,58	0,00	0,00	0,00	0,00	0,00	0,00	0,58	0,00
doc_3	0,00	0,00	0,00	0,00	0,00	0,00	1,58	0,00	1,58	0,00	0,00	0,00	0,00

Como discussão final sobre representações vetoriais para textos, é importante ressaltar que, embora a combinação do *tf-idf* crie boas representações para os textos, ainda é preciso considerar que os documentos em um *corpus* podem ter tamanhos diferenciados. Para corrigir distorções, é comum a realização de um procedimento de normalização (seguindo o princípio já apresentado na Seção 2.2.3), dando origem à medida *tf-idf normalizada* . Para isso, as seguintes equações podem ser aplicadas:

$$tf\text{-}idf_{norm}\,(doc_i, te_j) = \frac{tf\,(doc_i, te_j) * idf\,(te_j)}{\sqrt{\sum_{j=1}^{m} tf\,(doc_i, te_k)^2 * idf\,(te_k)^2}} \qquad \textbf{Equação 2-11}$$

ou alternativamente:

$$tf-idf_{norm}(doc_i, te_j) = \frac{tf(doc_i, te_j) * idf(te_j)}{\sum_{j=1}^{m} tf(doc_i, te_k)}.$$

Equação 2-12

Se essa representação for considerada, levando em conta a segunda alternativa de normalização, o conjunto de dados Y será como ilustrado na Tabela 2-18.

Tabela 2-18: Conjunto de dados Ψ considerando uma representação por *tf-idf normalizada*

	ador	**agrad**	**ambient**	**atend**	**cidad**	**com**	**delíc**	**fil**	**frit**	**músic**	**parmegian**	**qualidad**	**tranq**
Ψ	wte_1	wte_2	wte_3	wte_4	wte_5	wte_6	wte_7	wte_8	wte_9	wte_{10}	wte_{11}	wte_{12}	wte_{13}
doc_1	0,18	0,06	0,06	0,00	0,00	0,18	0,00	0,00	0,00	0,18	0,00	0,06	0,18
doc_2	0,00	0,08	0,08	0,23	0,23	0,00	0,00	0,00	0,00	0,00	0,00	0,08	0,00
doc_3	0,00	0,00	0,00	0,00	0,00	0,00	0,40	0,00	0,40	0,00	0,00	0,00	0,00

Os quatro tipos de representação vetorial permitem que os algoritmos apresentados neste livro, tanto para a resolução da tarefa de predição categórica quanto para a tarefa de agrupamento, possam ser aplicados.

2.6. Exemplo didático para preparação de dados não estruturados – do tipo texto

Apenas para fins ilustrativos, os resultados dos passos descritos na seção anterior são sequencialmente apresentados no Quadro 2-9, de forma anotada para indicar claramente as ações executadas em cada passo.

Quadro 2-9: Execução do processo de preparação do conjunto de dados Ψ

Documentos originais

doc1	`Ambiente agradável e tranquilo. Comida e música com qualidade. Adoramos o Filé à Parmegiana.`
doc2	`O Filé à Parmegiana da cidade. Ambiente agradável e qualidade no atendimento.`
doc3	`O Filé à Parmegiana com fritas é uma delícia.`

Análise lexical com eliminação de pontuação e padronização das letras

doc1	`(A)ambiente | agradável | e | tranquilo(.) | (C)comida | e | música | com | qualidade(.) | (A)adoramos | o | (F)filé | à | (P)parmegiana(.)`
doc2	`(O)o| (F)filé | à | (P)parmegiana | da | cidade(.) | (A)ambiente | agradável | e | qualidade | no | atendimento(.)`
doc3	`(O)o | (F)filé | à | (P)parmegiana | com | fritas | é | uma | delícia(.)`

Eliminação de termos irrelevantes

doc1	`ambiente | agradável | (e) | tranquilo | comida | (e) | música | (com) | qualidade | adoramos | (o) | filé | (à) | parmegiana`
doc2	`(o) | filé | (à) | parmegiana | (da) | cidade | ambiente | agradável | (e) | qualidade | (no) | atendimento`
doc3	`(o) | filé | (à) | parmegiana | (com) | fritas | (é) | (uma) | delícia`

Redução do termo ao seu radical

doc1	`ambient(e) | agrad(ável) | tranquil(o) | com(ida) | músic(a) | qualidad(e) | ador(amos) | fil(é) | parmegian(a)`
doc2	`fil(é) | parmegian(a) | cidad(e) | ambient(e) | agrad(ável) | qualidad(e) | atend(imento)`

doc3 `fil(é) | parmegian(a) | frit(as) | delíci(a)`

Criação do dicionário de termos

termo 1	ador	termo 8	fil
termo 2	agrad	termo 9	frit
termo 3	ambient	termo 10	músic
termo 4	atend	termo 11	parmegian
termo 5	cidad	termo 12	qualidad
termo 6	com	termo 13	tranquil
termo 7	delíc		

Criação de quatro representações para o *corpus* Ψ (binária, frequência de termos por documentos, *tf-idf* e *tf-idf* normalizada)

Ψ	wte_1	wte_2	wte_3	wte_4	wte_5	wte_6	wte_7	wte_8	wte_9	wte_{10}	wte_{11}	wte_{12}	wte_{13}
doc_1	1	1	1	0	0	1	0	1	0	1	1	1	1
doc_2	0	1	1	1	1	0	0	1	0	0	1	1	0
doc_3	0	0	0	0	0	0	1	1	1	0	1	0	0

Ψ	wte_1	wte_2	wte_3	wte_4	wte_5	wte_6	wte_7	wte_8	wte_9	wte_{10}	wte_{11}	wte_{12}	wte_{13}
doc_1	1	1	1	0	0	1	0	1	0	1	1	1	1
doc_2	0	1	1	1	1	0	0	1	0	0	1	1	0
doc_3	0	0	0	0	0	0	1	1	1	0	1	0	0

Ψ	wte_1	wte_2	wte_3	wte_4	wte_5	wte_6	wte_7	wte_8	wte_9	wte_{10}	wte_{11}	wte_{12}	wte_{13}
doc_1	1,58	0,58	0,58	0,00	0,00	1,58	0,00	0,00	0,00	1,58	0,00	0,58	1,58
doc_2	0,00	0,58	0,58	1,58	1,58	0,00	0,00	0,00	0,00	0,00	0,00	0,58	0,00
doc_3	0,00	0,00	0,00	0,00	0,00	0,00	1,58	0,00	1,58	0,00	0,00	0,00	0,00

Ψ	wte_1	wte_2	wte_3	wte_4	wte_5	wte_6	wte_7	wte_8	wte_9	wte_{10}	wte_{11}	wte_{12}	wte_{13}
doc_1	0,18	0,06	0,06	0,00	0,00	0,18	0,00	0,00	0,00	0,18	0,00	0,06	0,18
doc_2	0,00	0,08	0,08	0,23	0,23	0,00	0,00	0,00	0,00	0,00	0,00	0,08	0,00

doc_3	0,00	0,00	0,00	0,00	0,00	0,00	0,40	0,00	0,40	0,00	0,00	0,00	0,00

2.7. Exemplo prático para preparação de dados não estruturados – do tipo texto – em R

Para exemplificar os conceitos de pré-processamento de texto e preparação de um conjunto de dados Ψ, como discutido na Seção 2.5.2, usando R, considere um problema que envolve dados provenientes da rede social Twitter . O problema consiste em coletar dados referentes a postagens relacionadas com o evento *Restaurant Week na cidade de São Paulo*. Trata-se de uma semana de promoções organizada por um conjunto de restaurantes como estratégia para aumentar as vendas e divulgar suas marcas. É um evento que ocorre com sucesso em diversas capitais e também em algumas cidades do interior do país.

Um dos meios usados para divulgação do evento é o Twitter. A empresa organizadora tem um usuário nessa rede social chamado *@SaoPauloRW*, por meio do qual realiza a divulgação das promoções. O problema objetivado aqui pressupõe a aquisição de dados das postagens deste usuário para análise posterior.

Para manter o foco de estudo deste capítulo em funções relacionadas com o processamento de texto, explicações sobre como acessar a rede social e adquirir dados de *tweets* estão organizadas no Apêndice do livro. Isso significa que, para prosseguir com a execução dos comandos apresentados nesta seção, considera-se que os *tweets* já foram coletados e estão armazenados em uma variável da linguagem R.

A implementação das fases indicadas na Figura 2-8 é feita com o uso de funções disponibilizadas no pacote **tm** (Feinerer e Hornik, 2015; Feinerer *et al*., 2008) e, assim, é necessário que ele esteja disponível no ambiente R. As instruções para instalação dos pacotes e para carregamento dos dados de *tweets* (previamente coletados como especificado no Apêndice) são:

```
> install.packages(tm)
```

```
> library(tm)
> corpus <- Corpus(tweets)
```

A função `Corpus( )` coloca os textos dos *tweets* em uma estrutura de dados adequada para que outras funções do pacote **tm** sejam aplicadas. As fases do processo da Figura 2-8 são implementadas pela função **tm_map( )**. A sintaxe e os parâmetros para uso dessa função são mostrados no Quadro 2-10, e a implementação de cada uma das fases do processo é apresentada na sequência.

Quadro 2-10: Sintaxe e parâmetros da função *TM_MAP*

Sintaxe para aplicação da função que pré-processa os dados textuais
Usando a função `tm_map()` do pacote *tm*
Construindo o conjunto de dados Ψ `corpus_FI <- tm_map(corpus, operação)` • `corpus` é a variável em que os dados textuais estão armazenados. Eles podem estar no formato vetorial ou podem ser um *corpus* resultante de alguma execução precedente. • `operação` é a indicação de qual processamento deve ser executado sobre os textos – pode ser qualquer um dos processamentos envolvidos nas fases do processo da Figura 2-8. A função retorna o conjunto de dados Ψ, que representa o *corpus* no formato vetorial.

Para a análise lexical , considere as ações de capitalização, remoção de pontuação e remoção de números. Essas ações podem ser implementadas como segue (use a função `inspect( )` para visualizar o conteúdo produzido em cada um dos comandos):

```
> corpus_FI <- tm_map(corpus, content_transformer(tolower))
> corpus_FI <- tm_map(corpus_FI, removePunctuation)
> corpus_FI <- tm_map(corpus_FI, removeNumbers)
```

A remoção de termos irrelevantes (*stopwords*) é implementada a partir da escolha de uma lista de *stopwords*, dentre as várias disponibilizadas pelo pacote **tm**. No exemplo, a lista para o português foi escolhida:

```
> corpus_FI <- tm_map(corpus_FI, removeWords,
```

```
stopwords("portuguese"))
```

No caso de ser necessário adicionar novas palavras à lista de *stopwords*, a chamada da função `tm_map( )` deve incluir um vetor com as novas palavras. Um exemplo de como usar esse recurso é (uma variável auxiliar foi criada apenas para exemplificar esse comando):

```
> corpus_FI_2 <- tm_map(corpus_FI, removeWords,
c(stopwords("portuguese"),
c(qualidade, ambiente)))
```

O próximo passo do processo é a redução dos termos aos seus radicais da seguinte maneira (uma variável auxiliar foi criada para exemplificar esse passo, mas, no restante do exemplo, ela não será considerada para que a nuvem de *tags* usada na visualização dos dados seja criada com as palavras completas):

```
> corpus_FI_3 <- tm_map(corpus_FI, stemDocument)
```

Neste ponto, o *corpus* já está preparado para ser transformado em uma representação vetorial . Para construir uma representação baseada na frequência dos termos nos documentos, a função `TermDocumentMatrix()` deve ser usada. A sintaxe e parâmetros dessa função são apresentados no Quadro 2-11.

Quadro 2-11: Sintaxe e parâmetros da função TERMDOCUMENTMATRIX

Sintaxe para construção da representação vetorial, baseada na frequência dos termos, para um *corpus*
Usando a função `TermDocumentMatrix()` do pacote *tm*
Construindo a representação vetorial *tf* `corpus_tf <- TermDocumentMatrix(corpus, filtro)` • `corpus` é a variável na qual os dados textuais estão armazenados. • `filtro` permite selecionar termos com número mínimo de caracteres e com frequência mínima. A função retorna uma matriz de frequência de termos.

A função `TermDocumentMatrix( )` pode ser assim aplicada:

```
> corpus_tf <- TermDocumentMatrix(corpus_FI,
control = list(minWordLength = 1,
minDocFreq=1))
```

No pacote ***tm***, o uso da medida *tf-idf* , a partir da função `weightTfIdf( )`, deve ocorrer sobre o resultado gerado pela função `TermDocumentMatrix( )`. A sintaxe e parâmetros dessa função são apresentados no Quadro 2-12.

Quadro 2-12: Sintaxe e parâmetros da função WEIGHTTFIDF

Sintaxe para construção da representação vetorial, baseada na medida *tf-idf*, para um *corpus*
Usando a função `weightTfIdf()` do pacote *tm*
Construindo a representação vetorial *tf-idf* `corpus_tf_idf <- weightTfIdf (corpus_tf, tipo)` • `corpus_tf` é a matriz de frequências de termos do *corpus*. • `tipo` permite escolher entre gerar uma medida normalizada (TRUE) ou não (FALSE). A função retorna uma matriz com as medidas *tf-idf* dos termos.

O uso da função `weightTfIdf( )` pode ser implementado como segue:

```
> corpus_tf_idf <- weightTfIdf(corpus_tf, normalize=TRUE)
```

A partir dos dados representados no `corpus_tf` e no `corpus_tf_idf`, os algoritmos de mineração de dados podem ser aplicados para resolução de diferentes tipos de tarefas (classificação ou agrupamento, por exemplo). Também, procedimentos de análise mais simples podem ser aplicados, e visualizações interessantes sobre a informação contida nos dados textuais podem ser geradas.

Como exemplo de visualização, considere a geração de uma nuvem de palavras a partir da medida de frequência de termos do *corpus_tf*. Veja como gerar a nuvem de palavras no Apêndice deste livro. A nuvem resultante está

mostrada na Figura 2-10. Nessa figura, a frequência do termo é representada pelo tamanho da fonte de impressão das palavras e, portanto, como era de se esperar, o nome do usuário *saopaulorw* aparece com mais destaque, pois é o termo mais frequente no *corpus*. Os termos impressos nessa figura passaram por uma filtragem na qual apenas ocorrências de pelo menos 10 vezes foram selecionadas, e a impressão se deu a partir dos termos originais.

Figura 2-10: Nuvens de palavras gerada a partir da biblioteca `wordcloud()` para análise dos *tweets* do evento *Restaurante Week*

A partir do resultado da visualização pode-se inferir, por exemplo, que a promoção está surtindo mais efeito no jantar que no almoço – observe que a frequência da palavra “jantar” é maior. Também é possível verificar quais os pratos mais citados nos *tweets*, com destaque para os acompanhamentos “arroz”, “risoto”, “purê” e “batata”; para os pratos principais “cordeiro”, “frango”, “mignon”; e para os temperos “ervas” e “alho”.

2.8. Leituras adicionais

Os tipos de variáveis tratados neste capítulo seguiram uma taxonomia básica da estatística descritiva (quantitativas – discretas e contínuas, e qualitativas – nominais e ordinais). No entanto, para a área de análise de dados, outra taxonomia é muito usada: intervalares, binárias, categóricas, ordinais e em escalas de razão. Veja uma explicação bastante didática sobre essa taxonomia em Han, Kamber e Pei (2011). Ainda nessa mesma referência bibliográfica há uma discussão sobre construir matrizes de similaridades adequadas para lidar com conjuntos de dados nos quais há diferentes tipos de variáveis.

A criação de histogramas pressupõe o uso de faixas pelas quais é representada a distribuição de frequência de um conjunto de valores. No entanto, para os diferentes fins que um histograma ou uma distribuição de frequência podem ser usados (análise exploratória dos dados ou mesmo em estratégias de pré-processamento), nem sempre se tem disponível quantas faixas devem ser usadas e quais os limites dessas faixas. No entanto, é útil estabelecer as faixas de maneira sistematizada. Em Han, Kamber e Pei (2011), são apresentadas algumas estratégias para estabelecimento do número de faixas e de seus limites: largura-igual, frequência-igual, V-Optimal e MaxDiff.

Também relacionado com interpretação de histogramas está o Diagrama de Pareto . Esse diagrama tem como base o Princípio de Pareto – baseado na observação de que 80% das consequências estão relacionadas com 20% das causas. Há muitas aplicações para este princípio, em diferentes áreas do conhecimento. Uma literatura didática sobre a interpretação deste princípio é *O princípio 80/20*, de Richard Koch (Rio de Janeiro: Rocco, 2000).

Sobre tratamento de texto, para fins de análise automática, este capítulo considerou que um texto, ou um conjunto de textos, é um dado do tipo não estruturado. Contudo, a depender do tipo de análise considerada, a

organização de um texto pode ser vista sob um ângulo diferente – como um dado semiestruturado. De fato, do ponto de vista linguístico, elementos que compõem um texto (palavras, sequência de palavras, pontuação, uso de determinadas classes de palavras, parágrafos, tabelas etc.) representam de forma indireta algum conhecimento sobre o texto e seu conteúdo. Para uma discussão mais detalhada sobre esse e outros aspectos da tarefa de mineração de texto é indicado o estudo da obra de Feldman e Sanger (2007).

2.9. Exercícios

1. O conjunto de dados da Tabela 2-19, chamado *Fidelização*, trata de uma tentativa de obter dados das visitas dos clientes ao restaurante com o propósito de descobrir se existem fatores que contribuem para um cliente se fidelizar ao estabelecimento ou não. Os atributos descritivos de cada situação dizem respeito a diferentes tipos de avaliação realizadas em uma visita ao restaurante.

a) Realize uma análise exploratória dos dados de forma a verificar se conclusões sobre fidelização podem ser tiradas a partir dessa análise. Em sua análise, verifique as medidas de posição e separatrizes, medidas de dispersão, projete gráficos do tipo histograma e *boxplot* e realize análise de correlação.

b) Supondo que tarefas de mineração de dados precisam ser resolvidas sobre esse conjunto de dados, discorra sobre a necessidade de executar procedimentos de pré-processamento.

c) Prepare um programa em R que apresente as medidas e gráficos obtidos na análise exploratória e um novo conjunto de dados (pré-processado) mais adequado para ser minerado.

Tabela 2-19: Conjunto de dados *FIDELIZAÇÃO*

		x_{i1}	x_{i2}	x_{i3}	y
	RID	**AVALIAÇÃO 1**	**AVALIAÇÃO 2**	**AVALIAÇÃO 3**	**Situação**
$\mathbf{x}_{\rightarrow 1}$	R1	9,69	22,91	2,36	Fidelizado
$\mathbf{x}_{\rightarrow 2}$	R2	5,54	28,64	5,17	Fidelizado
$\mathbf{x}_{\rightarrow 3}$	R3	??	22,91	??	Fidelizado
$\mathbf{x}_{\rightarrow 4}$	R4	12,46	40,09	3,22	Fidelizado

$\mathbf{x}_{\rightarrow 5}$	R5	11,07	17,18	1,55	Fidelizado
$\mathbf{x}_{\rightarrow 6}$	R6	8,30	57,27	6,90	Fidelizado
$\mathbf{x}_{\rightarrow 7}$	R7	2,77	34,36	12,41	Não fidelizado
$\mathbf{x}_{\rightarrow 8}$	R8	5,54	17,18	6,01	Não fidelizado
$\mathbf{x}_{\rightarrow 9}$	R9	??	11,45	??	Não fidelizado
$\mathbf{x}_{\rightarrow 10}$	R10	8,30	??	??	Não fidelizado
$\mathbf{x}_{\rightarrow 11}$	R11	6,92	5,73	4,88	Não fidelizado
$\mathbf{x}_{\rightarrow 12}$	R12	13,84	11,45	0,83	Não fidelizado

2. Este problema consiste no reconhecimento de dígitos manuscritos (Kaynak, 1995), o qual é parte de um estudo maior, envolvendo imagens como dados de entrada.[14] Cada imagem de dígito tem um tamanho padronizado, com dimensão de 32 × 32 pixels, totalizando um conjunto com 43 ∗ 10 exemplares (distribuídos em conjuntos para treinamento e para teste). Considere ainda que a imagem é um mapa de bits, cada bit é um pixel da imagem valorado com zero (0) ou um (1), sendo que 1 significa que o pixel faz parte do traçado do dígito manuscrito. Adicionalmente, existe um segundo conjunto de dados, pré-processado, no qual as imagens originais foram representadas por imagens menores, de dimensão 8 × 8 pixels. Para conseguir a representação compacta, foi feita uma transformação sobre janelas de 4 × 4 pixels da imagem original. Cada uma das janelas de tamanho 4 foram transformadas em um único pixel cujo valor é a contagem dos pixels valorados com 1 na janela. Assim, para cobrir cada imagem de 32 × 32, foram necessárias 8 janelas de 4 × 4, o que gerou a representação na dimensão 8 × 8. Por fim, a matriz resultante é transformada em um vetor com 64 elementos ($\mathbb{R}^{64}$), em que cada elemento é um inteiro em [0, 16]. A Figura 2.11 ilustra o processo de transformação.

No conjunto de dados, o atributo de rótulo é adicionado à representação vetorial.

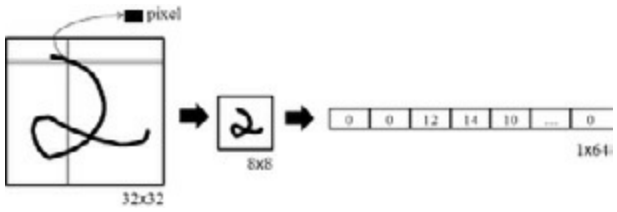


Figura 2-11: Transformação de uma imagem de 32 × 32 pixels para um vetor de 64 coordenadas

a) Para cada um dos dígitos, considerando o conjunto de dados pré-processado, realize (usando R) uma análise exploratória em cada um dos atributos – inclua gráficos *boxplot* em suas análises.

b) O pré-processamento com redução da quantidade de pixels é uma das possíveis estratégias para criar uma representação das imagens. Outra representação poderia ser feita com o uso de histogramas. Faça uma amostragem para cada um dos 10 dígitos e apresente o histograma de cada um deles, considerando o vetor de 64 elementos como entrada para o histograma. Discuta se essa representação poderia ser usada em um processo de mineração de dados realizado sobre esse conjunto de dados.

3. Para um projeto de mineração de dados, encomendado por uma instituição de ensino, considere que foi disponibilizado um conjunto de dados com os textos referentes às ementas das disciplinas. Os textos já passaram pelo processo de preparação, que envolve análise lexical, eliminação de termos irrelevantes e redução do termo ao seu radical. Os documentos textuais (ementa de cada disciplina) estão representados por simplificações: os termos estão representados pelas letras *A* e *B*, da seguinte forma:

Disciplina ١: *A A A B*

Disciplina ٢: *A A B*

Disciplina ٣: *A A*

Disciplina ٤: *B B*

A tarefa restante agora é a criação das representações vetoriais: binária, *tf*, *tf-idf* e *tf-idf* normalizado.

4. O *corpus* apresentado a seguir consiste em oito trechos de músicas populares nacionais. O objetivo aqui é realizar alguma tarefa de mineração de dados sobre essas músicas. Porém, o *corpus* não está representado como um conjunto de dados. Sua tarefa é prepará-lo de forma que um algoritmo de mineração de dados possa ser executado. Ao final do processo, quatro representações vetoriais são esperadas como conjuntos de dados: binária, *tf*, *tf-idf* e *tf-idf* normalizado.

L1: "...O teu amor pode estar...Do seu lado... "
L2: "...Eu quero amar você, e quando amanhecer eu quero acordar do seu lado."
L3: "Você abusou, tirou partido de mim, abusou"
L4: "Meu partido é um coração partido"
L5: "O nosso amor a gente inventa ... pra se distrair"
L6: "Grande pátria desimportante... Em nenhum instante ... Eu vou te trair"
L7: "Eu sou ... Eu sou a pátria que lhe esqueceu"
L8: "E o que dizem que foi tudo por causa de um coração partido"

CAPÍTULO 3

ANÁLISE PREDITIVA

A **análise preditiva** pode ser entendida como um processo que permite descobrir o relacionamento existente entre os exemplares de um conjunto de dados, descritos por uma série de características (atributos descritivos), e os rótulos a eles associados (atributo de classe). Nesse contexto, um exemplar no conjunto de dados é um evento no domínio de análise (um prato servido no restaurante ou uma instância de um anúncio publicitário, por exemplo) e o rótulo pode ser apresentado de duas formas: (i) como identificação da classe à qual o evento está associado, dentro de um número finito de classes existentes no domínio de análise (pratos adequados para consumir com vinho branco ou pratos adequados para consumir com vinho tinto, por exemplo); (ii) como um número ao qual o evento está associado, dentre um conjunto contínuo de valores possíveis para essa associação (número de visualizações de um anúncio, por exemplo). A primeira forma de apresentação dos rótulos define uma situação para a análise preditiva do tipo classificação (ou predição categórica), e a segunda, uma situação para a análise preditiva do tipo regressão (ou predição numérica).

Em geral, o relacionamento descoberto entre exemplares e rótulos constitui-se como um modelo de predição e pode ser apresentado na forma de funções ou organizado em estruturas de dados. O processo de construção do modelo de predição se dá por meio do ajuste de parâmetros realizado por um algoritmo, e, quando Aprendizado de Máquina é usado para isso, o processo é

indutivo e popularmente chamado de *treinamento* ou *aprendizado supervisionado* . Após sua determinação, o modelo preditivo pode ser usado para predizer o rótulo pertinente a exemplares antes desconhecidos, ou seja, que não fizeram parte do conjunto de dados usado para treinamento.

O processo de aplicação do modelo é popularmente conhecido como *teste* e consiste em apresentar um novo exemplar para o modelo, que lhe fornecerá um rótulo de acordo com o mapeamento previamente descoberto. Para os exemplos de uso supracitados, os modelos de predição podem ajudar um leigo a escolher o vinho que deve acompanhar determinado prato, mesmo que o prato seja novo no restaurante; ou a quantidade de visualizações de um novo anúncio publicitário pode ser estimada com base nas características que o descreve.

A análise preditiva é uma tarefa de mineração de dados aplicável em um grande número de domínios. Qualquer tentativa de nomeá-los, com certeza, resultará em um conjunto reduzido de possibilidades. Alguns exemplos de áreas nas quais a análise preditiva está presente são: análise de comportamento e expressão de emoções em redes sociais, realizada a partir do vocabulário usado nas manifestações de opiniões sobre produtos; na Biometria, com o reconhecimento de íris, impressão digital, face ou assinatura; na predição de subida ou queda de ações no mercado financeiro; na Biologia, mediante a classificação de novas espécies de organismos vivos; na Medicina, com a aplicação de modelos de predição categórica para auxiliar no diagnóstico de um tumor como sendo maligno ou benigno.

As estratégias algorítmicas aplicadas na resolução da tarefa de predição são baseadas na tomada de decisão sobre o uso de informação fornecida por atributos descritivos que possuem uma boa capacidade de discriminação dos dados em relação aos rótulos associados a eles. De modo geral, essas estratégias são implementadas de forma a minimizar erros de predição . Note que o processo de minimização do erro de predição é totalmente dependente da informação presente no conjunto de dados usado para treinamento. Assim,

conjuntos de dados de baixa qualidade não são capazes de fornecer informação de alta qualidade, sem a qual o modelo gerado será, muito provavelmente, também de baixa qualidade. Assim, o conjunto de dados usado na geração de um modelo de predição deve ser uma amostra de qualidade do conjunto universo do domínio de aplicação, caso contrário, o modelo gerado não será capaz de representar o relacionamento real existente entre a descrição dos eventos e os rótulos que devem ser associados a eles.

A fim de compreender melhor as questões envolvidas no processo de construção de um modelo preditivo, é necessário analisar o problema sob o ponto de vista da minimização do erro. A Figura 3-1 apresenta um esquema gráfico sobre diferentes **tipos de erros** e como modelos de predição e erros estão situados no **espaço de busca de modelos**.

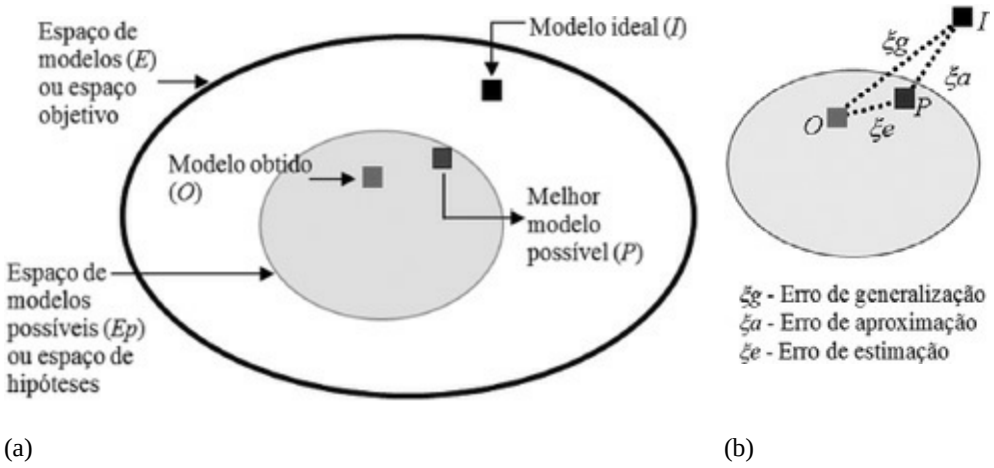


Figura 3-1: Espaço de modelos (a); erros de predição (b); baseado em Lima (2004)

Na Figura 3-1, *E* é todo o conjunto possível de diferentes modelos que podem ser estabelecidos no domínio de aplicação em estudo; *I* é um modelo ideal, ou seja, aquele que resolve o problema de predição considerando todo o universo de eventos que ocorrem no domínio de aplicação; *Ep* é o conjunto de todos os diferentes modelos que um algoritmo de geração de um preditor é capaz de alcançar, considerando seu conjunto de parâmetros e os exemplares

existentes no conjunto de dados usado no processo de treinamento (indução) do modelo; *P* é o melhor modelo que o algoritmo é capaz de alcançar; *O* é o modelo alcançado pelo algoritmo em uma execução de um processo de indução. Observe que, no exemplo aqui discutido, o algoritmo nunca conseguirá o modelo ideal por não ter condições para isso, uma vez que o modelo ideal está fora do seu escopo de busca. Os espaços de modelos e os modelos neles existentes se relacionam em termos de erros. Na Figura 3-1(b), tais relações são mostradas sob uma metáfora de distância. O erro de aproximação (ξa) é um problema que não pode ser resolvido para esta situação, ou seja, ele sempre existirá quando o modelo ideal estiver fora do escopo de busca do algoritmo; o erro de estimação (ξe) é um erro decorrente do processo de indução, ou seja, a escolha de determinado conjunto de parâmetros e de um conjunto de exemplares levou o algoritmo a encontrar um modelo aquém do que poderia conseguir; e o erro de generalização (ξg) é o erro que o modelo de predição comete em relação ao modelo ideal, e pode ser visto como uma composição dos outros dois erros.

Durante o processo de indução de um modelo preditivo e também ao seu término, é comum que seja feita uma estimativa do erro de generalização. Essa estimativa é realizada a partir do uso de um conjunto de exemplares de teste para o modelo gerado. Diz-se que se trata de uma estimativa porque é calculado a partir de uma amostra do conjunto de dados disponível, que não se constitui como o conjunto universo de exemplares possíveis no domínio sob estudo. O erro de generalização estimado é usado como forma de avaliação do modelo obtido.

Neste capítulo, são discutidos aspectos teóricos e práticos, necessários ao entendimento de como pode se dar a resolução da tarefa de análise preditiva. Para ilustrar formas de resolver a tarefa de predição categórica , são apresentados os algoritmos: *k*-vizinhos mais próximos (do inglês *k-nearest neighbors* – *k*-NN), *Perceptron* e *Multilayer Perceptron*, da classe de algoritmos conhecidos como redes neurais artificiais (do inglês *artificial*

neural networks), árvores de decisão (do inglês *decision trees*) e *Naïve*-Bayes. A predição numérica é representada aqui por técnicas de regressão linear, ilustrada por meio da estratégia de minimização dos erros quadrados, e regressão não linear a partir de uma generalização da mesma estratégia apresentada para o caso linear. Exemplos didáticos e práticos usando a linguagem R são fornecidos para cada um dos algoritmos. Ao final do capítulo, são sugeridos exercícios para fixação dos conteúdos e também leituras adicionais nas quais informações mais específicas sobre análise preditiva podem ser obtidas.

3.1. A tarefa de classificação

Segundo Rocha *et al*. (2012), “o termo *classe* deve ser usado quando existe informação sobre quantas e quais são as partições presentes em um conjunto de dados, bem como qual exemplar pertence à qual partição.” Assim, comumente denomina-se ***classificação*** o processo pelo qual se determina um mapeamento capaz de indicar a qual classe pertence qualquer exemplar de um domínio sob análise, com base em um conjunto de dados já classificado. Refinando a definição de conjunto de dados, estudada no Capítulo 1, diz-se que, formalmente, a tarefa de classificação pode ser descrita como a busca por uma função capaz de mapear um conjunto **X** de vetores de entrada (ou exemplares) $\boldsymbol{x}_{\rightarrow i} \in \mathrm{E}^d$ para um conjunto finito de rótulos *C* de cardinalidade *c*. A função F é então definida como $\mathrm{F} : \mathrm{E}^d \times W \rightarrow C$, em que *d* é a dimensão do espaço E, ou seja, o número de coordenadas do vetor $\boldsymbol{x}_{\rightarrow i}$, e *W* é um espaço de parâmetros ajustáveis por meio do algoritmo de indução supervisionada.

A tarefa de classificação pode ser dividida em, pelo menos, duas categorias: classificação binária e classificação multiclasse . Na classificação binária, a cardinalidade *c* é 2 e para os casos em que $c > 2$, o problema é considerado de múltiplas classes. Num contexto de um restaurante, por exemplo, um classificador binário poderia ser usado para ajudar um cliente a decidir por pagar ou não a taxa de 10% de serviço; já um classificador multiclasse poderia auxiliar o cliente sobre como classificar o serviço em um conjunto de opções {*excelente, bom, regular, ruim*}.

Particularidades em relação à construção de classificadores binários, composição de classificadores binários para resolução do problema de classificação multiclasse e avaliação de classificadores binários devem ser estudadas com atenção. Esse assunto é retomado nas Seções 3.3 e 3.4.

3.1.1. *k*-vizinhos mais próximos

O **algoritmo *k*-vizinhos mais próximos** ou *k*-NN implementa o aprendizado baseado em instâncias (*instance-based learning*). Isso significa que a classificação de um exemplar cuja classe é desconhecida é realizada a partir da comparação desse exemplar com aqueles que possuem uma classe já conhecida. O princípio usado nesse algoritmo é o de estocar o conjunto de treinamento e realizar comparações entre o exemplar de teste e os exemplares estocados a cada vez que um exemplar de teste é apresentado. Esse estilo de processamento pode dispender muito tempo dependendo da quantidade de exemplares do conjunto de dados de treinamento. Ainda, trata-se de um estilo de processamento conhecido como "avaliação preguiçosa" (*lazy evaluation*), já que não há um trabalho prévio de indução de um modelo.

Embora o processo de classificação possa ser custoso em algumas situações, a lógica que o implementa é simples. Um exemplar é classificado pela "votação da maioria", realizada junto aos seus vizinhos no conjunto de treinamento armazenado. A classe da maioria dos *k* exemplares mais próximos ao exemplar de teste é aquela que deve ser atribuída a ele. A relação de proximidade para identificação dos vizinhos é quantificada por uma métrica de distância. Diferentes métricas podem gerar diferentes resultados, e a métrica a ser usada deve ser adequada à natureza (tipo) dos dados sob análise. Portanto, antes de estudar o algoritmo *k-NN*, é útil estudar um pouco sobre métricas de distância.

Métricas de distância normalmente são usadas para medir similaridade ou dissimilaridade entre dois exemplares. Nesse contexto, os atributos descritivos dos exemplares são vistos como dimensões de um espaço métrico. Cada exemplar $\boldsymbol{x}_{\rightarrow i}$ do conjunto de dados é visto como um ponto distribuído nesse espaço de atributos, ou seja, cada exemplar $\boldsymbol{x}_{\rightarrow i} = \{\boldsymbol{x}_{i1}, \boldsymbol{x}_{i2}, ..., \boldsymbol{x}_{id}\}$, com $i = \{1, ..., n\}$ será representado no espaço conforme abstração[1] ilustrada na Figura 3-2. Assim, é possível calcular a distância entre os exemplares.

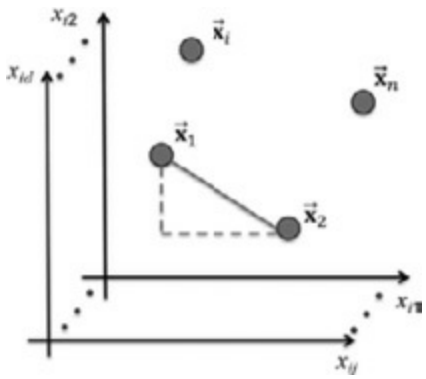


Figura 3-2: Gráfico de dispersão como abstração dos exemplares distribuídos no espaço dos atributos

Há, na literatura especializada, várias formas de medir distâncias entre dois exemplares com atributos descritivos do tipo numérico: Manhattan, euclidiana, Minkowski, Cosseno, Mahalanobis entre outras (Deza e Deza, 2009). Para atributo binário, é comum usar a distância de Hamming. As distâncias Manhattan e euclidiana são apresentadas aqui em mais detalhes.

A distância de Manhattan também é conhecida por *city-block*, *taxicab* e rectilinear. Ela quantifica distância como uma contagem de quadras percorridas durante um percurso em uma cidade. Foi assim chamada porque as ruas que cortam a ilha de Manhattan formam quadrados geometricamente perfeitos. Em uma cidade como essa, um percurso pode ser feito por diferentes caminhos, todos com a mesma distância. Tome como ilustração os exemplares $\boldsymbol{x}_{\rightarrow 1}$ e $\boldsymbol{x}_{\rightarrow 2}$ da Figura 3-2 e considere apenas dois atributos descritivos. A distância de Manhattan entre os dois pontos é a soma das diferenças absolutas de suas coordenadas (atributos). Pictoricamente, a distância está representada pelas linhas pontilhadas na referida figura. Formalmente, a distância de Manhattan é definida como: $dist\ x_{\rightarrow 1}, x_{\rightarrow 2} = |x_{11} - x_{21}| + |x_{12} - x_{22}|$.

O cálculo da distância de Manhattan permite dizer que: a distância $dist\ x_{\rightarrow 1}, x_{\rightarrow 2}$ é a mesma de $dist\ x_{\rightarrow 2}, x_{\rightarrow 1}$, aten- dendo à condição de simetria; $dist\ x_{\rightarrow 1}, x_{\rightarrow 2} \geq 0$, atendendo à condição de ser não negativa; $dist\ x_{\rightarrow 1}, x_{\rightarrow 1} = 0$, atendendo à condição de reflexividade, ou seja, é nula para pontos coincidentes. Essas condições devem ser satisfeitas para que a medida possa

ser considerada capaz de generalizar o conceito geométrico de distância (Deza e Deza, 2009). Para um cálculo genérico entre dois exemplares $\boldsymbol{x}\rightarrow_{\mathrm{p}}$ e $\boldsymbol{x}\rightarrow_{\mathrm{q}}$ quaisquer de um conjunto de dados com múltiplos atributos (*d*), tem-se, para a distância Manhattan:

$$dist_{\boldsymbol{x}_p,\boldsymbol{x}_q}=\sum_{j=1}^{d}\left|x_{pj}-x_{qj}\right| \qquad \textbf{Equação 3-1}$$

A distância euclidiana é a métrica mais comumente utilizada, pois possui a propriedade de representar a distância física entre pontos em um espaço *d*-dimensional. Na Figura 3-2, a representação da distância euclidiana se manifesta pela linha contínua entre os exemplares $\boldsymbol{x}\rightarrow_1$ e $\boldsymbol{x}\rightarrow_2$. Repare a figura geométrica que a linha contínua forma com as linhas tracejadas na referida figura: trata-se de um triângulo retângulo; a distância euclidiana é a hipotenusa desse triângulo; ou seja, vale o teorema de Pitágoras. O teorema diz que a soma dos quadrados dos catetos é igual ao quadrado da hipotenusa. Portanto, a distância euclidiana para os dois exemplos, $\boldsymbol{x}\rightarrow_1$ e $\boldsymbol{x}\rightarrow_2$, é: $dist_{\boldsymbol{x}_1,\boldsymbol{x}_2}=\sqrt{(x_{11}-x_{21})^2+(x_{12}-x_{22})^2}$. De maneira genérica, a distância euclidiana para dois exemplares $\boldsymbol{x}\rightarrow_{\mathrm{p}}$ e $\boldsymbol{x}\rightarrow_{\mathrm{q}}$ quaisquer é:

$$dist_{\boldsymbol{x}_p,\boldsymbol{x}_q}=\sqrt{\sum_{j=1}^{d}(x_{pj}-x_{qj})^2} \qquad \textbf{Equação 3-2}$$

O algoritmo que implementa a estratégia do *k*-NN é apresentado no Algoritmo 3-1. Antes de iniciar a execução do algoritmo, é necessário definir dois parâmetros: o valor de *k*, que define o número de vizinhos mais próximos consultados no processo de classificação; a medida de distância *dist*. Além da definição dos dois parâmetros, o algoritmo exige a disponibilização de um conjunto de treinamento $\mathbf{X}_{tr}$ e um conjunto de teste $\mathbf{X}_{te}$. No Algoritmo 3-1, o conjunto de teste possui apenas um exemplar,

porém, vários exemplares de teste podem compor esse conjunto, e a execução do algoritmo deve ser repetida para cada um desses exemplares.

Algoritmo 3-1: Algoritmo do *k*-NN , considerando um conjunto de teste com apenas um exemplar

Parâmetros de entrada:
- $\mathbf{X}_{tr}$: é um conjunto de exemplares rotulados de treinamento, ou seja, **X**tr = $\{(\boldsymbol{x}\rightarrow_i, y_i)\}$, $i = 1, ..., n$;
- $\mathbf{X}_{te}$: é um exemplar de teste com rótulo desconhecido, **X**te = $\{\boldsymbol{x}\rightarrow_{te}\}$;
- *k*: número de vizinhos para ser usado na comparação;
- *dist*: medida de distância;

Parâmetro de saída:
- $\hat{y}_{te}$: rótulo estimado para o exemplar de teste $\boldsymbol{x}\rightarrow_{te}$;

Passo 1: **para cada** exemplar $\boldsymbol{x}\rightarrow_i$ de **X**tr **faça**
Passo 1.1: calcule a distância *dist* entre $\boldsymbol{x}\rightarrow_i$ e $\boldsymbol{x}\rightarrow_{te}$;
Passo 2: ordene os exemplares de **X**tr, de forma ascendente, de acordo com as distâncias calculadas;
Passo 3: selecione, a partir do conjunto **X**tr ordenado, os *k* primeiros exemplares;
Passo 4: atribua, como rótulo estimado $\hat{y}_{te}$ de $\boldsymbol{x}\rightarrow_{te}$, o rótulo majoritário dos *k* primeiros exemplares selecionados no *Passo 3;*

De forma ilustrativa, na Figura 3-3, é mostrado o processo de classificação executado pelo *k*-NN. O conjunto de treinamento possui dados descritos por dois atributos, $\boldsymbol{x}\rightarrow_i = \{\boldsymbol{x}_{i1}, \boldsymbol{x}_{i2}\}$, cada exemplar *i* é um ponto no gráfico, e o atributo de classe (ou rótulo) está representado pelas formas geométricas círculo (C), quadrado (Q) e triângulo (T). A apresentação do exemplar de teste, de classe desconhecida (?), dispara a execução do laço do *Passo 1* do Algoritmo 3-1, as distâncias são calculadas, e o resultado ordenado dos *k* exemplares selecionados no *Passo 3* é representado na figura pelos círculos pontilhados.

Nesse exemplo, é possível analisar duas situações diferentes a partir de dois valores para *k*. Para $k = 1$ (e, nesse caso, também para $k = 2$), os *k*-exemplares do conjunto de treinamento mais próximos do exemplar de teste são da classe C, logo essa classe deve ser atribuída a $\hat{y}_{te}$. Para $k = 3$, o conjunto dos *k*-exemplares mais próximos de $\boldsymbol{x}\rightarrow_{te}$ inclui dois exemplares da

classe C e um exemplar da classe Q. Seguindo a estratégia de escolher a classe de maior frequência nesse conjunto, a classe C deve ser atribuída a $\hat{y}_{te}$.

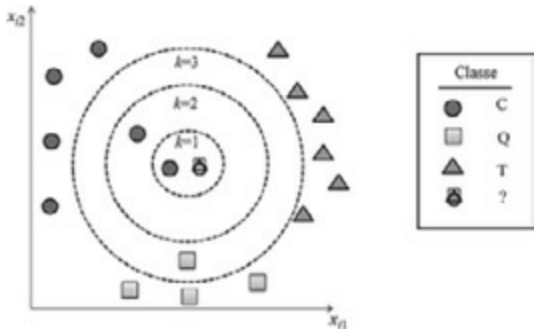


Figura 3-3: Exemplo ilustrativo para o *k-NN*

A escolha do parâmetro *k* deve ser feita com cuidado e, não raramente, exige um trabalho de inspeção dirigido ao problema em análise. A escolha de um *k* pequeno pode levar o algoritmo a uma alta sensibilidade a ruído; a escolha de um *k* grande pode levar a um aumento na diversidade de classes incluídas no conjunto dos *k* exemplares selecionados para a contagem da classe mais frequente.

3.1.1.1. Exemplo didático para o *k*-NN

Como exemplo didático para contextualizar a execução do algoritmo *k*-NN , considere o conjunto de dados *Pesquisa de Satisfação*, apresentado na Tabela 3-1, já com valores numéricos normalizados no intervalo [0,1] (veja, no Capítulo 2, como um procedimento de normalização é realizado). O conteúdo dessa tabela será usado como o conjunto de dados disponível para geração de um modelo de classificação com a execução do *k*-NN.

Para fins de análise, o conjunto de dados é apresentado também em um formato gráfico (Figura 3-4). Essa representação foi possível porque os exemplares, nesse conjunto, são descritos por dois atributos (`TEMPO` e `PREÇO`), e, portanto, a dimensão do espaço de atributos é igual a 2. A razão de construir esse gráfico é fornecer uma noção sobre a distribuição dos

exemplares, permitindo observar o quão similares (próximos) ou dissimilares (distantes) eles são.

Tabela 3-1: Conjunto de dados *PESQUISA DE SATISFAÇÃO*

		x_{i1}	x_{i2}	y_i
	RID	**TEMPO**	**PREÇO**	**CLASSE**
$\mathbf{x}\rightarrow_1$	R_1	0,70	0,40	SAT
$\mathbf{x}\rightarrow_2$	R_2	0,40	0,50	SAT
$\mathbf{x}\rightarrow_3$	R_3	0,61	0,40	SAT
$\mathbf{x}\rightarrow_4$	R_4	0,90	0,70	SAT
$\mathbf{x}\rightarrow_5$	R_5	0,80	0,30	SAT
$\mathbf{x}\rightarrow_6$	R_6	0,60	1,00	SAT
$\mathbf{x}\rightarrow_7$	R_7	0,20	0,60	INS
$\mathbf{x}\rightarrow_8$	R_8	0,40	0,30	INS
$\mathbf{x}\rightarrow_9$	R_9	0,61	0,20	INS
$\mathbf{x}\rightarrow_{10}$	R_{10}	0,60	0,43	INS
$\mathbf{x}\rightarrow_{11}$	R_{11}	0,50	0,10	INS
$\mathbf{x}\rightarrow_{12}$	R_{12}	1,00	0,20	INS

R_{ID}: identificação de uma refeição
TEMPO: tempo médio de preparo da refeição no último mês
PREÇO: preço das refeições servidas no último mês
SAT: satisfeito
INS: insatisfeito

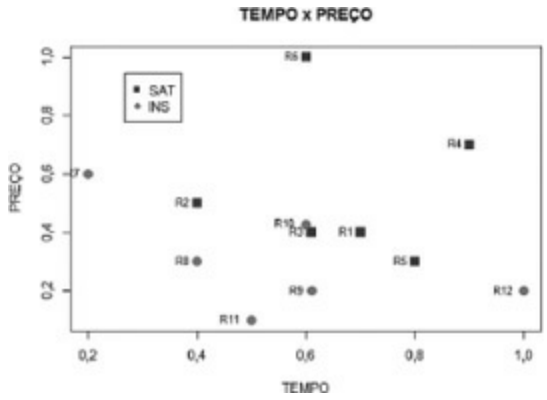


Figura 3-4: Distribuição dos exemplares do conjunto de dados *PESQUISA DE SATISFAÇÃO*

Como não há um exemplar de classe desconhecida, a fim de ilustrar o exemplar de teste do algoritmo *k*-NN, suponha que o primeiro exemplar ($\boldsymbol{x}\rightarrow_1$ ou R_1) seja de teste (ou seja, imagine que o rótulo seja desconhecido) e que deva ser classificado seguindo as estratégias do algoritmo. Além do exemplar desconhecido, considere a distância euclidiana e o número de vizinhos mais próximos igual a 3 ($k = 3$).

A aplicação da distância euclidiana entre os exemplares $\boldsymbol{x}\rightarrow_1$ e $\boldsymbol{x}\rightarrow_2$, como um exemplo da primeira iteração do *Passo* 1 do Algoritmo 3-1, resulta em:

$$dist_{\boldsymbol{x}_1,\boldsymbol{x}_2} = \sqrt{(x_{11}-x_{21})^2+(x_{12}-x_{22})^2} = \sqrt{(0{,}70-0{,}40)^2+(0{,}40-0{,}50)^2} = 0{,}32$$ **Equação 3-3**

e cálculos similares devem ser executados comparando $\boldsymbol{x}\rightarrow_1$ com todos os demais exemplares do conjunto de dados. A Tabela 3-2 traz os resultados obtidos após o cálculo de todas as distâncias necessárias; a Tabela 3-3 traz os mesmos resultados, porém ordenados. Em destaque nessas figuras, estão as três menores distâncias obtidas.

Tabela 3-2: Distâncias de $\boldsymbol{x}\rightarrow_1$ para todos os demais exemplares do conjunto de dados *PESQUISA DE SATISFAÇÃO*

dist	$\boldsymbol{x}\rightarrow_2$	$\boldsymbol{x}\rightarrow_3$	$\boldsymbol{x}\rightarrow_4$	$\boldsymbol{x}\rightarrow_5$	$\boldsymbol{x}\rightarrow_6$	$\boldsymbol{x}\rightarrow_7$	$\boldsymbol{x}\rightarrow_8$	$\boldsymbol{x}\rightarrow_9$	$\boldsymbol{x}\rightarrow_{10}$	$\boldsymbol{x}\rightarrow_{11}$	$\boldsymbol{x}\rightarrow_{12}$
$\boldsymbol{x}\rightarrow_1$	0,32	**0,09**	0,36	**0,14**	0,61	0,54	0,32	0,22	**0,10**	0,36	0,36

Tabela 3-3: Distâncias de $\boldsymbol{x}_{\rightarrow 1}$ para todos demais exemplares do conjunto de dados *PESQUISA DE SATISFAÇÃO*, porém ordenadas de forma ascendente

dist	$\mathbf{x}_{\rightarrow 3}$	$\mathbf{x}_{\rightarrow 10}$	$\mathbf{x}_{\rightarrow 5}$	$\mathbf{x}_{\rightarrow 9}$	$\mathbf{x}_{\rightarrow 2}$	$\mathbf{x}_{\rightarrow 8}$	$\mathbf{x}_{\rightarrow 12}$	$\mathbf{x}_{\rightarrow 4}$	$\mathbf{x}_{\rightarrow 11}$	$\mathbf{x}_{\rightarrow 7}$	$\mathbf{x}_{\rightarrow 6}$
$\mathbf{x}_{\rightarrow 1}$	**0,09**	**0,10**	**0,14**	0,22	0,32	0,32	0,36	0,36	0,36	0,54	0,61

Por fim, faz-se a classificação do exemplar de entrada considerando a classe mais frequente nos *k*-vizinhos mais próximos. Dado que a distribuição de classes no conjunto dos exemplares *k*-vizinhos mais próximos é {`SAT`, `INS`, `SAT`}, a classe de $\boldsymbol{x}_{\rightarrow 1}$ (ou R_1) é `SAT`.

3.1.1.2. Exemplo prático para o *k*-NN em R

Para ilustrar a implementação do algoritmo *k-NN* em R, será usada a função `knn( )`, disponível no pacote ***class*** (Venables e Ripley, 2002). Essa é uma das funções mais simples disponíveis para implementar esse algoritmo. Funções mais complexas, que implementam diferentes formas de comparação de exemplares, estão disponíveis no ambiente R.[2] Um resumo dos parâmetros que a função `knn( )` recebe para seu uso são apresentados no Quadro 3-1.

Quadro 3-1: Sintaxe e parâmetros da função *k-NN*

Sintaxe para aplicação da função que implementa o algoritmo *k-NN*
Usando a função *knn*( ) do pacote *class*

Fazendo a predição

```
y_estimado <- knn (conjunto_treinamento, exemplar_teste, rótulos, k)
```

- `dados_treinamento` é um ***data.frame*** que contém dados numéricos com atributos descritivos usados como conjunto de treinamento.
- `exemplar_teste` é um ***data.frame*** que contém dados numéricos com atributos descritivos para realização do teste.
- `rótulos` é um vetor com as classes para cada exemplar do conjunto de dados para treinamento.
- `k` é uma variável do tipo inteiro que indica o número de vizinhos mais próximos a serem consultados.
- `y_estimado` é a variável que recebe a saída da função – a classe do `exemplar_teste`.

A função retorna a variável `y_estimado` que contém a classe do exemplares em `exemplar_teste`.

Vale chamar a atenção para o fato de que a função `knn( )` exige que o conjunto de atributos descritivos esteja armazenado em uma variável distinta daquela na qual está armazenado o conjunto de rótulos correspondente a cada exemplar do conjunto de treinamento. Sendo assim, para o caso desse exemplo, os comandos de leitura da base de dados precisam assumir que, para a variável `treinamento`, apenas os atributos `TEMPO` e `PREÇO` devem ser considerados; e para a variável `rótulos`, apenas o atributo `CLASSE` deve ser considerado. Também, para manter a coerência com o exemplo discutido na Seção 3.1.1.1, o primeiro exemplar do conjunto de dados deve ser considerado para o conjunto de teste, e seus atributos descritivos devem ser armazenados na variável `exemplar_teste`. A sequência de instruções que segue realiza as leituras necessárias para instanciar cada uma das variáveis requeridas na aplicação da função `knn( )`.

Por exemplo, considerando que o conjunto de dados foi carregado na variável `pesquisa`, e que a mesma tem, portanto, número de linhas e colunas iguais a $n = 12$ e $d = 4$, respectivamente, o conjunto de treinamento deve ser formado pelas linhas de 2 a 12 (pois o primeiro exemplar será deixado para o teste) e colunas de 2 a 3 (pois a primeira coluna é o atributo identificador, e a última é o atributo de classe). Portanto, o conjunto de treinamento será separado como:

```
> pesquisa <-
read.table("C:\\Users\\LivroDM\\Predicao\\pesquisa_satisfacao.csv"
header=TRUE,sep=";")
> treinamento <- pesquisa[2:12,2:3]
> rotulos <- pesquisa[2:12,4:4]
```

O exemplar de teste, por sua vez, será formado pelo primeiro exemplar do conjunto de dados sem a apresentação do atributo classificatório. Ou seja:

```
> teste <- pesquisa[1:1,2:3]
```

Enfim, o uso da função `knn( )` deve ser implementado da seguinte forma:

```
> y_estimado <- knn(treinamento, teste, rotulos, 3)
```

A saída gerada contém a informação sobre a classe atribuída ao exemplar de teste e também uma lista (na variável `Levels`) das possíveis classes que o exemplar poderia assumir. Dado que a execução, para esse exemplo, considera o conjunto de dados *Pesquisa de Satisfação* (Tabela 3-1) e que o conjunto de dados para teste é composto de apenas um elemento, tem-se como saída:

```
> y_estimado
[1] SAT
Levels: INS SAT
```

3.1.2. Rede Neural Artificial – *Perceptron* e *Multilayer Perceptron*

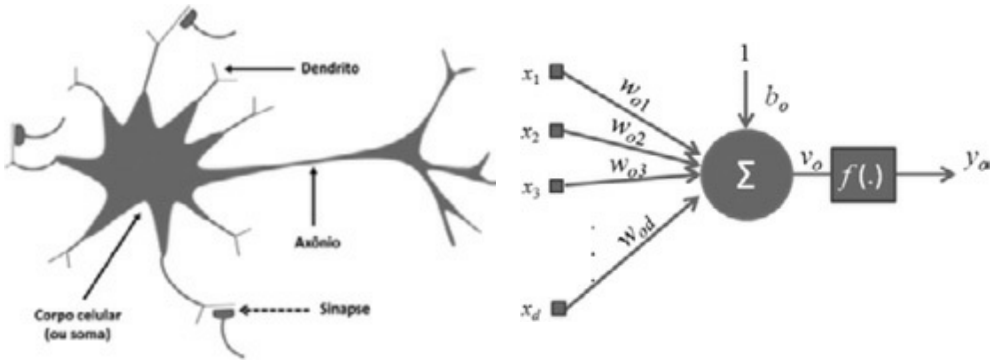
O interesse em desvendar os mistérios do funcionamento do cérebro é bastante antigo. E, mesmo com os mais recentes avanços nessa área, há questões para as quais explicações precisas ainda não existem. Por exemplo, o caso de *Phineas P. Gage*, o homem que teve seu cérebro atravessado por uma barra de ferro em uma obra civil (Damásio, 2012) e sobreviveu sem nenhuma perda de memória ou motora, mas teve sua personalidade alterada; ou ainda, o caso de pessoas que sofrem acidentes vasculares cerebrais que causam avarias em parte do cérebro e que, com o passar do tempo, voltam à sua vida normal.

Mas muito já se sabe sobre o cérebro e seu funcionamento. É de conhecimento pleno atualmente que o elemento principal do cérebro é o neurônio e que a capacidade de formação de redes de neurônios que trocam sinais via sinapses é o fator responsável pela mais importante característica dos humanos: o aprendizado . O aprendizado confere aos humanos a capacidade de lidar com informações imprecisas e de formar generalizações

para essas informações. Por exemplo, uma pessoa que aprende a dirigir no Brasil e se habilita sob as regras de trânsito brasileiras é capaz de dirigir na Inglaterra, mesmo lá sendo necessário inverter “a mão”. Isso se deve à capacidade do ser humano de adaptar seu conhecimento, generalizando-o de acordo com as variações impostas pelo ambiente. A simulação do funcionamento dessas unidades de processamento – chamadas de neurônios – por meio de um computador permite, inclusive, contribuir para o entendimento dos mistérios do cérebro, embora, para os propósitos de mineração de dados, a vantagem de usar tal simulação é poder construir modelos capazes de resolver tarefas de classificação ou de agrupamento.

A simulação de um neurônio se dá por meio de um modelo formal baseado na fisiologia básica de um neurônio biológico, como mostrado na Figura 3-5(a). De acordo com essa figura, um neurônio é dividido em três partes: o corpo celular ou soma, os dendritos e o axônio. A aproximação entre o axônio de um neurônio e o dendrito de outro desencadeia um processo de reações, elétricas ou químicas, chamadas de sinapses. A sinapse então gera um impulso nervoso que define o fluxo de informação entre um axônio e vários dendritos. No corpo celular, os impulsos vindos dos dendritos são combinados e formam o sinal de entrada do neurônio. A depender da força desse sinal, o neurônio é capaz de impulsioná-lo pelo axônio, dando continuidade ao fluxo de informação. Ainda, a sinapse pode ser excitatória ou inibitória, o que indica haver uma alteração nos sinais transmitidos por axônios e recebidos pelos dendritos.

O **neurônio artificial** é, na realidade, uma função que mapeia entradas e saídas. O primeiro modelo formal de neurônio foi proposto em 1943, como resultado de pesquisa interdisciplinar realizada por um psiquiatra e neuroanatomista, chamado Warren Sturgis McCulloch, e um matemático, chamado Walter Harry Pitts Jr.. Esse modelo era bastante simplificado e seguia uma série de regras específicas que limitavam seu uso para aplicações reais (para mais informações, veja Haykin, 2008; Fausett, 1993).



(a) (b)

Figura 3-5: (a) Fisiologia básica de um neurônio biológico; (b) Estrutura de um neurônio artificial do tipo *Perceptron*

Um modelo de neurônio mais interessante é o ***Perceptron*** (Figura 3-5(b)), proposto por Frank Rosemblatt, em meados de 1955. É possível fazer um paralelo entre o funcionamento do neurônio artificial *Perceptron* e um neurônio biológico: os dendritos, ou cada atributo descritivo $\{\boldsymbol{x}_{i1}, \boldsymbol{x}_{i2}, ..., \boldsymbol{x}_{ij}, ..., \boldsymbol{x}_{id}\}$ de um exemplar de entrada $\boldsymbol{x}_{\rightarrow i}$, estão associados a um neurônio *o* por meio dos pesos sinápticos ($\mathbf{w}_{\rightarrow O}$); o núcleo do corpo celular é representado por uma função somatória que combina as entradas do neurônio; ao resultado do somatório ou campo induzido (v_o), é aplicada a função de ativação *f(.)* gerando um valor de saída (y_o) no axônio do neurônio.

A grande questão que envolve o uso de um neurônio artificial na resolução de um problema é determinar cada uma das variáveis que compõem o modelo neuronal de acordo com as especificações do problema. Cada um dos atributos descritivos de um exemplar de entrada é associado a cada uma das entradas (x_j, $j = 1 ... d$) do neurônio, possibilitando a apresentação de uma informação referente ao problema, e, portanto, o valor da variável *d* depende do problema sob resolução; a função de ativação é escolhida pelo projetista que configura o neurônio artificial, observando também algumas características do problema; cada uma das sinapses (ou pesos sinápticos) (w_{oj}, $j = 1 ... d$) é ajustada por meio de um algoritmo de

treinamento (ou de aprendizado), de tal forma que a saída do neurônio (y_o) seja a saída esperada na resolução do problema.

Opcionalmente, um neurônio *Perceptron* pode ter uma entrada extra chamada *bias*, cujo valor (também chamado de valor de ativação) é sempre 1 e cujo peso associado é denotado por b_o. Na existência do *bias*, o algoritmo de treinamento faz o ajuste do peso b_o com o ajuste do conjunto de pesos $\mathbf{w}\rightarrow_o$. A existência do *bias* aumenta ou diminui o sinal no campo induzido, dependendo se o valor do peso associado é positivo ou negativo, respectivamente.

Para entender como um neurônio pode resolver tarefas de classificação de dados, considere o simples problema de implementação de uma porta lógica AND . A Tabela 3-4 ilustra o problema informando cada uma das entradas da porta lógica e as respectivas saídas esperadas. Um neurônio projetado para resolver esse problema receberá x_1 e x_2 como entradas,[3] ponderará essas entradas por meio da aplicação dos pesos sinápticos, as combinará no somatório e aplicará a função de ativação para produzir uma saída (y^), que deve ser igual à saída desejada (y).

Tabela 3-4: Função lógica AND representando um problema a ser resolvido por um neurônio artificial do tipo *Perceptron*

x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1

Para que o neurônio possa ser preparado para resolver o problema proposto, é necessário definir uma função de ativação para ele. Uma função degrau, ilustrada na Figura 3-6, pode ser usada. Ela produzirá a saída 1

quando o campo induzido do neurônio for maior que 0, e produzirá o valor 0 caso contrário. A regra de processamento do neurônio pode então ser definida: o cálculo do sinal que entra no neurônio *o* é $v_o = \sum_{j=1}^{d} (x_j * w_{oj}) + b_o$ e a saída do neurônio *o* é dada por $y_o = f(v_o)$.

Definidas a função de ativação e a regra de processamento do neurônio, resta agora definir como o ajuste dos pesos sinápticos (o aprendizado) deve ser feito. Há diferentes princípios de aprendizado que podem ser aplicados a um neurônio, e um deles ajusta o conjunto de pesos sinápticos, iterativamente, a partir do erro $e_o(t)$, produzido no tempo *t*, entre a saída desejada (y) e a saída produzida pelo neurônio *o* (y_o). Ou seja, a regra de aprendizado do *Perceptron*, definida por Rosemblatt, é $\Delta w_{oj}(t) = \eta * x_j * e_o(t)$ em que $e_o(t) = y - y_o(t)$, x_j é a entrada associada ao peso w_{oj} sendo processada pelo neurônio no tempo *t*, e η é uma taxa de aprendizado que tem a função de ponderar o ajuste de pesos de forma que o aprendizado não se torne especializado no exemplar de entrada. Caso o *bias* esteja sendo usado no neurônio, o ajuste de pesos sinápticos deve também ser aplicado a ele por meio de $\Delta b_o(t) = \eta * e_o(t)$. A partir do cálculo das variações Δ, os pesos podem ser efetivamente atualizados por meio da adição de Δ aos seus valores atuais.

Figura 3-6: Função de ativação do tipo degrau

As atualizações dos pesos são realizadas com o objetivo de minimizar o erro produzido pelo neurônio e representam, portanto, o aprendizado (nesse caso, supervisionado). Esse processo é chamado também de treinamento , por se tratar de um processo iterativo que deve ser repetido muitas vezes

(épocas), até que o erro produzido seja nulo ou suficientemente pequeno. O Algoritmo 3-2 implementa o treinamento supervisionado para o *Perceptron*. Note que, embora o *Perceptron* seja apresentado aqui como o modelo de um neurônio, uma rede de neurônios do tipo *Perceptron* pode ser construída, e o Algoritmo 3-2 se adéqua a essa situação. O conceito de modelagem de uma rede neural será deixado para o momento de apresentação da rede neural artificial *Multilayer Perceptron* (MLP), realizada mais adiante.

A Tabela 3-5 mostra momentos do treinamento (passagem de alguns exemplares em algumas épocas) de um *Perceptron* para o problema da porta lógica AND, seguindo o Algoritmo 3-2. Para esse exemplo, o conjunto de pesos foi inicializado em 0, a taxa de aprendizado η foi definida como 0,1, o erro e_{final} foi definido com 0, e a função de ativação $f(v_o)$ definida como a função degrau da Figura 3-6. Nessa tabela, ainda foi considerada a variável *i* para indicar o exemplar escolhido (a ordem de apresentação dos exemplares dentro do laço do *Passo 2.1* é aleatória) para ser apresentado – *Passo 2.1* do algoritmo de treinamento.

Algoritmo 3-2: Algoritmo de treinamento (ou aprendizado) supervisionado para o *Perceptron*

Parâmetros de entrada:

- $\mathbf{X}_{tr}$: um conjunto de exemplares rotulados de treinamento, ou seja, $\mathbf{X}$tr = $\{(\mathbf{x}\rightarrow_i, y_i)\}$, $i = 1, ..., n$;
- $\mathbf{w}\rightarrow_o$: um conjunto de pesos sinápticos tipicamente inicializado aleatoriamente ou com valores iguais a 0.
- b_o: peso *bias* inicializado seguindo a mesma estratégia usada para $\mathbf{w}\rightarrow_o$.
- $f(\cdot)$: função de ativação.
- η: taxa de aprendizado inicializada com um número no intervalo]0,1].
- *épocas*: número máximo de épocas de treinamento (opcional se houver e_{final}).
- $\varepsilon_{\text{final}}$: erro máximo permitido (um valor pequeno, opcional se houver *épocas*).

Parâmetros de saída:

- $\mathbf{w}\rightarrow_o$: conjunto de pesos sinápticos ajustados.
- b_o: peso *bias* ajustado.

Passo 1: defina $t = 0$.
Passo 2: **enquanto** $t <$ épocas **ou** $\xi > e_{final}$ **faça**
Passo 2.1: $\xi = 0$;

Passo 2.2: **para cada** par $(\boldsymbol{x}_{\rightarrow i}, y_i)$ **faça**
Passo 2.2.1: calcule a saída do neurônio: $y_o = f(v_o) = f(\sum_{j=1}^{d}(x_j * w_{oj}) + b_o)$;
Passo 2.2.2: calcule o erro entre a saída obtida e a saída desejada: $e_o = y_i - y_o$;
Passo 2.2.3: atualize o erro acumulado na iteração $\xi = \xi + e_o^2$;
Passo 2.2.4: ajuste os pesos sinápticos $w_{oj} = w_{oj} + \eta * x_j * e_o$ e $b_o = b_o + \eta * e_o$;
Passo 2.3: $t = t + 1$;
Passo 2.4: $\xi = \frac{1}{2} * \xi$;
Passo 2.5: decremente a taxa de aprendizado η (usando uma função de decrescimento); // opcional
Passo 2.6: teste condições de parada

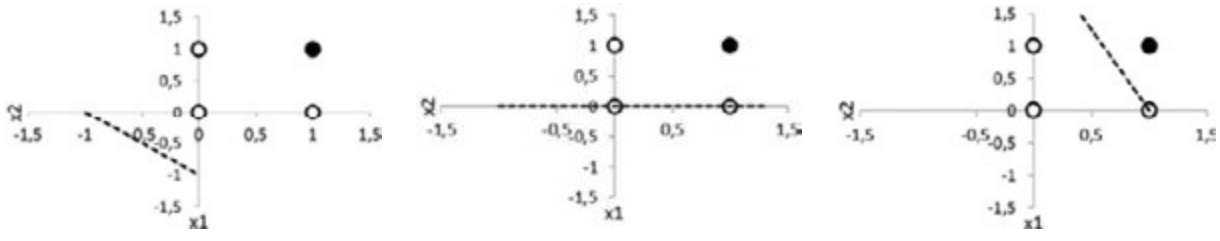
Tabela 3-5: Amostras da execução do treinamento de um *Perceptron* aplicado ao problema da porta lógica AND

Iteração		**Entradas**			**Pesos Atuais**			**Saídas**		**Erro**	**Ajustes**			**Pesos Alterados**		
t	i	x_{i1}	x_{i2}	yi	w_{o1}	w_{o2}	w_{ob}	v_o	y_o	$y_i - y_o$	Δw_{o1}	Δw_{o2}	Δb_o	w_{o1}	w_{o2}	b^o
–	–	–	–	–	0	0	0	–	–	–	–	–	–	–	–	–
0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
.....																
0	4	1	1	1	0	0	0	0	0	1	0	0,1	0,1	0	0,1	0,1
.....																
5	4	1	1	1	0	0	0	0	0	1	0,1	0	0,1	0,1	0	0,1
.....																
10	4	1	1	1	0,5	0,2	-0,5	0,2	1	0	0	0	0	0,5	0,2	-0,5

A execução parcial do Algoritmo 3-2 permite observar que, como o ajuste de pesos sinápticos é feito em função do erro cometido na saída do neurônio, quando o erro é nulo, nenhum ajuste de pesos é necessário, pois o neurônio respondeu com a saída desejada. A exemplo disso, veja que em $t = 0$ e $i = 1$, a saída calculada ($y_o = f(1 * 0 + 0 * 0 + 0) = 0$, *Passo 2.2.1*) é igual a desejada (y_i), portanto, o erro é zero ($e = y_i - y_o = 0 - 0 = 0$, *Passo 2.2.2*). Por outro lado, em $i = 4$, da mesma época t, o erro é 1, e os pesos devem ser atualizados (por exemplo, o peso w_{o2} tem ajuste $\Delta w_{o2} = 0{,}1 * 1 * 1 = 0{,}1$, e peso alterado

igual a w_{o2} = 0 + 0,1 = 0,1 – *Passo 2.2.4*). Um neurônio terá conseguido aprender a solução para o problema da porta lógica AND quando, dentro de uma época ou dentro de um limite de erro permitido, ele responder corretamente para todas as entradas presentes no conjunto de treinamento.

Interessante ainda notar que o processamento de um neurônio *Perceptron* pode ser interpretado geometricamente como resultando em um hiperplano de separação do espaço *d*-dimensional dos exemplares de entrada. Considerando o problema da porta lógica AND, para a qual o espaço dos exemplares de entrada é 2-dimensional (x_1, x_2) (Figura 3-7), o processamento do neurônio resulta em uma reta de separação do espaço. De fato, quando os atributos descritivos de um exemplar ou simplesmente entradas x_1 e x_2 são apresentadas ao neurônio, seu sinal no campo induzido é calculado como (x_1 * w_1) + (x_2 * w_2) + b; se esse sinal é maior que 0, a resposta do neurônio é 1, se o sinal é menor ou igual a 0, a resposta do neurônio é 0 (considerando a função de ativação como mostrado na Figura 3-6). Logo, tem-se que o limite de separação entre as duas classes é dado pela equação de reta: (x_1 * w_1) + (x_2 * w_2) + b = 0, em que w_1, w_2 e b são constantes determinadas no algoritmo de treinamento. Para o treinamento realizado sobre o problema da porta lógica AND, os pontos de interseção da reta nos eixos x_1 e x_2, segundo as informações da Tabela 3-5, serão, respectivamente, $x_1 = -b/w_1 = 0{,}5/0{,}5 = 1{,}0$ e $x_2 = -b/w_2 = 0{,}5/0{,}2 = 2{,}5$ (reta plotada na Figura 3-7(c)). Como ilustração, as retas de separação obtidas após a primeira e a segunda épocas do treinamento são mostradas na Figura 3-7(a) e (b).



(a) t = 1 e i = 4 (b) t = 5 e i = 4 (c) t = 10 e i = 4

Figura 3-7: Representação geométrica do problema da porta lógica AND: os círculos brancos representam a classe 0 e o círculo preto representa a classe 1; a reta pontilhada é a reta de separação que o neurônio representa

Após o treinamento, a equação do hiperplano de separação pode ser usada para classificar exemplares novos, para os quais a informação de rótulo é desconhecida. O hiperplano de separação é considerado o modelo de predição. O algoritmo de aplicação do *Perceptron* para classificação de novos exemplares é apresentado no Algoritmo 3-3.

Algoritmo 3-3: Algoritmo para classificação de novos exemplares a partir de um *Perceptron* treinado

Parâmetros de entrada:
- $\mathbf{X}_{te}$: um exemplar de teste com rótulo desconhecido, **X**te = {$\boldsymbol{x}\rightarrow_{te}$);
- $\mathbf{w}\rightarrow_{o}$: conjunto de pesos sinápticos obtido ao final do processo de treinamento;
- b_o: peso *bias* obtido ao final do processo de treinamento;
- $f(\cdot)$: função de ativação usada no treinamento

Parâmetro de saída:
- $\hat{y}_{te}$ rótulo estimado para o exemplar de teste $\boldsymbol{x}\rightarrow_{te}$;

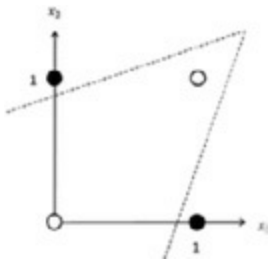
Passo 1: calcule a saída do neurônio $y_o = f(v_o) = f(\sum^{d}_{j=1}(x_j * w_{oj}) + b_o)$;
Passo 2: faça $\hat{y}_{te} = y_o$

O uso do neurônio *Perceptron* como modelado até o momento está restrito a conjuntos de dados com exemplares organizados em classes que podem ser separadas por uma reta (ou um hiperplano), ou seja, problemas linearmente separáveis. Em situações nas quais essa restrição não é obedecida, a aplicação do *Perceptron* como gerador de um modelo de decisão fica prejudicada. Para exemplificar esse problema, considere a modelagem da porta lógica OU EXCLUSIVO, também denominada XOR, ilustrada na Figura 3-8(a). A separação dos exemplares de cada classe (classe 0 e classe 1) é não linearmente separável, como mostrado na Figura 3-8(b). Nessa figura, é possível notar que a separação das classes é feita a partir da combinação de duas retas e, para que isso seja possível, é necessário

combinar mais de um neurônio *Perceptron*, criando uma rede de neurônios organizados, por exemplo, em múltiplas camadas – *Multilayer Perceptron*.

x_1	x2	y
0	0	0
0	1	1
1	0	1
1	1	0

(a)



(b)

Figura 3-8: (a) Problema XOR; (b) Representação geométrica do problema XOR

A partir da possiblidade da combinação de neurônios em uma rede de múltiplas camadas , é possível a obtenção de estruturas mais complexas, como a arquitetura para rede neural ilustrada na Figura 3-9(a). Nessa figura, aparece o conceito de camadas: a camada de entrada é formada por unidades responsáveis apenas por receber os valores descritivos dos exemplares processados na rede neural, não havendo processamento em tais unidades; na camada de saída, estão os neurônios que processam informação recebida das camadas escondidas e fornecem a resposta da rede neural para cada exemplar a ela apresentado; e, no meio, estão as camadas escondidas, compostas por neurônios que processam a informação descritiva dos exemplares, implementando mapeamentos especiais que possibilitam a resolução de problemas complexos. Essas camadas possuem uma série de neurônios que recebem sinal dos neurônios da camada anterior e enviam sinal para os neurônios da próxima camada, caracterizando um fluxo de informação do tipo *feedforward* (alimentação para frente).

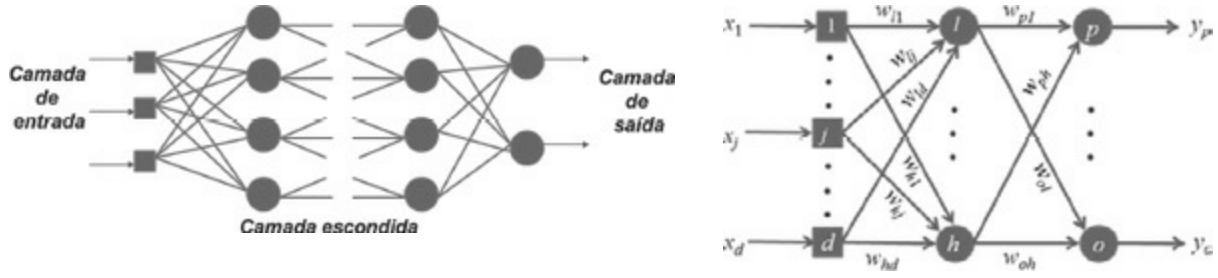


Figura 3-9: (a) Arquitetura genérica de uma rede neural de múltiplas camadas; (b) Arquitetura da rede *Multilayer Perceptron* com uma única camada escondida

Perceba que, agora, o ajuste de pesos proposto por Rosemblatt com respeito a cada neurônio precisa ser modificado. Originalmente, o ajuste de pesos do *Perceptron* é feito com base na informação sobre o erro cometido na sua saída e também considerando a informação de entrada na rede neural. Agora, na nova arquitetura, o erro (calculado na camada de saída) não é mais associado apenas às entradas da rede, mas também ao processamento realizado nas camadas escondidas; e, ainda, não há como calcular o erro ocorrido nos neurônios das camadas escondidas de forma direta, uma vez que não existe a resposta desejada para tais neurônios. Sendo assim, uma nova forma de implementar o ajuste de pesos precisa ser utilizada, e o algoritmo para ajuste de pesos nessa arquitetura é conhecido como aprendizado por retropropagacão do erro (em inglês, *error backpropagation*), proposto por Rumelhart, Hinnton e Willians (1986).

O funcionamento desse algoritmo de treinamento é composto por três fases: na primeira, um exemplar do conjunto de dados usado para treinamento da rede neural é apresentado na camada de entrada da rede e é propagado para as próximas camadas por meio do processamento nos neurônios (já conhecido do leitor – pois são neurônios do tipo *Perceptron*) até que, na camada de saída, o resultado final é produzido; na segunda fase, a resposta produzida na camada de saída é comparada com a resposta desejada, o erro obtido é calculado e, então, propagado de volta até a camada de

entrada, propiciando que as alterações necessárias nos pesos das sinapses sejam calculadas; na terceira fase, o ajuste de pesos é de fato realizado a partir das alterações calculadas na segunda fase.

Para entender em detalhes como o processo descrito ocorre, considere a arquitetura da **rede neural artificial *Multilayer Perceptron*** mostrada na Figura 3-9(b). Nessa figura, um neurônio em uma camada pode ser identificado como *d*, *h* e *o*, a depender se estão localizados respectivamente nas camadas de entrada, escondida e de saída. Ainda, *d*, *h* e *o* significam a quantidade de neurônios nas respectivas camadas. A rede neural referente a tal arquitetura possui apenas uma camada escondida.

A propagação da informação sobre um exemplar do conjunto de dados de treinamento ($\boldsymbol{x}\rightarrow_i$) inicia com a instanciação dos neurônios de entrada (*d*), de acordo com os valores dos atributos descritivos do exemplar. Em seguida, tais valores são propagados para todos os neurônios da próxima camada (*h*), considerando o conjunto de pesos ($\mathbf{w}\rightarrow_h$) associados a cada neurônio. Então, o campo induzido para cada neurônio *h* é dado por $v_h = \sum^{d}_{j=1} (x_j * w_{hj})$, e a saída produzida por um neurônio *h* é dada por $z_h = f(v_h)$. A saída do neurônio *h* deve ser propagada para a próxima camada (a camada de saída, neste exemplo), de forma que o campo induzido para cada neurônio de saída *o* e a saída produzida por ele sejam, respectivamente: $u_o = \sum^{h}_{l=1} (z_l * w_{ol})$ e $y_o = f(u_o)$.

A partir dos resultados produzidos na camada de saída, o erro para cada neurônio de saída *o* pode ser calculado fazendo $e_o = y_{io} - y_o$, sendo y_{io} a saída desejada em cada neurônio[4] *o* para a entrada $\boldsymbol{x}\rightarrow_i$. O erro produzido deve ser retropropagado para que a correção de pesos seja realizada. A retropropagação do erro se dá a partir do cálculo de uma *informação de erro* (δ) e da inserção dessa informação na regra de alteração de pesos. A informação de erro de cada neurônio de saída *o* é calculada a partir do erro produzido na sua saída e da derivada de sua função de ativação no ponto

referente ao valor de seu campo induzido: $\delta = e_o * f'(u_o)$. Então, a regra para cálculo da alteração de pesos dos neurônios da camada de saída é $\Delta w_{oh} = \eta * z_h * \delta_o$; e a atualização efetiva é $w_{oh} = w_{oh} + \Delta w_{oh}$.

Para os neurônios da camada escondida, é preciso considerar a informação de erro δ_o calculada na camada de saída. Dado que cada neurônio da camada escondida tem uma parcela de contribuição na produção dos erros gerados nos neurônios da camada de saída, as alterações de seus pesos devem considerar a combinação de todos δ_o. Assim, a informação do erro a ser considerada para o ajuste de pesos dos neurônios da camada escondida é $\delta_h = \sum^{o}_{p=1} (\delta_p * w_{ph}) * f'(v_h)$, o cálculo da alteração de pesos para os neurônios da camada escondida é $\Delta w_{hd} = \eta * x_d * \delta_h$, e a atualização efetiva é $w_{hd} = w_{hd} + \Delta w_{hd}$. Note que a diferença entre a alteração de pesos da camada de saída e da camada escondida é que, para essa última, a informação de erro deve considerar a soma do que ocorre em todos os neurônios da camada de saída.

Para o caso de haver mais camadas escondidas na arquitetura da rede neural *Multilayer Perceptron*, a regra de alteração de pesos de camada escondida deve ser replicada. Também note que o peso *bias* não foi considerado na discussão sobre o algoritmo de retropropagação de erro. Se o *bias* é desejado, as regras de alteração para ele devem ser derivadas seguindo a lógica aplicada aos demais pesos. De fato, o algoritmo de treinamento de rede neural por retropropagação do erro tem como objetivo minimizar o erro quadrado total cometido pela rede neural, dado por $\xi = 1/2 \sum^{o}_{p=1} e^2_p$. O processo de treinamento da rede neural *Multilayer Perceptron* e o procedimento de aplicação da rede para novos exemplares de classe desconhecida são apresentados, respectivamente, no Algoritmo 3-4 e no Algoritmo 3-5.

Algoritmo 3-4: Algoritmo de treinamento (ou aprendizado) supervisionado Retropropagação do Erro para a rede neural Multilayer Perceptron

---

Parâmetros de entrada:

- $\mathbf{X}_{tr}$: um conjunto de exemplares rotulados de treinamento, ou seja, $\mathbf{X}$tr = $\{(\boldsymbol{x}\rightarrow_i, y_i)\}$, $i = 1, ..., n$;
- $\mathbf{w}\rightarrow_h$: um conjunto de pesos sinápticos para os neurônios (h) da camada escondida inicializado aleatoriamente;
- $\mathbf{w}\rightarrow_o$: um conjunto de pesos sinápticos para os neurônios (o) da camada de saída inicializado aleatoriamente;
- $f(\cdot)$: função de ativação – derivável em todo o seu domínio;
- η: taxa de aprendizado inicializada com um número no intervalo]0,1];
- *épocas*: número máximo de épocas (opcional, se houver e_{final});
- e_{final}: erro máximo permitido (um valor pequeno e opcional, se houver *épocas*);

Parâmetros de saída:
- $\mathbf{w}\rightarrow_h$: conjunto de pesos sinápticos para os neurônios (h) da camada escondida ajustados;
- $\mathbf{w}\rightarrow_o$: conjunto de pesos sinápticos para os neurônios (o) da camada de saída ajustados;

Passo 1: defina t = 0;
Passo 2: **enquanto** $t <$ *épocas* ou $\xi > e_{\text{final}}$ **faça**
Passo 2.1: $\xi = 0$;
Passo 2.2: **para cada** par $(\boldsymbol{x}\rightarrow_i, y_i)$, **faça** // y_i deve ser codificado em um vetor se houver mais de um neurônio na camada de saída
Passo 2.2.1: calcule as saídas dos neurônios da camada escondida $z_h = f(v_h) = f(\sum^{\text{d}}_{j=1}(x_j * w_{hj}))$;
Passo 2.2.2: calcule as saídas dos neurônios da camada de saída $y_o = f(u_o) = f(\sum^{\text{h}}_{l=1}(z_l * w_{ol}))$;
Passo 2.2.3: calcule os erros nos neurônios na camada de saída, $e_o = y_{io} - y_o$;
Passo 2.2.4: atualize o erro quadrado total na camada de saída $\xi = \xi + \sum^{\text{o}}_{p=1} e^2_p$;
Passo 2.2.5: calcule a informação de erro em cada neurônio na camada de saída, $\delta_o = e_o * f'(u_o)$;
Passo 2.2.6: determine os ajustes de pesos dos neurônios da camada de saída $\Delta w_{oh} = \eta * z_h * \delta_o$;
Passo 2.2.7: calcule a informação de erro em cada neurônio na camada escondida, $\delta_o = \sum^{\text{o}}_{p=1}(\delta_p * w_{ph}) * f'(v_h)$;
Passo 2.2.8: determine os ajustes de pesos dos neurônios da camada de escondida $\Delta w_{hd} = \eta * x_d * \delta_h$;
Passo 2.2.9: aplique as alterações de pesos $w_{hd} = w_{hd} + \Delta w_{hd}$ e $w_{oh} = w_{oh} + \Delta w_{oh}$;
Passo 2.3: decremente a taxa de aprendizado η (usando uma função de decrescimento); // opcional
Passo 2.4: t = t + 1;
Passo 2.5: $\xi = \frac{1}{2} * \xi$;
Passo 2.6: teste condições de parada.

Algoritmo 3-5: Algoritmo para classificação de novos exemplares a partir de uma rede neural *Multilayer Perceptron* treinada

---

Parâmetros de entrada:
- $\mathbf{X}_{te}$: um exemplar de teste com rótulo desconhecido, $\mathbf{X}$te = $\{\boldsymbol{x}\rightarrow_{te}\}$;
- $\mathbf{w}\rightarrow_h$: conjunto de pesos sinápticos da camada escondida, obtido ao final do processo de treinamento;
- $\mathbf{w}\rightarrow_o$: conjunto de pesos sinápticos da camada de saída, obtido ao final do processo de treinamento;

- $f(\cdot)$: função de ativação usada no treinamento;

Parâmetros de saída:
- $\hat{y}_{te}$: saída estimada para o exemplar de teste $\boldsymbol{x}_{te}$;

Passo 1: calcule a saída dos neurônios da camada escondida $z_h = f(\sum_{j=1}^{d}(x_j * w_{hj}))$;
Passo 2: calcule a saída dos neurônios da camada de saída $y_o = f(\sum_{j=1}^{h}(z_j * w_{ol}))$;
Passo 3: faça $\hat{y}_{te} = y_o$ para cada neurônio da camada de saída.

Importante salientar que a função de ativação usada no neurônio de uma rede neural *Multilayer Perceptron* treinada com o algoritmo de retropropagação do erro deve ser derivável em todos os seus pontos, caso contrário, o cálculo da informação de erro não seria possível. Também, é desejável que a função seja não linear, de forma a possibilitar a geração de superfícies de decisão não lineares, em vez de serem capazes apenas de gerar hiperplanos de separação. Uma função de ativação tipicamente usada nessa rede neural é a função logística $y = 1/(1 + \exp(-av))$ na qual a é o parâmetro de inclinação da função e v é o argumento de entrada. Outras funções possíveis para uso nessa rede neural podem ser estudadas em (Fausett, 1993; Haykin, 2008).

3.1.2.1. Exemplo didático para *Perceptron* e *Multilayer Perceptron*

Para exemplificar o uso de redes neurais artificiais na resolução da tarefa de classificação, o conjunto de dados *Pesquisa de Satisfação* (Tabela 3-1) será submetido à análise do neurônio *Perceptron* e da rede neural artificial *Multilayer Perceptron*.[5]

Inicialmente, considere o neurônio *Perceptron*, cujo conjunto de pesos é inicializado com valores iguais a 0; com unidade de *bias* igual a 1; valor para a taxa de aprendizado determinado em 1; função de ativação degrau e uma passagem aleatória dos exemplares do conjunto de treinamento dentro de cada época. O resultado do treinamento, considerando várias épocas de treinamento, está ilustrado na Tabela 3-6. Note que o rótulo de cada padrão de treinamento era, originalmente, um atributo do tipo categórico e foi

transformado em numérico (com domínio {0,1}), para que pudesse ser tratado pelo algoritmo de treinamento. A interpretação geométrica do resultado do treinamento está apresentada na Figura 3-10, na qual a classe `SATISFAÇÃO` (y = 1) é representada por quadrados, e a classe `INSATISFAÇÃO` (y = 0) é representada por círculos.

Tabela 3-6: Resultados para algumas épocas de treinamento do *Perceptron*, considerando o conjunto de dados *PESQUISA DE SATISFAÇÃO*

Iteração		**Entradas**			**Pesos Atuais**			**Saídas**		**Erro**	**Ajustes**			**Pesos Alterados**		
t	i	x_{i1}	x_{i2}	y_i	w_{o1}	w_{o2}	b_o	v_o	y_i	$y_i - y_o$	Δw_{o1}	Δw_{o2}	Δb_o	w_{o1}	w_{o2}	b_o
–	0	–	–	–	0	0	0	–	–	–	–	–	–	–	–	–
0	1	0,7	0,4	1	0,45	0,40	–0,60	–0,13	0,00	1,00	0,07	0,04	0,10	0,52	0,44	–0,50
.....																
10	5	0,4	0,5	1	0,52	0,44	–0,50	–0,07	0,00	1,00	0,04	0,05	0,10	0,56	0,49	–0,40
.....																
100	7	0,61	0,4	1	0,56	0,49	–0,40	0,14	1,00	0,00	0,00	0,00	0,00	0,56	0,49	–0,40
.....																
1000	9	0,9	0,7	1	0,56	0,49	–0,40	0,45	1,00	0,00	0,00	0,00	0,00	0,56	0,49	–0,40
.....																
10000	11	1	0,2	0	0,48	0,39	–0,60	–0,04	0,00	0,00	0,00	0,00	0,00	0,48	0,39	–0,60

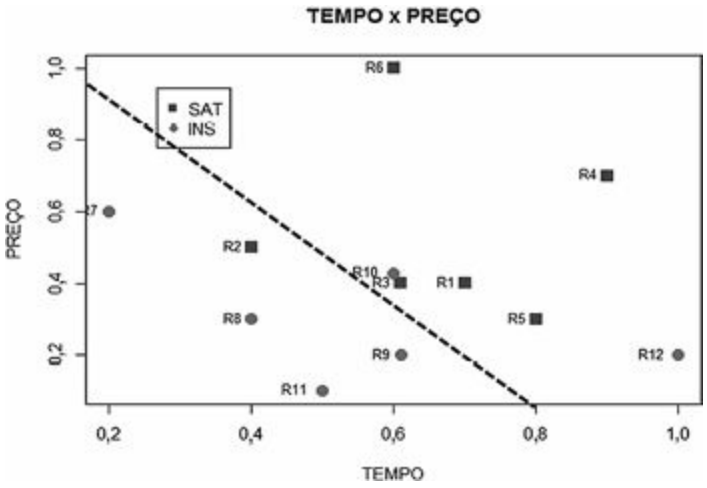
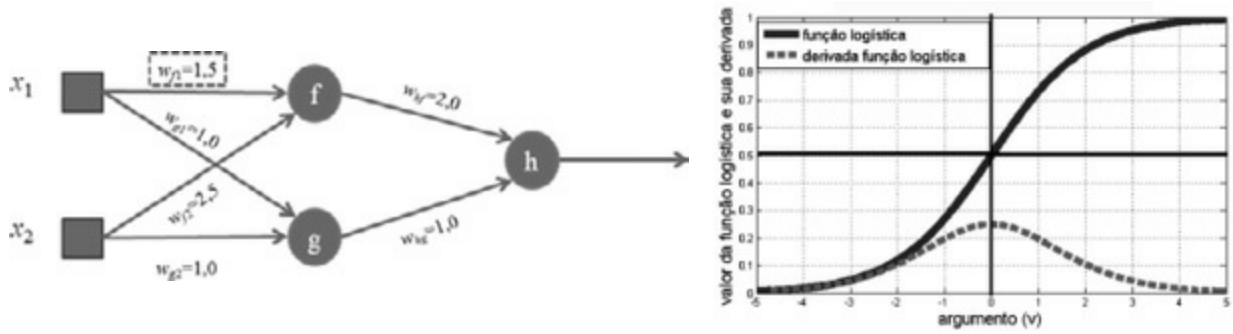


Figura 3-10: Interpretação geométrica do *Perceptron* treinado para o conjunto de dados *PESQUISA*

A exemplificação do teste do *Perceptron* será feita considerando o exemplar (0,90; 0,70) do conjunto de dados *Pesquisa de Satisfação* e os pesos obtidos após a interrupção do treinamento (Tabela 3-6). Representando o teste por meio da instanciação da equação do hiperplano de separação (equivalente a executar os passos do Algoritmo 3-3), tem-se: (0,90 ∗ 0,48) + (0,70 ∗ 0,39) – 0,60 = 0,11. O valor encontrado para a saída do neurônio é maior que 0 e, portanto, $y = 1$, e a classe à qual o exemplar de teste pertence é a classe `SATISFAÇÃO`.

Note, na Figura 3-10, que há exemplares classificados erroneamente pela reta de separação. O erro ocorre porque o *Perceptron* está modelado de uma forma que só consegue resolver problemas linearmente separáveis, e, neste problema, essa restrição não vale.

Para este exemplo didático, a arquitetura ilustrada na Figura 3-11(a) é adotada para a rede neural artificial *Multilayer Perceptron*, e uma única iteração do *Passo 2* do Algoritmo 3-4 é realizada para ilustrar a alteração de um peso sináptico (w_{f1}). O treinamento completo pode ser observado pelo leitor a partir da implementação em R, explicada na próxima seção. Os pesos sinápticos foram inicializados com valores aleatórios, a função de ativação dos neurônios é a função logística (Figura 3-11(b)) e a taxa de aprendizado é inicializada em 0,1. A mesma transformação para o atributo rótulo e o mesmo exemplar de teste, considerados no caso do *Perceptron*, devem ser considerados para a rede neural artificial *Multilayer Perceptron*.



(a) (b)

Figura 3-11: (a) arquitetura inicial para a MLP; (b) função de ativação logística e sua derivada

Para facilitar a demonstração do processo de treinamento da rede neural artificial MLP, acompanhe o grafo de fluxo[6] (Figura 3-12) e os cômputos realizados (Figura 3-13) no cálculo das variáveis envolvidas no processo. No grafo de fluxo, variáveis com valores dizem respeito a inicializações ou aos cálculos sob demonstração, e variáveis sem valores dizem respeito àquelas computadas no próximo passo. Na Figura 3-12, está ilustrado o grafo de fluxo correspondente a apresentação do exemplar de teste.

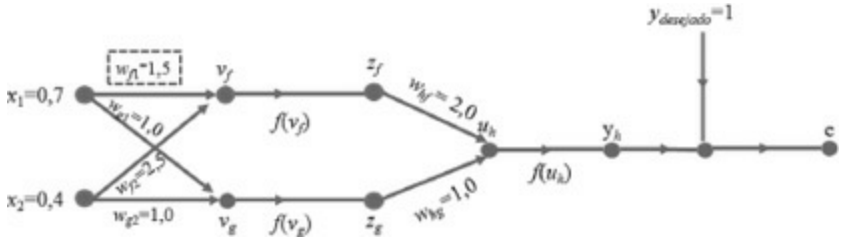


Figura 3-12: Grafo de fluxo correspondente à apresentação do exemplar de teste à rede MLP

O valor dos campos induzidos v_f e v_g para os neurônios f e g da camada escondida são respectivamente $v_f = \sum_{j=1}^{2} x_j * w_{fj} = (0,7 * 1,5) + (0,4 * 1,5) \cong 2,1$ e $v_g = \sum_{j=1}^{2} x_j * w_{gj} = (0,7 * 1,0) + (0,4 * 1,0) \cong 1,1$; tais valores devem ser aplicados na função de ativação para gerar as saídas dos neurônios, $z_f = f(v_f) = f(2,1) \cong 0,9$ e $z_g = f(v_g) = f(1,1) \cong 0,8$ (*Passo 2.2.1* do Algoritmo 3-4). Então, o *Passo 2.2.2* do algoritmo pode ser realizado, computando o campo induzido u_h e o valor de saída y_h do neurônio h da camada de saída. Tais cômputos são, respectivamente, $u_h = \sum_{l=1}^{2} z_l * w_{hl} = (0,9 * 2,0) + (0,8 * 1,0) = 2,6$ e $y_h = f(u_h) = f(2,6) \cong 0,9$. Enfim, o valor obtido na saída do neurônio h é comparado ao valor de saída desejado y por meio do cálculo do erro, conforme *Passo 2.2.3*, totalizando $e_h = y_{\text{desejado}} - y_h = 1,0 - 0,9 = 0,1$. O grafo

de fluxo para a fase de propagação do sinal pela rede é mostrado na Figura 3-13.

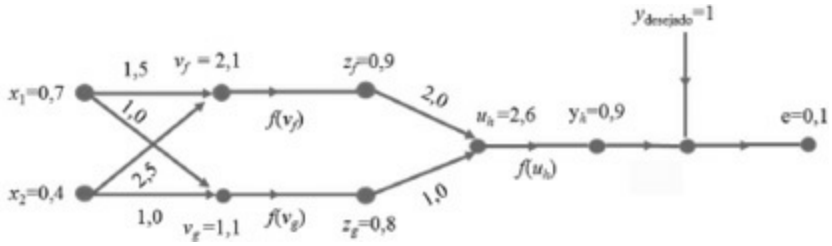


Figura 3-13: Grafo de fluxo ao final do passo de propagação do sinal pela rede

Em caso de treinamento completo, seria ainda necessário o cálculo do erro médio quadrado, como definido no *Passo 2.2.4* e no *Passo 2.5*. Caso esse erro seja menor ou igual ao erro máximo permitido, diz-se que a rede neural convergiu, seu processo de treinamento pode ser finalizado, e ela está pronta para ser aplicada na classificação de novos exemplares com rótulos desconhecidos.

Após a fase de propagação do sinal gerando a resposta da rede, é preciso propagar de volta (retropropagar) o erro obtido na saída da rede a fim de realizar as atualizações dos pesos. Nesse exemplo, deseja-se atualizar apenas o peso w_{f1} na camada escondida. Portanto, é preciso calcular a informação de erro associada para então calcular o ajuste do peso. As variáveis envolvidas neste ajuste estão destacadas na Figura 3-14,e, para o ajuste, os *Passos* de *2.2.5* a *2.2.9* do Algoritmo 3-4 devem ser executados.

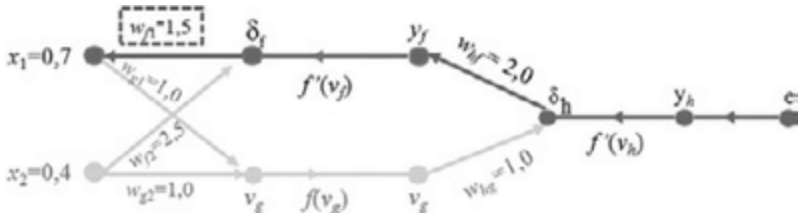


Figura 3-14: Grafo de fluxo para a retropropagação do erro com a finalidade de ajuste do peso w_{f1}

Em sequência, inicialmente calcula-se a informação de erro δ_h do neurônio da camada de saída, conforme *Passo 2.2.5*, resultando em $\delta_h = e_h *$

$f'(u_h) = 0{,}1 * f'(2{,}6) = 0{,}1 * 0{,}09 = 0{,}009$. Agora, com a informação de erro do neurônio da camada de saída, calcula-se a informação de erro para o neurônio da camada escondida referente ao peso atualizado (*Passo 2.2.7*): $\delta_f = \delta_h * w_{hf} * f'(v_f) = 0{,}009 * 2{,}0 * 0{,}1 = 0{,}0018$. E, finalmente, encontra-se o ajuste que deverá ser aplicado a w_{f1}: $\Delta w_{f1} = \eta * x_j * \delta_f = 1{,}0 * 0{,}7 * 0{,}0018 = 0{,}0013$. Este valor de ajuste será adicionado ao valor atual do peso, definindo, assim, um novo valor para ele (*Passo 2.2.9*): $w_{f1} = w_{f1} + \Delta w_{f1} = 1{,}5 + 0{,}0013 = 1{,}5013$.

Com isso, a exemplificação do ajuste do peso sináptico com o algoritmo de retropropagação de uma rede neural artificial *Multilayer Perceptron* é encerrada. Novamente, deve-se ressaltar que, para um treinamento completo, todos os pesos devem ser ajustados iterativamente até um dos critérios – número máximo de épocas ou erro médio quadrado máximo permitido – ser satisfeito. E, embora este exemplo não tenha apresentado todo o treinamento da rede neural, apenas para finalidade de exemplificação, um exemplo de fronteira de separação (não linear) possível de ser descoberta pela rede é mostrada na Figura 3-15. O procedimento de aplicação de uma rede MLP é exemplificado na Seção 3.3.4.

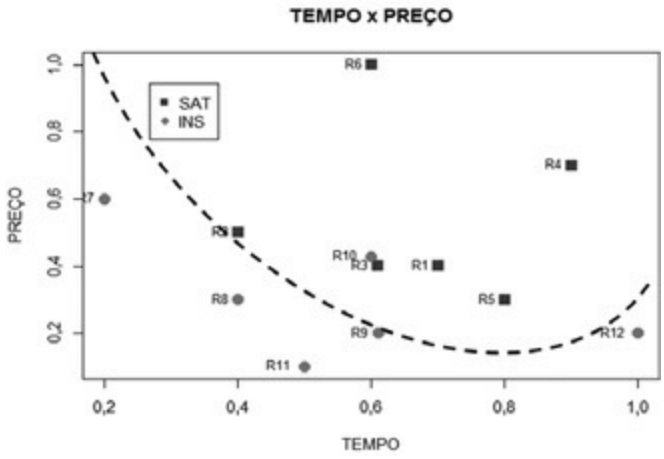


Figura 3-15: Exemplo de fronteira de separação que deverá ser encontrada pela rede MLP

3.1.2.2. Exemplo prático para o *Multilayer Perceptron* em R

Os pacotes em R mais utilizados para implementação de Redes Neurais Artificiais são **nnet** (Venables e Rippley, 2002), **neuralnet** (Fritsch, Guenther e Suling, 2012) e ***RSNNS*** – *Stuttgart Neural Network Simulator* (Bergmeir e Benitex, 2012). Todos eles disponibilizam, na forma de funções, uma série de arquiteturas de redes neurais artificiais; porém, nenhum implementa o *Perceptron*. A razão para tal, segundo entendimento dos autores deste livro, pode ser que a implementação de um *Perceptron* é relativamente simples e, além disso, a sua limitação, por operar apenas com problemas linearmente separáveis.

Nesta seção, a implementação em R para a rede neural artificial *Multilayer Perceptron* é apresentada sob o uso do pacote ***RSNNS***. O pacote ***RSNNS*** dispõe de implementações para construção dos conjuntos de treinamento e teste, para normalização de dados e para análise de desempenho de classificadores (matriz de confusão – veja Seção 3.3.2).

Inicialmente, é preciso inicializar o pacote e carregar a base de dados *Pesquisa de Satisfação*, a qual deve estar armazenada em um arquivo do tipo .csv, para fins de execução, conforme explicado aqui:

```
> install.packages("RSNNS")
> library("RSNNS")
> d <-
read.table("C:\\Users\\LivroDM\\Predicao\\pesquisa_satisfacao.csv"
header=TRUE,sep=";")
> X <- d[2:3]
> Y <- decodeClassLabels(d[,4])
```

Então, conjuntos de treinamento e de teste devem ser criados. Para tanto, use a função `splitForTrainingAndTest( )`,que recebe como parâmetros a variável com os exemplares, a variável com os rótulos e a proporção do conjunto de dados que deve compor o conjunto de teste.

```
> d_separado <- splitForTrainingAndTest(X, Y, ratio=0.10)
```

O conjunto de dados desse exemplo já se encontra normalizado; porém, caso não fosse esse o caso, o seguinte comando poderia ser aplicado para normalizar os dados:

```
> d_normalizado <- normTrainingAndTestSet(d_separado,
dontNormTargets = TRUE,
type = "0_1")
```

A função para treinamento chama-se `mlp( )` e é apresentada no Quadro 3-2. Para finalidade de exemplificação, considere o treinamento da rede com a base de dados preparada, conforme explicação anterior, e com uso de todos os parâmetros expostos no Quadro 3-2:

```
> modelo_mlp <- mlp(d_separado$inputsTrain,
d_separado$targetsTrain, size=c(3),
maxit=5000, initFunc="Randomize_Weights",
learnFunc="Std_Backpropagation", learnFuncParams=c(0.1),
hiddenActFunc="Act_Logistic",
shufflePatterns=TRUE, linOut=TRUE,
inputsTest=d_separado$inputsTest,
targetsTest=d_separado$targetsTest
)
```

Quadro 3-2: Sintaxe e parâmetros da função *MLP*

Sintaxe para a aplicação da função que implementa o treinamento de uma rede neural *Multilayer Perceptron*
Usando a função *mlp( )* do pacote *RSNNS*
Fazendo o treinamento da rede neural artificial *Multilayer Perceptron* `modelo_mlp <- mlp(x, y, size, maxit, initFunc, learnFunc, learnFuncParams, hiddenActFunc, shufflePatterns, linOut, inputsTest, targetsTest)` • `x` exemplares do conjunto de treinamento – atributos descritivos. • `y` rótulos dos exemplares do conjunto de treinamento. • `size` número de neurônios na camada escondida. Pode ser definido como um vetor para o caso de mais de uma camada escondida. O valor numérico e escalar define o número de neurônios na camada escondida. • `maxit` número máximo de iterações (épocas). • `initFunc` define como os pesos devem ser inicializados (opções: `Randomize_Weights` ou a definição de um vetor de valores desejados para a iniciação). • `learnFunc` estabelece o algoritmo de aprendizado (opções: `Std_Backpropagation; BackpropBatch; Quickprop`).

- `learnFuncParams` valor para a taxa de aprendizado.
- `hiddenActFunc` define o tipo de função de ativação do neurônios da camada escondida (`Act_Logistic`).
- `shufflePatterns` parâmetro para definir se os exemplares do conjunto de treinamento devem ou não ser embaralhados (opções `TRUE` ou `FALSE`) a cada época.
- `linOut` define qual deve ser a função de ativação para os neurônios da camada de saída, linear ou logística (opções: `TRUE` (para linear) ou `FALSE` (para logística)).
- `inputTest` refere-se aos padrões separados para teste da rede.
- `targetsTest` refere-se às saídas desejadas dos padrões de teste, usados apenas para aferir o desempenho da predição.

A função retorna a variável `modelo_mlp`, que contém um objeto com todas as variáveis definidas e ajustadas no processo de treinamento da rede neural artificial *Multilayer Perceptron.*

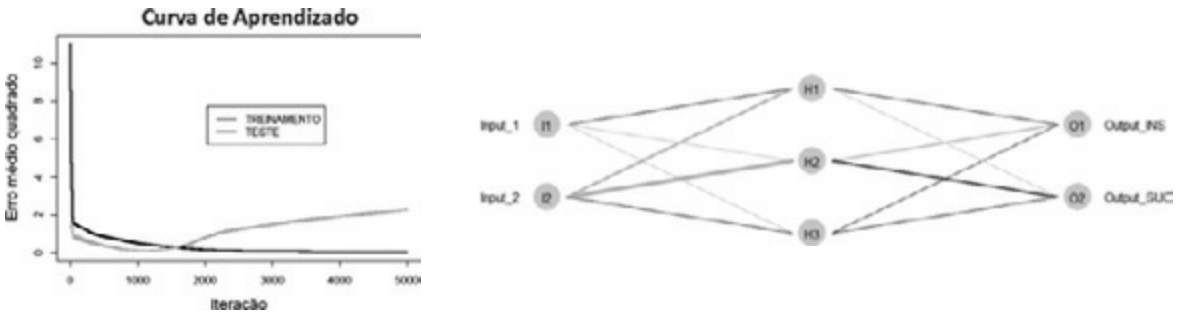
Neste exemplo, a quantidade de camadas escondidas e o número de neurônios foram escolhas arbitrárias; no entanto, é preciso fazer experimentos com outros valores para encontrar a melhor parametrização para a arquitetura da rede neural. O resultado do treinamento, o `modelo_mlp`, é um objeto que resume a parametrização escolhida para o treinamento da rede e detalhes referentes ao processo de treinamento da mesma. Observe tais detalhes no *script:*

```
> modelo_mlp
Class: mlp->rsnns
Number of inputs: 2
Number of outputs: 2
Maximal iterations: 5000
Initialization function: Randomize_Weights
Initialization function parameters: -0.3 0.3
Learning function: Std_Backpropagation
Learning function parameters: 0.1
Update function: Topological_Order
Update function parameters: 0
Patterns are shuffled internally: TRUE
Compute error in every iteration: TRUE
Architecture Parameters:
$size
[1] 3

All members of model:
[1]"nInputs" "maxit" "initFunc" "initFuncParams"
[5]"learnFunc" "learnFuncParams" "updateFunc" "updateFuncParams"
```

```
[9]“shufflePatterns” “computeIterativeError” “snnsObject”
“archParams”
[13]“IterativeFitError” “IterativeTestError” “fitted.values”
“fittedTestValues”
[17]“nOutputs”
```

Veja, a seguir, um *script* em `R` para a plotagem da curva de aprendizado (Figura 3-16(a)). Para construir a curva, é necessário acessar os valores nas variáveis `IterativeFitError` e `IterativeTestError`, que representam, respectivamente, o erro médio quadrado a cada iteração para o conjunto de treinamento e para o conjunto de teste. Acesse a Seção 3.3.2 para mais detalhes sobre esta medida de erro e também sobre o significado das curvas da Figura 3-16. As curvas obtidas nessa execução não são necessariamente as mesmas obtidas em outra execução, devido à aleatoriedade da inicialização de pesos.

```
> plot(modelo_mlp$IterativeFitError,type=“n”,main=“Curva de
Aprendizagem”,
xlab=“Iteração”,ylab=“Erro médio quadrado”,cex.lab=1.5,
ylim=c(0,10))
> lines(modelo_mlp$IterativeFitError,col=“1”,lwd=3,cex=2)
> lines(modelo_mlp$IterativeTestError,col=“2”,lwd=3)
>
legend(locator(1),c(“TREINAMENTO”,“TESTE”),lty=c(1,1),col=c(1,2))
```



(a) (b)

Figura 3-16: (a) Curvas de aprendizagem obtidas durante o treinamento da rede neural artificial *Multilayer Perceptron;* (b) arquitetura da rede neural resultante do treinamento: a largura das arestas é proporcional aos valores dos pesos

Outra informação interessante para análise durante ou após o treinamento da rede neural é o conjunto de valores assumidos pelos pesos sinápticos. Para obtê-los, acesse a variável `modelo_mlp$weightMatrix`. Por fim, a título de ilustração, sugere-se o uso da função `plot.nnet( )`, a qual requer a instalação dos pacotes **devtools** (Wickham e Chang, 2015), **scales** (Wickham, 2015) e **reshape** (Wickham, 2007), que permitem imprimir um gráfico como o da Figura 3-16(b), mostrando a arquitetura final da rede neural. Para isso, use `plot.nnet(modelo_mlp)`.

3.1.3. Árvores de decisão

Árvore de decisão é uma das técnicas mais populares de mineração de dados. Mais comumente usada para resolver a tarefa de classificação de dados, a árvore de decisão consiste em uma coleção de nós internos e nós folhas, organizados em um modelo hierárquico (da mesma forma que se organizam as estruturas de dados do tipo árvore). No contexto da resolução da tarefa de classificação, uma árvore de decisão representa o modelo capaz de guiar a tomada de decisão sobre a determinação da classe à qual um exemplar pertence.

A Figura 3-17 traz um exemplo de um modelo classificador hipotético, na forma de uma árvore de decisão, para escolha de pratos em um restaurante. Os nós internos da árvore (temperatura e refeição) dizem respeito a atributos descritivos de uma ocasião em que alguém está no restaurante para realizar uma refeição; de cada nó interno, saem subárvores relacionadas com os valores que o atributo de seu nó raiz pode assumir; cada nó folha (tipo de prato) representa uma decisão sugerida pelo modelo classificador para a ocasião, sugestão esta, presente no atributo rótulo do conjunto de dados usado para induzir o modelo. Assim, essa árvore de decisão está classificando as ocasiões em: apropriadas para consumo de pratos quentes ou apropriadas para consumo de pratos frios. Segundo o modelo representado pela árvore, se

a ocasião se refere à temperatura alta e à hora do jantar, deve-se optar por um prato frio.

A construção desse modelo, ou seja, a construção da árvore, é realizada por meio de um algoritmo que iterativamente analisa os atributos descritivos de um conjunto de dados previamente rotulado, constituindo o processo de aprendizado do modelo classificador. O Algoritmo 3-6 ilustra o processo básico de construção de uma árvore de decisão.

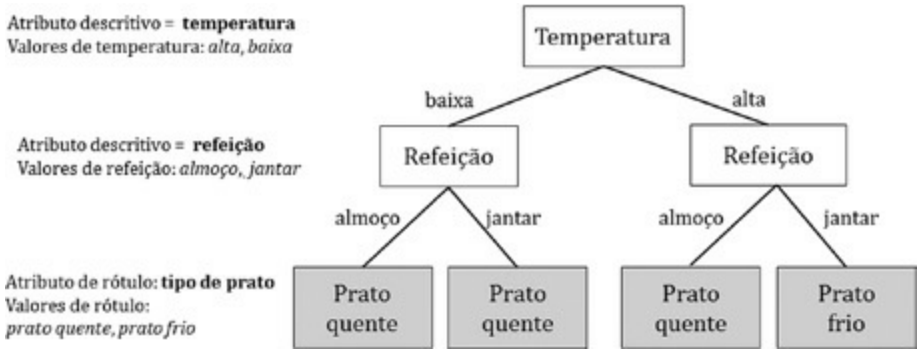


Figura 3-17: Exemplo de um classificador hipotético, na forma de uma árvore de decisão

Algoritmo 3-6: Algoritmo básico para construção de uma árvore de decisão

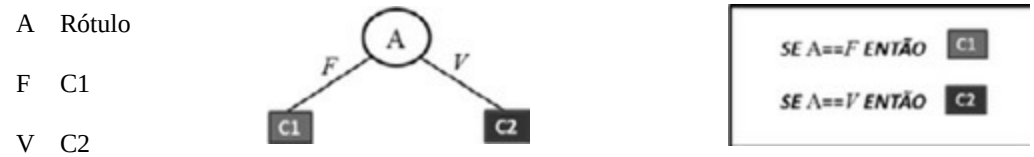
Parâmetros de entrada:
- $\mathbf{X}_{tr}$: um conjunto de exemplares rotulados de treinamento, ou seja, $\mathbf{X}$tr = $\{(\boldsymbol{x}_{\rightarrow i}, y_i)\}$, $i = 1, ..., n$;
- *L:* lista de *d* atributos descritivos de um exemplar em $\mathbf{X}_{\text{tr}}$;

Parâmetro de saída:
- estrutura de dados hierárquica organizada como um classificador (a árvore de decisão);

Função árvore de decisão **($\mathbf{X}_{tr}$, *L*);**
Passo 1: crie um nó *N*;
Passo 2: **se** todos os exemplares $\boldsymbol{x}_{\rightarrow i}$ em $\mathbf{X}_{tr}$ possuem o mesmo valor para o atributo rótulo y_i **então**
Passo 2.1: **retorne** *N* como um nó folha rotulado com a classe y_i;
Passo 3: **se** *L* é vazia **então**
Passo 3.1: **retorne** *N* como um nó folha rotulado com a classe y_i mais frequente em $\mathbf{X}_{tr}$;
Passo 4: **senão**
Passo 4.1: *A* = primeiro atributo descritivo em *L*;
Passo 4.2: associe *A* ao nó *N*;
Passo 4.3: *L* = *L* – *A*;
Passo 4.4: **para** cada valor distinto *v* do atributo *A*, considerando os exemplares em $\mathbf{X}_{tr}$ **faça**
Passo 4.4.1: $\mathbf{X}_{tr-v}$ = subconjunto de dados composto por exemplares em que *A* = *v*;

Passo 4.4.2: **se** $\mathbf{X}_{tr-v}$ é vazio **então**
Passo 4.4.2.1: **retorne** *N* como um nó folha, rotulado com a classe y_i mais frequente em $\mathbf{X}_{tr-v}$;
Passo 4.4.3: **senão**
Passo 4.4.3.1: associe a *N* uma subárvore retornada por árvore de decisão **(**$\mathbf{X}_{tr-v}$, *L***);**
Passo 4.5: **retorne** N;

A fim de compreender melhor o Algoritmo 3-6, considere um conjunto de dados simples (Figura 3-18(a)), com dois exemplares ($\boldsymbol{x}_{\rightarrow 1}$ e $\boldsymbol{x}_{\rightarrow 2}$), descritos por um único atributo descritivo *A* que pode assumir dois valores (V ou F), e um atributo de rótulo com duas classes distintas (C1 e C2). Seguindo os passos do algoritmo, a lista *L* será formada pelo único atributo descritivo, e o *Passo 4.4* será executado apenas uma vez. Como o atributo *A* tem dois valores possíveis, duas chamadas recursivas serão feitas no *Passo 4.4.3.1*. O caso-base da recursão implementado no *Passo 2* será executado para as duas chamadas. A árvore resultante desta execução é mostrada na Figura 3-18(b).



(a) (b) (c)

Figura 3-18: Exemplo de árvore de decisão (b) gerada a partir da execução do Algoritmo 3-6 para o conjunto de dados simples (a). Também são apresentas duas regras SE ENTÃO (c), derivadas da árvore de decisão

A árvore de decisão é um modelo que pode ser interpretado como regras *SE ENTÃO*. Na ilustração da Figura 3-18(c), é possível observar que duas regras foram derivadas da árvore de decisão. Cada uma das regras derivadas da árvore é, na realidade, uma série de condições verificadas sobre os atributos que compõem os nós internos da árvore (premissas) que levam a uma tomada de decisão, representada nos nós folhas da árvore (conclusões). Cada caminho percorrido em profundidade na árvore de decisão gera uma

regra SE ENTÃO ; ou alternativamente, regras do tipo SE ENTÃO SENÃO podem ser projetadas para cada subárvore do nó raiz.

Neste ponto, já é possível perceber como se dá a aplicação da árvore de decisão como um modelo de classificação. A partir do recebimento de um exemplar de teste, a árvore de decisão deve ser percorrida da raiz até uma de suas folhas. A cada nó interno, o atributo descritivo associado deve ser verificado no exemplar de teste, e, dependendo do valor de tal atributo, uma das subárvores do nó será escolhida. Então, o processo se repete até que o nó folha, com o rótulo atribuído ao exemplar de teste, seja alcançado.

A popularidade do uso de árvores de decisão como apoio à tomada de decisão e descoberta de conhecimento se deve principalmente ao fato de o conhecimento gerado e representado no modelo ser de fácil entendimento, e, assim, as tomadas de decisão podem ser claramente justificadas.

Uma árvore de decisão é naturalmente adequada para uso com atributos categóricos; porém, atributos numéricos podem também ser usados, desde que seja realizada uma transformação nos dados, de forma a criar intervalos em seu domínio de valores. A fim de estudar diferentes formas de tratar os valores dos atributos na construção de uma árvore de decisão, considere dois atributos descritivos de um funcionário do restaurante: “estado civil” e “pretensão salarial”. O atributo “estado civil” pode assumir três valores: “viúvo”, “solteiro” ou “casado”. Seguindo a estratégia apresentada no Algoritmo 3-6, três ramos devem ser criados a partir de um nó associado a esse atributo. Entretanto, estratégias alternativas podem ser usadas, se forem vantajosas para a resolução do problema de classificação. Na Figura 3-19, são mostradas quatro formas diferentes de tratar esse atributo, geradas seguindo o uso de três ramos (a) ou fazendo agrupamentos de valores (b). No caso da escolha por agrupamento de valores, a premissa da regra SE ENTÃO, a ser derivada a partir de um ramo, deve incluir o uso da operação lógica conjuntiva OU (SE `estado civil =viúvo` OU `estado civil = casado` ... ENTÃO ...).

(a) (b)

Figura 3-19: Exemplos de tratamento para atributos do tipo categórico cujo domínio inclui mais de dois valores

Um procedimento semelhante ao aplicado com o atributo categórico pode tratar um atributo do tipo numérico. O atributo “pretensão salarial” deve ser tratado a partir da criação de intervalos. Esses intervalos devem também ser criados de maneira a maximizar os benefícios na construção dos classificadores, e uma medida apropriada, por exemplo, a medida de entropia, pode ser usada para esse fim (veja uma explicação sobre isso em Han, Kamber e Pei, 2011). A Figura 3-20 mostra dois exemplos de como os intervalos podem ser criados: (a) usando um valor único que divide o domínio do atributo em duas partes ou (b) usando faixas de valores.

(a) (b)

Figura 3-20: Exemplo de tratamento para atributos do tipo numérico

O exemplo usado para ilustrar o Algoritmo 3-6 é bastante simples. O algoritmo, no entanto, permite a resolução de problemas mais complexos com a inserção do uso de um critério que permita escolher, de maneira mais eficiente, o atributo associado a um nó interno da árvore. Uma pequena modificação no *Passo 4.1* do Algoritmo 3-6, apresentado na Figura 3-21, implementa esse critério.

[...

Passo 4.1: A = **função *seleção_de_atributo*** ($\mathbf{X}_{tr}$, *L*);

...]

Figura 3-21: Alteração do *Passo 4.1* do Algoritmo 3-6 para permitir o uso de um critério de seleção de atributo para um nó interno da árvore de decisão

Assim, de acordo com a modificação apresentada, em cada iteração do algoritmo, é necessário aplicar uma função aos atributos disponíveis na lista *L*, de forma que a saída da função seja a indicação de um atributo que, naquele momento, é o mais adequado para ser associado ao nó da árvore. Essas funções são baseadas em critérios que avaliam o quão bom cada atributo é no papel de dividir o conjunto de dados, considerando a distribuição de classes presentes no conjunto. Rokack e Maimon (2008) apresentam uma série de critérios de divisão de atributos. Um resumo desses critérios é mostrado no Quadro 3-3. O conceito de impureza usado em vários critérios de escolha de atributos, a medida de entropia da Teoria da Informação, e sua aplicação para elaboração do critério de ganho de informação, são discutidos com mais detalhes na sequência deste texto.

Quadro 3-3: Caracterização dos critérios de seleção de atributos, aplicáveis na construção de árvores de decisão, baseado em Rockack e Maimon (2008)

Critério	**Resumo**	**Referência**
Ganho de informação	Usa a medida de entropia, da Teoria da Informação, como meio para análise do grau de impureza das partições geradas a partir da análise dos valores de um atributo descritivo.	(Quinlan, 1987)
Índice Gini	Usa um critério baseado em impureza para analisar as diferenças entre as distribuições de probabilidade dos valores dos atributos de classe (rótulos).	(Breiman *et al*., 1984) (Gelfand *et al*., 1991)
Likelihood Ration Chi-Squared Statistics	Mede a significância estatística do critério de ganho de informação.	(Attneave, 1959)
DKM	Critério baseado em impureza, especialmente projetado para atributos de classes binárias. Gera árvores menores que o ganho da informação e o índice Gini.	(Dietterich, Kearns e Mansour, 1996) (Kearns e Mansour, 1999)
Raio do ganho	Normaliza o critério de ganho de informação. Deve ser usado para selecionar atributos que obtiveram um ganho de informação médio.	(Quinlan, 1988)

Twoing	Alternativa ao índice Gini, melhorando o desempenho quando o atributo de classe tem um domínio grande.	(Breiman *et al.*, 1984)
ORT	Critério binário baseado no cálculo do ângulo entre vetores que representam a distribuição de classes nas partições geradas.	(Fayyad e Irani, 1992)
Kolmogorov_Smirnov	Critério binário baseado na distância Kolmogorov_Smirnov. Possui uma versão estendida para tratar múltiplas classes e dados faltantes.	(Friedman, 1977) (Rounds, 1980) (Utgoff e Clouse, 1996)
AUC *Splitting*	Baseado na área sob a curva ROC (Seção 3.3.2) referente ao uso de cada um dos atributos descritivos.	(Ferri *et al.*, 2002)

No contexto da tarefa de classificação, o conceito de **impureza** está relacionado com a variabilidade de classes presentes em uma partição do conjunto de dados. Quanto mais classes diferentes existirem na partição, mais impura ela será. Uma partição pura é aquela que só contém exemplares de uma única classe. Formalmente, no contexto de indução de um modelo de classificação como uma árvore de decisão, considerando um conjunto de dados de treinamento $\mathbf{X}_{tr}$, uma medida de impureza para esse conjunto pode ser representada por um vetor de probabilidades $\mathbf{p}\rightarrow$ do atributo de classe *y* (Rockack e Maimon, 2008). Assim:

$$\vec{\mathbf{p}}_y(\mathbf{X}_{tr}) = \left(\frac{|\sigma_{y=c_1}\mathbf{X}_{tr}|}{|\mathbf{X}_{tr}|}, \ldots, \frac{|\sigma_{y=c_K}\mathbf{X}_{tr}|}{|\mathbf{X}_{tr}|}, \ldots, \frac{|\sigma_{y=c_K}\mathbf{X}_{tr}|}{|\mathbf{X}_{tr}|} \right) \qquad \textbf{Equação 3-4}$$

em que $|\cdot|$ é a cardinalidade de um conjunto, σ_θ é um operador de seleção, e θ é o predicado de seleção de tuplas (nesse contexto, exemplares), definido sobre o atributo de classe *y*, cujo domínio é formado por *K* classes distintas.

A partir da definição de uma medida de impureza, é possível perceber que, se o vetor $\mathbf{p}\rightarrow_y(\mathbf{X}\text{tr})$ é composto por uma única componente de valor 1, significa que só há uma classe no conjunto de treinamento. Pode-se dizer, portanto, que esse conjunto de treinamento é uma partição pura. O nível de impureza de uma partição é máximo quando todas as componentes do vetor de probabilidades são iguais. Por exemplo, se $\mathbf{p}\rightarrow_y(\mathbf{X}\text{tr})$ = (0,25, 0,25, 0,25,

0,25) e |**X**tr| = 100, então o conjunto de dados de treinamento possui quatro classes distintas, K = 4, com 25 exemplares associados a cada classe. Os exemplares estão igualmente distribuídos em cada uma das classes existentes.

Ainda, outra forma de medir a impureza de uma partição de dados é usando a medida de entropia. A **entropia** pode ser vista, de maneira simplificada, como uma forma de medir a desordem em um sistema fechado. No caso do contexto de classificação de dados, pode-se usar a medida de entropia para medir a impureza de uma partição de dados: quanto mais desorganizada ela estiver em relação às classes, ou seja, quanto mais exemplares de classes diferentes nela existirem, com probabilidades parecidas, mais desorganizada e impura ela é e mais alta é sua entropia. A Figura 3-22 ilustra duas situações: (a) uma situação de alta entropia; (b) uma situação de baixa entropia.

Figura 3-22: Entropia *versus* desordem: (a) mistura desorganizada e com alta entropia; (b) mistura sem desordem e com baixa entropia

Para medir a entropia de uma partição de dados, considere novamente o contexto de indução de um modelo de classificação e o conjunto de dados de treinamento **X**tr. A entropia é dada por:

$$Entropia(\mathbf{X}_{tr}) = -\sum_{c_k=1}^{K} p_{c_k} \log_2(p_{c_k})$$

Equação 3-5

em que p_{ck} é a probabilidade da classe c_k ocorrer em **X**tr determinada como $\frac{|\sigma_{y=c_k}\mathbf{X}_{tr}|}{|\mathbf{X}_{tr}|}$.

Quanto maior for a entropia, maior será a desordem. Assim, é possível concluir que, quanto maior for a entropia de uma partição de dados, mais esforço será necessário para organizá-la (determinar quais dados pertencem a quais classes). Já é possível antever que a medida de entropia pode ajudar a decidir qual atributo do conjunto de dados é melhor para ser usado como guia para a criação de partições.

A fim de verificar esse fato, considere um conjunto de dados que possui o atributo *A*, e que esse atributo pode assumir *v* valores distintos. Se esse atributo for usado para particionar o conjunto de dados, *v* partições serão geradas ($\mathbf{X}_{tr\_1}$,$\mathbf{X}_{tr\_2}$, ..., $\mathbf{X}_{tr\_v}$). A partição $\mathbf{X}_{tr\_1}$ conterá apenas os exemplares do conjunto de dados que possuem o valor $v = 1$ para o atributo *A*; a partição $\mathbf{X}_{tr\_2}$ conterá apenas os exemplares do conjunto de dados que possuem o valor $v = 2$ para o atributo *A* e assim por diante. Para saber o quão bom é esse atributo para a criação de partições no contexto da resolução da tarefa de classificação, é útil verificar o quão pura é cada uma das partições geradas. Essa verificação pode ser feita por meio do cálculo da entropia de cada uma das partições: *Entropia*($\mathbf{X}_{tr\_1}$), *Entropia*($\mathbf{X}_{tr\_2}$), … *Entropia*($\mathbf{X}_{tr\_v}$).

Se todas as medidas de entropia resultarem em 0, todas as partições serão consideradas puras, e saberemos que o atributo *A* é bom para ser usado como critério para particionar os dados, já que as partições que ele gera possuem, cada uma, exemplares de uma única classe. Na indução de uma árvore de decisão, o uso desse atributo levaria à finalização da construção do modelo classificador. Se algumas das medidas de entropia resultarem em valores maiores que 0, é sinal de que as partições contêm exemplares de classes diferentes. Dessa análise, deriva o conceito de “informação necessária ” (do inglês, *expected information requirements*). A “informação necessária”

expressa a quantidade de esforço ainda necessário para chegar a partições puras a partir do conjunto de partições atuais. Ela pode ser calculada como:

$$Info.Necessária_A(\mathbf{X}_{tr}) = \frac{|\mathbf{X}_{tr\_1}|}{\mathbf{X}_{tr}} Entropia(\mathbf{X}_{tr\_1}) + \frac{|\mathbf{X}_{tr\_2}|}{\mathbf{X}_{tr}} Entropia(\mathbf{X}_{tr\_2}) + \ldots + \frac{|\mathbf{X}_{tr\_v}|}{|\mathbf{X}_{tr}|} Entropia(\mathbf{X}_{tr\_v}) \quad \textbf{Equação 3-6}$$

Enfim, com base na "informação necessária", define-se a medida de **ganho de informação** como:

$$Ganho(A) = Info.Necessária(\mathbf{X}_{tr}) - Info.Necessária_A(\mathbf{X}_{tr}) \quad \textbf{Equação 3-7}$$

em que $Info.Necessária(\mathbf{X}_{tr})$ é a entropia de $\mathbf{X}_{tr}$. Essa medida pode ser interpretada como o ganho de informação obtido ao usar o atributo *A* para particionar o conjunto $\mathbf{X}_{tr}$. O atributo de maior ganho de informação deve ser o selecionado.

Um modelo classificador na forma de uma árvore de decisão, como visto, pode ser entendido como uma série de regras SE ENTÃO. Quanto maior a árvore de decisão gerada, maior o número de regras que derivam dela. Uma árvore com um grande número de nós resulta em um modelo com um grande número de comparações, de forma que seu poder de generalização pode ficar prejudicado (veja a definição de sobreajuste na Seção 3.3.1). Uma forma de melhorar essa limitação é executando procedimentos de poda da árvore.

Estratégias de poda têm como objetivo a eliminação de ramos da árvore que podem ter sido gerados por ocasião da existência de dados ruidosos ou mesmo por consequência do fenômeno de sobreajuste , o que implicará a geração de um modelo pouco confiável para classificação de novos exemplares. Uma forma de analisar esse fato é pensar que, quanto mais profundo for um nó, menor será o número de exemplares que chegam até ele. Com a aplicação de estratégias de poda, é possível eliminar alguns ramos da

árvore, permitindo melhorar sua capacidade de generalização. No entanto, as estratégias para executar essa tarefa precisam também se preocupar em não aumentar demais o erro de generalização (Figura 3-1). As estratégias são definidas como pré-poda ou pós-poda, e há muitas delas disponíveis na literatura especializada.

Na literatura, os tipos de atributos e as métricas de seleção de atributos para compor a árvore de decisão definem a taxonomia de algoritmos para indução da árvore de decisão. Por exemplo, o algoritmo mais conhecido é o algoritmo ID3 (Quinlan, 1986), que usa a medida de ganho de informação, não executa poda na árvore e não foi concebido originalmente para tratar atributos numéricos. Já o algoritmo C4.5 (Quinlan, 1993), por exemplo, lida com atributos do tipo categório e numérico, apli- ca um procedimento de poda e utiliza o raio do ganho de informação na seleção dos atributos que compõem os nós da árvore. Outro exemplo é o algoritmo de árvore de decisão conhecido como CART (*Classification and Regression Trees*) (Breiman, 1984), que lida com todos os tipos de atributos, realiza podas, porém, produz apenas árvores binárias, utilizando, originalmente, a medida Twoing para seleção de atributos.

3.1.3.1. Exemplo didático para o algoritmo árvore de decisão

Para exemplificar a execução do Algoritmo 3-6, considere o conjunto de dados *Qualidade de Serviço* (Tabela 3-7). Cada atributo descritivo ("experiência dos profissionais" – EP, "qualidade da refeição" – QR, "localização do estabelecimento" – LE) dos exemplares do conjunto de dados pode assumir um de dois valores.

Tabela 3-7: Conjunto de dados *QUALIDADE DE SERVIÇO*

	x_{i1}	x_{i2}	x_{i3}	y_i
	EP	**QR**	**LE**	**R**
$\mathbf{x}_{\rightarrow 1}$	MUITA	BOA	RUIM	Lucro
$\mathbf{x}_{\rightarrow 2}$	MUITA	BOA	BOA	Lucro
$\mathbf{x}_{\rightarrow 3}$	POUCA	BOA	BOA	Lucro
$\mathbf{x}_{\rightarrow 4}$	POUCA	ÓTIMA	RUIM	Prejuízo
$\mathbf{x}_{\rightarrow 5}$	POUCA	ÓTIMA	BOA	Prejuízo
$\mathbf{x}_{\rightarrow 6}$	MUITA	ÓTIMA	BOA	Lucro
$\mathbf{x}_{\rightarrow 7}$	MUITA	ÓTIMA	RUIM	Prejuízo
$\mathbf{x}_{\rightarrow 8}$	POUCA	BOA	RUIM	Prejuízo

A Figura 3-23 apresenta os detalhes de execução do Algoritmo 3-6 sobre o conjunto de dados *Qualidade de Serviço*, para construção parcial da subárvore mais à esquerda do modelo classificador. A árvore de decisão para esse conjunto de dados terá o atributo EP em seu nó inicial (ou raiz), por ser o primeiro atributo descritivo que aparece no conjunto de dados (*Passo 4.1* e *4.2* do Algoritmo 3-6). Esse atributo pode assumir os valores "MUITA" ou "POUCA", os quais foram arbitrariamente associados como ramo esquerdo e

ramo direito da árvore, respectivamente, a partir das chamadas recursivas da função árvore de decisão (*Passo 4.4.3.1*).

Função árvore de decisão (Xtr, L);				
	Chamada da função: 1	**Chamada da função: 1.1**	**Chamada da função: 1.1.1**	**Chamada da função: 1.1.1.1**
Passos	X_{tr} = {$\boldsymbol{x}\rightarrow_1$, $\boldsymbol{x}\rightarrow_2$, $\boldsymbol{x}\rightarrow_6$, $\boldsymbol{x}\rightarrow_4$, $\boldsymbol{x}\rightarrow_5$, $\boldsymbol{x}\rightarrow_6$, $\boldsymbol{x}\rightarrow_7$, $\boldsymbol{x}\rightarrow_8$} *L*= {EP,QR,LE}	$X_{tr\_EP=MUITA}$ = {$\boldsymbol{x}\rightarrow_1$, $\boldsymbol{x}\rightarrow_2$, $\boldsymbol{x}\rightarrow_6$, $\boldsymbol{x}\rightarrow_7$} *L*= {QR,LE}	$X_{tr\_EP=MUITA,QR=ÓTIMA}$ = {$\boldsymbol{x}\rightarrow_6$, $\boldsymbol{x}\rightarrow_7$} *L*={LE}	$X_{tr\_EP=MUITA,QR=ÓTIMA\ LE=RUIM}$ = {$\boldsymbol{x}\rightarrow_7$} *L*={ }
1	nó *N*	*N* = EP(MUITA)	*N* = EP (MUITA) → QR (ÓTIMA)	*N* = EP(MUITA) → QR (ÓTIMA) → LE(RUIM)
2	condição = falso	condição = falso	condição = falso	*N* = EP(MUITA) → QR (ÓTIMA) → LE(RUIM) → PREJUÍZO
3	condição = falso	condição = falso	condição = falso	
4.1	A={EP}	A={QR}	A={LE}	
4.2	*N* = EP (nó raíz)	*N* = EP(MUITA) → QR	*N* = EP(MUITA) → R(ÓTIMA) → LE	
4.3	*L*={QR,LE}	*L*={LE}	*L*={ }	
4.4	EP_v1 = MUITA EP_v2 = POUCA	QR_v1 = ÓTIMA QR_v2 = BOA	LE_v1 = RUIM LE_v2 = BOA	
4.4.1	$X_{tr\_EP=MUITA}$ = {$\boldsymbol{x}\rightarrow_1$, $\boldsymbol{x}\rightarrow_2$, $\boldsymbol{x}\rightarrow_6$, $\boldsymbol{x}\rightarrow_7$} $X_{tr\_EP=POUCA}$ = {$\boldsymbol{x}\rightarrow_3$, $\boldsymbol{x}\rightarrow_4$, $\boldsymbol{x}\rightarrow_5$, $\boldsymbol{x}\rightarrow_8$}	$X_{tr\_EP=MUITA,QR=ÓTIMA}$ ={$\boldsymbol{x}\rightarrow_6$, $\boldsymbol{x}\rightarrow_7$} $X_{tr\_EP=MUITA,QR=BOA}$ = {$\boldsymbol{x}\rightarrow_1$, $\boldsymbol{x}\rightarrow_2$}	$X_{tr\_EP=MUITA,QR=ÓTIMA,LE=RUIM}$ = {$\boldsymbol{x}\rightarrow_7$}	
4.4.2	condição = falso condição = falso	condição = falso condição = falso	condição = falso	
4.4.3.1	*EP* (subárvore esquerda: MUITA) *EP* (subárvore	*EP*,*QR* (subárvore esquerda: ÓTIMA) *EP*,*QR* (subárvore direita:	*EP*,*QR*,*LE* (subárvore esquerda: RUIM)	

direita: BOA)
POUCA)

Figura 3-23: Construção parcial da subárvore mais à esquerda da solução obtida com a execução do Algoritmo 3-6 no conjunto de dados *QUALIDADE DE SERVIÇO*

A continuação da construção da árvore exige que chamadas recursivas empilhadas sejam também executadas. Assim, as chamadas `EP,QR(subárvore direita: BOA)` e `EP(subárvore direita: POUCA)` devem ser executadas, e ambas podem incluir novas chamadas recursivas na pilha. O processo completo de construção dessa árvore pode ser obtido a partir da execução da implementação descrita na próxima seção.

É importante lembrar que a seleção do nó que comporá a árvore em determinado momento da execução do algoritmo de construção pode ser diferente do exposto no exemplo anterior, se outro critério de seleção de atributo for aplicado.

A árvore completa é o modelo de classificação construído para predizer se, a partir da qualidade do serviço prestado em determinada ocasião, o restaurante terá lucro ou prejuízo. A construção desse modelo permitiu que o conhecimento escondido no conjunto de dados (experiências passadas ocorridas no restaurante) fosse evidenciado na estrutura hierárquica. Agora, a partir das regras SE ENTÃO, que podem ser extraídas dessa árvore, uma pessoa que deseja abrir um restaurante pode se beneficiar do conhecimento prévio dos requisitos necessários para que se tenha lucro no empreendimento; ou, ainda, um consultor pode usar esse modelo para mostrar ao dono do estabelecimento de onde as experiências organizadas no conjunto de dados vieram e quais são os fatores que levam seu empreendimento a ter prejuízo.

3.1.3.2. Exemplo prático para árvore de decisão em R

Há uma série de pacotes que implementam árvores de decisão em R. Adicionalmente, a árvore resultante como modelo classificador também pode ser interpretada de forma gráfica. Portanto, há também pacotes específicos

para a visualização do resultado. Para o exemplo discutido aqui, será usada a função árvore de decisão chamada `rpart( )`, disponível no pacote ***rpart*** (Therneau, Atkinson e Ripley, 2014). O significado do nome da função `rpart( )` vem de *Recursive Partitioning and Regression Trees*, portanto, vê-se que há duas abordagens implementadas neste pacote: uma modelagem feita para classificação (como no Algoritmo 3-6) e outra específica para resolução de problemas de regressão (Seção 3.2).

A função `rpart( )`está apresentada em detalhes no Quadro 3-4. Ela é responsável por implementar a fase de indução do modelo. A função responsável pela predição, `predict( )`, a partir da apresentação de um exemplar de rótulo desconhecido, é apresentada no Quadro 3-5.

Quadro 3-4: Sintaxe e parâmetros da função RPART

Sintaxe para aplicação da função que constrói uma árvore de decisão
Usando a função `rpart( )` do pacote *rpart*
Construindo uma árvore de decisão `modelo_ad <- rpart(fórmula, dados, método, controle, parâmetros)` • `fórmula` define qual é o atributo de classe e quais são os atributos descritivos do conjunto de dados. Por exemplo: $y \sim x_{i1} + x_{i2} + x_{i3}$. • `dados` é um conjunto de dados armazenado como um ***data.frame.*** • `método` é o parâmetro que define se árvore será usada como classificação (“*class*”) ou regressão (“*anova*”). • `controle` é a variável na qual são armazenados os parâmetros que controlam o crescimento da árvore (veja (Therneau, Atkinson e Ripley, 2014)). • `parâmetros` é a variável na qual é definido o critério de seleção de atributos. A função retorna a variável `modelo_ad` com um objeto de árvore de decisão.

Quadro 3-5: Sintaxe e parâmetros da função PREDICT

Sintaxe para a aplicação da função que classifica um exemplar a partir de uma árvore de decisão
Usando a função `predict( )` do pacote *rpart*
Usando uma árvore de decisão `y_estimado <- predict(modelo_ad, exemplares_teste, tipo)` • `modelo_ad` é o resultado da construção da árvore de decisão, retornado pela função **rpart( )**. • `exemplares_teste` é a variável que contém os atributos descritivos dos exemplares de teste, considerando os

mesmos atributos descritivos usados na geração de `modelo_ad`.

- `tipo` especifica como será apresentado o resultado da predição: um vetor para valores numéricos ("*vector*"), classe para valores categóricos ("*class*") ou a probabilidade de cada classe ("*prob*").

A função retorna uma variável `y_estimado` que contém os rótulos dos exemplares de teste.

Um exemplo do uso da função `rpart( )` é apresentado a seguir, considerando o conjunto de dados *Qualidade de Serviço*, predição categórica (*class*), quantidade mínima de exemplares em um nó para que ele possa ter subárvores (1) e critério para seleção de atributos baseado na medida de ganho de informação (*information*).

```
> install.packages("rpart")
> library("rpart")
> servico <-
read.table("C:\\Users\\LivroDM\\Predicao\\servico.csv",
header=TRUE,sep=";")
> modelo_ad <- rpart(R ~ EP + QR + LE, data = servico,
method = "class", control = rpart.control( minsplit = 1), parms =
list(split = "Information"))
```

É possível extrair informações básicas do modelo de predição gerado digitando o nome da variável que o armazena (`modelo_ad`), e o resultado apresentado, neste caso, é:

```
n = 8

node), split, n, loss, yval, (yprob)
* denotes terminal node

1) root 8 4 Lucro (0.5000000 0.5000000)
2) EP=MUITA 4 1 Lucro (0.7500000 0.2500000)
4) QR=BOA 2 0 Lucro (1.0000000 0.0000000) *
5) QR=ÓTIMA 2 1 Lucro (0.5000000 0.5000000)
10) LE=BOA 1 0 Lucro (1.0000000 0.0000000) *
11) LE=RUIM 1 0 Prejuízo (0.0000000 1.0000000) *
3) EP=POUCA 4 1 Prejuízo (0.2500000 0.7500000)
6) QR=BOA 2 1 Lucro (0.5000000 0.5000000)
12) LE=BOA 1 0 Lucro (1.0000000 0.0000000) *
13) LE=RUIM 1 0 Prejuízo (0.0000000 1.0000000) *
```

```
7) QR=ÓTIMA 2 0 Prejuízo (0.0000000 1.0000000) *
```

Para auxiliar na interpretação desses resultados do modelo, note que o valor de `n` é referente ao número de exemplares no conjunto de dados usado para induzir a árvore e na linha `node), split, n, loss, yval, (yprob)` cada item é um tipo de informação: `node)` refere-se ao nó analisado, sendo que as numerações seguem uma leitura "em largura" da árvore (repare que a indentação do resultado ajuda a observar a hierarquia da árvore); `split` é o par atributo/valor considerado na separação dos exemplares; `loss` é uma sequência numérica na qual o primeiro número é a quantidade de exemplares analisados no nó, e o segundo é a quantidade de exemplares que não pertencem à classe majoritária naquele nó; `yval` refere-se ao valor da classe (ou saída) predominante no nó em análise; `(yprob)`é a probabilidade de cada classe presente no nó em análise.

Por exemplo, o resultado `3) EP=POUCA 4 1 Prejuízo (0.2500000 0.7500000)`, significa que está sendo analisado o nó `3`), cujo atributo/valor é `EP=POUCA`; na sequência `4 1` indica que existem quatro exemplares chegando ao nó, e um deles não pertence à classe majoritária `Prejuízo`; desta sequência, a classe predominante é `Prejuízo`; e, por fim, a probabilidade de cada classe respeitando `(0.2500000 0.7500000)`, respectivamente.

Em caso de uma interpretação mais ampla, o resultado impresso na linha de comando anterior traz a apresentação do número de exemplares mapeados em cada nó da árvore. Pelo nó raiz, passam todos os exemplares do conjunto de dados de indução da árvore (oito exemplares). Desse total, quatro exemplares são mapeados para o nó `EP=MUITA` (nó 2) e quatro são mapeados para o nó `EP=POUCA` (nó 3). A indentação do resultado indica o nível em que cada nó está na árvore. E os "*" indicam quais nós são folhas.

Embora o resultado obtido no exemplo permita a interpretação do modelo preditor, uma compreensão mais detalhada do resultado pode ser obtida por meio de uma representação gráfica. Essa representação pode ser feita por diferentes bibliotecas do ambiente R, incluindo a ***rpart***. Neste exemplo, foi

usada a função `rpart.plot( )`, que permite a visualização de qualquer resultado do pacote `rpart( )`(Milborrow, 2014). No Quadro 3-6 é apresentada a função `rpart.plot( )`, e na Figura 3-24 é mostrado o resultado da representação gráfica gerada.

Quadro 3-6: Sintaxe e parâmetros da função RPART.PLOT

Sintaxe para a aplicação da função que gera uma representação gráfica para uma árvore de decisão
Usando a função `rpart.plot( )` do pacote *rpart.plot*
Visualizando uma árvore de decisão `plot <- rpart.plot(modelo_ad, dígitos, tipo)` • `modelo_ad` é o resultado da construção da árvore de decisão, retornado pela função **rpart( )**. • `dígitos` é a quantidade de casas decimais plotadas. • `tipo` especifica como será apresentado o resultado da visualização e pode assumir valores de 0 a 4 (veja Milborrow, 2014). A função retorna uma representação gráfica para a árvore de decisão armazenada em `modelo_ad`.

Este pacote precisa ser instalado usando o comando `install.packages("rpart.plot")`, e a biblioteca deve ser chamada com o comando `library("rpart.plot")` para a execução do experimento. O uso da função `rpart.plot( )` é da seguinte forma: `plot_modelo <- rpart.plot(modelo_ad, type = 3)`.

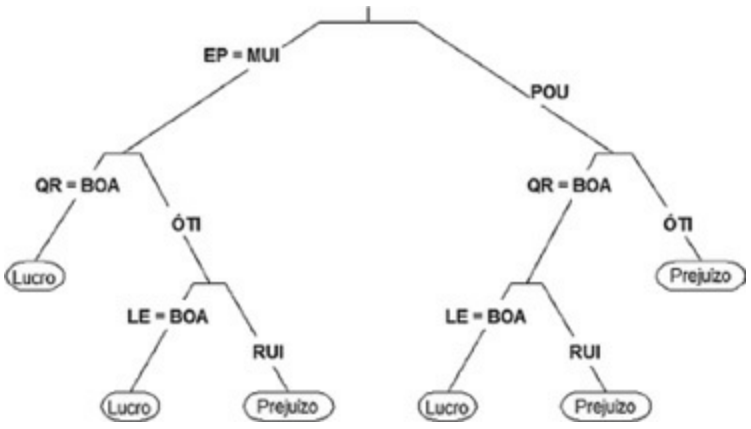


Figura 3-24: Representação gráfica para a árvore de decisão gerada sobre o conjunto de dados *QUALIDADE DE SERVIÇO*, usando a função RPART.PLOT

3.1.4. *Naïve* Bayes

O algoritmo de classificação ***Naïve* Bayes** , assim como árvore de decisão, é um dos mais utilizados em classificação de dados por apresentar bom desempenho em problemas de classificação quando usado tanto com dados categóricos quanto com dados numéricos. Trata-se de um classificador estatístico , baseado no Teorema de Bayes, de Thomas Bayes (Bellhouse, 2004), que recebe o nome de ingênuo (*naïve*) porque presume que, se o valor de um atributo exerce algum efeito sobre a distribuição de classes existente no conjunto, esse efeito é independente dos valores assumidos por outros atributos e de seus respectivos efeitos sobre a mesma distribuição de classes.

Em um processo de classificação no qual um exemplar com rótulo desconhecido seja apresentado ao classificador, o algoritmo *Naïve* Bayes tomará a decisão sobre a qual classe o exemplar $\boldsymbol{x}_{\rightarrow i} = \{x_{i1}, x_{i2}, ..., \boldsymbol{x}_{id}\}$ deve estar associado, por meio do cálculo de probabilidades condicionais, ou seja, as probabilidades de ele pertencer a cada uma das c_K classes (probabilidade da classe *K*, dado o exemplar $\mathbf{x}_{\rightarrow}$, $P\ (c_K|\mathbf{x}_{\rightarrow})$) existentes no conjunto de dados usado para treinamento, da seguinte maneira:

$$P(c_K|\vec{\mathbf{x}}_i) = P(c_K)\prod_{j=1}^{d} P(x_{ij}|c_K)$$

Equação 3-8

sendo que:

- $P(c_K)$ é a probabilidade da classe c_K no conjunto de dados de treinamento, ou seja, a razão entre número de exemplares pertencentes à classe c_K e o número total de exemplares no conjunto, dada por $|c_K|/|\mathbf{X}\text{tr}|$ em que $|\cdot|$ significa a cardinalidade do conjunto.
- $P\ (x_{ij}|c_K)$ é a probabilidade do valor de atributo x_{ij} dada a distribuição de classe c_k, ou seja, a razão do número de exemplares com valor de atributo igual a x_{ij} pelo número total de exemplares da classe c_K. Note

que, se o atributo j for categórico, $P\ (x_{ij}|c_K) = |(\sigma x \rightarrow_i [_j]=_{xij}\ (c_K))|/|c_K|$, em que $\mathbf{x} \rightarrow_i [j]$ significa o valor do exemplar $\mathbf{x} \rightarrow_i$ no atributo j. Por outro lado, caso o atributo seja de natureza contínua, então a probabilidade deve presumir uma distribuição de probabilidade Gaussiana (Han, Kamber e Pei, 2011).

A Equação 3-8, segundo o Teorema de Bayes, ainda deveria ser dividida por $P(\mathbf{x} \rightarrow)$, mas como essa probabilidade é constante para todas as classes, só é necessário analisar o numerador da razão. Então, a classificação de $\mathbf{x} \rightarrow$ será feita a partir da maximização $P(c_K) \prod^{d}_{j=1} P\ (x_{ij}|c_K)$, de forma que o exemplar será associado à classe de maior probabilidade, ou seja, o exemplar de teste será associado à classe p, se, e somente se, $P(c_p|\mathbf{x} \rightarrow) > P(c_q|\mathbf{x} \rightarrow)$ para $1 \leq p, q \leq$ K e $p \neq q$.

O Algoritmo 3-7 descreve os passos para implementação da classificação com *Naïve* Bayes. Os parâmetros de entrada são o conjunto de treinamento e o exemplar com o rótulo desconhecido. A análise do exemplar desconhecido é feita mediante o cálculo das probabilidades de cada classe, dados os valores dos atributos do exemplar (*Passo 2*). Para isso, encontram-se todos exemplares do conjunto de treinamento (*Passo 2.1.1*), respeitando o valor do atributo analisado. Em seguida (*Passos 2.2.2*, *2.2.3*), faz-se o cálculo das probabilidades das classes dado o exemplar $\mathbf{x} \rightarrow_{te}$ que precisa ser classificado (Equação 3-8). Note que, se o atributo for numérico, será necessário modificar, a forma como as probabilidades são calculadas. Ao final, a classe de maior probabilidade condicional é dada como resposta à classificação de $\mathbf{x} \rightarrow_{te}$, e $\hat{y}_{te}$ recebe o rótulo correspondente.

Algoritmo 3-7: Algoritmo para classificação *Naïve* Bayes

Parâmetros de entrada:

- $\mathbf{X}_{tr}$: um conjunto de treinamento com exemplares rotulados, $\mathbf{X}_{tr} = \{(\boldsymbol{x} \rightarrow_i, y_i)\}$, $i = 1, ..., n$;
- $\mathbf{X}_{te}$: um exemplar de teste com rótulo desconhecido, $\mathbf{X}_{te} = \{\mathbf{x} \rightarrow_{te}\}$;

Parâmetros de saída:
- $\hat{y}_{te}$: rótulo estimado para o exemplar de teste $\mathbf{x} \rightarrow_{te}$.

Passo 1: calcule a probabilidade (*a priori*) para cada classe *K*, ($P(c_K)$);
Passo 2: **para cada** classe *K* **faça**
Passo 2.1: inicialize uma variável para o produtório (*prod*) com 1;
Passo 2.2: **para cada** atributo descritivo *j* de $\mathbf{x} \rightarrow_{te}$ **faça**
Passo 2.2.1: recupere os exemplares $\mathbf{x} \rightarrow_i$ de $\mathbf{X}_{\text{tr}}$ seguindo ($\sigma x \rightarrow_i [_j]=_{xtej}$ ($\mathbf{X}_{tr}$));
Passo 2.2.2: organize os exemplares recuperados no *Passo 2.2.1* de acordo com seus rótulos y_i;
Passo 2.2.3: calcule $P(x_{ij}|c_K)$, sendo que o rótulo y_i equivale a uma classe c_K, e multiplique a probabilidade obtida por *prod*;
Passo 2.3: faça $P(c_K|\ \mathbf{x} \rightarrow_{te}) = P(c_K) * prod$;
Passo 3: encontre *K* que maximiza $P(c_K|\ \mathbf{x} \rightarrow_{te})$ e o retorne como rótulo $\hat{y}_{te}$ de $\mathbf{x} \rightarrow_{te}$;

Para casos nos quais o valor do atributo é numérico, o cálculo de $P\ (x_{ij}|c_K)$ deverá ser substituído por:

$$P(x_{ij}|c_K) = g(x_{ij}, \mu_{c_k}, \sigma_{\mu_{c_k}}) \qquad \textbf{Equação 3-9}$$

em que $\propto c_K$ é a média dos valores do atributo *j*, considerando os exemplares da classe c_K, e $\sigma_{\propto cK}$, o desvio-padrão. Para valores contínuos, presume-se que os valores dos atributos tenham uma distribuição Gaussiana com média $\propto c_K$ e desvio $\sigma_{\propto cK}$ definida por

$$g(x_{ij}, \mu_{c_K}, \sigma_{\mu_{c_K}}) = \frac{1}{\sqrt{2\pi\sigma_{\mu_{c_K}}}} e^{-\frac{(x_{ij}-\mu_{c_K})^2}{2\sigma_{\mu_{c_K}}^2}} \qquad \textbf{Equação 3-10}$$

Como será possível perceber na execução do exemplo com o conjunto de dados *Planejamento de Vendas* na próxima seção, e também pela Equação 3-8, caso o valor de um atributo do exemplar a ser classificado não exista no conjunto de treinamento, a probabilidade $P\ (x_{ij}|c_K)$ será zero; e, consequentemente, o produto de probabilidades baseado na hipótese de

independência condicional de cada atributo descritivo (Equação 3-7) retornará uma probabilidade zero. A solução para evitar probabilidade zero pode ser a estimação de probabilidade conhecida por *Correção de Laplace* ou *Estimador Laplaciano* (veja mais informações em Han, Kamber e Pei, 2011). Isto consiste em supor que a quantidade de exemplares do conjunto de treinamento seja muito grande, e a adição de uma unidade nas contagens de exemplares nas classes não prejudique a estimação da probabilidade.

3.1.4.1. Exemplo didático para *Naïve* Bayes

Para exemplificar a aplicação do *Naïve* Bayes , considere um cenário relacionado com o planejamento de vendas do restaurante. O objetivo aqui é saber sob que condições se vende mais *Feijoada* ou *Filé* à *parmegiana*. O conjunto de dados criado neste cenário está ilustrado na Tabela 3-8 e é denominado *Planejamento de Vendas*. Esse conjunto de dados relaciona características referentes ao clima (previsão, temperatura, humidade e vento) com o prato que tem mais saída no restaurante diante de tal condição climática. O conjunto de dados é totalmente rotulado e, por isso, pode ser usado como um conjunto de treinamento para a construção de um classificador.

Uma nova situação $\mathbf{x}_{\rightarrow te}$, que não está presente na Tabela 3-9, será o objetivo de classificação, ou seja, a pergunta é qual prato (*Feijoada* ou *Filé à parmegiana*) venderá mais na ocorrência de tais variáveis climáticas.

Tabela 3-8: Conjunto de dados *PLANEJAMENTO DE VENDAS*

	x_{i1}	x_{i2}	x_{i3}	x_{i4}	y_i
	PREVISÃO	**TEMPERATURA**	**HUMIDADE**	**VENTO**	**VENDER**
$\mathbf{x}_{\rightarrow 1}$	chuva	frio	normal	sim	parmegiana
$\mathbf{x}_{\rightarrow 2}$	chuva	moderado	alta	sim	parmegiana
$\mathbf{x}_{\rightarrow 3}$	sol	quente	alta	não	parmegiana

$\mathbf{x}_{\rightarrow 4}$	sol	quente	alta	sim	parmegiana
$\mathbf{x}_{\rightarrow 5}$	sol	moderado	alta	não	parmegiana
$\mathbf{x}_{\rightarrow 6}$	nublado	frio	normal	sim	feijoada
$\mathbf{x}_{\rightarrow 7}$	nublado	quente	alta	não	feijoada
$\mathbf{x}_{\rightarrow 8}$	nublado	quente	normal	não	feijoada
$\mathbf{x}_{\rightarrow 9}$	nublado	moderado	alta	sim	feijoada
$\mathbf{x}_{\rightarrow 10}$	chuva	frio	normal	não	feijoada
$\mathbf{x}_{\rightarrow 11}$	chuva	moderado	alta	não	feijoada
$\mathbf{x}_{\rightarrow 12}$	chuva	moderado	normal	não	feijoada
$\mathbf{x}_{\rightarrow 13}$	sol	frio	normal	não	feijoada
$\mathbf{x}_{\rightarrow 14}$	sol	moderado	normal	sim	feijoada

Na solução com o algoritmo *Naïve* Bayes, a primeira ação a fazer é calcular a probabilidade de cada classe ($P(c_{feijoada})$ e $P(c_{filé\ à\ parmegiana})$). O conjunto de dados possui 14 exemplares, a classe *feijoada* possui 9 exemplares, e a classe *filé à parmegiana* possui 5 exemplares. Assim, tem-se $P(c_{feijoada}) = 9/14 = 0{,}64$ e $P(c_{filé\ à\ parmegiana}) = 5/14 = 0{,}36$.

Tabela 3-9: Exemplar com rótulo desconhecido

	x_{te1}	x_{te2}	x_{te3}	x_{te4}	$\hat{y}$
	PREVISÃO	**TEMPERATURA**	**HUMIDADE**	**VENTO**	**COMPRAR**
$\mathbf{x}_{\rightarrow te}$	sol	frio	normal	sim	?

O passo seguinte consiste em avaliar cada um dos atributos descritivos no cálculo das probabilidades condicionais de cada classe. Para o primeiro atributo (PREVISÃO), o exemplar $\mathbf{x}_{\rightarrow te}$ tem o valor *sol*. Do conjunto de dados

de treinamento, são então separados os cinco exemplares $\{\mathbf{x}_{\rightarrow 3}, \mathbf{x}_{\rightarrow 4}, \mathbf{x}_{\rightarrow 5}, \mathbf{x}_{\rightarrow 13}, \mathbf{x}_{\rightarrow 14}\}$. Destes, dois $\{\mathbf{xx}_{\rightarrow 13}, \mathbf{x}_{\rightarrow 14}\}$ possuem o rótulo *feijoada*, e três $\{\mathbf{x}_{\rightarrow 3}, \mathbf{x}_{\rightarrow 4}, \mathbf{x}_{\rightarrow 5}\}$ possuem o rótulo *filé à parmegiana*. Assim, $P(sol|_{Cfeijoada}) = 2/9 = 0{,}22$ e $P(sol|_{Cfilé\ à\ parmegiana}) = 3/5 = 0{,}6$.

O processo deve ser repetido para cada um dos atributos, e, ao final, todas as probabilidades devem ser multiplicadas. Então, $P(feijoada|\mathbf{x}_{\rightarrow te}) = 9/14 * 2/9 * 3/9 * 6/9 * 3/9 = 0{,}011$ e $P(parmegiana|\mathbf{x}_{\rightarrow te}) = 5/14 * 3/5 * 1/5 * 1/5 * 3/5 = 0{,}005$. A classe de maior probabilidade é feijoada, e, portanto, $y\hat{}_{te}$ = *feijoada.*

3.1.4.2. Exemplo prático para *Naïve* Bayes em R

Nesta seção, é apresentada a solução computacional, implementada em R, para resolver o problema discutido na Seção 3.1.4.1, a partir da aplicação de *Naïve* Bayes . O pacote ***e1071***[7] foi escolhido para a realização desta implementação (Meyer *et al.*, 2014). Este pacote precisa ser instalado usando o comando `install.packages("e1071")`, e a biblioteca deve ser chamada com o comando `library(e1071)` para a execução do experimento.

Como a solução agora é feita de maneira computacional, os dados precisam ser importados para dentro do ambiente R. Considere que os dados estão disponíveis como exemplificado na Tabela 3-8, porém, em formato de uma planilha (no formato .xlsx). A leitura dos dados a partir desse formato é feita com uso da função `read.xlsx( )` do pacote **xlsx** (Dragulescu, 2014). Consulte o Apêndice para mais detalhes sobre a importação de dados com esta biblioteca.

```
> planejamento <-
read.xlsx("C:\\Users\\LivroDM\\Predicao\\planejamento.xlsx",
"Sheet1",header=TRUE)
```

Agora, com os dados importados, a função `naiveBayes( )` pode ser executada. Embora esse classificador também não apresente uma fase de

aprendizado (como o *k*-NN), e nem o resultado gere um modelo, a função se apropria desses conceitos para separar o momento em que encontra as probabilidades condicionais a partir da análise dos exemplares presentes no conjunto de treinamento, da fase em que calcula as probabilidades condicionais referentes à pertinência dos exemplares de teste a cada uma das classes presentes no problema. No Quadro 3-7, é explicada a função que implementa a primeira fase (treinamento). No Quadro 3-8, está ilustrada a função que implementa a fase de teste (ou predição).

Note que a função `naiveBayes( )` espera que as classes dos exemplares sejam apresentadas de maneira separada. Isto pode ser feito utilizando um índice que indica a coluna da variável `planejamento` que deve ser associada à variável `rótulo` (`planejamento[,5]`), ou associando o nome do cabeçalho à variável (`planejamento$VENDER`). A instrução completa para esta fase de treinamento, contemplando a apresentação do rótulo de maneira separada é, da seguinte forma:

```
> modelo_NB <- naiveBayes(planejamento[1:4],planejamento[,5])
```

Quadro 3-7: Sintaxe e parâmetros da função NAIVEBAYES

Sintaxe para a função que implementa o algoritmo *Naïve* Bayes
Usando a função `naiveBayes( )` do pacote `e1071`
Construindo o classificador `modelo_naiveBayes <- naiveBayes(dados_treinamento, rótulo, laplace)` • `dados_treinamento` é um ***data.frame*** ou matriz contendo os exemplares de treinamento. • `rótulo` é um vetor com as classes para cada exemplar do conjunto de treinamento. • `laplace` é um número real positivo que controla a suavização de Laplace. O valor *default* é 0, desabilitando a suavização de Laplace. A função retorna um modelo de *Naïve* Bayes com um objeto com os parâmetros do modelo.

Quadro 3-8: Sintaxe e parâmetros da função PREDICT

Sintaxe para a função que implementa a predição a partir do modelo *Naïve* Bayes

Usando a função `predict( )` do pacote *e1071*

Fazendo a predição com o *Naïve* Bayes

```
y_estimado <- predict(modelo_naiveBayes, exemplar_teste, tipo, laplace)
```

- `modelo_naiveBayes` é o modelo construído pela função `naiveBayes()`.
- `exemplar_teste` é o exemplo de teste.
- `tipo` é um parâmetro com dois valores possíveis (`"class"` ou `"raw"`) e define se o resultado é apresentado apenas pela classe de maior probabilidade (*class*) ou pela probabilidade de cada classe (*raw*).
- `laplace` é um número real positivo que controla a suavização de Laplace. O valor *default* é 0, desabilitando a suavização de Laplace.

A função retorna um vetor com a classe estimada e sua probabilidade ou as classes e as respectivas probabilidades, dependendo do parâmetro `tipo`.

Então, a classificação pode ser feita com a função `predict( )`. O exemplar de teste deve estar em um formato de matriz ou ***data.frame***. Para resolver o problema desse experimento, optou-se por usar ***data.frame***, e a importação do exemplar de teste para esse tipo de variável é feita da seguinte forma:

```
> exemplar_teste <- data.frame(PREVISAO="sol",TEMPERATURA="frio",
HUMIDADE="normal",VENTO="sim")
```

Agora, finalmente a função *predict* pode ser aplicada. Nesse caso, optou-se por usar o *tipo class* e sem adição de *Correção de Laplace*. As instruções para classificação e apresentação do resultado são da seguinte forma:

```
> p <- predict(modelo_NB, exemplar_teste, type="class")
> p
[1] feijoada
Levels: feijoada parmegiana
```

Se a opção de classificação fosse por apresentar a probabilidade (*raw*), o resultado seria:

```
> p
feijoada parmegiana
[1,] 0.6756757 0.3243243
```

O resultado apresentado pelo R é equivalente ao resultado obtido no exemplo didático da seção anterior; no entanto, eles são apresentados após a realização de uma normalização: $P(\text{feijoada}|\mathbf{x}\rightarrow_{te}) = 0{,}011/(0{,}011 + 0{,}005) \cong 0{,}68$ e $P(\text{parmegiana}|\mathbf{x}\rightarrow_{te}) = 0{,}005/(0{,}011 + 0{,}005) \cong 0{,}31$. Note que, então, a soma das probabilidades do dado de teste pertencer a cada uma das classes possíveis resulta em 1.

3.2. A tarefa de regressão

Na resolução da tarefa de classificação de dados, o objetivo é predizer o rótulo para um exemplar qualquer que não pertence ao conjunto de dados de treinamento. Portanto, o uso de um modelo preditivo *f* promove a atribuição de um rótulo *y* a um exemplar $\mathbf{x}\rightarrow$ qualquer, ou seja, $y = f(\mathbf{x}\rightarrow)$, sendo *y* uma variável do tipo categórico. Por outro lado, quando *y* é do tipo numérico (contínuo ou discreto), diz-se ter um problema de **regressão** ou **predição numérica** .

Basicamente, os modelos de regressão podem ser divididos em: *linear simples* ou *multivariado*, ou *não linear simples* ou *multivariado*. A diferença entre regressão do tipo *linear* e regressão do tipo *não linear* está na função *f* a ser utilizada: por exemplo, uma função que representa a equação da reta ou do plano se aplica para regressão *linear*, e uma função que representa uma equação exponencial se aplica para regressão *não linear*. O tipo *simples* ou *multivariado* é definido a partir da quantidade de atributos descritivos utilizados para estimar o valor de *y*: um único atributo é utilizado no tipo *simples*, e mais de um atributo é usado no tipo *multivariado*.

Em se tratando de aplicação, a regressão é usada para estimar valores a partir de um conjunto de dados históricos. Isto é o que acontece, por exemplo, em problemas de indicadores econômicos ou de mercado futuro, nos quais se tenta prever o próximo valor analisando os dados de algumas variáveis (atributos descritivos) historicamente armazenadas em um conjunto de dados. Para o caso do restaurante, cenário deste livro, exemplos poderiam ser a estimação da quantidade de bebidas que deve ser estocada para determinado período ou o número de clientes que provavelmente comparecerá ao restaurante em um dia especial.

Para decidir entre usar uma regressão linear ou não linear, geralmente se faz uma análise inicial dos dados, de forma a verificar o tipo de distribuição que os atributos assumem. Usar recursos de visualização de dados, a exemplo

de um gráfico de dispersão, pode ser de grande auxílio. No caso da regressão não linear, ainda é preciso verificar qual seria a melhor função de ajuste a ser usada, como polinomial, potência, logarítmica etc.

A solução para a tarefa de regressão pode ser obtida a partir de métodos estatísticos baseados em premissas e condições relacionadas com o tipo de distribuição dos dados, ou de técnicas de aprendizado indutivo, que não necessitam de informação prévia sobre o tipo de distribuição dos dados. Para cobrir esse assunto, nesta seção são apresentadas soluções baseadas em métodos estatísticos com soluções analíticas para regressão linear e não linear.

3.2.1. Regressão linear

A **regressão linear** consiste em uma análise estatística que envolve duas variáveis: a de resposta, explicada, dependente ou, como definido aqui, rótulo de um exemplar (y); e a preditora, explicativa, independente ou, como aqui definido, conjunto de atributos descritivos ($\mathbf{x}\rightarrow$). Um modelo de regressão linear considera que o valor da variável de resposta (ou dependente) y pode ser estimado por uma combinação linear das variáveis explicativas (ou independentes) $\mathbf{x}\rightarrow$. Sendo assim, o modelo de regressão para uma única variável preditora x, ou regressão linear simples, pode ser definido pela seguinte equação da reta:

$$y = a + bx \quad \textbf{Equação 3-11}$$

em que a e b são coeficientes de regressão e especificam o intercepto do eixo y e a inclinação da reta, respectivamente. Os coeficientes de regressão podem também ser entendidos como pesos. Para o caso de regressão linear multivariada, a Equação 3-11 deve ser estendida para equação de plano ou hiperplano.

Na solução da tarefa de regressão, deve-se, então, encontrar valores para os coeficientes de regressão, de forma que a reta (ou o plano/hiperplano) se

ajuste aos valores assumidos pelas variáveis nos exemplares de um conjunto de dados. O melhor ajuste da reta pode ser encontrado, por exemplo, pelo método dos *mínimos quadrados*, o qual minimiza o erro entre os valores das variáveis nos exemplares do conjunto de dados e os valores estimados pelo regressor. Considere que o conjunto de dados disponível para treinamento seja $\mathbf{X}_{tr} = \{(\mathbf{x}\rightarrow_i, y_i)\}$, $i = 1, ..., n$. Os coeficientes de regressão podem ser estimados usan- do o método dos mínimos quadrados, traduzido pelas seguintes equações:

$$b = \frac{\sum_{i=1}^{n}(x_i - \mu_x)(y_i - \mu_y)}{\sum_{i=1}^{n}(x_i - \mu_x)^2}$$ **Equação 3-12**

$$a = \mu_y - b\mu_x$$ **Equação 3-13**

em que μ_x é a média dos valores de $x_1, x_2, ..., x_n$ e μ_y é a média dos valores de $y_1, y_2, ..., y_n$. Um procedimento para implementar o método dos mínimos quadrados é apresentado no Algoritmo 3-8.

Algoritmo 3-8: Algoritmo para modelagem de Regressão Linear Simples, usando o método dos mínimos quadrados

---

Parâmetros de entrada:
- **X**tr: um conjunto de treinamento com exemplares rotulados, $\mathbf{X}_{tr} = \{(\mathbf{x}\rightarrow_i, y_i)\}$, $i = 1, ..., n$;

Parâmetros de saída:
- coeficientes de regressão *a* e *b*;

Passo 1: calcule a média da variável independente (**x**) e a média da variável dependente (**y**), considerando os valores em $\mathbf{X}_{tr}$;
Passo 2: inicialize somadores ($soma_1$, $soma_2$) com zero;
Passo 3: **para cada** exemplar *i* em $\mathbf{X}_{tr}$ **faça**
Passo 3.1: $soma_1 = soma_1 + (x_i - \mu_x)(y_i - \mu_y)$;
Passo 3.2: $soma_2 = soma_2 + (x_i - \mu_x)$;
Passo 4: faça $b = soma_1 / soma_2$;

Passo 5: faça a = $\mu_y - b\mu_x$;

3.2.1.1. Exemplo didático para regressão linear

Como exemplo didático para a regressão linear, considere que o proprietário de um restaurante deseja aumentar as vendas investindo em propaganda na rádio da cidade. Considere também que o gasto neste tipo de publicidade é calculado pelo número de inserções do anúncio na programação da rádio durante o mês. Com o cuidado de mensurar o efeito desses anúncios, o proprietário do restaurante somou, ao final dos meses em que fez o investimento com o anúncio, o número de vendas do prato *Filé à Parmegiana*. Uma amostra dos resultados obtidos está ilustrada na Tabela 3-10, e esses resultados são usados nesta seção para ilustrar a obtenção de uma modelagem com regressão linear.

Embora a solução por regressão linear esteja sendo usada para o presente problema, na prática, é importante ter alguma pista de como os dados estão distribuídos de forma a escolher entre usar a regressão linear ou a não linear. Isso pode ser feito com a plotagem dos dados em um gráfico de dispersão.[8] Para o conjunto de dados *Planejamento de Propagandas* da Tabela 3-10, os dados estão distribuídos como mostra a Figura 3-25.

Tabela 3-10: Conjunto de dados *PLANEJAMENTO DE PROPAGANDAS*

	ID	x_{i1}	y_i
	PERÍODO	**# Inserções de anúncios**	**# Filés à Parmegiana vendidos**
$\mathbf{x}_{\rightarrow 1}$	mês 1	30	430
$\mathbf{x}_{\rightarrow 2}$	mês 2	21	335
$\mathbf{x}_{\rightarrow 3}$	mês 3	38	520
$\mathbf{x}_{\rightarrow 4}$	mês 4	42	490
$\mathbf{x}_{\rightarrow 5}$	mês 5	37	470
$\mathbf{x}_{\rightarrow 6}$	mês 6	20	210

$\mathbf{x}_{\rightarrow 7}$	mês 7	8		195
$\mathbf{x}_{\rightarrow 8}$	mês 8	17		270
$\mathbf{x}_{\rightarrow 9}$	mês 9	35		400
$\mathbf{x}_{\rightarrow 10}$	mês 10	25		480

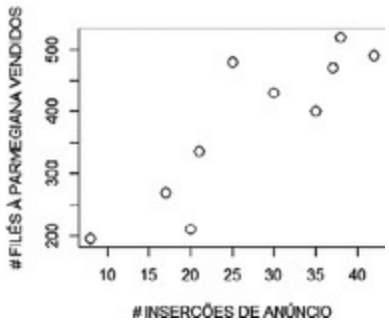


Figura 3-25: Gráfico de dispersão para o conjunto de dados *Planejamento de Propaganda*

A solução para o cálculo dos coeficientes de regressão é apresentada na Tabela 3-11, na qual as primeiras duas colunas (*I* e *II*) dizem respeito aos exemplares de treinamento (Tabela 3-10), e as colunas *III*, *IV*, *V e VI*, aos cálculos intermediários necessários no processo do Algoritmo 3-8. Os resultados referentes às variáveis $soma_1$ e $soma_2$ do Algoritmo 3-8 são mostrados à direita na última linha na tabela.

Tabela 3-11: Dados de execução do Algoritmo 3-8 sobre o conjunto de dados *PLANEJAMENTO DE PROPAGANDA*

I	*II*	*III*	*IV*	*V*	*VI*
x	***y***	**(*x* – μ*x*)**	**(*y* – μ*y*)**	**(*x* – μ*x*)(*y* – μ*y*)**	**(*x* – μ*x*)2**
30	430	2,7	50	135	7,29
21	335	–6,3	–45	283,5	39,69
38	520	10,7	140	1498	114,49
42	490	14,7	110	1617	216,09
37	470	9,7	90	873	94,09
20	210	–7,3	–170	1241	53,29
8	195	–19,3	-185	3570,5	372,49

	17	270	–10,3	-110	1133	106,09
	35	400	7,7	20	154	59,29
	25	480	–2,3	100	–230	5,29
médias (μ)	27,3	380		**somas** (Σ)	10275	1068,1

O cálculo dos coeficientes de regressão *b* e *a* (*Passos 4 e 5*) resultam em 9,62 e 117,38, respectivamente, de forma que o modelo linear para o conjunto de dados *Planejamento de Propaganda* é

$$y = 117{,}38 + 9{,}62x \qquad \textbf{Equação 3-14}$$

Esse modelo pode agora ser aplicado para estimar a venda do prato *Filé à Parmegiana* se certo investimento em propaganda for realizado. Por exemplo, suponha que o proprietário do restaurante esteja disposto a investir na inserção de 50 anúncios no próximo mês. Com esse modelo em mãos, ele pode estimar que a venda do *Filé à parmegiana* chegará a 117,38 + 9,62 * 50 ≅ 598 pratos.

3.2.1.2. Exemplo prático para regressão linear em R

No ambiente R, a análise preditiva numérica por regressão linear pode ser implementada com a função `lm( )` (Chambers, 1992; Wilkinson e Rogers, 1973), cuja descrição é apresentada no Quadro 3-9.

Quadro 3-9: Sintaxe e parâmetros da função LM

Sintaxe para a função que cria um modelo preditivo por regressão linear
Usando a função `lm( )` – função nativa no *R*
Construindo um modelo preditivo por regressão linear `modelo_linear <- lm(fórmula, dados)`

- `fórmula` é a definição dos atributos descritivos (variáveis independentes) e rótulo (dependente) para geração do modelo.
- `dados` é o conjunto de treinamento, em uma variável ***data.frame***.

A função retorna o modelo preditivo.

A leitura do conjunto de dados e a separação dos atributos nas variáveis dependente e independente (ou atributos descritivos e de rótulo) *x* e *y* devem ser realizadas da seguinte forma:

```
> bd <- read.table("C:\\Users\\LivroDM\\Predicao\\Propaganda.csv",
header=TRUE,
sep=";")
```

`> x <- bd[,1]` ou, alternativamente, `x <- bd$ANUNCIO`

`> y <- bd[,2]` ou, alternativamente, `y <- bd$PARMEGIANA`

E a aplicação da função que constrói o modelo preditivo é:

```
modelo_linear <- lm(y~x,bd)
```

Para visualizar os coeficientes do modelo, a variável `modelo_linear` deve ser chamada, e então as informações serão listadas como segue:

```
> modelo_linear

Call:
lm(formula = y ~ x)

Coefficients:
(Intercept) x
117.38 9.62
```

Depois de construído, o modelo pode ser utilizado na predição propriamente dita. A função responsável por implementar essa ação é `predict( )`, cuja sintaxe e parâmetros são apresentados no Quadro 3-10.

Quadro 3-10: Sintaxe e parâmetros para a função PREDICT

Sintaxe para a função que aplica um modelo preditivo construído via regressão linear
Usando a função `predict( )` – função nativa no *R*
Usando um modelo preditivo construído via regressão linear

```
y_estimado <- predict(modelo_linear, exemplar_teste)
```

- `modelo_linear` é o modelo resultante da função `lm( )`.
- `exemplar_teste` é o exemplar (atributos descritivos apenas) para o qual se deseja fazer a predição.

A função retorna a predição para um exemplar de teste ou para um conjunto de exemplares de teste.

Por exemplo, considerando o exemplo anterior sobre a possibilidade de uma inserção de 50 anúncios, o `predict( )` deve ser feito da seguinte forma:

```
> exemplares_teste <- data.frame(x=bd[,1])
> y_estimado <- predict(modelo_linear, exemplares_teste)
```

Uma representação gráfica para o modelo de regressão obtido pode ser realizada por meio de um gráfico no qual os valores das variáveis preditivas do conjunto de dados, em conjunto com os valores preditos, são plotados. Os comandos para criar tal visualização estão listados a seguir, e o gráfico resultante é mostrado na Figura 3-26.

```
>
plot(x,y,xlab="ANUNCIO",ylab="PARMEGIANA",type="p",col="blue",pch=1
> points(x,y_estimado,type="p",col="red",pch=16)
>
legend(locator(1),c("y","y_estimado"),pch=c(1,16),col=c("blue","red
```

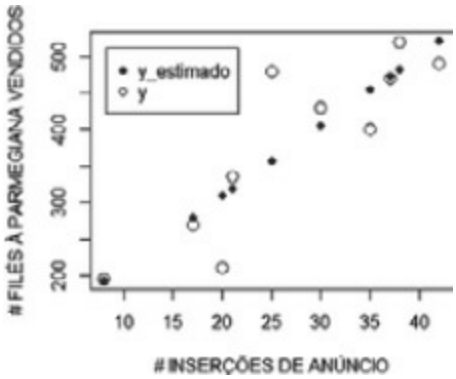


Figura 3-26: Representação gráfica para o modelo de regressão linear obtido para o conjunto PLANEJAMENTO DE PROPAGANDA

Para executar o teste do novo exemplar, como no exemplo didático na seção anterior:

```
> exemplar_teste <- data.frame(x=50)
> y_estimado <- predict(modelo_linear, exemplar_teste)
> y_estimado
1
598.3714
```

3.2.2. Regressão não linear

Em algumas situações, a aproximação das relações entre a variável dependente (y) e as variáveis independentes (x) de um conjunto de dados não segue uma relação linear. A razão desse comportamento, em muitos casos, pode ser observada a partir de gráficos que permitam visualizar o fenômeno ou então com a ajuda de medidas que avaliem a qualidade da regressão.

Uma regressão não linear é modelada por uma função, combinação não linear de parâmetros do modelo, e, assim como no caso da regressão linear, também depende das variáveis dependentes e independentes. Exemplos típicos de funções não lineares são exponencial, logarítmica e de potência. Essas funções têm como característica principal a capacidade de linearização, isto é, o modelo não linear produzido por elas pode ser reduzido a um modelo linear por meio de um ajuste (apropriado) nos parâmetros. Para finalidade de exemplo, considere a função não linear exponencial; ela é descrita como:

$$y = a * e^{bx}$$ **Equação 3-15**

Agora, aplicando-se logaritmo em ambos os lados da Equação 3-15, obtém-se a seguinte função linear:

$$\ln y = \ln a + bx$$ **Equação 3-16**

Em aspectos práticos, a linearização permite que o cálculo dos coeficientes de regressão seja facilitado. Isso significa que o algoritmo e as

equações definidas para a regressão linear podem ser reaproveitados, com a ressalva de que, para esse caso, os valores da variável *y* devem ser transformados em ln *y*, representado aqui por $y*$. O intercepto de $y*$ será ln *a*, chamado aqui de $a*$. Assim, a Equação 3-16 pode ser reescrita como:

$$y* = a* + bx$$ **Equação 3-17**

Vale notar que a Equação 3-17 é semelhante à Equação 3-11, com a diferença das transformações em *y* e *a*. A fim de realizar a regressão não linear, o Algoritmo 3-8 deve ser ligeiramente modificado: um *Passo 0* deve ser inserido calculando $y* = \ln y$; no *Passo 1*, o cálculo da média da variável dependente (μ_y) deve ser substituído pelo cálculo da média sobre o logaritmo de *y* (μ_{y*}). Como o intercepto $a*$ é igual a ln *a*, após o cálculo do coeficiente $a*$, no *Passo 5*, deve ser acrescentado um novo passo, *Passo 6*, fazendo $a = e^{a*}$.

3.2.2.1. Exemplo didático para regressão não linear

O conjunto de dados *Planejamento de Propagandas* também pode ser usado para o exemplo didático da regressão não linear . A solução para o cálculo dos coeficientes de regressão é apresentada na Tabela 3-12; e o cálculo dos coeficientes de regressão *a* e *b* resultam em 163,59 e 0,029, respectivamente, de forma que o modelo não linear para o conjunto de dados *Planejamento de Propaganda* é $y = 163{,}59 * e^{0{,}029x}$. Nesse caso, a predição para o investimento em 50 anúncios é aproximadamente 697 pratos de *Filé à parmegiana*.

Tabela 3-12: Dados de execução do Algoritmo 3-8 modificado para atender aos requisitos da regressão não linear sobre o conjunto de dados PLANEJAMENTO DE PROPAGANDA

I	*II*	*III*	*IV*	*V*	*VI*
x	**y***	**(*x* – μ*x*)**	**(y* – μy*)**	**(*x* – μ*x*)(y* – μy*)**	**(*x* – μ*x*)2**
30	6,064	2,7	0,178	0,479	7,29

	21	5,814	–6,3	–0,072	0,454	39,69
	38	6,254	10,7	0,368	3,933	114,49
	42	6,194	14,7	0,308	4,530	216,09
	37	6,153	9,7	0,266	2,585	94,09
	20	5,347	–7,3	–0,539	3,936	53,29
	8	5,273	–19,3	–0,613	11,836	372,49
	17	5,598	–10,3	–0,288	2,965	106,09
	35	5,991	7,7	0,105	0,810	59,29
	25	6,174	–2,3	0,288	–0,661	5,29
médias (μ)	27,3	5,886		**somas** (Σ)	30,87	1068,1

3.2.2.2. Exemplo prático em R para regressão não linear

A regressão não linear também é implementada pela função `lm( )`. Porém, nesse caso, é necessário aplicar a função logarítmica sobre *y*. Então, tem-se:

```
> modelo_nao_linear <- lm(log(y)~x, bd),
```

lembrando que as variáveis *x* e *y* devem ser lidas como na explicação da Seção 3.2.1.2. Também, os valores referentes aos coeficientes do modelo gerado via regressão não linear podem ser exibidos da seguinte forma:

```
> modelo_nao_linear

Call:
lm(formula = ln(y) ~ x)

Coefficients:
(Intercept) x
5.09733742 0.002889849.
```

Transformando o valor do intercepto com `exp(5.09733742)`, conforme previsto no novo *Passo 6* do Algoritmo 3-8, o valor estimado deve ser calculado como:

```
> y_estimado <- 163.59 * exp(0.029 * x)
```

```
> y_estimado_50 <- 163.59 * exp(0.029 * 50)
> y_est <- 163.59 * exp(0.029 * bd[,1])
```

e a plotagem do gráfico (Figura 3-27) deve ser:

```
>
plot(bd[,1],bd[,2],xlab="ANUNCIO",ylab="PARMEGIANA",type="p",col="
cex=1.5)
> points(bd[,1],y_est,type="p",col="red",pch=16)
>
legend(locator(1),c("y","y_estimado"),pch=c(1,16),col=c("blue","re
```

Para executar o teste do novo exemplar, como no exemplo didático na seção anterior:

```
> exemplar_teste <- data.frame(x=50)
> y_estimado <- 163.59*exp(0.029*exemplar_teste)
> y_estimado
x
697.4029
```

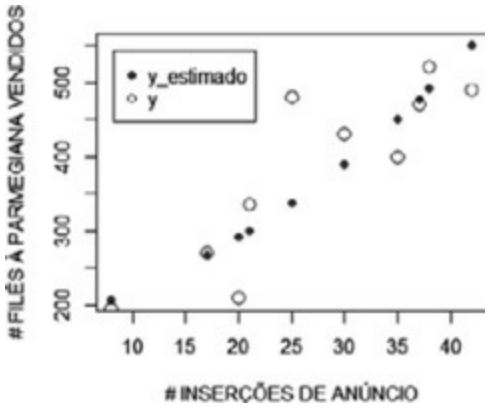


Figura 3-27: Representação gráfica para o modelo de regressão não linear obtido para o conjunto PLANEJAMENTO DE PROPAGANDA

3.3. Metodologia de construção e avaliação de modelos preditivos

Nas seções anteriores, foram apresentadas várias técnicas capazes de construir modelos preditivos, tanto para a predição categórica quanto para a predição numérica. Não é suficiente, no entanto, se preocupar apenas com a execução do algoritmo que implementa tais técnicas e gera um (qualquer) modelo. É necessário se preocupar também com o processo de construção dos modelos , o qual deve ser baseado em estratégias que possibilitem a geração de bons modelos preditivos. Há diferentes maneiras de implementar o processo de construção de modelos preditivos, e esta seção é dedicada a apresentar algumas delas.

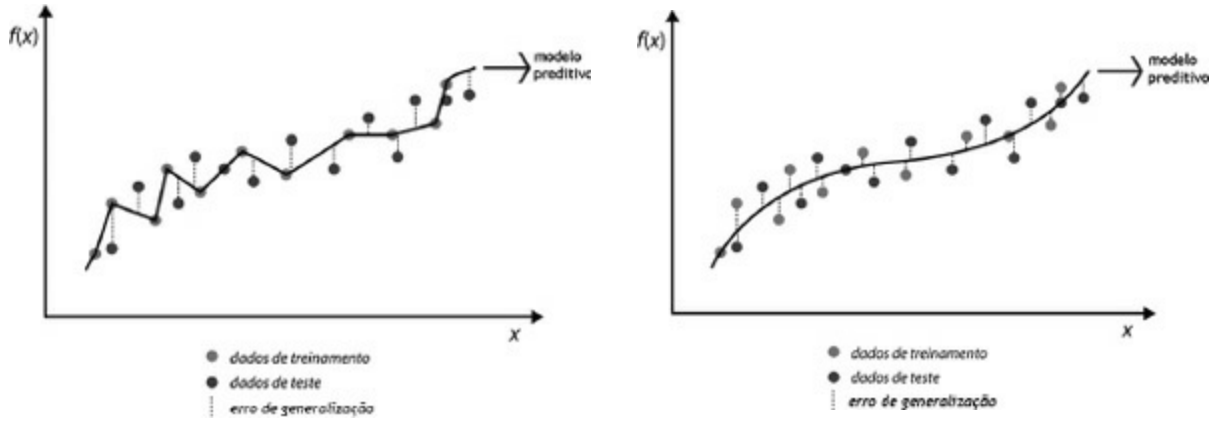
Além disso, é preciso saber avaliar o que é um “bom” modelo preditivo. Também não é suficiente apenas desenvolver um processo de geração de modelos, com base em uma das estratégias aqui descritas. É preciso ter condições de avaliar a qualidade dos modelos gerados diante de cada contexto de aplicação e da tarefa de mineração a ser resolvida (classificação ou regressão). Algumas medidas para avaliação de modelos preditivos são também apresentadas nesta seção.

3.3.1. Estratégias para treinamento, validação e teste

A primeira lição a ser aprendida é que, independentemente da medida de avaliação a ser usada para atestar a qualidade de um modelo, não é adequado avaliá-lo por seu desempenho em relação aos exemplares apresentados no processo de treinamento (indução). É sempre necessário saber como o modelo se comporta quando aplicado a exemplares que ainda não conhece, ou seja, não usados no processo de sintonização de seus parâmetros. O motivo para essa ressalva é que modelos preditivos, a depender de como são

gerados, podem levar à manifestação de um fenômeno bastante conhecido, o **sobreajuste** (do inglês *overfitting*).

O fenômeno de sobreajuste acontece quando o modelo preditivo é gerado de forma a representar os exemplares usados para sua geração com uma fidelidade mais alta que o necessário. Esses modelos, se avaliados em relação ao seu desempenho nesses exemplares, receberão avaliações muito boas. Porém, de fato, modelos que se sobreajustam aos exemplares usados durante o processo de indução geralmente não são capazes de fornecer respostas adequadas para os novos exemplares, ou seja, perdem desempenho no que diz respeito ao erro de generalização; e, se avaliados por seu desempenho nos novos exemplares, receberão avaliações muito ruins. Lidar com esse problema é um dos objetivos da aplicação das estratégias aqui apresentadas. A Figura 3-28 mostra a representação de modelos preditivos: (a) com um erro de generalização alto porque está sobreajustado aos exemplares de treinamento; (b) com um erro de generalização menor e mais adequado para um modelo preditivo.



(a) (b)

Figura 3-28: Exemplos de modelo preditivo: (a) com sobreajuste; (b) sem sobreajuste

São apresentadas, nesta seção, as seguintes estratégias para a geração e avaliação de modelos preditivos: resubstituição, *holdout*, validação cruzada e

bootstrap. A primeira é uma exceção ao objetivo delineado nesta discussão, usada apenas para a verificação da implementação de uma técnica ou para estudo do seu potencial para a geração de modelos preditivos em algum tipo específico de aplicação. Leituras adicionais sobre esse assunto são indicadas na Seção 3.4.

3.3.1.1. Resubstituição

Trata-se de uma estratégia otimista – a mais otimista de todas as estratégias, na qual as medidas de avaliação dos classificadores são aplicadas no próprio conjunto de dados usado para indução do modelo. Nessa estratégia, as medidas de avaliação geram estimativas do **erro de resubstituição** e são úteis para dois tipos de avaliação:

- Para realizar uma verificação da codificação desenvolvida para determinada técnica, de forma que, a partir de um conjunto de dados conhecido, para o qual se sabe que modelos preditivos eficientes podem ser facilmente obtidos, possa ser verificado se os modelos gerados alcançam bom desempenho. Dessa forma, a geração do modelo e a avaliação de seu desempenho podem ser feitas usando o mesmo conjunto de dados, e, no caso da obtenção de avaliações ruins, poder-se-á inferir que a codificação pode estar com problemas.
- Para avaliar se a técnica escolhida para resolução de uma tarefa preditiva, em determinado contexto, tem potencial para gerar bons resultados. Nesse caso, se já no erro de resubstituição os modelos preditivos gerados alcançarem baixas avaliações, poder-se-ia inferir que a técnica escolhida para gerá-los não está adequada à complexidade daquele contexto.

3.3.1.2. *Holdout*

Na sua forma mais simples, a estratégia ***holdout*** pressupõe a criação de dois subconjuntos de dados disjuntos, a partir do conjunto de dados disponível para uso na indução do modelo. Um dos subconjuntos será usado para treinamento (indução) do modelo preditivo, e o segundo, para teste após o término do treinamento e, consequentemente, para aplicação das medidas de avaliação do modelo. Tradicionalmente, os dois subconjuntos são gerados de forma que 70% dos exemplares do conjunto de dados sejam alocados para o subconjunto de treinamento , e os 30% restantes sejam alocados no subconjunto de teste . Alternativamente, as porcentagens 60% e 40% podem ser usadas. Os exemplares a serem alocados em cada um dos subconjuntos devem ser escolhidos aleatoriamente.

A execução dessa estratégia uma única vez, embora capaz de produzir uma avaliação mais confiável que a estratégia de resubstituição, não é suficiente. A aleatoriedade imposta na geração dos subconjuntos de dados pode levar a situações que beneficiariam o teste do modelo; sabendo que o conjunto de dados é uma amostra do universo de dados possíveis em certo contexto, é interessante maximizar as chances de gerar avaliações estatisticamente confiáveis. Então, a estratégia *holdout* deveria ser aplicada *N* vezes, considerando os mesmos conjuntos de parâmetros determinados no algoritmo de geração dos modelos, porém, variando a composição dos subconjuntos de treinamento e teste. A avaliação do modelo deveria ser gerada pela aplicação de medidas estatísticas (média, desvio-padrão, intervalos de confiança) ao conjunto de avaliações obtido para os modelos gerados nas *N* execuções da estratégia.

Ainda, outra alternativa deve ser considerada: o uso de um subconjunto de validação . Nesse caso, o conjunto de dados deve ser dividido em três subconjuntos distintos, usando proporções como 70% ou 60% (treinamento), 15% ou 20% (validação) e 15% ou 20% (teste). Os papéis dos conjuntos de treinamento e de teste continuam os mesmos, sendo a novidade, aqui, o papel do conjunto de validação. O conjunto de validação será usado apenas para

avaliar o desempenho do modelo preditivo enquanto estiver sendo gerado, fazendo um papel de conjunto de teste durante o treinamento. Os exemplares presentes no conjunto de validação não devem ser usados para realizar nenhum tipo de correção ou ajuste de parâmetros no modelo.

O desempenho do modelo nos subconjuntos de treinamento e de validação deve melhorar durante o processo. No entanto, após um tempo necessário para estabilização do modelo, o desempenho no subconjunto de treinamento deve melhorar cada vez mais, caminhando para a manifestação do fenômeno de sobreajuste. O início dessa manifestação é marcado por uma série de pioras no desempenho do modelo no conjunto de validação, indicando que o erro de generalização está subindo. Nesse momento, o processo de indução do modelo pode ser interrompido de forma a gerar um modelo de predição com uma capacidade adequada de generalização. A Figura 3-29 mostra uma situação típica referente à análise do desempenho do modelo nos conjuntos de treinamento e validação durante o processo de indução (veja o exemplo prático da rede neural artificial *Multilayer Perceptron*, Seção 3.1.2.2).

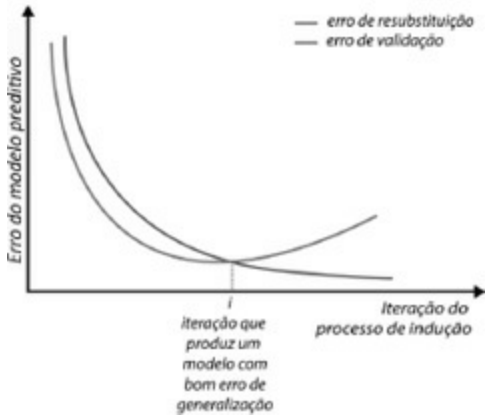


Figura 3-29: Comportamento do erro de resubstituição e do erro de validação durante um processo de indução de um modelo de predição

Para o caso de problemas nos quais o conjunto de dados possui uma distribuição de classes desbalanceada, é interessante realizar um controle na criação dos subconjuntos de treinamento, validação e teste. Esse controle

deve garantir que a distribuição de classes associadas aos exemplares alocadas nos subconjuntos siga as mesmas proporções existentes no conjunto de dados original.

3.3.1.3. Validação cruzada

Na estratégia de **validação cruzada**, todos os exemplares farão parte, em algum momento, do conjunto de dados usado no teste do modelo preditivo. Para implementar essa situação, o conjunto de dados será dividido em *K* subconjuntos disjuntos, com alocação aleatória de exemplares para cada subconjunto (podendo ser aplicado aqui um controle referente à distribuição de classes, como já explicado). Assim, o conjunto de dados **D** será dividido nos subconjuntos[9] $\mathbf{D}_1 \ldots \mathbf{D}_k \ldots \mathbf{D}_K$.

Então, mantendo constante o conjunto de parâmetros livres da técnica de indução de modelo adotada, a geração de *K* modelos preditivos será realizada da seguinte forma: um dos subconjuntos será reservado para ser usado como conjunto de teste, e os *K-1* restantes comporão o conjunto de treinamento; esse procedimento será repetido *K* vezes, alterando o subconjunto reservado para teste do modelo. Para cada execução do procedimento, um modelo preditivo é gerado, e, a ele, devem ser aplicadas as medidas de avaliação de desempenho desejadas. A Figura 3-30 mostra um esquema gráfico referente à execução da estratégia de validação cruzada.

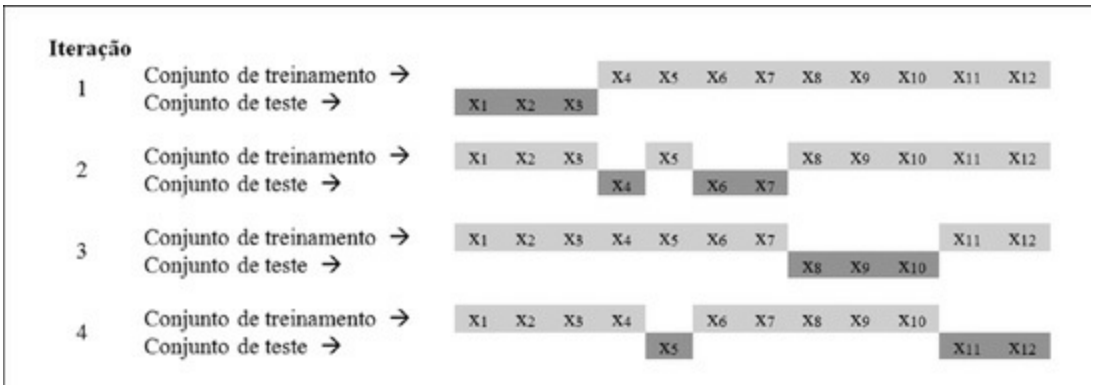


Figura 3-30: Esquema gráfico referente à execução da estratégia de validação cruzada com *K* = 4, para um conjunto de dados com 12 exemplares

A avaliação final referente à geração dos modelos preditivos pode ser feita, pelo menos, de duas diferentes formas: (a) aplicando medidas estatísticas, como média, desvio-padrão e intervalo de confiança ao conjunto de *K* avaliações obtidas; (b) somando-se o desempenho obtidos pelos *K* modelos gerados e dividindo essa soma pelo número de exemplares do conjunto de dados original.

Há também uma estratégia de validação cruzada conhecida como ***leave-one-out***. É, na realidade, a estratégia de validação cruzada na qual o número de subconjuntos é igual ao número de exemplares existentes no conjunto de dados disponível para ser usado no processo de indução (*K* = número de exemplares). Assim, a cada geração e avaliação de um modelo de predição, um exemplar é usado para teste, enquanto todos os outros exemplares presentes no conjunto de dados são usados para treinamento. Essa variação da validação cruzada é ideal para uso quando o conjunto de dados é pequeno. A Figura 3-31 mostra um esquema da execução dessa estratégia considerando um conjunto de dados com 12 exemplares.

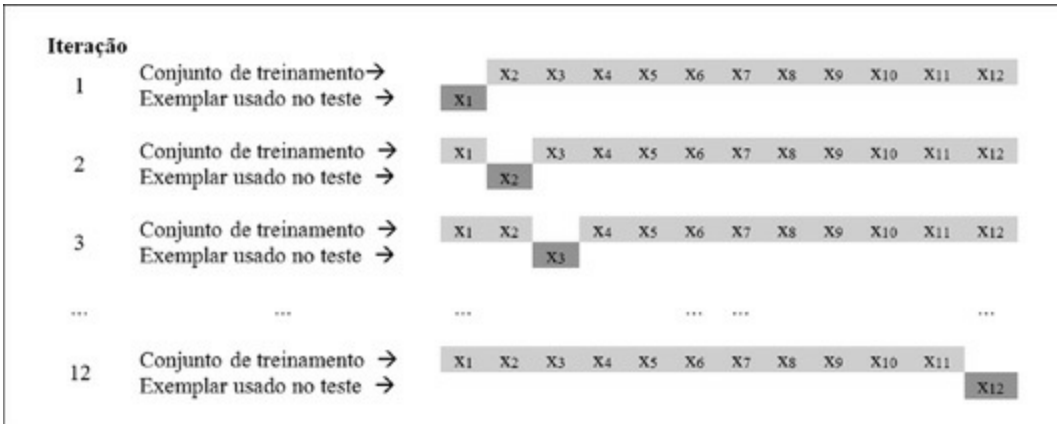


Figura 3-31: Esquema gráfico referente à execução da estratégia *leave-one-out* para um conjunto de dados com 12 exemplares

3.3.1.4. *Bootstrap*

A estratégia ***bootstrap*** é muito similar à estratégia *holdout*. Ela também usa dois subconjuntos de dados, um para treinamento e outro para teste.

Porém, durante o processo de formação dos subconjuntos, exemplares já sorteados para fazerem parte de um deles podem ser novamente sorteados, com igual probabilidade. Trata-se, portanto, de uma estratégia que permite a reposição.

É preciso então estabelecer um método para guiar o processo de formação dos subconjuntos. Han, Kamber e Pei (2011) apresentam uma estratégia na qual o subconjunto de treinamento é formado por *n* exemplares, sendo *n* o número de exemplares do conjunto de dados original. Dado que a escolha dos *n* exem- plares permite a reamostragem dos mesmos, é muito provável que alguns exemplares sejam escolhidos mais de uma vez para compor o subconjunto de treinamento. Os exemplares não escolhidos comporão o subconjunto de teste. Essa abordagem é denominada *.632 bootstrap*, visto que, em média, 63,2% dos exemplares comporão o subconjunto de treinamento.[36] Assim como na estratégia *holdout*, esse processo deve ser executado várias vezes, considerando os mesmos conjuntos de parâmetros determinados no algoritmo de geração dos modelos, para que vários modelos preditivos sejam gerados a partir de diferentes composições de subconjuntos de treinamento e de teste.

Para chegar a uma avaliação sobre o desempenho médio dos modelos preditivos gerados, sugere-se usar uma média ponderada dos erros de resubstituição e dos erros calculados usando o conjunto de teste. O erro de resubstituição deve ter um peso menor que o erro de teste, sugerindo-se usar 0,368 e 0,632 como pesos, respectivamente.

Devido à possibilidade de usar uma estratégia de sorteio com reposição na composição do subconjunto de treinamento, essa estratégia se torna interessante para uso em contextos nos quais o conjunto de dados disponível para indução do modelo preditivo é pequeno.

3.3.2. Medidas de avaliação

A medida mais comumente usada na avaliação de classificadores é a acurácia ou taxa de classificações corretas. A acurácia é dada por:

$$Acurácia = |y - f(\varkappa) = 0|, \qquad \textbf{Equação 3-18}$$

em que $|\cdot|$ representa a contagem de vezes em que $\cdot$ é verdadeiro, f é o modelo preditivo, $\varkappa$ é o subconjunto de dados sob o qual o modelo está sendo avaliado, $f(\cdot)$ é a classificação fornecida pelo modelo preditivo para cada um dos exemplares, e y é a classe esperada como resposta.

A acurácia de um classificador pode também ser escrita em termos do erro de generalização, ξ_g, e uma função de perda binária e, assim, ser interpretada como a probabilidade de ocorrer uma classificação correta. Assim:

$$Acurácia_g = 1 - \xi_g \qquad \textbf{Equação 3-19}$$

$$\xi_g(f(\varkappa), D) = \sum_{\langle \vec{x}_i, y_i \rangle \in \varkappa} D(\vec{\mathbf{x}}_i, y_i) * \wp(f(\vec{\mathbf{x}}_i), y_i), \qquad \textbf{Equação 3-20}$$

$$\wp(f(\vec{\mathbf{x}}_i), y_i) = \begin{cases} 0 \ if \ y_i = f(\vec{\mathbf{x}}_i) \\ 1 \ if \ y_i \neq f(\vec{\mathbf{x}}_i) \end{cases} \qquad \textbf{Equação 3-21}$$

em que D é a distribuição de probabilidade que gera o conjunto de dados original, e $\mathbf{x}\rightarrow$ é um exemplar de $\varkappa$.

Para avaliação de modelos preditivos gerados para resolução da tarefa de regressão, a avaliação se dá por meio de uma função de perda contínua, capaz de medir o erro entre a resposta obtida pelo modelo preditivo e a resposta esperada. Algumas funções de perda comumente usadas nesse contexto são:

$$Erro\ absoluto = \Sigma_{<\vec{x}_i, y_i> \in \varkappa} |y_i - f(\vec{\mathbf{x}}_i)|$$ **Equação 3-22**

$$Erro\ quadrado = \Sigma_{<\vec{x}_i, y_i> \in \varkappa} (y_i - f(\vec{\mathbf{x}}_i))^2$$ **Equação 3-23**

$$Erro\ absoluto\ médio = \Sigma_{<\vec{x}_i, y_i> \in \varkappa} |y_i - f(\vec{\mathbf{x}}_i)| / m$$ **Equação 3-24**

$$Erro\ quadrado\ médio = \Sigma_{<\vec{x}_i, y_i> \in \varkappa} (y_i - f(\vec{\mathbf{x}}_i))^2 / m$$ **Equação 3-25**

em que *m* é a quantidade de exemplares existentes em $\varkappa$.

Durante o processo de avaliação de um classificador é útil analisar o tipo de erro que o modelo está cometendo. A depender da natureza dos erros cometidos é possível entender se há classes mais difíceis de serem tratadas pelo classificador e, principalmente no caso de classificadores binários, é possível descobrir que, embora a acurácia possa estar alta, o classificador não está respondendo adequadamente. Uma ferramenta adequada para realizar esse tipo de análise é a matriz de confusão.

De forma geral, a **matriz de confusão** tem dimensões C × C, em que C é o número de classes presentes no problema de classificação sendo resolvido. As linhas dessa matriz são indexadas seguindo as "classes esperadas" (y), e as colunas são indexadas seguindo as "classes preditas" ($f(x)$ ou $\hat{y}$). Cada célula da matriz é um contador que deve ser incrementado a depender do resultado da comparação de $f(x)$ e y. As respostas corretas do classificador geram os valores que entram na diagonal principal da matriz. A Figura 3-32 ilustra o procedimento de preenchimento de uma matriz de confusão, de acordo com a resolução de um problema de classificação hipotético.

Note que, no caso apresentado na Figura 3-32, o classificador apresenta dificuldades em lidar com os exemplares da Classe 3. Esse fato pode ser um indicativo de que a Classe 3 apresenta uma complexidade maior que as

demais classes e que alguma ação de pré-processamento de dados ou de refinamento do modelo preditivo deve ser realizada.

Problema sob resolução	
f(x)	y
1	1
1	1
1	1
2	1
2	2
2	2
2	2
3	1
3	2
3	3

Classe Esperada (y)	Classe Predita (f(x))		
	Classe 1	Classe 2	Classe 3
Classe 1	3	1	1
Classe 2		3	1
Classe 3			1

Figura 3-32: Preenchimento de uma matriz de confusão

A matriz de confusão é especialmente útil para avaliação de classificadores binários. Nesse caso, uma taxonomia é atribuída às suas células, conforme ilustrado na Figura 3-33. Lembrando que, nos classificadores binários, as duas classes presentes no problema devem ser definidas como “classe positiva” e “classe negativa”.

Classe Esperada (y)		*Classe Predita (f(x))*	
		positivo	*negativo*
positivo		*Verdadeiros positivos (VP)*	*Falsos negativos (FN)*
negativo		*Falsos Positivos (FP)*	*Verdadeiros negativos (VN)*

Figura 3-33: Matriz de confusão para o caso de um problema de classificação binária

Cada uma das células da matriz de confusão possui um significado que pode indicar problemas maiores ou menores nos resultados do classificador, a

depender do contexto de aplicação no qual o classificador foi aplicado. Os significados das células da matriz de confusão para o problema binário são:

- Verdadeiro positivo (VP): classificação correta na classe positiva – o exemplar pertence à classe **positiva**, e o classificador o classificou como pertencente à classe **positiva**.
- Falso positivo (FP): classificação incorreta na classe positiva – o exemplar pertence à classe **negativa**, mas o classificador o classificou como pertencente à classe **positiva**.
- Verdadeiro negativo (VN): classificação correta na classe negativa – o exemplar pertence à classe **negativa**, e o classificador o classificou como pertencente à classe **negativa**.
- Falso negativo (FN): classificação incorreta na classe negativa – o exemplar pertence à classe **positiva,** mas o classificador o classificou como pertencente à classe **negativa**.

Para compreender melhor a utilidade da análise dos erros dentro da matriz de confusão, considere o seguinte problema: um restaurante indiano está inserindo novos itens bem apimentados em seu cardápio, e o gerente do restaurante decidiu fazer uma degustação às cegas, porém, não quer oferecer um novo prato apimentado para clientes que possuem um perfil de pedidos referentes a pratos sem pimenta. Então, ele recorre a um classificador binário para tomar a decisão sobre servir (classe positiva) ou não servir (classe negativa) os novos pratos para determinados clientes – visto que existem pessoas alérgicas a pimenta, é importante que exista uma preocupação especial com essa tomada de decisão. Dentre os 22.400 pedidos existentes no subconjunto de teste usado na aferição de desempenho do classificador, apenas 2.900 não incluem pratos apimentados, sendo que, desses, 90% referem-se a clientes que possuem alergia à pimenta – o gerente não tem

acesso a essas informações e deposita total confiança no software classificador à sua disposição.

Na aferição de desempenho do classificador, usando a medida de acurácia, obtivera-se ≅ 84% de classificações corretas, ou seja, o classificador respondeu corretamente para 18.872 casos, dos 22.400 apresentados a ele. A matriz de confusão que representa o desempenho do classificador implementado no software desse restaurante apresenta a configuração mostrada na Figura 3-34. Note que, embora o classificador tenha sido avaliado com ≅ 84% de acurácia, ele só foi capaz de classificar corretamente ≅ 53% dos pedidos sem pratos apimentados. Assim, ≅ 47% dos pedidos sem pratos apimentados foram classificados como falsos positivos. Sabendo que 90% deles podem ser de pessoas alérgicas, pode-se estimar que a chance de o classificador indicar comida apimentada para uma pessoa alérgica à pimenta é de ≅ 43%. Ou seja, esse classificador não deve ser considerado bom, principalmente no contexto no qual está sendo usado.

		Classe Predita (f(x))	
		*Pedidos **com** pratos apimentados*	*Pedidos **sem** pratos apimentados*
Classe Esperada (y)	*Pedidos **com** pratos apimentados (servir)*	*17.312*	*2.188*
	*Pedidos **sem** pratos apimentados (não servir)*	*1.340*	*1.560*

Figura 3-34: Matriz de confusão para o software classificador do restaurante indiano

Obviamente, o melhor caso esperado para uma matriz de confusão é o preenchimento apenas de sua diagonal principal, o que resultaria em 100% de acurácia, levando apenas a verdadeiros positivos e verdadeiros negativos. Porém, em situações reais, esse cenário é muito difícil de ser obtido. Para avaliar adequadamente os diferentes cenários possíveis, há uma série de medidas de avaliação derivadas de uma matriz de confusão para problemas de classificação binária:

- Sensibilidade ou **revocação** (do inglês, *recall*) ou taxa de verdadeiros positivos: porcentagem de verdadeiros positivos dentre todos os exemplos cuja classe esperada é a classe positiva (VP/(VP + FN)).
- Especificidade ou taxa de verdadeiros negativos: proporção de rejeições corretas, ou seja, porcentagem de verdadeiros negativos dentre todos os exemplos cuja classe esperada é a classe negativa (VN / (FP + VN)).
- Taxa de falsos positivos: porcentagem de falsos positivos dentre todos os exemplos cuja classe esperada é a classe negativa (FP / (VN + FP)).
- Taxa de falsas descobertas: porcentagem de falsos positivos dentre os exemplos classificados como positivos (FP / (VP + FP)).
- Preditividade positiva ou **precisão** (do inglês, *precision*): porcentagem de acertos ou verdadeiros positivos dentre todos os exemplos classificados como positivos (VP / (VP + FP)).
- Preditividade negativa: porcentagem de rejeições ou de verdadeiros negativos dentre todos os exemplos classificados como negativos (VN / (VN + FN)).
- *F-score* : faz uma relação entre a precisão e a revocação: 2/((1/revocação) + (1/precisão)).

A análise do contexto do restaurante indiano sob a visão proporcionada por cada uma dessas medidas é deixada como exercício para o leitor.

Outro recurso útil para análise de classificadores binários são os gráficos ROC (*Receiver Operating Characteristics*). Em Fawcett (2006) e Prati, Batista e Monard (2008), os autores apresentam discussões detalhadas sobre várias análises que podem ser feitas com ouso desse recurso. O **gráfico ROC** é construído como a relação entre taxa de falsos positivos (eixo x) e a taxa verdadeiros positivos (eixo y), e ilustra como um classificador se posiciona em relação à sua inabilidade em evitar falsos alarmes e sua habilidade em detectar a classe positiva corretamente. Assim, um classificador é

representado como um ponto no gráfico, de forma que um ponto na origem representa um classificador incapaz de classificar qualquer exemplo como pertencente à classe positiva; e um ponto no canto superior direito do gráfico, só capaz de classificar os exemplos como pertencentes à classe positiva. O classificador perfeito é aquele cujo ponto que o representa no gráfico está localizado no canto superior esquerdo. A Figura 3-35 traz um exemplo de um gráfico ROC com a representação de alguns classificadores.

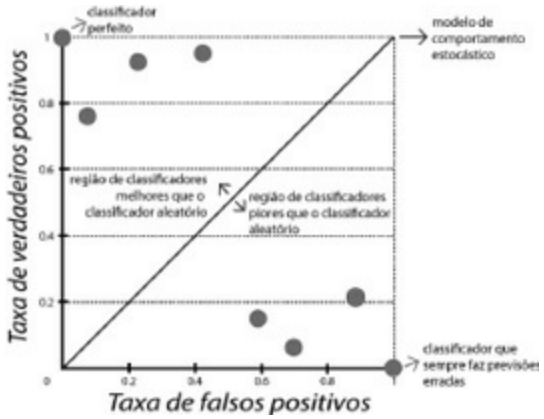


Figura 3-35: Exemplos de um gráfico ROC, no qual cada ponto no gráfico representa um classificador

3.3.3. Exemplo didático para construção e avaliação de modelos preditivos

A fim de ilustrar alguns dos conceitos discutidos na seção anterior, considere dois tópicos a serem explorados para ajudar na aferição da qualidade de modelos preditivos: a escolha da estratégia de treinamento, validação e teste, e a medida aplicada para avaliar o desempenho do modelo.

No exemplo didático apresentado para o algoritmo *k*-NN com a base *Pesquisa de Satisfação*, o exemplar identificado por `R1` foi tomado como exemplar de teste para ilustrar os passos do algoritmo em um processo de classificação. Naquele momento, pode ter parecido estranho classificar um exemplar que já possuia a informação de classe. Nesta seção, foi visto que se trata de uma estratégia para analisar o desempenho de um modelo. Entretanto, apenas fazer um teste simples como esse não é suficiente, pois

imagine que justamente o exemplar escolhido para teste esteja em uma região do espaço em que a classificação seja facilmente resolvida. Uma estratégia mais robusta de avaliação deve ser empregada, como a validação cruzada do tipo *leave-one-out*. Uma vez que se trata de um conjunto de dados com poucos exemplares, a estratégia se apresenta apropriada.

Seguindo essa estratégia, para fins de exemplificação, considere o uso da medida de avaliação *Acurácia* (Equação 3-18), e o próprio exemplo didático apresentado para o *k*-NN (Seção 3.1.1.1), no qual a distância adotada foi a euclidiana, $k = 3$, e o conjunto de dados é o *Pesquisa de Satisfação*. A classe esperada para `R1` é `SAT`, e a classe predita pelo modelo é `SAT` (*Acurácia* = 1). Para os demais testes, os resultados são apresentados na Figura 3-36. Ao final do processo, a acurácia média é calculada como forma de avaliar o desempenho do classificador sobtido com a configuração de parâmetros escolhida.

teste	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}
y	1	1	1	1	1	1	2	2	2	2	2	2
y^	1	2	2	1	1	1	2	2	1	1	2	1
Acurácia	1	0	0	1	1	1	1	1	0	0	1	0

Acurácia média = 58,33%

Figura 3-36: Esquema gráfico referente à execução da estratégia *leave-one-out* sobre a aplicação do *k*-NN no conjunto de dados *PESQUISA DE SATISFAÇAO*

Já nos exemplos didáticos construídos para ilustrar a predição numérica – regressão linear e regressão não linear (Seção 3.2.1.1 e Seção 3.2.2.1) –, dois preditores foram construídos para o mesmo problema, o *Planejamento de Propagandas*. Uma questão que pode ser levantada é qual dos dois modelos de predição é o mais adequado para o problema, e a resposta pode ser encontrada por meio da aplicação de medidas de avaliação. O resultado da aplicação das medidas de erro absoluto e erro quadrado às predições realizadas pelos dois modelos, considerando uma estratégia de resubstituição,

é apresentado na Tabela 3-13. Analisando os erros obtidos, o modelo de regressão linear é considerado o mais adequado.

Tabela 3-13: Resultado comparativo de regressão linear e regressão não linear

x	**y**	**saída estimada – ŷ**		**erro absoluto – ea**		**erro quadrado – eq**	
# inserções do anúncio	**# Filés à parmegiana vendidos**	**ŷ_linear**	**ŷ_nlinear**	**ea_linear**	**ea_nlinear**	**eq_linear**	**eq_nlinear**
30	430	409,14	388,45	20,86	41,55	435,14	1726,75
21	335	321,50	299,21	13,50	35,79	182,30	1280,75
35	520	457,83	449,06	62,17	70,94	3865,11	5032,65
42	490	526,00	550,13	36,00	60,13	1295,71	3615,57
37	470	477,31	475,87	7,31	5,87	53,38	34,51
20	210	311,76	290,66	101,76	80,66	10355,10	6506,02
8	195	194,90	205,23	0,10	10,23	0,01	104,75
17	270	282,55	266,44	12,55	3,56	157,40	12,66
35	400	457,83	449,06	57,83	49,06	3344,31	2406,77
25	480	360,45	336,01	119,55	143,99	14292,20	20731,87
		erros médios		43,16	50,18	3398,07	4145,23

3.3.4. Exemplo prático para a construção e avaliação de modelos preditivos em R

No experimento feito na Seção 3.1.2.2 com o conjunto de dados *Pesquisa de Satisfação*, um procedimento para a criação dos conjuntos de treinamento e de teste (`splitForTrainingAndTest( )`) foi realizado. Nesse ponto já se sabe que essa estratégia consiste na estratégia *holdout*. No entanto, caso fosse necessário utilizar a metodologia de validação cruzada sob a variação *leave-one-out* ou *k-fold*, poder-se-ia utilizar o pacote `cvTools( )` (Alfons, 2012). Nesse pacote, a função `cvFolds( )` permite gerar uma distribuição de exemplares para os diferentes subconjuntos (*folds*) desejados. O resultado da

função é uma matriz que armazena, em cada linha, duas informações: o número do subconjunto e o índice do exemplar (posição no conjunto de dados original) associado a ele. A sintaxe e parâmetros da função `cvFolds( )` são apresentados no Quadro 3-11.

Quadro 3-11: Sintaxe e parâmetros da função *CVFOLDS*

Sintaxe para a função para o método de validação cruzada *k-folds*
Usando a função `cvFolds( )` do pacote *cvTools*
Construindo os conjuntos (*folds*) para validação cruzada `folds <- cvFolds(N, K, tipo)` • `N` é o número de exemplares disponível no conjunto de dados. • `K` é a quantidade de subconjuntos (*folds*) que se deseja gerar. • `tipo` é o método que pode ser usado para executar a distribuição dos exemplares nos subconjuntos (*folds*): (*consecutive*, *random* (*default*), *interleaved*). A função retorna um vetor com a distribuição dos exemplares nos subconjuntos (*folds*).

Para demonstrar o uso da função `cvTools( )`, considere o conjunto de dados *Pesquisa de Satisfação* (Tabela 3-1). Desde que esse conjunto possui 12 exemplares, a sintaxe para a construção dos subconjuntos (*folds*), considerando $K = 4$ e a escolha aleatória dos exemplares, e o resultado da execução da função são:

```
> folds <- cvFolds(12, K = 4, type = "random")
> folds

4-fold CV:
Fold Index
1 12
2 1
3 7
4 8
1 9
2 4
3 10
4 6
1 2
2 11
```

```
3 3
4 5
```

Para acessar os subconjuntos formados (`folds`), deve-se usar `folds$which`, sendo que `which( )` é uma função utilizada para retornar o índice de *arrays*. Por outro lado, para recuperar o valor dos índices dos exemplares, representado no resultado por `Index`, deve-se usar `folds$subsets`. Então, para a construção efetiva de cada subconjunto de exemplares, deve-se definir como condicional de pesquisa no `folds$which` o número do subconjunto sendo construído. A instrução a seguir recupera os índices dos exemplares que fazem parte do subconjunto 1:

```
> folder_1 <- folds$subsets[which(folds$which == 1)]
```

Agora, `folder_1` é um conjunto de índices de exemplares que pode ser usado para recuperar os exemplares em si. Por exemplo, considerando que `X` é o conjunto de dados original, a recuperação dos valores para a formação do subconjunto de dados `folder_1` será:

```
> X_folder_1 <- X[folds$subsets[which(folds$which == 1)],]
```

A fim de implementar a estratégia de validação cruzada, sugere-se usar esse subconjunto (`X_folder_1`) como subconjunto de teste da primeira execução, repetindo o processo com outros valores para `folds$which` para as demais execuções. Seguindo essa sugestão, o subconjunto de treinamento será formado com o restante dos exemplares (que não entraram no subconjunto de teste). A construção desse subconjunto, por sua vez, é realizada recuperando os índices dos exemplares dos outros subconjuntos da variável `folds`, ou seja, os exemplares usados no treinamento do modelo podem ser recuperados da seguinte maneira:

```
> Xtrain <- X[folds$subsets[which(folds$which != 1)],]
```

Ainda, o pacote **RSNNS** (Bergmeir e Benitez, 2012), usado na implementação apresentada na Seção 3.1.2.2, dispõe de funcionalidades de construção de matriz de confusão (`confusionMatrix( )`) e gráficos ROC (`plotROC( )`). A sintaxe e parâmetros relacionados com cada uma dessas funções são apresentados, respectivamente, no Quadro 3-12 e no Quadro 3-13.

Quadro 3-12: Sintaxe e parâmetros para a função CONFUSIONMATRIX

Sintaxe para a função que gera a matriz de confusão
Usando a função `confusionMatrix( )` do pacote *RSNNS*
Construindo uma matriz de confusão `matriz <-confusionMatrix(saídas_desejadas, saídas_estimadas)` • `saídas_desejadas` é um vetor com as saídas desejadas, geralmente do conjunto de teste. • `saídas_estimadas` é um vetor com as saídas estimadas pelo classificador, geralmente do conjunto de teste. A função retorna um objeto contendo a matriz de confusão.

Quadro 3-13: Sintaxe e parâmetros para a função PLOTROC

Sintaxe para gerar a função que gera um gráfico com a curva ROC
Usando a função `plotROC ( )` do pacote *RSNNS*
Plotando a curva ROC `roc <-plotROC(saídas_desejadas, saídas_estimadas)` • `saídas_desejadas` é um vetor com as saídas desejadas, geralmente do conjunto de teste. • `saídas_estimadas` é um vetor com as saídas estimadas pelo classificador, geralmente do conjunto de teste. A função gera um gráfico ROC.

O uso dessas funções e seus respectivos resultados (incluindo o gráfico ROC na Figura 3-37, gerado apenas em relação ao teste aplicado no classificador), considerando o discutido no exemplo prático da Seção 3.1.2.2, é como segue:

```
> predictions <- predict(modelo_mlp,d_separado$inputsTest)
> confusionMatrix(d_separado$targetsTest,predictions)

predictions
```

```
targets 1 2
1 1 0
2 0 1

> plotROC(d_separado$targetsTest,predictions)
```

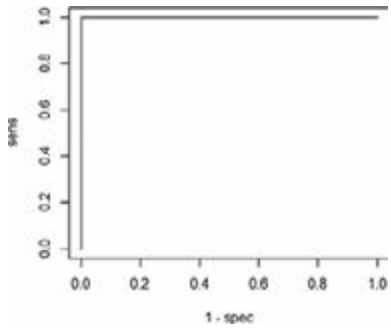


Figura 3-37: Gráfico com a curva ROC para a predição no conjunto de teste

3.4. Leituras adicionais

Os estudos realizados no contexto de análise preditiva estão relacionados com diferentes áreas do conhecimento. Na área de Redes Neurais Artificiais , por exemplo, a discussão sobre problemas de classificação foi muito importante no início do seu desenvolvimento. O conceito de separabilidade linear está muito relacionado com ambas as áreas, Redes Neurais Artificiais e Análise Preditiva, e é indicado ao leitor que se aprofunde no tema. Uma discussão bastante didática sobre esse tema é apresentada em Fausett (1993).

Outro tópico que precisa ser explorado com cuidado são as medidas de distância . Alguns dos algoritmos usados para indução de modelos preditivos têm a sua tomada de decisão baseada no conceito de distâncias (isso vale também para o caso de algoritmos usados na resolução da tarefa de agrupamento – Capítulo 4). A apresentação de inúmeras medidas de distância é encontrada em Deza e Deza (2009).

Na discussão sobre metodologias para construção e avaliação de modelos preditivos, realizadas neste capítulo, foram abordadas algumas estratégias referentes ao uso de conjuntos de treinamento, validação e teste. Essas estratégias são discutidas em mais detalhes por Han, Kamber e Pei (2011), principalmente no que diz respeito ao tamanho dos conjuntos obtidos na estratégia *bootstrap*. Também, foram apresentados alguns recursos especialmente úteis para a avaliação de classificadores binários, entre eles, o Gráfico ROC. Sobre essa forma de avaliação, é importante dizer que a discussão apresentada aqui está bastante resumida. Esse gráfico tem o potencial de gerar muito conhecimento para a avaliação de classificadores se interpretado sob diferentes óticas. Em Fawcett (2006), é apresentada uma introdução ao uso desse recurso de avaliação na qual o autor discute: o espaço ROC (equivalente ao gráfico ROC apresentado neste livro), a curva ROC e suas propriedades, o ROC *convex hull*, a área sob a curva ROC (muito

conhecida sob seu acrônico em inglês – AUC) e como trabalhar com o espaço ROC na avaliação de classificadores multiclasse.

Nesse contexto de avaliação de classificadores, há de se considerar que muitos dos problemas reais a serem resolvidos sob a perspectiva de mineração de dados apresentarão uma amostragem (um conjunto de dados) desbalanceada em relação à distribuição de classes. Medidas como o *f-score* são mais adequadas para a avaliação de situações nas quais há desbalanceamento de classes; contudo, alterações das estratégias de avaliação podem ser realizadas para melhorar as avaliações e, consequentemente, a própria concepção dos modelos preditivos. Um exemplo útil nesse contexto é usar uma matriz de custo associada aos erros cometidos pelo modelo. Veja uma descrição sobre esse assunto em Witten, Frank e Hall (2011) e Beyan e Fisher (2014).

Ainda, é necessário mencionar que há técnicas usadas para a construção de modelos preditivos, capazes de gerar apenas classificadores binários . Essas técnicas, no entanto, podem ser usadas na solução de problemas multiclasse, mas, nesse caso, os modelos binários precisam ser construídos e combinados dentro de estratégias, especialmente modeladas para suprir a limitação de tratamento do problema multiclasse. As estratégias mais conhecidas são: *um-versus-um* e *um-versus-todos*. Em Lima (2004), são discutidas essas e outras possibilidades correlatas a esse tema.

Para melhorar o desempenho de modelos preditivos, há a possibilidade de construção de *essembles* e misturas de especialistas . *Essembles* são compostos por um conjunto de múltiplos modelos preditivos, construídos para resolver um mesmo problema. A partir da composição das saídas individuais de tais modelos, obtém-se uma única saída para o problema, geralmente mais assertiva que cada uma das saídas individuais. Misturas de especialistas também envolvem a construção de múltiplos modelos preditivos, contudo, nesse caso, cada modelo é responsável por resolver uma parte do problema sob análise. Discussões mais específicas sobre esse tema

são apresentadas em Lima (2004), Hansen e Salamon (1990), Jacobs *et al*. (1991) e Jordan e Jacobs (1994).

Outro ponto muito importante na construção de modelos preditivos é a estratégia usada para definir os parâmetros dos modelos (os parâmetros livres dos algoritmos, como o *k*, no *k-NN*, ou o número de neurônios na camada escondida da rede neural artificial *Multilayer Perceptron*). Para cada parâmetro de cada algoritmo, com certeza, há uma série de estudos propostos na literatura especializada. É importante salientar que todos os parâmetros dos algoritmos precisam ser estudados com cuidado, e seus efeitos no desempenho do modelo devem sempre ser explorados.

Por fim, há ainda uma série de assuntos interessantes, relacionados com a análise preditiva, não tratados neste livro. Alguns desses assuntos constituem-se como tópicos muito conhecidos e muito usados atualmente, como Máquinas de Vetores Suporte (Lima, 2004; Haykin, 2008; Cristianini e Shawe-Taylor, 2000); Classificação Multirótulo (Tsoumakas e Katakis, 2007); *Deep Learning* (Arel, Rose e Karnowski, 2010); *Active Learning* (Settles, 2010); *Transfer Learning* (Pan e Yang, 2010; Dai *et al*., 2007).

3.5. Exercícios

1. Para cada um dos conjuntos de dados citados a seguir, disponíveis no repositório UCI – *Machine Learning* (Lichman, 2013), responda: (a) para resolver o problema de predição categórica ou predição numérica (identifique o problema) relacionado com o contexto de cada conjunto, qual algoritmo, dentre os tratados neste capítulo, você entende ser o mais adequado? Justifique sua resposta considerando a teoria que você aprendeu e argumente em favor de sua escolha, relacionando aspectos do algoritmo escolhido com características do conjunto de dados presentes na respectiva descrição ou em artigos científicos correlatos a ele (tanto a descrição quanto as referências bibliográficas dos artigos correlatos podem ser também acessadas no repositório); (b) considerando que se trata de um problema de predição, seja qual for a técnica escolhida para resolvê-lo, será necessário avaliar os resultados obtidos. Comente sobre uma forma adequada de estimar o erro de predição para o problema em questão em cada conjunto de dados.

a) Bank Marketing Data Set (Moro, Cortez e Rita, 2014) **(http://archive.ics.uci.edu/ml/datasets/Bank+Marketing)**

b) Computer Hardware Data Set (Lichman, 2013) **(http://archive.ics.uci.edu/ml/datasets/Computer+Hardware)**

c) Libras Movement Data Set (Dias *et al*., 2009) **(http://archive.ics.uci.edu/ml/datasets/Libras+Movement)**

d) Zoo Data Set (Lichman, 2013) **(http://archive.ics.uci.edu/ml/datasets/Zoo)**

2. Os gestores de uma faculdade se interessaram em construir um modelo preditivo para que os alunos do terceiro ano de um curso de graduação o utilizassem com o objetivo de estimar se conseguiriam se formar em cinco

anos (dentro do prazo) ou em sete anos (fora do prazo). O comportamento do aluno, então, é usado como dado – cada aluno é um exemplar no conjunto de dados. As variáveis (atributos descritivos) disponíveis para análise são notas, horas de atividades realizadas extra classe (atividades complementares) e horas de estágio. A preparação do conjunto de dados se deu a partir dos alunos já formados, para que, desta forma, se soubesse a situação de cada um ao final dos cinco (ou sete) anos do curso. Desses alunos, os valores para os seguintes atributos foram coletados: nota média em todas as disciplinas cursadas, com ou sem aprovação (ME); total de horas em atividades complementares (AC); total de horas de estágio (ES) e a situação final em relação à formatura (SIT), assumindo D, quando o aluno se formou dentro do prazo, e F, quando o aluno se formou fora do prazo. O conjunto de dados está disponível na Tabela 3-14.

Tabela 3-14: Conjunto de dados *COMPORTAMENTO DO ALUNO*

	x_{i1}	x_{i2}	x_{i3}	y_i
	ME	**AC**	**ES**	**SIT**
$\mathbf{x}\rightarrow_1$	6,0	150	300	D
$\mathbf{x}\rightarrow_2$	7,0	165	150	D
$\mathbf{x}\rightarrow_3$	6,2	100	380	D
$\mathbf{x}\rightarrow_4$	5,5	150	301	D
$\mathbf{x}\rightarrow_5$	4,8	110	308	D
$\mathbf{x}\rightarrow_6$	3,5	88	50	F
$\mathbf{x}\rightarrow_7$	4,0	150	100	F
$\mathbf{x}\rightarrow_8$	6,0	120	80	F

Para esse problema e conjunto de dados, pede-se:

a) O modelo resultante da execução do algoritmo de indução de árvore de decisão, usando como critério de seleção de atributo para compor um nó interno: o primeiro atributo da lista de atributos; o melhor atributo da lista de atributos em termos de ganho de informação.

b) Uma interpretação para a importância de cada um dos atributos descritivos sob o ponto de vista da tomada de decisão de um aluno em relação ao que fazer para finalizar sua graduação dentro do prazo.

c) A realização de um teste sobre os modelos preditores obtidos, considerando a situação de um aluno que possui os seguintes valores para os atributos descritivos: 4, 130 e 200.

d) Exercite o mesmo processo de análise preditiva, porém, agora usando o algoritmo *Naïve* Bayes.

e) Compare os dois processos: indução da árvore de decisão e execução do processo *Naïve* Bayes, ressaltando similaridades e diferenças.

3. Um professor, preocupado com o desempenho final dos alunos em sua disciplina, a qual tem altos índices de reprovação, decidiu monitorar o desempenho de oito turmas em termos de horas (média) de estudo, registrado em um Ambiente Virtual de Apoio ao Ensino Presencial, e a média final da turma ao fim do semestre. Esses dados estão disponíveis na Tabela 3-15. Com eles, o professor deseja estimar a média da turma do semestre corrente, já que a hora média de estudo de cada aluno pode ser facilmente extraída pelo sistema. Portanto, pede-se que você modele o problema, usando regressão linear e regressão não linear (exponencial), e apresente:

a) A equação de cada modelo.

b) Um gráfico formado pelos exemplares do conjunto de dados no eixo x e o valor estimado pelo modelo no eixo y.

c) Uma avaliação dos modelos obtidos considerando erro médio e erro médio quadrado.

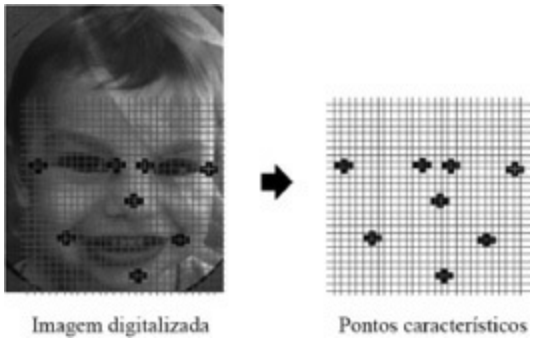
d) Previsão da média da turma na prova final, considerando que a turma tem 2,3 horas de estudo, e usando o modelo que melhor representa o conjunto de dados.

Tabela 3-15: Conjunto de dados *ACOMPANHAMENTO DOS ALUNOS*

	x_{i1}	y
	Horas (média) de estudo da turma	**Nota média da turma**
$\mathbf{x}_1$	1,0	4,2
$\mathbf{x}_2$	1,5	4,5
$\mathbf{x}_3$	3,0	4,8
$\mathbf{x}_4$	4,0	5,2
$\mathbf{x}_5$	4,5	6,0
$\mathbf{x}_6$	5,0	6,2
$\mathbf{x}_7$	6,0	7,0
$\mathbf{x}_8$	6,5	7,5

4. Reconhecimento facial é um tipo de aplicação na qual se pretende identificar pessoas em uma imagem digital que contém faces. Para o reconhecimento, é fundamental que medidas de faces sejam coletadas. Como exemplo, considere o processo de reconhecimento de faces em 2D, ilustrado na Figura 3-38. A partir de técnicas de processamento de imagens digitais, são extraídos os pontos característicos da face, que, por sua vez, permitem o cálculo de medidas armazenadas em um conjunto de dados como atributos descritivos de faces na Figura 3-38(b). Esse conjunto de dados pode ser usado como entrada para um algoritmo que resolve um problema de classificação. O Facebook, por exemplo, faz uso do

reconhecimento facial para marcar rostos em fotografias postadas na rede social.



(a)

OLHO ESQUERDO	OLHO DIREITO	NARIZ	BOCA	CLASSE
6	6	1	2	F
1	3	3	4	F
2	10	8	4	N
8	8	6	4	N

(b)

Figura 3-38: (a) processo de medição da face humana; (b) medidas extraídas

Como exercício, considere o conjunto de dados apresentado na Figura ٣٨-٣, no qual estão armazenadas as medidas entre os cantos dos olhos, entre a ponta do nariz e o queixo e entre os cantos (esquerdo e direito) da boca. Cada exemplo deste conjunto de dados é rotulado como sendo face – F ou não face – N. Então, aplique o algoritmo *k*-NN para classificar um novo exemplar (Tabela ١٦-٣). No seu experimento, considere a distância euclidiana e k = ٣.

Tabela 3-16: Exemplar de teste: nova face?

	OLHO ESQUERDO	OLHO DIREITO	NARIZ	BOCA	CLASSE
$\mathbf{x}_{\rightarrow teste}$	2	6	3	8	?

5. No exemplo didático para a rede neural artificial *Multilayer Perceptron*, foi apresentado o cálculo para o ajuste no peso sináptico w_{f1}. Agora, faça os cálculos para a atualização dos demais pesos sinápticos, ainda considerando uma iteração do treinamento.

6. No Capítulo 2, foi solicitado que o leitor fizesse uma análise exploratória sobre o conjunto de dados *Optical Recognition of Handwritten Digits* (Kaynak, 1995), disponível no repositório UCI *Machine Learning* (Lichman, 2013). A descrição desse conjunto de dados informa que ele está disponível na forma "bruta" (*raw*) e na forma pré-processada, essa última com dimensionalidade reduzida. E ainda, os dados já estão separados em dois conjuntos, uma para treinamento e um para teste. Sobre esse conjunto de dados, realize um experimento comparativo (use o R para isso) sobre o algoritmo *k*-NN e a rede neural artificial *Multilayer Perceptron*. O resultado de cada modelo classificador obtido deve ser apresentado sob a forma de uma matriz de confusão. Na sequência, discuta os valores obtidos em cada matriz. Lembre-se de que a escolha dos parâmetros de cada algoritmo faz parte do problema, e a apresentação dos resultados obtidos no processo de escolha dos parâmetros, na forma de tabelas ou gráficos, é incentivada nesse exercício.

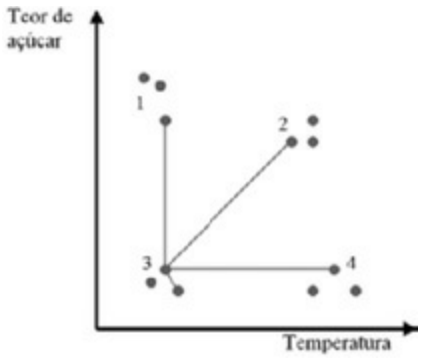
7. O problema dessa questão consiste na construção de um modelo preditivo que auxilie na avaliação de um carro antes da compra. Um conjunto de dados nesse contexto está disponível no repositório UCI – *Machine Learning* (Lichman, 2013) – *Car Evaluation Data Set* https://archive.ics.uci.edu/ml/datasets/Car+Evaluation. Usando R, desenvolva classificadores capazes de realizar essa avaliação. Use a estratégia de *cross-validation* (use 4-*folds*) e os algoritmos de árvore de decisão e *Naïve* Bayes para analisar o conjunto de dados. Faça análises comparativas dos resultados obtidos. O resultado individual de cada classificador deve ser apresentado em um diagrama de caixa (*boxplot*).

CAPÍTULO 4

ANÁLISE DE AGRUPAMENTO

A **análise de agrupamentos** (comumente chamada de *clustering* ou, simplesmente, “agrupamento ”) pode ser entendida como um processo que permite descobrir relações existentes entre exemplares de um conjunto de dados descritos por uma série de características (atributos descritivos). De forma diferente da análise preditiva, não há a presença de um rótulo associado (atributo de classe) ao exemplar, de forma que as relações procuradas serão reveladas por meio do estudo apenas das descrições sobre os eventos presentes no domínio de análise (por exemplo, dados que descrevem os pedidos realizados em um restaurante ou que descrevem o processo de prestação de serviço do restaurante).

Em geral, as análises realizadas pelos algoritmos que implementam estratégias para agrupamento buscam por similaridades ou diferenças entre exemplares, quantificadas por meio de medidas de distância (quanto menor for a distância entre dois exemplares, maior será a similaridade), tal que exemplares similares sejam associados a um mesmo grupo, e exemplares dissimilares, a grupos diferentes. Ao final da execução de um algoritmo de agrupamento, uma estrutura de grupos é formada de maneira que a similaridade intragrupos tenha sido maximizada, e a similaridade intergrupos tenha sido minimizada. A Figura 4-1 ilustra como ocorre a correspondência entre medidas de distância entre exemplares de um conjunto de dados e a formação de grupos.



Grupo 1: alimentos doces que devem ser servidos frios.

Grupo 2: alimentos doces que devem ser servidor quentes.

Grupo 3: alimentos salgados que devem ser servidos frios.

Grupo 4: alimentos salgados que devem ser servidos quentes.

Figura 4-1: Correspondência entre medidas de distância e a formação de grupos

Nessa figura, os alimentos estão representados por duas características: temperatura em que devem ser servidos e teor de açúcar. Repare que alimentos similares possuem valores similares para seus atributos descritivos, e a noção de distância (menor entre exemplares similares) pode ser percebida pela observação do comprimento das linhas entre os exemplares.

A descoberta de um agrupamento se constitui, portanto, como um processo de tomada de decisão sobre a associação de um exemplar a um grupo, sendo que a similaridade entre os exemplares escolhidos para compor um mesmo grupo seja maximizada, isto é, a tarefa de análise de agrupamento é frequentemente modelada por meio de um problema de minimização de medidas de distância intragrupos. Quando o processo de minimização é feito por um algoritmo de Aprendizado de Máquina, é considerado um processo indutivo e não supervisionado – também conhecido como *treinamento não supervisionado* ou *aprendizado não supervisionado* .

O modelo de grupos gerado no processo descrito é uma solução na tarefa de mineração de dados de análise de agrupamento. De fato, a saída de um algoritmo de agrupamento de dados constitui-se como um fim em si, não sendo necessariamente previsto um uso do modelo em um processo de *teste*, como se faz na análise preditiva. Ao receber o modelo de grupos, o analista de dados tem em mãos a descoberta do conhecimento que deve ser útil para entendimento do perfil dos eventos sob análise (por exemplo, em um

restaurante italiano, o modelo de grupos pode dividir os pedidos em quatro principais grupos: pedidos que incluem massas ao molho à carbonara e com frutos do mar; pedidos que incluem massas ao molho de tomate; pedidos que incluem massas ao molho de carne; e pedidos que incluem pizzas).

A partir da descoberta desse conhecimento, porém, se faz útil a realização de um segundo estudo, no qual se buscam explicações que ajudem os usuários do modelo a entender os motivos pelos quais os subconjuntos de exemplares são similares. Essa é uma análise posterior à tarefa de análise de grupos e, geralmente, exige que especialistas no domínio de aplicação trabalhem com analistas de dados. No caso do restaurante, por exemplo, uma análise cuidadosa dos atributos descritivos que mais pesaram para a tomada de decisão sobre a indicação dos tais quatro grupos, bem como a observação dos valores assumidos por tais atributos, pode levar à conclusão de que os exemplares foram colocados em tais grupos por conta da harmonização de seus ingredientes com os elementos que compõem as bebidas que os clientes apreciaram com cada prato – nesse caso, respectivamente, vinho branco, vinho rosé, vinho tinto, cervejas e refrigerantes.

O leitor pode estar se perguntando se esse modelo de grupos não poderia ter um uso para além dessa análise. A resposta é “sim” e, neste caso, se encaixaria o uso do modelo de grupos em um processo de *teste*, no qual novos exemplares, até então desconhecidos no contexto, podem ser associados a algum grupo por similaridade. Por exemplo, se uma massa com um novo e exótico molho passa a ser servida no restaurante, muito provavelmente os clientes não saberão exatamente como escolher o melhor tipo de bebida para acompanhá-la, e, na ausência do *maître*, o modelo de grupos pode ser usado para verificar a qual grupo o novo molho melhor se encaixa (por meio da minimização da distância entre os valores dos atributos que descrevem o novo molho e os dos atributos que descrevem cada um dos grupos) e, então, indicar o tipo de bebida que melhor harmoniza com ele, por similaridade.

A quantidade de domínios de aplicação da análise de agrupamento é muito vasta; além disso, assim como no caso de análise preditiva, tentativas de nomeá-los resultarão em um conjunto reduzido de possibilidades. Alguns exemplos de domínio de aplicação seriam: análise de perfil de usuários e perfil de itens para sistemas de recomendação; análise de padrões de comportamento de multidões; identificação de grupos de risco para empresas seguradoras; análise de emoções em redes sociais; reconhecimento de padrões em imagens de satélite ou imagens médicas; análise de padrões de cliques em páginas da internet etc.

Durante o processo de indução de um modelo de grupos, e também ao seu término, é comum fazer algumas estimativas de qualidade do modelo em formação ou formado. Dado que na execução da tarefa de agrupamento tem-se apenas informação descritiva sobre os dados, essas estimativas são realizadas de forma a validar o modelo sendo formado. Essa validação pode ser implementada de diferentes maneiras, desde que se tenha como princípio avaliar se o problema de maximização de similaridade intragrupos e minimização de similaridade intergrupos está sendo resolvido a contento.

Neste capítulo, são discutidos aspectos teóricos e práticos, necessários ao entendimento de como pode se dar a resolução da tarefa de análise de agrupamento. As estratégias de resolução dessa tarefa podem ser divididas em diferentes categorias. As categorias tratadas aqui são: estratégias hierárquicas, estratégias por partição, estratégias baseadas em densidade e estratégias conexionistas. Para cada uma delas, respectivamente, é apresentado um algoritmo: métodos AGNES e DIANA, *k*-médias, DBSCAN e Mapas Auto-Organizáveis. Exemplos didáticos e implementações em R são fornecidos para cada um dos algoritmos. Ao final do capítulo, são sugeridos alguns exercícios para fixação dos conteúdos apresentados e leituras adicionais, nas quais informações mais específicas sobre análise de agrupamento podem ser obtidas.

4.1. A tarefa de agrupamento

Segunda Rocha *et al*. (2012), “o termo ‘grupo’ deve ser usado quando não existe qualquer informação sobre como é a organização dos dados.” Assim, comumente, denomina-se **agrupamento** o processo pelo qual se estuda as relações de similaridade entre os exemplares, determinando como estão organizados em grupos. Assumindo a definição de conjunto de dados, do Capítulo 1, diz-se que, formalmente, a tarefa de agrupamento pode ser descrita como a busca por uma função G, capaz de mapear um conjunto $\mathbf{X}$ de vetores de entrada (exemplares) $\mathbf{x}\rightarrow \in E^d$ para um conjunto finito de grupos que minimiza a distância *dist* entre elementos dentro dos grupos. A função G é definida como G: $E^d \times W \rightarrow C$, em que d é a dimensão do espaço E, ou seja, o número de coordenadas do vetor $\mathbf{x}\rightarrow$, W é um espaço de parâmetros ajustáveis por meio de um algoritmo de indução não supervisionada e $W = arg\_min_W\ dist(\mathbf{x}\rightarrow_p, \mathbf{x}\rightarrow_q)$, sendo p e q os índices de dois exemplares quaisquer e distintos associados a um mesmo grupo.

Também, tradicionalmente, as seguintes condições devem ser respeitadas na resolução da tarefa de agrupamento, em que c é o número de grupos no modelo de grupos resultante:

$$C_k \neq \emptyset, k = 1, \ldots, c;$$

$$\bigcup_{k=1}^{c} C_k = \mathbf{X};$$

$$C_k \cap C_l = \emptyset,\ k, l = 1, \ldots, c \text{ e } k \neq l.$$

As estratégias de resolução da tarefa de agrupamento podem ser divididas em categorias, a depender do tipo de regras usadas em sua concepção, e várias categorias podem ser encontradas na literatura da área de Mineração de Dados. Neste livro, três delas são exploradas:

- Estratégias hierárquicas : os exemplares são divididos hierarquicamente em grupos. Se uma abordagem *top-down* for adotada, o processo de análise se inicia colocando todos os exemplares em um único grupo e, iterativamente, dividindo um grupo em grupos menores, até que cada exemplar seja isolado em um grupo. Essa abordagem também pode ser encontrada na literatura como abordagem *divisiva* . Por outro lado, se uma abordagem *bottom-up* for adotada, o processo de análise inicia com cada exemplar em um grupo separado e, iterativamente, aglomera grupos similares, até que um único grupo com todos os exemplares seja formado. Essa abordagem também pode ser encontrada na literatura como abordagem *aglomerativa* . Os métodos AGNES e DIANA (Kaufman e Rousseeuw, 1990) são os representantes dessa categoria de estratégias neste livro e são discutidos na Seção 4.1.1.
- Estratégias por partição : partições (que representam os grupos) do espaço, no qual o conjunto de dados está inserido, são criadas de acordo com um critério de particionamento. Então, iterativamente, os exemplares são realocados entre as partições de forma que o modelo de grupos mude e se ajuste melhor ao objetivo de maximização de similaridade intragrupo. O algoritmo *k*-médias (MacQueen, 1967; Lloyd, 1982) é o representante dessa categoria de estratégias neste livro e é discutido na Seção 4.1.2.
- Estratégias baseadas em densidade : grupos com um (ou poucos) exemplar(es) são inicialmente formados e, iterativamente, recebem mais exemplares (localizados na vizinhança do grupo) e crescem até que um limiar seja atingido. O algoritmo DBSCAN (Ester *et al*., 1996) é o representante dessa categoria de estratégias neste livro e é discutido na Seção 4.1.3.

Além dos algoritmos citados, a rede neural artificial Mapas Auto-Organizáveis (Kohonen, 1982), também conhecida como *Self Organizing*

Maps (SOM), é discutida neste livro, na Seção 4.1.4, como um algoritmo conexionista para resolução da tarefa de agrupamento. De fato, SOM tem ainda outra aplicação – a visualização de conjuntos de dados de alta dimensão, já que é capaz de executar projeções dos dados em espaços de baixa dimensão.

4.1.1. Agrupamento hierárquico – AGNES e DIANA

As **estratégias de agrupamento hierárquicas** podem ser implementadas a partir de uma de duas abordagens: aglomerativa ou divisiva. A primeira, aglomerativa, pode ser implementada pelo método AGNES (do inglês, *A Glomerative NESting*), e a segunda, divisiva, pode ser implementada pelo método DIANA (do inglês, *DIvisive ANAlysis*). Esses métodos são apresentados nesta seção a fim de ilustrar como se dá o processo de agrupamento sob estratégias hierárquicas. Contudo, existem vários outros métodos que implementam agrupamento hierárquico. Algumas leituras adicionais sobre esses métodos são sugeridas na Seção 4.3.

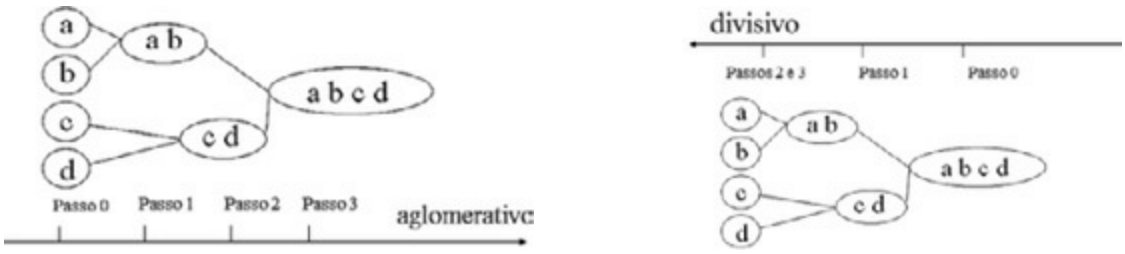
Basicamente, o agrupamento hierárquico é obtido a partir da criação de uma estrutura em árvore (uma estrutura de dados hierárquica) na qual os exemplares $\boldsymbol{x}_{\rightarrow i}$ de um conjunto de dados são definidos como nós folhas, e os nós internos revelam uma organização baseada na similaridade entre os exemplares. Comumente, a árvore é representada graficamente por um dendograma, como será apresentado neste texto. Em cada nível da árvore, tem-se uma organização dos dados em grupos. O processo de construção da árvore pode ser feito da forma aglomerativa ou da forma divisiva, usando, respectivamente, os métodos AGNES e DIANA:

- Método AGNES: a construção da árvore é iniciada pelos nós folhas. Cada exemplar (cada um associado a uma folha) é alocado, inicialmente, a um grupo distinto, ou seja, cada exemplar representa um grupo distinto. Então, iterativamente, os grupos são fundidos a partir de

algum critério de similaridade aplicado aos exemplares que os constituem. Assim, pares de grupos serão fundidos se seus exemplares forem os mais similares entre si, considerando todo o conjunto de grupos disponíveis. Esse processo dá origem aos nós internos da árvore e deve ser repetido até que não haja grupos para serem aglomerados, ou seja, até que todos os exemplares formem um único grupo no nó raíz da árvore.

- Método DIANA : a construção da árvore é iniciada pelo nó raiz. Todos os exemplares são alocados, inicialmente, a um único grupo. Então, iterativamente, os grupos são divididos de acordo com algum critério de dissimilaridade, aplicado aos exemplares que os constituem. Assim, enquanto houver grupos formados por mais de um exemplar, dois grupos distintos são criados a cada divisão, dando origem aos demais nós internos da árvore. Nos casos em que uma divisão gera um grupo com um único exemplar, o nó correspondente é um nó folha da árvore. O processo deve ser repetido até que cada exemplar do conjunto de dados esteja alocado a um grupo distinto.

Na Figura 4-2, são mostrados dois esquemas gráficos da execução dos dois processos de agrupamento hierárquico para um conjunto de dados que possui quatro exemplares.



(a) (b)

Figura 4-2: Esquema gráfico para ilustrar o processo de agrupamento hierárquico aglomerativo (a) e divisivo (b)

O critério de similaridade (ou dissimilaridade) deve ser aplicado sempre a pares de exemplares ou pares de grupos e pode ser implementando usando uma medida de distância, de forma que, quanto menor for o valor que quantificar a distância, mais similares serão os exemplares sob comparação. No caso dos exemplos deste livro, a distância euclidiana é utilizada (Capítulo 3).

Para a aplicação da medida de distância a pares de exemplares, faz-se útil a criação de uma matriz de similaridade que relaciona todos os exemplares do conjunto de dados. Trata-se de uma matriz de tamanho $n \times n$, em que n é o número de exemplares no conjunto de dados. Cada célula dessa matriz corresponde à medida de distância $dist\ \mathbf{x}\rightarrow_p, \mathbf{x}\rightarrow_q$ calculada a partir dos exemplares $\mathbf{x}\rightarrow_p$ e $\mathbf{x}\rightarrow_q$. A diagonal principal dessa matriz deve sempre assumir o valor 0 (zero), pois representa a distância de um exemplar a ele mesmo, e a similaridade entre eles é máxima (ou seja, são iguais). Além disso, a matriz é simétrica, pois a distância entre dois exemplares p e q quaisquer é sempre igual, ainda que a ordem de apresentação na função de distância seja diferente ($dist\ \mathbf{x}\rightarrow_p, \mathbf{x}\rightarrow_q = dist\ \mathbf{x}\rightarrow_q, \mathbf{x}\rightarrow_p$). A estrutura dessa matriz é apresentada na Figura 4-3.

$$MS = \begin{bmatrix} 0 & & & & \\ dist_{\vec{x}_2,\vec{x}_1} & 0 & & & \\ dist_{\vec{x}_3,\vec{x}_1} & dist_{\vec{x}_3,\vec{x}_2} & 0 & & \\ \vdots & \vdots & \vdots & 0 & \\ dist_{\vec{x}_n,\vec{x}_1} & dist_{\vec{x}_n,\vec{x}_2} & \cdots & dist_{\vec{x}_n,\vec{x}_{n-1}} & 0 \end{bmatrix}$$

Figura 4-3: Representação de uma matriz de similaridade (MS)

Para aplicação da medida de distância a pares de grupos,[1] quatro abordagens são possíveis:

- Menor distância (*single-linkage* ou *single-link* ou *agrupamento de vizinhos*): a distância entre dois grupos é dada de acordo com a distância entre os seus exemplares mais próximos.

- Maior distância (c*omplete-linkage* ou *complete-link*): a distância entre dois grupos é dada de acordo com a distância entre os seus exemplares mais distantes.
- Distância média (*average-linkage* ou *average-link*): a distância entre dois grupos é dada pela média das distâncias entre todos os exemplares de um grupo e todos os exemplares do outro grupo.
- Distância de centroides : a distância entre dois grupos é dada de acordo com a distância entre seus centroides.

A partir dos conceitos referentes à matriz de similaridade e abordagem de aplicação de medidas de distância entre grupos, é possível apresentar os algoritmos para os métodos AGNES (Algoritmo 4-1) e DIANA (Algoritmo 4-2).

Algoritmo 4-1: Algoritmo para agrupamento hierárquico aglomerativo – método AGNES

Parâmetros de entrada:
- $\mathbf{X}_{tr}$: um conjunto de exemplares não rotulado de treinamento, ou seja, $\mathbf{X}_{tr} = \{\mathbf{x}_{\rightarrow i}\}$, $i = 1, ..., n$;
- *dist*: medida de distância;
- abordagem para cálculo da distância entre grupos;

Parâmetro de saída:
- ***D***: árvore (ou dendograma) representando a organização dos exemplares em grupos;

Passo 1: calcule a matriz de similaridade (MS); % opcional
Passo 2: aloque cada exemplar $\mathbf{x}_{\rightarrow i}$ de $\mathbf{X}_{tr}$ em um grupo distinto, criando os nós folhas da árvore ***D***;
Passo 3: **enquanto** há possibilidade de fusão de grupos **faça**
Passo 3.1: verifique a distância entre todos os pares de grupos, usando a matriz de similaridade ou calculando a distância entre os centroides dos grupos (a depender da abordagem escolhida);
Passo 3.2: encontre o par de grupos mais similares e os transforme em um único grupo, criando um nó interno na hierarquia da árvore ***D***;

Algoritmo 4-2: Algoritmo para agrupamento hierárquico divisivo – método DIANA

Parâmetros de entrada:
- $\mathbf{X}_{tr}$: um conjunto de exemplares não rotulado de treinamento, ou seja, $\mathbf{X}_{tr} = \{\mathbf{x}_{\rightarrow i}\}$, $i = 1 ..., n$;
- *dist*: medida de distância;

- abordagem para cálculo da distância entre grupos;

Parâmetro de saída:
- ***D***: árvore (ou dendograma) representando a organização em grupos;
Passo 1: calcule a matriz de similaridades (MS); % opcional
Passo 2: aloque todos os exemplares $\mathbf{x}_{\rightarrow i}$ de $\mathbf{X}_{tr}$ em um único grupo, criando o nó raiz da árvore ***D***;
Passo 3: **enquanto** há possibilidade de divisão de grupos **faça**
Passo 3.1: selecione o grupo ***G***, de maior dispersão[2] dentre todos os grupos disponíveis para divisão;
Passo 3.2: dentro de ***G***, encontre o exemplar $\mathbf{x}_{\rightarrow dif}$ menos similar a todos os outros (de maior distância aos demais exemplares de seu grupo);
Passo 3.3: transfira $\mathbf{x}_{\rightarrow dif}$ de ***G*** para um novo grupo $\boldsymbol{G_{split}}$;
Passo 3.4: **enquanto** ocorrer transferência de ***G*** para $\boldsymbol{G_{split}}$ **faça**
Passo 3.4.1: dentro de ***G***, encontre o exemplar $\mathbf{x}_{\rightarrow dif}$ menos similar a todos os outros (de maior distância aos demais exemplares de seu grupo);
Passo 3.4.2: **se** a similaridade entre $\mathbf{x}_{\rightarrow dif}$ e $\boldsymbol{G_{split}}$ é maior que a similaridade entre $\mathbf{x}_{\rightarrow dif}$ e $(\boldsymbol{G} - \mathbf{x}_{\rightarrow dif})$ **então**
Passos 3.4.2.1: transfira $\mathbf{x}_{\rightarrow dif}$ de ***G*** para $\boldsymbol{G_{split}}$;
Passo 3.5: organize a árvore ***D***, criando dois novos nós na sua hierarquia (para $\boldsymbol{G_{split}}$ e ***G***), iniciando pelo grupo de maior dispersão, sendo que grupos formados por mais de um exemplar devem constituir-se como um nó interno da árvore ***D***, e grupos formados por apenas um exemplar devem constituir-se como um nó folha na árvore ***D***.

Para exemplificar a execução de um processo para agrupamento hierárquico, considere o disposto no Algoritmo 4-1 – método AGNES, a abordagem para cálculo da distância entre grupos "menor distância", e o conjunto de dados de seis exemplares distribuídos no espaço euclidiano, como apresentado na Figura 4-4.

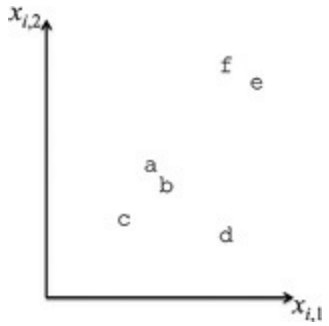


Figura 4-4: Conjunto de dados com seis exemplares (letras) dispostos no espaço euclidiano

O *Passo 1* deste processo será omitido, já que não se têm as apresentação dos valores dos atributos x_{i1} e x_{i2} que descrevem os exemplares, e as distâncias entre os elementos devem ser observada visualmente na Figura 4-4. No *Passo 2*, os exemplares do conjunto de dados são alocados em grupos distintos {{a}, {b}, {c}, {d}, {e}, {f}}. Note que esse passo é equivalente a encontrar os grupos cujos elementos possuem distância 0 (zero) entre si. A primeira iteração do laço (*Passo 3*) pressupõe a identificação dos dois exemplares mais similares. Na investigação visual, vê-se facilmente que {a} e {b} são os grupos mais próximos e, portanto, devem ser agrupados. Nesse ponto da execução do algoritmo, os seguintes grupos são formados: {{a,b}, {c}, {d}, {e}, {f}}. A segunda aglomeração se dá com os grupos {e} e {f}, resultando nos grupos ({a,b}, {c}, {d}, {e,f}). Na iteração seguinte, os grupos mais próximos, {c} e {a,b}, serão fundidos, resultando na seguinte organização de grupos: {{a,b,c}, {d}, {e,f}}. Nas iterações restantes, as seguintes organizações de grupos serão obtidas: {{a,b,c,d}, {e,f}}, e, então, {{a,b,c,d,e,f}}.

Na Figura 4-5, são mostradas duas representações gráficas nas quais é possível observar as organizações de grupos formadas a partir do início de execução do laço (*Passo 3*). A primeira representação (Figura 4-5(a)) usa o próprio espaço euclidiano, e a segunda representação (Figura 4-5(b)) usa um dendograma . As espessuras e tonalidades das linhas indicam a correspondência entre as duas figuras no que diz respeito aos grupos formados durante o processo de agrupamento hierárquico.

(a) (b)

Figura 4-5: Representações gráficas para o resultado do processo implementado pelo método AGNES: (a) representação no espaço euclidiano; (b) representação em dendograma

Importante notar que a visualização em dendograma organiza os exemplares por similaridade, de maneira bastante útil. A partir desta organização, é possível decidir qual é a distância que se quer assumir como o limiar admitido para formar um grupo e, consequentemente, determinar o número de grupos (*k*) desejados para a organização final do agrupamento. Alternativamente, a decisão pode ser tomada exclusivamente a partir do número desejado de grupos. Na Figura 4-6, são apresentados cortes no dendograma que representam a escolha sobre a distância máxima dentro de um grupo e o número de grupos resultantes dessa decisão. Observe na figura que cada corte leva a uma organização de grupos diferentes: o corte em $k = 1$ gera o grupo {{a,b,c,d,e,f}}; $k = 2$ gera os grupos {{a,b,c,d},{e,f}}; $k = 3$ gera os três grupos {{a,b,c},{d},{e,f}}; $k = 4$ gera os quatro grupos, {{a,b},{c},{d},{e,f}}; $k = 5$ gera os grupos {{a,b},{c},{d},{e},{f}}. O caso de $k = 6$ não está representado, pois, na realidade, ele leva à apresentação do conjunto de dados original, sem uma organização em grupo.

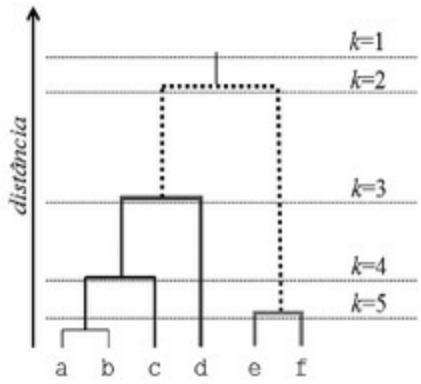


Figura 4-6: Exemplificação de cortes no dendograma

4.1.1.1. Exemplo didático para AGNES

Como exemplo didático para contextualizar o uso do algoritmo de agrupamento de dados hierárquico, considere a abordagem aglomerativa com *menor distância* e o seguinte cenário: com o objetivo de descobrir o perfil de seus clientes, o gerente de um restaurante deseja fazer uma análise, verificando a relação entre a quantidade média de vezes em que o cliente frequenta o restaurante no mês, o valor médio gasto por ele nesse mesmo período e a forma de pagamento predominantemente utilizada. Como resultado dessa verificação, o gerente espera descobrir os perfis de comportamento dos seus clientes e usar a informação para criar formas de tratamento diferenciadas, direcionando as campanhas publicitárias do estabelecimento. O objetivo do gerente é aumentar a frequência do cliente, fidelizando-o e fazendo-o consumir mais, ainda que seja necessário tomar medidas adicionais como firmar novas parcerias com diferentes operadoras de cartões de crédito.

Uma amostra dos dados que o gerente tem sobre seus clientes é listada na Tabela 4-1 (a). A frequência é dada pelo número médio de vezes que cada cliente visita o restaurante por mês, o consumo é o gasto médio do cliente por mês e o pagamento diz respeito à forma como o cliente efetua os pagamentos na maioria das vezes em que visita o restaurante (sendo 1 – cartão de crédito e 2 – cartão de débito). O conjunto original foi pré-processado, e, portanto, os atributos quantitativos numéricos foram normalizados, e o atributo qualitativo nominal foi codificado (*string* de bits, veja Capítulo 2). O conjunto de dados resultante, usado na execução do exemplo, é mostrado na Tabela 4-1 (b). A matriz de similaridade é mostrada na Tabela 4-2.

Tabela 4-1: Conjunto de dados *PERFIL DE CONSUMO*: (a) conjunto original; (b) conjunto pré-processado

		x_{i1}	x_{i2}	x_{i3}
	ID	**FREQUÊNCIA**	**CONSUMO**	**PAGAMENTO**
$\mathbf{x}_{\rightarrow 1}$	cli1	3	120,5	1
	cli2	3	130,0	1

		x_{i1}	x_{i2}	x_{i3}
	ID	**FREQ*norm***	**CONS*norm***	**PAG*c***
$\mathbf{x}_{\rightarrow 1}$	cli1	0,67	0,67	0
	cli2	0,67	0,77	0

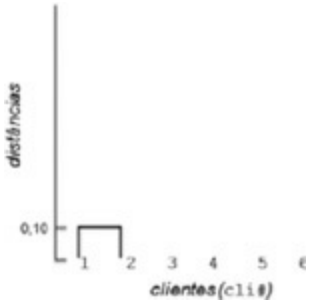
$\mathbf{x}_{\rightarrow 2}$						$\mathbf{x}_{\rightarrow 2}$			
$\mathbf{x}_{\rightarrow 3}$	cli3	4	150,8	1		$\mathbf{x}_{\rightarrow 3}$	cli3	1,00	1,00	0
$\mathbf{x}_{\rightarrow 4}$	cli4	2	75,6	2		$\mathbf{x}_{\rightarrow 4}$	cli4	0,33	0,18	1
$\mathbf{x}_{\rightarrow 5}$	cli5	3	88,5	2		$\mathbf{x}_{\rightarrow 5}$	cli5	0,67	0,32	1
$\mathbf{x}_{\rightarrow 6}$	cli6	1	58,8	2		$\mathbf{x}_{\rightarrow 6}$	cli6	0,00	0,00	1

ID: identificação de um cliente.
FREQUÊNCIA: número médio de vezes que o cliente frequenta o estabelecimento.
CONSUMO: valor médio consumido pelo cliente em um mês.
PAGAMENTO: forma de pagamento usada nas compras (duas formas possíveis).

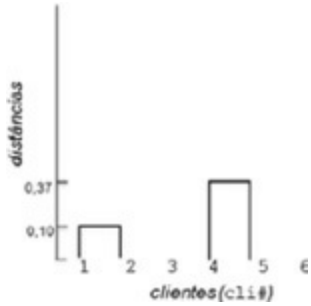
Para o conjunto de dados *Perfil de Consumo*, a matriz de similaridade é dada na Tabela 4-2. Seguindo os passos apresentados no Algoritmo 4-1, as organizações de grupos obtidas a cada iteração do laço (*Passo 3*) são: {{`cli1`, `cli2`}, {`cli3`}, {`cli4`}, {`cli5`}, {`cli6`}}; {{`cli1`, `cli2`}, {`cli3`}, {`cli4`, `cli5`}, {`cli6`}}; {{`cli1`, `cli2`}, {`cli3`}, {`cli4`, `cli5`, `cli6`}}; {{`cli1`, `cli2`, `cli3`}, {`cli4`, `cli5`, `cli6`}}; {{`cli1`, `cli2`, `cli3`, `cli4`, `cli5`, `cli6`}}. Veja cada uma dessas organizações representadas por meio de dendogramas na Figura 4-7.

Tabela 4-2: Matriz de similaridade para os exemplares do conjunto de dados *PERFIL DE CONSUMO*

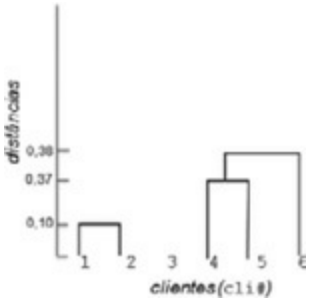
	`cli1`	`cli2`	`cli3`	`cli4`	`cli5`	`cli6`
`cli1`	0,0					
`cli2`	0,10	0,0				
`cli3`	0,47	0,40	0,0			
`cli4`	1,16	1,21	1,46	0,0		
`cli5`	1,06	1,10	1,25	0,37	0,0	
`cli6`	1,38	1,43	1,73	0,38	0,74	0,0



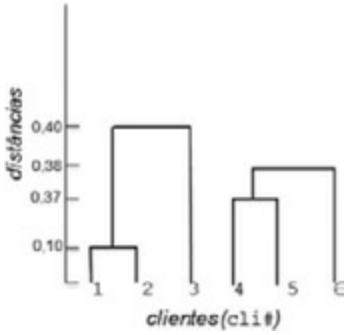
(a) Clientes mais similares: `cli1`, `cli2`; distância de 0,10.



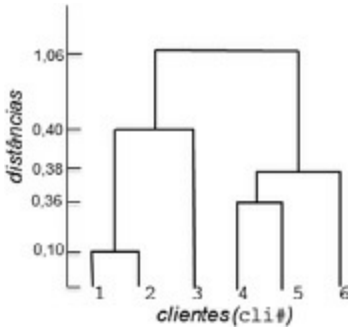
(b) Clientes mais similares `cli4`, `cli5`; distância de 0,37.



(c) Cliente `cli6` mais similar a `cli4`, já agrupado com `cli5`; distância de 0,38.



(d) Cliente `cli3` mais similar a `cli2`, já agrupado com `cli1`; distância de 0,40.



(e) Cliente `cli1` mais similar a `cli5`; distância de 1,06. Neste caso, `cli1` já está agrupado com `cli2` e `cli3`, e `cli5` já está agrupado com `cli4` e `cli6`.

Figura 4-7: Sequência de organizações de grupos representadas por dendogramas

Nesse exemplo, a descoberta de dois grupos pode ser uma boa escolha considerando o cenário do gerente do restaurante. Assumindo essa tomada de decisão, um corte no dendograma em qualquer distância no intervalo 0,40 < *distâncias* ≤ 1,06 (Figura 4-7(d)) permite a geração dos grupos: {`cli1`,`cli2`, `cli3`}, {`cli4`,`cli5`, `cli6`}.

Nesse ponto do exemplo de contexto, a dificuldade está em como interpretar o resultado fornecido pelo algoritmo de agrupamento: Qual grupo de clientes deverá ser alvo de qual estratégia de publicidade? Trata-se de uma pergunta que deve ser respondida a partir de uma análise mais detalhada sobre que características descrevem cada um dos grupos de clientes formados e como tais perfis de clientes devem ser tratados. O exemplo didático, em sua

simplicidade, torna essa tarefa relativamente fácil: um grupo de clientes que costuma visitar o restaurante mais vezes, fazendo pedidos mais caros e pagando com cartão de crédito; outro grupo de clientes que vai ao restaurante com menos frequência, consome menos e realiza os pagamentos com cartão de débito. Talvez o gerente do restaurante queira aproveitar o perfil dos clientes do primeiro grupo e possibilitar pagamentos parcelados no cartão de crédito, apostando que esse perfil é de clientes que poderiam consumir ainda mais mediante tais condições.

4.1.1.2. Exemplo prático do AGNES em R

O agrupamento hierárquico aglomerativo em R é realizado em dois passos. No primeiro deles, calcula-se a matriz de similaridade com o uso da função `dist( )`, disponível no pacote ***stats*** (R Core Team, 2015), nativo no R. A sintaxe e parâmetros para a função `dist( )` estão apresentados no Quadro 4-1.

Quadro 4-1: Sintaxe e parâmetros da função DIST

Sintaxe para aplicação da função que constrói a matriz de similaridade
Usando a função `dist( )` do pacote *stats*
Construindo a matriz de similaridade `matriz_similaridade <- dist(dados, método)` • `dados` é um ***data.frame*** que contém dados numéricos para realização dos cálculos de distância. • `método` é a definição de qual é a medida de distância a ser aplicada, podendo ser euclidiana (“*euclidean*”) ou outras, como Manhattan, cosseno etc. A função retorna uma variável contendo a matriz de similaridade considerando todos os exemplares em `dados`.

No segundo passo, a construção da hierarquia com a organização dos grupos é realizada com o uso da função `hclust( )`, também disponível no pacote ***stats***. A sintaxe e os parâmetros para a função `hclust( )` são apresentados no Quadro 4-2.

Quadro 4-2: Sintaxe e parâmetros da função HCLUS

Sintaxe para aplicação da função que implementa agrupamento hierárquico aglomerativo
Usando a função `hclust( )` **do pacote *stats***
Construindo o agrupamento hierárquico aglomerativo `agrupamento <- hclust(matriz_similaridade, método)` • `matriz_similaridade` é a matriz de similaridade entre os exemplares de um conjunto de dados. • `método` é a definição de qual é a abordagem a ser utilizada para calcular a distância entre grupos, podendo ser menor distância ("*single*"), maior distância ("*complete*") ou distância média ("*average*"). A função retorna uma variável contendo informações sobre a organização de grupos encontrada.

O uso das funções apresentadas no Quadro 4-1 e Quadro 4-2, considerando o conjunto de dados *Perfil de Consumo* (Tabela 4-1(b)), é:

```
> consumo <- read.table
("C:\\Users\\LivroDM\\Agrupamento\\consumo.csv",
header=FALSE,sep=";")
> ms <- dist(consumo , method = "euclidean")
> agrupamento <- hclust(ms, method="single")
```

A visualização do resultado de agrupamento por meio de dendogramas pode ser feita com uso da função `plot( )`, como especificado a seguir. A Figura 4-8(a) mostra o dendograma gerado.

```
> plot (agrupamento, main = "Agrupamento Hierárquico
Aglomerativo",
xlab = "Clientes", ylab = "Distância")
```

Ainda, é possível representar os grupos no dendograma. Dois resultados são possíveis a partir do uso de funções do pacote ***stats***: o primeiro é obtido por meio do uso da função `cutree( )`, que gera um vetor no qual cada elemento identifica o grupo ao qual cada exemplar do conjunto de dados foi atribuído; e o segundo, obtido por meio da função `rect.hclust( )`, que gera o dendograma segmentado de acordo com os grupos formados. A sintaxe e os

parâmetros para as funções `cutree( )` e `rect.hclust( )` são mostrados no Quadro 4-3 e Quadro 4-4, respectivamente.

Quadro 4-3: Sintaxe e parâmetros da função CUTREE

Sintaxe para aplicação da função que implementa representação de grupos do dendograma – considerando geração de vetores
Usando a função `cutree( )` do pacote *stats*

Construindo representações de grupos – geração de vetores

```
grupos <- cutree(agrupamento, k)
```

- `agrupamento` é a variável resultante da aplicação da função `hclust( )`.
- `k` é um número inteiro que representa a quantidade de segmentos (grupos) desejados.

A função retorna uma variável com os exemplares do conjunto de dados separados em *k* grupos.

Quadro 4-4: Sintaxe e parâmetros da função RECT.HCLUST

Sintaxe para aplicação da função que implementa representação de grupos no dendograma – considerando cortes no dendograma
Usando a função `rect.hclust( )` do pacote *stats*

Construindo representações de grupos – cortes no dendrograma

```
rect.hclust(agrupamento, k, borda)
```

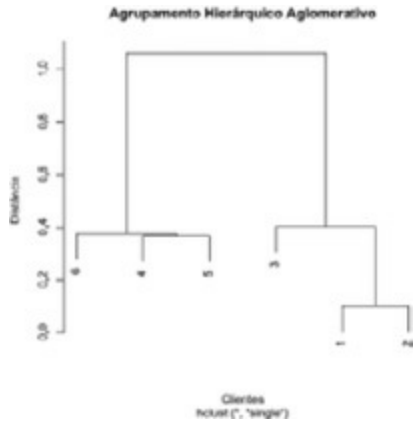
- `agrupamento` é a variável resultante da aplicação da função `hclust( )`.
- `k` é um número inteiro que representa a quantidade de segmentos (grupos) desejados.
- `borda` é a cor da linha desenhada na visualização do dendrograma.

A função imprime o dendrograma com corte no número de grupos desejado.

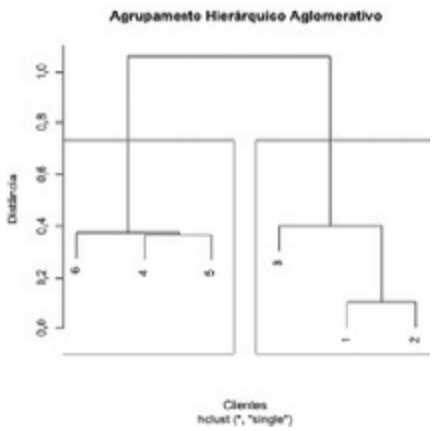
O uso das funções apresentadas no Quadro 4-3 e Quadro 4-4 é:

```
> grupos <- cutree(agrupamento,k=2)
> grupos
[1] 1 1 1 2 2 2
> rect.hclust( agrupamento , k=2, border="red")
```

e a visualização resultante pode ser observada na Figura 4-8(b).



(a) Visualização do dendograma referente ao agrupamento hierárquico realizado pela função HCLUST



(b) Dendograma segmentado gerado a partir da função RECT.HCLUST

Figura 4-8: Dendogramas gerados sobre o conjunto *PERFIL DE CONSUMO*

4.1.2. Agrupamento por partição – *k*-médias

Dentro do conjunto de algoritmos para agrupamento de dados que seguem a estratégia de **agrupamento por partição** , o *k*-médias (do inglês *k-means*) é o mais popular. Esse algoritmo tem como objetivo encontrar partições no conjunto de dados de forma que *k* grupos disjuntos de exemplares sejam descobertos, sendo que *k* é um parâmetro de entrada para o algoritmo. A busca pela descoberta do conjunto de *k* partições é iterativa e classicamente iniciada a partir da escolha aleatória de *k* vetores distintos que têm o papel de representar centroides (ou protótipos) para grupos. Como esses vetores devem representar grupos de exemplares, eles devem estar localizados no mesmo espaço, ou seja, os vetores *k* possuem *d* coordenadas, em que *d* é o número de atributos descritivos do conjunto de dados. A partir da escolha inicial para os centroides, o algoritmo procede verificando quais exemplares são mais similares a quais centroides, ajustando os centroides aos exemplares mais similares a eles e, dessa forma, estabelecendo as partições. Os passos desse processo são descritos no Algoritmo 4-3.

Algoritmo 4-3: Algoritmo para agrupamento por partição – *k*-médias

Parâmetros de entrada:

- $\mathbf{X}_{tr}$: conjunto de exemplares não rotulado de treinamento, ou seja, $\mathbf{X}_{tr} = \{\mathbf{x}\rightarrow_i\}$, $i = 1, ..., n$;
- k: o número de partições (ou grupos) a serem descobertos;
- *dist*: medida de distância;

Parâmetro de saída:
- k vetores que representam centroides de grupos, representando as partições descobertas;
Passo 1: escolha aleatoriamente um conjunto de vetores distintos para representar os centroides, ou seja, $\mathbf{C} = \{(\boldsymbol{c}\rightarrow_p), p = 1, ..., k\}$
Passo 2: **enquanto** houver alterações nas associações dos exemplares aos grupos representados por cada centroide **faça**
Passo 2.1: verifique a distância $dist(\mathbf{x}\rightarrow_i, \mathbf{c}\rightarrow_p)$ para cada exemplar i em n e cada centroide p em k;
Passo 2.2: associe cada exemplar $\mathbf{x}\rightarrow_i$ ao vetor $\boldsymbol{c}\rightarrow_p$ que minimiza $dist(\mathbf{x}\rightarrow_i, \mathbf{c}\rightarrow_p)$, formando cada uma das k partições do conjunto $\mathbf{X}_{tr}$;
Passo 2.3: atualize o conjunto de centroides $\mathbf{C}$, de forma que cada vetor $\mathbf{c}\rightarrow$ seja a média dos vetores $\mathbf{x}\rightarrow$ associados a ele.

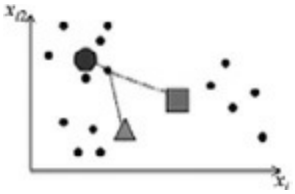
Esse processo está ilustrado na Figura 4-9. Um conjunto de dados com 15 exemplares não rotulados é apresentado na Figura 4-9(a). Cada exemplar é representado por um ponto nessa figura. Nesse exemplo, o número de partições procurado é três ($k = 3$), e a distância escolhida foi a euclidiana. Na Figura 4-9(b), os três vetores de centroides são aleatoriamente inicializados no espaço dos exemplares e estão arbitrariamente representados pelas formas geométricas círculo, quadrado e triângulo. Ainda nessa figura, a distância euclidiana entre um dos exemplares e os três centroides é representada por meio de linhas pontilhadas. Na Figura 4-9(c), a associação de cada exemplar a uma partição é representada por meio da forma geométrica do seu respectivo centroide. Cada partição é composta pelos exemplares mais próximos a cada um dos centroides. Finalmente, na Figura 4-9(d), os centroides são reposicionados na média dos exemplares a eles associados, de forma a, de fato, posicionarem-se no centro de cada partição. Para calcular a nova posição de cada centroide, suas coordenadas recebem o valor médio das coordenadas dos exemplares associados a ele (ou o valor da mediana ou o valor da moda – veja observações sobre isso na Seção 4.3), seguindo:

$$\vec{\mathbf{c}}_p = \frac{1}{G}\sum_{g=1}^{G}\vec{\mathbf{x}}_g$$

Equação 4-1

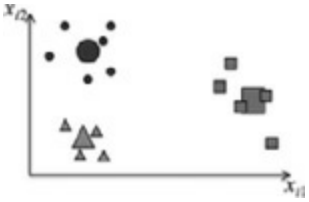
em que $\mathbf{x}\rightarrow_g$ é um exemplar do conjunto de dados associado ao centroide $\mathbf{c}\rightarrow_p$, sendo que p = 1 ... k, e G é o número de exemplares associados a um centroide.

(a) Conjunto de exemplares submetido a um processo de análise de agrupamento.



(b) *Passo 1*: inicialização dos centroides, sendo k = 3; *Passo 2.1*: verificação das distâncias entre exemplares do conjunto de dados e vetores centroides.

(c) *Passo 2.2*: associação dos exemplares aos centroides mais próximos.



(d) *Passo 2.3*: atualização dos centroides para a média dos grupos que eles representam.

Figura 4-9: Ilustração dos passos do algoritmos *k*-médias

As ações dos *Passos 2.1*, *2.2* e *2.3* (Algoritmo 4-3) devem se repetir até que a atualização dos centroides se estabilize, ou seja, até que não haja mais alterações nas associações dos exemplares aos grupos representados por cada centroide. Alternativamente, duas ações podem ser implementadas de modo diferente: a inicialização dos centroides pode ser feita a partir da seleção aleatória de *k* exemplares; a condição de parada do processo pode ser substituída por um número máximo de iterações.

O particionamento produzido pelo algoritmo *k*-médias, representado pelos centroides em suas configurações finais, é dependente de dois fatores que representam as duas grandes dificuldades desse algoritmo: (a) a escolha do número de partições, de fato uma informação *a priori*, fornecida pelo usuário; (b) a inicialização dos vetores (centroides), que pode comprometer o resultado final do algoritmo. A primeira dificuldade pode ser minimizada por estratégias que permitem comparar os resultados produzidos a partir de diferentes execuções do algoritmo com diferentes valores atribuídos a *k*. A segunda dificuldade pode ser amenizada com a realização de análises de densidade e de distâncias (Ostrovsky *et al*., 2006; Arthur e Vassilvitskii, 2007). Por fim, é interessante salientar que o *k*-médias tem um viés para a segmentação de grupos compactos (Seção 4.2) e de distribuição esférica.

4.1.2.1. Exemplo didático para o *k*-médias

O problema didático para ser resolvido pelo *k*-médias trata do particionamento de um conjunto de dados referente a avaliações de clientes com relação ao atendimento no restaurante. Nesse caso, está sendo considerado o tempo de preparo e as notas que os clientes atribuem à refeição. O conjunto de dados *Eficiência de Atendimento*, já normalizado, é apresentado na Tabela 4-3. O problema consiste em identificar os perfis de atendimento e caracterizar cada perfil de acordo com os valores que os atributos descritivos assumem em cada um. A depender do conhecimento descoberto, ações de adequação de atendimento podem ser realizadas. No caso deste exemplo, deseja-se descobrir dois perfis de atendimento ($k = 2$), e a distância adotada é a euclidiana.

Tabela 4-3: Conjunto de dados *EFICIÊNCIA DE ATENDIMENTO*

		x_{i1}	x_{i2}
	ID	**TEMPO**	**QUALIDADE**
$\mathbf{x}_{\rightarrow 1}$	atend1	0,00	0,57

$\mathbf{x}_{\rightarrow 2}$	atend2	0,50	0,43
$\mathbf{x}_{\rightarrow 3}$	atend3	1,00	0,00
$\mathbf{x}_{\rightarrow 4}$	atend4	0,50	0,86
$\mathbf{x}_{\rightarrow 5}$	atend5	1,00	1,00
$\mathbf{x}_{\rightarrow 6}$	atend6	1,00	0,57

ID: identificação de um atendimento.
TEMPO: tempo de preparo de uma refeição.
QUALIDADE: nota que o cliente atribui à refeição consumida.

Como descrito no Algoritmo 4-3, o primeiro passo é definir os *k* vetores de centroides iniciais. Aleatoriamente, os seguintes vetores foram determinados: $\mathbf{c}_{\rightarrow 1}$ ={0, 0,29} e $\mathbf{c}_{\rightarrow 2}$ ={1, 0,71}. Seguindo o processo iterativo do *k*-médias, o conjunto de dados é particionado e os centroides são reposicionados até que atinjam estabilidade. Para ilustrar o cálculo de distância dos exemplares aos centroides (*Passo 2.1* no Algoritmo 4-3), o exemplar $\mathbf{x}_{\rightarrow 1}$ do conjunto de dados foi escolhido. As distâncias entre esse exemplar e os centroides $\mathbf{c}_{\rightarrow 1}$ e $\mathbf{c}_{\rightarrow 2}$ são:

$$dist_{\vec{x}_1,\vec{c}_1} = \sqrt{(x_{11}-c_{11})^2+(x_{12}-c_{12})^2} = \sqrt{(0-0)^2+(0{,}57-0{,}29)^2} = 0{,}28$$

$$dist_{\vec{x}_1,\vec{c}_2} = \sqrt{(x_{11}-c_{21})^2+(x_{12}-c_{22})^2} = \sqrt{(0-1)^2+(0{,}57-0{,}71)^2} = 1{,}01$$

Após o cálculo da distância (considerando cada um dos centroides), o exemplar $\mathbf{x}_{\rightarrow 1}$ é associado ao cen- troide com a menor distância, nesse caso: $\mathbf{c}_{\rightarrow 1}$. Todas as distâncias e associações entre exemplares e centroides obtidas ao final da execução da primeira iteração (*t* = 1) são apresentadas na Tabela 4-4. As células sombreadas representam os agrupamentos formados na iteração.

Tabela 4-4: Resultados referentes a distâncias e agrupamento entre exemplares e centroides ao final da primeira iteração

	atend1	atend2	atend3	atend4	atend5	atend6
$\mathbf{c}_{\rightarrow 1}$	0,28	0,52	1,04	0,76	1,23	1,04
$\mathbf{c}_{\rightarrow 2}$	1,01	0,57	0,71	0,52	0,29	0,14

De acordo com o exposto na Tabela 4-4, os grupos estão organizados da seguinte forma:

- Grupo representado pelo centroide $\mathbf{c}_{\rightarrow 1}$: G_1 = {atend1, atend2}.
- Grupo representado pelo centroide $\mathbf{c}_{\rightarrow 2}$: G_2 = {atend3, atend4, atend5, atend6}.

A partir dessa organização dos exemplares nos dois grupos, os centroides devem ser atualizados de forma a se tornarem os vetores médios de cada partição (Equação 4-1). As atualizações dos centroides $\mathbf{c}_{\rightarrow 1}$ e $\mathbf{c}_{\rightarrow 2}$ seguem o exposto na Tabela 4-5. Com os centroides atualizados, a condição de parada deve ser observada. Seguindo o exposto no Algoritmo 4-3, o algoritmo deve continuar enquanto ocorrerem, na iteração, alterações na distribuição dos exemplares entre os grupos. Considerando que, nessa iteração, houve alterações (na realidade, a primeira formação de grupo foi realizada), o algoritmo ainda não pode ser considerado estabilizado e será necessário executar novas iterações do algoritmo, agora considerando os valores de centroides atualizados. O leitor pode realizar as próximas iterações como um exercício.

Tabela 4-5: Alteração de centroides na primeira iteração do *k*-médias

ID	TEMPO	QUALIDADE	ID	TEMPO	QUALIDADE
atend1	0,00	0,57	atend3	1,00	0,00
atend2	0,50	0,43	atend4	0,50	0,86
$\mathbf{c}_{\rightarrow 1}$	0,25	0,5	atend5	1,00	1,00

atend6	1,00	0,57
$\mathbf{c}_{\rightarrow 2}$	0,88	0,61

4.1.2.2. Exemplo prático para o *k*-médias em R

A apresentação da implementação do algoritmo *k*-médias em R está baseada no uso da função `kmeans( )`, disponível no pacote ***stats*** (R Core Team, 2015). Outras implementações para este algoritmo estão disponíveis no repositório CRAN (*Comprehensive R Archive Network*). A sintaxe e os parâmetros referentes ao uso da função `kmeans( )` são apresentados no Quadro 4-5.

Quadro 4-5: Sintaxe e parâmetros da função KMEANS

Sintaxe para uma função que implementa o algoritmo *k*-médias
Usando a função `kmeans( )` do pacote *stats*
Construindo o agrupamento por particionamento `grupos <-kmeans(dados, k, iteração)` • `dados` é um **data.frame** contendo dados numéricos para serem submetidos ao processo de agrupamento. • `k` é um número inteiro que indica o número de grupos a ser descoberto. • `iteração` é o número máximo de iterações permitido no algoritmo. A função retorna uma variável contendo informações sobre a organização de grupos encontrada.

O uso da função `kmeans( )`, considerando o conjunto de dados *EFICIÊNCIA DE ATENDIMENTO* e *k* = 2, é da seguinte forma:

```
> atendimento =
read.table("C:\\Users\\LivroDM\\Agrupamento\\eficiencia_atendiment
csv", header=FALSE, sep=";")
> perfis <- kmeans(atendimento,2,iter.max=5)
```

Após a descoberta dos grupos, a função `kmeans( )` retorna uma variável com as seguintes informações: o número de exemplares associados a cada grupo (`perfis$size`), os centroides finais na forma de uma matriz de médias

dos exemplares associados a cada grupo (`perfis$centers`) e o índice do grupo ao qual cada exemplar foi associado (`perfis$cluster`). A sintaxe para acessar o conteúdo da variável `perfis$cluster` é exemplificada a seguir, e o resultado significa que o primeiro, segundo e quarto exemplares pertencem ao grupo 2 e o terceiro, quinto e sexto exemplares pertencem ao grupo 1.

```
> atendimento_perfis <- perfis$cluster
[1] 2 2 1 2 1 1
```

4.1.3. Agrupamento por densidade – DBSCAN

As estratégias de **agrupamento por densidade** são especialmente úteis para aplicação em conjuntos de dados com um grande número de exemplares e também para descoberta de grupos de formatos arbitrários. Um dos principais algoritmos que segue a estratégia de analisar aspectos relacionados com densidade para determinar os grupos é o DBSCAN (do inglês, *Density Based Spatial Clustering of Applications with Noise*). Esse algoritmo, proposto nos anos 1990 por Ester *et al*. (1996), é caracterizado por guiar o processo de descoberta de grupos com base na densidade de exemplares existentes na vizinhança[3] de um exemplar já pertencente a um grupo. A densidade nessa vizinhança deve ser igual ou maior que um limite mínimo para que tais exemplares possam ser considerados parte do grupo associado ao exemplar referencial.[4] Esse algoritmo é atrativo porque, como seu próprio nome diz, é capaz de lidar com ruído presente nos dados. Os passos que implementam a estratégia usada no DBSCAN são mostrados no Algoritmo 4-4.

Algoritmo 4-4: Algoritmo para agrupamento por densidade – *DBSCAN*

Parâmetros de entrada:
- $\mathbf{X}_{tr}$: é um conjunto de exemplares não rotulado de treinamento, ou seja, $\mathbf{X}_{tr} = \{\mathbf{x}_{\rightarrow i}\}$, $i = 1, ..., n$;
- *minEx*: número mínimo de exemplares exigido na verificação da densidade de vizinhança;
- *r*: raio de vizinhança de um exemplar;
- *dist*: medida de distância entre dois exemplares;

Parâmetro de saída:
- *G*: uma lista com os exemplares agrupados ou indicação de exemplares definidos como ruído pelo algoritmo;

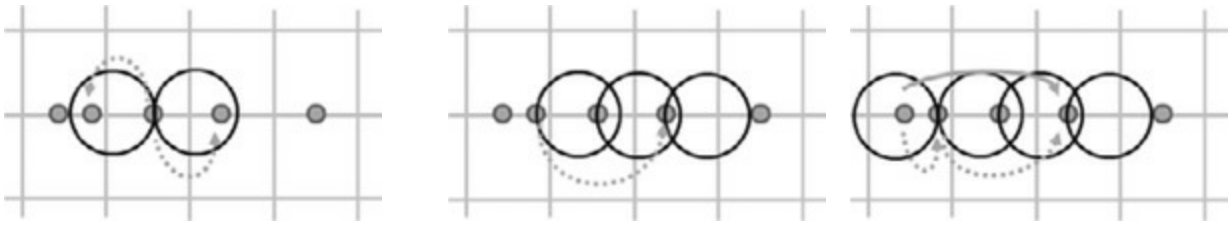
Passo 1: G[1 .. n] = *na;*[5]
Passo 2: *id_grupo* = 0;
Passo 3: **para** todo $\mathbf{x}\rightarrow_i$ que ainda não pertence a um grupo e não está determinado como ruído **faça**
Passo 3.1: lista1 = { };
Passo 3.2: lista1 = {($\mathbf{x}\rightarrow_i$ e todos os exemplares de $\mathbf{X}_{tr}$ na vizinhança de raio *r* de $\mathbf{x}\rightarrow_i$}, considerando a distância *dist*;
Passo 3.3: **se** *lista1* indicar menos exemplares na vizinhança de $\mathbf{x}\rightarrow_i$ que *minEx*
então
Passo 3.2.1: $\mathbf{x}\rightarrow_i$ é considerado ruído e *G*[*i*] = ruído;
senão
Passo 3.2.2: *id_grupo* = *id_grupo* + 1;
Passo 3.2.3: G[*lista1*] = *id_grupo*;
Passo 3.2.4: *lista1* = *lista1* – $\mathbf{x}\rightarrow_i$;
Passo 3.2.5: **para** todo exemplar $\mathbf{x}\rightarrow_j \in$ *lista1* **faça**
Passo 3.2.5.1: lista2 = { };
Passo 3.2.5.2: lista2 = {$\mathbf{x}\rightarrow_i$ e todos os exemplares de $\mathbf{X}_{tr}$ na vizinhança de raio *r* de $\mathbf{x}\rightarrow_j$}, considerando a distância *dist*;
Passo 3.2.5.3: **se** *lista2* contém menos exemplares que *minEx*
então
Passo 3.2.5.3.1: *lista1* = *lista1* - $\mathbf{x}\rightarrow_j$;
senão
Passo 3.2.5.3.2: *lista1* = *lista1* - $\mathbf{x}\rightarrow_j$;
Passo 3.2.5.3.3: **para** todo $\mathbf{x}\rightarrow_k \in$ *lista2* ainda não pertencente a um grupo **faça**
Passo 3.2.5.3.3.1: **se** $\mathbf{x}\rightarrow_k$ <> *ruído* **então**
Passo 3.2.5.3.3.1.1: lista1 = *lista1* + $\mathbf{x}\rightarrow_k$;
Passo 3.2.5.3.3.2: G [*k*] = *id_grupo*.

Para o entendimento pleno do algoritmo DBSCAN, é necessário definir alguns conceitos. O primeiro diz respeito à classificação dos exemplares pertencentes a um grupo como “exemplares internos” (do inglês, *core datapoints*) ou “exemplares de borda” (do inglês, *border datapoints*). Os exemplares internos são aqueles que, considerando um raio de vizinhança *r* (Algoritmo 4-4), possuem uma quantidade maior de exemplares em sua vizinhança; enquanto os exemplares de borda possuem uma quantidade bem

menor de exemplares em sua vizinhança (de raio *r*). O parâmetro *minEx* (Algoritmo 4-4) estabelece uma quantidade mínima de exemplares que deve existir na vizinhança de um exemplar $\mathbf{x}\rightarrow$ para que ele possa ser considerado um “exemplar interno”.

O segundo conceito diz respeito ao “alcance por densidade ”. Trata-se de um conceito que permite verificar se um exemplar $\mathbf{x}\rightarrow_p$ é diretamente alcançável por densidade, a partir de outro exemplar $\mathbf{x}\rightarrow_q$. Um exemplar $\mathbf{x}\rightarrow_p$ é diretamente alcançável por densidade a partir de $\mathbf{x}\rightarrow_q$ se $\mathbf{x}\rightarrow_p$ pertencer à vizinhança de $\mathbf{x}\rightarrow_q$ considerando um raio *r*, e $\mathbf{x}\rightarrow_q$ é um “exemplar interno” considerando um valor *minEx*. Também, diz-se que um exemplar $\mathbf{x}\rightarrow_p$ é alcançável por densidade a partir de $\mathbf{x}\rightarrow_q$ se existir uma sequência de exemplares $\{\mathbf{x}\rightarrow_i \ldots \mathbf{x}\rightarrow_{k-1} \ldots \mathbf{x}\rightarrow_k\}$ entre $\mathbf{x}\rightarrow_p$ e $\mathbf{x}\rightarrow_q$ tal que $\mathbf{x}\rightarrow_1$ é diretamente alcançável por densidade a partir de $\mathbf{x}\rightarrow_q$, $\mathbf{x}\rightarrow_{k-1}$ é diretamente alcançável por densidade a partir de $\mathbf{x}\rightarrow_1$, $\mathbf{x}\rightarrow_k$ é diretamente alcançável por densidade a partir de $\mathbf{x}\rightarrow_{k-1}$ e $\mathbf{x}\rightarrow_p$ é diretamente alcançável por densidade a partir de $\mathbf{x}\rightarrow_k$.

O terceiro conceito diz respeito à “conexão por densidade ”. Com esse conceito, é possível verificar se dois exemplares são conectados por densidade. Um exemplar $\mathbf{x}\rightarrow_p$ é conectado por densidade a um exemplar $\mathbf{x}\rightarrow_q$ se existir um terceiro exemplar $\mathbf{x}\rightarrow_k$ a partir do qual tanto $\mathbf{x}\rightarrow_p$ quanto $\mathbf{x}\rightarrow_q$ são alcançáveis por densidade. A Figura 4-10 mostra situações gráficas em que os conceitos de “alcance por densidade” e “conexão por densidade” são ilustrados.



(a) Diretamente alcançáveis por densidade (b) Alcançáveis por densidade (c) Conectados por densidade

Figura 4-10: Representações gráficas dos conceitos de “alcance por densidade” e “conexão por densidade”: os círculos maiores representam a vizinhança de raio *r* de um exemplar (os círculos pequenos – exemplares); as flechas representam “alcance por densidade” ou “conexão por densidade”

Ao olhar com mais cuidado para as variáveis *lista1* e *lista2*, no Algoritmo 4-4, é possível notar que elas dizem respeito, respectivamente, ao conjunto de exemplares alcançáveis por densidade a partir do exemplar $\mathbf{x}_{\rightarrow i}$ referente ao contexto do laço relacionado com o *Passo 3*, e diretamente alcançáveis por densidade a partir do exemplar $\mathbf{x}_{\rightarrow j}$ referente ao contexto do laço relacionado com o *Passo 3.2.5*. E, ainda, cada grupo formado em *G* é composto por um conjunto de exemplares “conectados por densidade”. Por fim, vale lembrar que os exemplares não “conectados por densidade”, com respeito aos valores estabelecidos para *r* e *minEx*, são considerados ruído.

Uma das dificuldades na aplicação desse algoritmo é a escolha apropriada para os parâmetros *r* e *minEx*. No próprio artigo no qual o DBSCAN foi proposto (Ester *et al.*, 1996), os autores apresentam uma heurística para automaticamente determinar tais valores. Tal heurística se constitui em um procedimento que procura dicas sobre como é a distribuição de densidade do conjunto de dados e é baseada no conceito de distância dos vizinhos mais próximos.

4.1.3.1. Exemplo de didático para o DBSCAN

Como exemplo didático para a estratégia implementada pelo DBSCAN , considere o conjunto de dados *Análise de Vendas*, apresentado na Tabela 4-6. Trata-se de um conjunto de dados referente a valores gastos por mesa no restaurante e número de pessoas presentes na mesa.

Tabela 4-6: Conjunto de dados *ANÁLISE DE VENDAS* (valores originais e valores normalizados)

			x_{i1}	x_{i2}
VALOR GASTO	**# DE PESSOAS**		**ID**	**VALOR GASTO**	**# DE PESSOAS**

(valor original)	(valor original)			(valor normalizado)	(valor normalizado)
432	7	$\mathbf{x}_1$	m_1	10	10
287	3	$\mathbf{x}_2$	m_2	6,6	4
146	3	$\mathbf{x}_3$	m_3	3,4	4
201	3	$\mathbf{x}_4$	m_4	4,7	4
227	3	$\mathbf{x}_5$	m_5	5,3	4
40,5	1	$\mathbf{x}_6$	m_6	1	1
118	2	$\mathbf{x}_7$	m_7	2,8	2,5
97	2	$\mathbf{x}_8$	m_8	2,3	2,5
126	2	$\mathbf{x}_9$	m_9	2,9	2,5
43	1	$\mathbf{x}_{10}$	m_{10}	1,05	1
161	3	$\mathbf{x}_{11}$	m_{11}	3,8	4
83	2	$\mathbf{x}_{12}$	m_{12}	1,9	2,5

ID: identificação de uma mesa.
VALOR GASTO: valor normalizado do gasto em uma mesa.
#PESSOAS: valor normalizado do número de pessoas presentes em uma mesa.

Para executar, passo a passo, o Algoritmo 4-4, presuma que o número mínimo de exemplares exigido na verificação da densidade de vizinhança de um exemplar seja 2 (*minEx* = 2), o raio de vizinhança seja 1 (*r* = 1) e a distância seja a euclidiana. A Tabela 4-7 mostra como o conteúdo das variáveis *G*, *id_grupo*, *lista1* e *lista2* variam no decorrer da execução do algoritmo. A fim de facilitar o entendimento da execução desse exemplo, as distâncias euclidianas entre os exemplares são listadas na Tabela 4-8.

No *Passo 1*, a lista *G* com os exemplares agrupados indica que todos ainda não pertencem a um grupo (todos os exemplares estão valorados como *na*, significando “não agrupado”). No *Passo 2*, uma variável de controle para os identificadores dos grupos é inicializada em zero (*id_grupo*).

Na primeira iteração do *Passo 3*, a vizinhança do primeiro exemplar é estudada. Isso significa que será medida a distância de m_1 para os demais exemplares do conjunto de dados (resultado apresentado na primeira linha da Tabela 4-8). Como não há nenhum exemplar na sua vizinhança de raio *r*, m_1 é considerado um ruído, e a variável *G* é atualizada para refletir essa condição do primeiro exemplar. Na segunda iteração do *Passo 3*, a vizinhança do segundo exemplar do conjunto de dados é estudada. A mesma situação é observada, pois não há exemplares na vizinhança de raio *r* do segundo exemplar.

Uma nova situação é observada na terceira iteração do *Passo 3*, pois o estudo da vizinhança de raio *r* do terceiro exemplar (*Passo 3.2*) coloca (m_3 e m_{11}) na *lista 1*. Nesse caso, a condição de densidade mínima é atendida, e o primeiro grupo é estabelecido. A variável *G* é atualizada para refletir a organização do grupo 1. Então, a vizinhança de raio *r* do exemplar m_{11} do conjunto de dados precisa ser estudada. A vizinhança desse exemplar contém o m_3, m_4 e m_{11} (*lista 2*). Portanto, a densidade mínima é alcançada. Como m_3 e m_{11} já estão no grupo (grupo 1), o m_4 passa a compor o mesmo grupo.

Tabela 4-7: Execução passo a passo do Algoritmo 4-4

Passo 1
G = {*na*, *na*, *na*, *na*, *na*, *na*, *na*, *na*, *na*, *na*, *na*, *na*}

Passo 2
id_grupo = 0

Passo 3 – primeira iteração
lista1 = {m_1}
G = {*ruído*, *na*, *na*, *na*, *na*, *na*, *na*, *na*, *na*, *na*, *na*, *na*}

Passo 3 – segunda iteração
lista1 = {m_2}
G = {*ruído*, *ruído*, *na*, *na*, *na*, *na*, *na*, *na*, *na*, *na*, *na*, *na*}

Passo 3 – terceira iteração
lista1 = {m_3, m_{11}}
id_grupo = 1
G = {*ruído*, *ruído*, *1*, *na*, *na*, *na*, *na*, *na*, *na*, *na*, *1*, *na*}

lista1 = {m_{11}}

lista2 = {m_3,m_4,m_{11}}

lista1 = {m_4}

G = {*ruído, ruído, 1, 1, na, na, na, na, na, na, 1, na*}

lista2 = {m_4,m_5,m_{11}}

lista1={m_5}

G = {*ruído, ruído, 1, 1, 1, na, na, na, na, na, 1, na*}

lista2 = {m_4,m_5 }

lista1={ }

Passo 3 – quarta iteração

lista1 = {m_6,m_{10}}

id_grupo = 2

G = {*ruído, ruído, 1, 1, 1, 2, na, na, na, 2, 1, na*}

lista1 = {m_{10}}

lista2 = {m_6,m_{10}}

lista1 = {}

Passo 3 – quinta iteração

lista1 = {m_7,m_8,m_9,m_{12}}

id_grupo = 3

G = {*ruído, ruído, 1, 1, 1, 2, 3, 3, 3, 2, 1, 3*}

lista1 = {m_8,m_9,m_{12}}

lista2 = {m_7,m_8,m_9,m_{12}}

lista1 = {m_9,m_{12}}

lista2 = {m_7,m_8,m_9,m_{12}}

lista1 = {m_{12}}

lista2 = {m_7,m_8,m_9,m_{12}}

lista1 = { }

Tabela 4-8: Distância euclidiana entre pares de exemplares do conjunto de dados ANÁLISE DE VENDAS

	m_1	m_2	m_3	m_4	m_5	m_6	m_7	m_8	m_9	m_{10}	m_{11}	m_{12}
m_1	0											
m_2	6,8	0										
m_3	8,9	3,2	0									
m_4	8,0	1,9	1,3	0								
m_5	7,6	1,3	1,9	0,6	0							
m_6	12,7	6,3	3,8	4,7	5,2	0						
m_7	10,3	4,0	1,6	2,4	2,9	2,3	0					
m_8	10,7	4,5	1,8	2,8	3,3	1,9	0,5	0				
m_9	10,3	3,9	1,5	2,3	2,8	2,4	0,1	0,6	0			
m_{10}	12,6	6,3	3,8	4,7	5,2	0,05	2,3	1,9	2,3	0		
m_{11}	8,6	2,8	0,4	0,9	1,5	4,1	1,8	2,1	1,7	4,0	0	
m_{12}	11,0	4,9	2,1	3,1	3,7	1,7	0,9	0,4	1	1,7	2,4	0

Note que uma verificação sobre condição de ruído é realizada neste ponto do algoritmo. Isso ocorre porque, se o exemplar em análise já tiver sido classificado como ruído antes, significa que sua vizinhança não é densa, mas agora ele deixa de ser ruído e entra no grupo na condição de exemplar de borda. Sendo um exemplar de borda, ele não precisa ter sua vizinhança analisada. Como m_4 não foi previamente identificado como ruído, sua vizinhança deverá ser analisada e, por isso, ele entrará na *lista 1*. O processo então se repete até que a *lista 1* fique vazia.

Como m_4 e m_5 já foram agrupados, a iteração seguinte, a quarta, analisará o exemplar m_6. E, ao final desta iteração, o exemplar e seu vizinho, m_6 e m_{10}, são definidos em um novo grupo.

Na quinta e última iteração do *Passo 3*, um novo grupo é formado. A vizinhança do sétimo exemplar, m_7, do conjunto de dados é densa, contendo os exemplares m_8, m_9 e m_{12}. O algoritmo procede então associando esses exemplares ao grupo 3 e passa a estudar a vizinhança de cada um deles. Todos eles possuem vizinhança densa, porém, em relação a eles próprios apenas, não levando à inserção de nenhum novo exemplar ao grupo.

Analisando os grupos formados pelo DBSCAN, a Figura 4-11 e a indicação de ruído, algumas conclusões podem ser formuladas: mesas com

muitas pessoas (sete pessoas) são eventos raros no restaurante; normalmente, as mesas são ocupadas por duas ou três pessoas (considerando os valores originais do conjunto de dados); uma mesa com três pessoas com gasto no montante de aproximadamente R$280,00 também não é comum no restaurante. A partir dos três grupos formados, o gerente do restaurante consegue estabelecer perfis “de mesas” e pode direcionar o atendimento quando percebe uma mesa com determinado perfil. Além disso, o gerente do restaurante pode usar dados adicionais para enriquecer o conhecimento sobre os três perfis de mesas encontrados, observando, por exemplo, se mesas ocupadas por duas pessoas são ou não geralmente ocupadas por casais. Se essa conclusão for obtida, promoções especiais para casais podem ser lançadas.

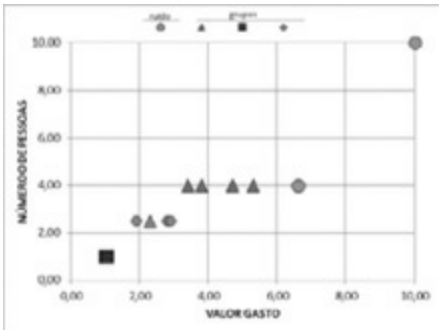


Figura 4-11: Representação gráfica do conjunto de dados ANÁLISE DE VENDAS e da organização de grupos descoberta pelo DBSCAN com *r* = 1 e *minEx* = 2

4.1.3.2. Exemplo prático para o DBSCAN em R

O exemplo prático com o DBSCAN será demonstrado com uso da função `dbscan()`, disponível no pacote ***fpc*** (Henning, 2014). A parametrização deste algoritmo é descrita no Quadro 4-6.

Quadro 4-6: Sintaxe e parâmetros da função DBSCAN

Sintaxe para aplicação da função que implementa DBSCAN
Usando a função `dbscan( )` **do pacote *fpc***
Gerando os agrupamentos `grupos <- dbscan(dados, raio, mínimo_exemplares)`

- `dados` é o conjunto de dados agrupado, disponível como um ***data.frame***.
- `raio` é o parâmetro que determina a vizinhança acessível de um exemplar – nomeado como *eps*.
- `mínimo_exemplares` é o parâmetro exigido na verificação da densidade de vizinhança de um exemplar – nomeado como *MinPts*.

A função retorna um objeto com os grupos descobertos e também indicação de exemplares classificados como ruído.

A sequência de comandos necessária para executar o DBSCAN é:

```
> install.packages("fpc")
> library("fpc")
> vendas =
read.table("C:\\Users\\LivroDM\\Agrupamento\\vendas.csv",
header=TRUE,sep=";")
> dbs <- dbscan(vendas,eps=1,MinPts=2)
```

Na variável `dbs$cluster`, são armazenados os indicadores de grupos para cada exemplar do conjunto de dados, sendo que o indicador 0 significa que o exemplar foi considerado ruído

```
> dbs$cluster
[1] 0 0 1 1 1 2 3 3 3 2 1 3
```

Por fim, para visualizar os resultados, é possível gerar uma plotagem, cuja sintaxe é como segue, e o resultado ilustrado na Figura 4-12.

```
> plot(dbs, vendas)
```

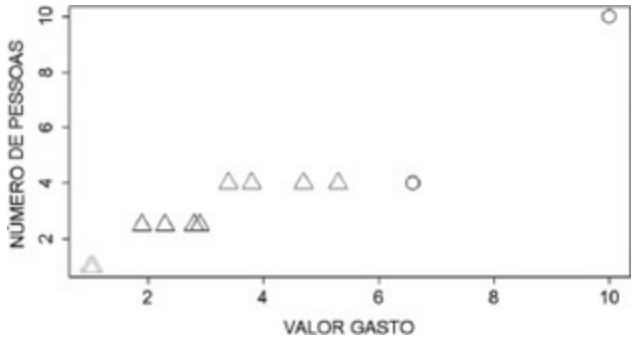


Figura 4-12: Agrupamento do conjunto de dados ANÁLISE DE VENDAS

4.1.4. Mapas Auto-organizáveis

O funcionamento do cérebro desperta o interesse de pesquisadores de diversas áreas, e o estudo dos processos cerebrais inspirou a geração de diferentes modelos matemáticos para análise de dados – as diferentes arquiteturas de Redes Neurais Artificiais. Teuvo Kohonen desenvolveu um estudo, na década de 1980, que deu origem a uma das mais famosas arquiteturas de Redes Neurais Artificias – os **Mapas Auto--organizáveis** (do inglês, *Self Organizing-maps* – SOM) (Kohonen, 1982). Os estudos de Kohonen são baseados sobretudo no princípio da formação de mapas de unidades cerebrais (neurônios) que, de forma auto-organizada, estabelecem uma ordenação espacial que permite representar informação. Importante dizer que essa ordenação espacial se refere ao efeito consequente da forma como neurônios respondem a um estímulo, e muda com o tempo de aprendizado do significado daquele estímulo.

Com base nessa inspiração biológica, e também em outras propriedades existentes no cérebro, a arquitetura da rede neural artificial SOM foi proposta, e essa arquitetura é discutida aqui como ferramenta para a resolução da tarefa de agrupamento de dados. Nesse modelo, os exemplares de um conjunto de dados (de alta dimensão – ou seja, muitos atributos descritivos) são projetados em um espaço de baixa dimensão, e o processo de criação dessas projeções é planejado para manter, tanto quanto possível, as relações topológicas existentes entre os exemplares, de forma que exemplares similares (próximos) na representação original do conjunto de dados sejam projetados em regiões próximas na projeção em baixa dimensão em construção.

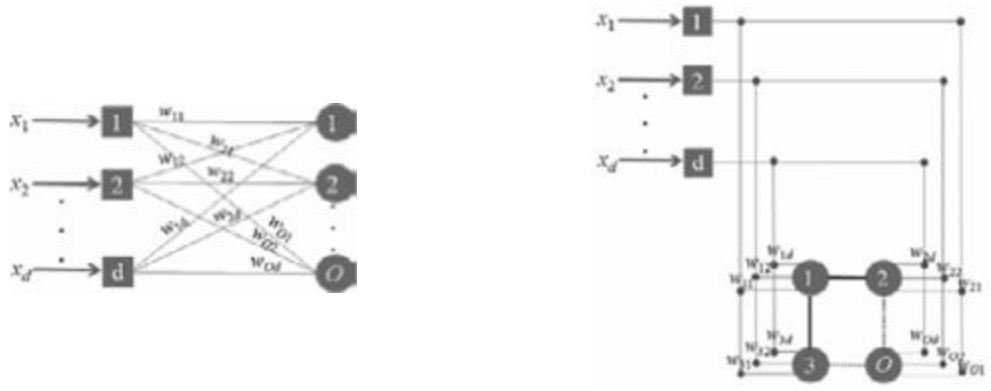
Duas arquiteturas de rede SOM são apresentadas na Figura 4-13. Na entrada da arquitetura (camada de entrada), há uma série de neurônios sensoriais responsáveis por apresentar um estímulo (exemplar) à rede neural. O número de neurônios sensoriais é igual ao número de atributos descritivos dos exemplares presentes no conjunto de dados sob análise. Todos os

atributos descritivos de um exemplar são apresentados a todos os neurônios da saída da arquitetura (camada de saída), e essa apresentação é representada na arquitetura pelas ligações entre os neurônios das duas camadas. Os neurônios da camada de saída formam o mapa, que alude aos mapas formados no cérebro humano e é organizado a partir do processamento da informação presente nos estímulos. A arquitetura de um SOM unidimensional é mostrada na Figura 4-13(a), e a arquitetura de um SOM bidimensional, na Figura 4-13(b).

A análise das arquiteturas ilustradas na Figura 4-13 permite a introdução de dois conceitos: espaço de entrada (ou espaço dos dados, espaço vetorial) e espaço de saída (ou espaço de características, espaço matricial).

O espaço de entrada é definido de acordo com o conjunto de dados submetido à análise pelo SOM, no qual cada exemplar é visto como um vetor de *d*-coordenadas (*d* é o número de atributos descritivos no conjunto de dados), e esse espaço é *d*-dimensional. Observe que cada um dos neurônios na camada de saída do SOM (no mapa) recebe *d* sinapses ponderadas (w_{Od}). Essas sinapses representam o relacionamento existente entre cada neurônio e os exemplares do conjunto de dados – cada *d* sinapse do neurônio deve ser vista como uma coordenada de um vetor que representa o neurônio no espaço de entrada. Assim, deduz-se que cada neurônio na camada de saída do SOM é representado por um vetor *d*-dimensional no espaço definido pelo conjunto de dados. No espaço de entrada valem as relações entre os vetores, e, portanto, trata-se de um espaço vetorial no qual medidas de distâncias vetoriais podem ser aplicadas. Tais relações podem ser observadas na Figura 4-14. Note que, se medidas de distâncias vetoriais podem ser aplicadas, é possível medir a distância existente entre exemplares, entre neurônios e entre exemplares e neurônios. Na Figura 4-14(a), é apresentado um conjunto de dados composto por 75 exemplares (cruzes), com dois atributos descritivos, e, portanto, o espaço de entrada é considerado bidimensional. Uma rede neural SOM com 25 neurônios (círculos) é apresentada na Figura 4-14(b).

Note que os neurônios residem no mesmo espaço bidimensional dos exemplares.



(a) SOM unidimensional

(b) SOM bidimensional

Figura 4-13: Exemplos de arquiteturas de um SOM

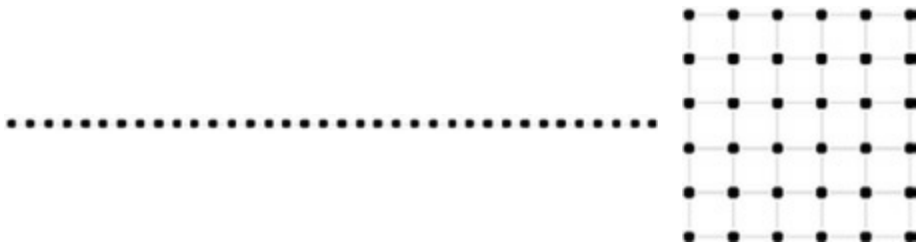
(a) Conjunto de dados em um espaço de entrada (vetorial) bidimensional

(b) Conjunto de dados e neurônios de um SOM no espaço de entrada (vetorial) bidimensional

Figura 4-14: Representação gráfica para o espaço de entrada (vetorial) de um SOM. No espaço de entrada residem tanto os exemplares do conjunto de dados (cruzes) quanto os neurônios do SOM (círculos)

O espaço de saída é definido por uma estrutura de vizinhança topológica, estabelecida sobre os neurônios da camada de saída de um SOM. É uma estrutura criada pelo analista de dados que aplica o SOM sobre um conjunto de dados. Trata-se de um espaço matricial no qual apenas relações entre neurônios se estabelecem – cada neurônio da camada de saída de um SOM é associado a uma posição em uma matriz. A dimensão dessa matriz é a

dimensão associada ao SOM (como na Figura 4-13), ou seja, se o espaço matricial é definido sobre uma matriz unidimensional, diz-se então que se trata de um SOM unidimensional, e assim por diante. Espaços matriciais de quaisquer dimensões podem ser estabelecidos; no entanto, espaços uni e bidimensionais são os mais usados e os que mais se aproximam das inspirações originais usadas na concepção desse modelo de análise de dados. Na Figura 4-15, são mostradas algumas situações que ilustram espaços de saída e como eles se relacionam com espaços de entrada bidimensionais.



(a) SOM unidimensional

(b) SOM bidimensional

(c) SOM unidimensional em um espaço de entrada bidimensional

(d) SOM bidimensional em um espaço de entrada bidimensional

Figura 4-15: Espaços de saída (matricial) e as relações existentes entre os espaços vetorial e matricial, representados pelos neurônios da camada de saída

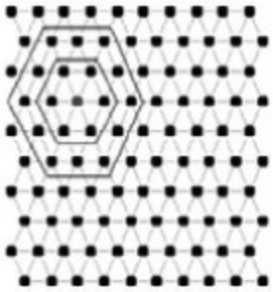
Definições de espaços matriciais são graficamente apresentadas na Figura 4-15(a, b). No primeiro caso (a), é apresentada uma estrutura matricial unidimensional com 36 posições (ou seja, 36 neurônios). O segundo caso (b) diz respeito a uma estrutura matricial bidimensional, também com 36 posições (neurônios), sendo que a matriz estabelecida tem dimensões 6 × 6.

A Figura 4-15(c) mostra um SOM, no espaço dos dados, que já passou por um processo de treinamento. Importante entender que os neurônios nessa figura e na Figura 4-15(a) são exatamente os mesmos, ou seja, um mesmo neurônio tem representações em dois espaços diferentes. No espaço dos dados (espaço vetorial), o vetor de *d*-coordenadas representa o neurônio, e distâncias vetoriais podem ser medidas a partir desse vetor. No espaço de saída (matricial), esse mesmo vetor possui um índice que indica qual a sua posição (posição do neurônio) na matriz. Na Figura 4-15(c, d), as relações de vizinhança dos neurônios existentes no espaço matricial são representadas pelas linhas. Observe que as relações de distâncias matriciais entre dois neurônios não são necessariamente as mesmas que as de distâncias vetoriais entre o mesmo par de neurônios. Analisando com mais detalhes esse fenômeno na Figura 4-16, observa-se que os três neurônios mais próximos ao neurônio 1 no espaço matricial são os neurônios 2, 3 e 4. Porém, no espaço vetorial, os três neurônios mais próximos ao neurônio 1 são os neurônio 2, 9 e 10. A esse fenômeno, dá-se o nome de distorção topológica. A organização matricial unidimensional dos neurônios executa “distorções” para conseguir aproximar um conjunto de dados que possui topologia bidimensional. A mesma análise deve ser feita para o caso da Figura 4-15(d), porém, nesse caso, menos distorções topológicas serão encontradas, uma vez que a organização matricial do SOM é bidimensional, ou seja, é a mesma dimensão da organização topológica do conjunto de dados.

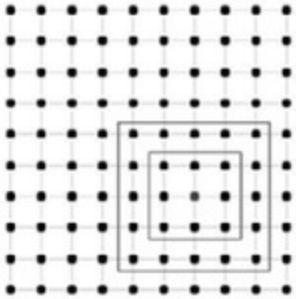
(a) (b)

Figura 4-16: Detalhe sobre distorção topológica

Ainda, espaços de saída bidimensionais ou de dimensões mais altas precisam assumir um padrão de relação de vizinhança , conhecido como *lattice* . Existem dois padrões mais comumente usados: retangular ou hexagonal. A Figura 4-17 ilustra cada um desses *lattices*, e sua aplicação em espaço de dimensões mais altas é uma generalização do que é ilustrado nessa figura. O *lattice* estabelece uma regra para identificação de quais neurônios são vizinhos no espaço matricial. Ainda, na Figura 4-17, a ideia de raio de vizinhança é ilustrada. Observe que todos os neurônios localizados a uma distância matricial ($da = r$) de um neurônio são considerados seus vizinhos de raio (r). Na vizinhança hexagonal, um neurônio possui seis vizinhos de raio $r = 1$ e 12 vizinhos de raio $r = 2$. Na vizinhança retangular, um neurônio possui oito vizinhos de raio $r = 1$ e 16 vizinhos de raio $r = 2$. Por fim, o número de neurônios em cada dimensão do espaço de saída também precisa ser determinado. No caso da Figura 4-17, os mapas possuem 10×10 neurônios.



(b) *Lattice* hexagonal



(c) *Lattice* retangular

Figura 4-17: Padrões de *lattice* para espaços matriciais bidimensionais

A fim de produzir uma representação para o conjunto de dados (um mapa de características), o SOM precisa aprender as relações que existem entre os exemplares, ou seja, o processo de auto-organização dos neurônios precisa ocorrer para que sua ordenação reflita tais relações. De forma resumida, a auto-organização consiste em um processo iterativo no qual, a cada época, os

exemplares são individualmente apresentados aos neurônios, e esses respondem a cada um dos exemplares de formas diferentes. Para cada exemplar, um dos neurônios responderá da maneira mais adequada, e ele e seus vizinhos na vizinhança topológica (no espaço matricial) serão ajustados (no espaço vetorial) para melhorar sua capacidade de responder adequadamente àquele estímulo em uma próxima época.

O processo de aprendizado de um SOM é mostrado no Algoritmo 4-5. Esse algoritmo usa uma série de conceitos que precisam ser cuidadosamente estudados para que se tenha o completo entendimento do processo nele implementado. Cada um dos conceitos usados no algoritmo, ainda não apresentados neste livro, são discutidos na sequência.

As primeiras decisões a serem tomadas quando o SOM está sendo projetado são quantos neurônios serão usados na camada de saída e como estarão organizados no espaço matricial. Essas decisões são tomadas pelo analista de dados e devem ter como base algumas diretrizes: é interessante que a rede tenha neurônios suficientes para lhe dar condições de aproximar a densidade dos dados, ou seja, poucos neurônios na camada de saída podem impossibilitar a organização dos neurônios no mapa, de modo que reflitam adequadamente a informação de que há regiões do espaço dos dados nas quais existe uma concentração maior de exemplares; por outro lado, uma quantidade muito grande de neurônios pode tornar o algoritmo ineficiente em termos de tempo de treinamento e também levar a uma configuração final no mapa em que muitos neurônios não são utilizados; ainda, devido a características do processo de aprendizado do SOM, quando esse possui uma estrutura matricial de dimensão muito alta, será necessário usar mais neurônios para que o mapa não se torne rígido e consiga se adaptar à distribuição dos exemplares. A distribuição dos neurônios pelas dimensões do mapa também é importante e também deve ser cuidadosamente projetada, observando se a distribuição usada está de acordo com a dimensão matricial pretendida.

Algoritmo 4-5: Algoritmo para treinamento de um SOM

Parâmetros de entrada:
- $\mathbf{X}_{tr}$: um conjunto de dados de treinamento não rotulado, ou seja, $\mathbf{X}_{tr} = \{\mathbf{x}\rightarrow_i\}$, $i = 1, ..., n$;
- **W**: é o conjunto de pesos sinápticos, $\mathbf{W} = \{\mathbf{w}\rightarrow_l\}$, $l = 1, ..., O$;
- *O:* número de neurônios no mapa;
- *mapa*: configurações do mapa (número de neurônios por dimensão do mapa, *lattice*);
- *dist*: uma medida de distância vetorial, aqui, considerada a euclidiana;
- *da*: uma medida de distância matricial;
- η: taxa de aprendizado inicial;
- $\nu_{l,c}$: função de vizinhança topológica entre o neurônio l e o neurônio mais próximo ao exemplar c;
- r: raio de vizinhança;
- $t_{máximo}$: número de iterações (ou época);
- $e_{máximo}$: valor do erro máximo esperado para alterações nos pesos;

Parâmetro de saída:
- **W**: conjunto de pesos sinápticos ajustados;

Passo 1: defina $t = 0$;
Passo 2: **enquanto** t < $t_{máximo}$ **ou** e > $e_{máximo}$ não são alcançados **faça**
Passo 2.1: **para** cada $\mathbf{x}\rightarrow_i \in \mathbf{X}_{tr}$ **faça**
Passo 2.1.1: calcule $dist(\mathbf{x}\rightarrow_i, \mathbf{w}\rightarrow_l)$, em que $\mathbf{w}\rightarrow_l \in \mathbf{W}$ e $l = 1 .. O$;
Passo 2.1.2: determine $BMU = arg\ min_l\ dist(\mathbf{x}\rightarrow_i, \mathbf{w}\rightarrow_l)$;
Passo 2.1.3: determine o ajuste de pesos como $\mathbf{w}\rightarrow_l = \mathbf{w}\rightarrow_l + \nu_{l,c} * \eta * (\mathbf{x}\rightarrow_i - \mathbf{w}\rightarrow_l)$ para o neurônio *BMU* e seus vizinhos
topológicos;
Passo 2.2: $e = ||\mathbf{W}(t + 1) - \mathbf{W}(t)||$;
Passo 2.3: $t = t + 1$;
Passo 2.4: ajuste a taxa de aprendizado (η) e o raio de vizinhança (r), se for o caso.

Classicamente, as medidas de distância usadas no SOM são a distância euclidiana para o espaço vetorial, e a distância de Manhattan para o espaço matricial. Inicialmente, o SOM conta com um conjunto de pesos inicializado aleatoriamente, dentro do espaço de entrada e com valores pequenos. Iniciar o conjunto de pesos significa posicionar os neurônios dentro do espaço vetorial. Como alternativa, outras medidas de distâncias e outros métodos de inicialização de pesos podem ser considerados. A discussão dos parâmetros de entrada no Algoritmo 4-5 “taxa de aprendizado” e “função de vizinhança ” deve se juntar à discussão sobre a dinâmica de treinamento do SOM.

Finalmente, os dois últimos parâmetros de entrada do algoritmo são os critérios de parada do treinamento: número máximo de épocas de treinamento ($t_{máximo}$) e erro máximo esperado ($e_{máximo}$) são os mais comuns. O número máximo de épocas deve ser alto, para permitir que os neurônios se ajustem, de forma precisa, aos exemplares do conjunto de dados. Além disso, o treinamento do SOM é caracterizado por duas fases: a fase de ordenação e a fase de refinamento ou de convergência. Na primeira fase, é esperado que parâmetros como “taxa de aprendizado ” e “raio de vizinhança” assumam valores altos, como será discutido mais à frente. Isso permitirá que os neurônios se organizem de maneira a aprender a topologia do conjunto de dados. A fase de convergência, caracterizada pela redução gradual dos parâmetros citados, permite que os neurônios refinem seu posicionamento no espaço dos dados, procurando representar, tão fielmente quanto possível, a distribuição de densidade dos dados. Já o erro máximo esperado pode ser usado como um limiar máximo de erro aceito na avaliação do mapa gerado. Esse erro pode ser medido como a quantidade de variação observada nos pesos dos neurônios, ou seja, variações muito pequenas significam que os neurônios assumem ajustes muito pequenos, indicando a convergência do algoritmo. O erro máximo esperado pode ainda ser medido por meio do erro de quantização (Rocha *et al.*, 2012), o qual está intimamente ligado à representação da distribuição de densidade dos dados. O alcance desse limiar, antes do alcance no número máximo de épocas, viabiliza a interrupção adiantada do processo de treinamento.

Como pode ser observado no Algoritmo 4-5, o processo de treinamento do SOM pressupõe um laço de controle de sua duração (as épocas – *Passo 2*). Uma época é definida como a apresentação de todos os exemplares ($\mathbf{x}\rightarrow_i$) do conjunto de treinamento ($\mathbf{X}_{tr}$) para a rede neural (*Passo 2.1*). Então, para cada um dos exemplares, executam-se os *Passos 2.1.1* a *2.1.3*.

No *Passo 2.1.1*, calcula-se a distância vetorial (*dist*) entre o exemplar sob análise no momento e todos os neurônios da camada de saída do SOM.

Observe que calcular a distância vetorial entre esses dois elementos (exemplar e neurônio) significa executar o cálculo da distância usando o vetor do exemplar $\mathbf{x}\rightarrow_i$ e o vetor de pesos de um neurônio $\mathbf{w}\rightarrow_l$. Uma vez que as distâncias entre o exemplar de entrada e todos os neurônios do mapa já estão calculadas, é preciso encontrar qual é o neurônio mais próximo do vetor de entrada (*Passo 2.1.2*), ou seja, encontrar qual neurônio minimiza a função de distância *dist*($\mathbf{x}\rightarrow_i$, $\mathbf{w}\rightarrow_l$). Lembre-se que dois vetores que se encontram próximos no espaço dos dados são parecidos entre si, logo, o neurônio que minimiza a distância é o mais próximo (ou similar) ao exemplar sob análise, e diz-se que ele é o BMU (*Best Match Unit*) para o exemplar. Essa estratégia implementa um **aprendizado** do tipo **competitivo** , ou seja, os neurônios da rede neural estão "competindo" para representar o exemplar.

Então, após a determinação do BMU para o exemplar sob análise, o ajuste dos pesos será realizado (*Passo 2.1.3*). Tal ajuste é aplicado ao neurônio vencedor (*BMU*) e também aos seus neurônios vizinhos no espaço matricial, ou seja, seus vizinhos topológicos. O fato de o ajuste ser feito no BMU e nos seus vizinhos caracteriza esse **aprendizado** como sendo **cooperativo** (além de competitivo). O ajuste de pesos é feito de tal maneira que os neurônios ajustados serão reposicionados no espaço dos dados, de forma a se aproximarem do exemplar, tornando-se mais similares a ele. Uma operação vetorial implementa o reposicionamento do neurônio como $\mathbf{w}\rightarrow_l = \mathbf{w}\rightarrow_l + \eta * (\mathbf{x}\rightarrow_i - \mathbf{w}\rightarrow_l)$ (um termo foi suprimido nesta formulação em relação àquela apresentada no Algoritmo 4-5 e ele será discutido na sequência).

A taxa de aprendizado é um termo que regulariza a magnitude da alteração de pesos. Ela pode ser vista como um termo de ponderação para o tamanho do deslocamento que o neurônio sofrerá no ajuste de seus pesos. A taxa de aprendizado deve variar no intervalo]0,1]. Ela deve assumir valores altos na fase de ordenação do treinamento do SOM, pois, dessa forma, permite deslocamentos mais acentuados, possibilitando que os neurônios assumam uma organização próxima à topologia do conjunto de dados. Na

fase de refinamento, a taxa de aprendizado deve ter o seu valor diminuído, pois, assim, contribui para a estabilização dos pesos dos neurônios, permitindo apenas que deslocamentos suaves sejam realizados.

Ainda, como determinado no Algoritmo 4-5, o ajuste de pesos deve ser aplicado aos vizinhos matriciais do neurônio vencedor. A determinação de quais são os vizinhos matriciais é feita com a análise da estrutura matricial, definida no início do algoritmo, e com a análise tanto da função de vizinhança ($v_{l,c}$) adotada, quanto do valor do raio de vizinhança (r) vigente na época em execução. A função de vizinhança comumente adotada em implementações para o SOM é a gaussiana, definida como:

$$v_{l,c} = \exp\left(-\frac{da(l,c)^2}{2r^2}\right)$$ **Equação 4-2**

em que $l = \{1 .. O\}$, c é o BMU e r é o raio de vizinhança (largura da função gaussiana) escolhido no algoritmo. A depender do valor assumido por r, essa função pode permitir que todos os neurônios da camada de saída de um SOM tenham seus pesos ajustados. No entanto, a intensidade do ajuste sofrido pelo neurônio decresce conforme $da(l, c)$ aumenta.

Como parâmetro de saída do Algoritmo 4-5, é retornado o conjunto de pesos finais do SOM (**W**), que permite verificar a forma como os neurônios da camada de saída respondem aos estímulos após o processo de aprendizado. Contudo, as configurações básicas do espaço de saída (mapa) usadas durante o algoritmo devem ser conhecidas, se houver o objetivo de analisar seus resultados em relação a informações de vizinhança topológica – geralmente feito quando se pretende visualizar o mapa gerado.

O mapa de características resultante da aplicação do SOM sobre um conjunto de dados apresenta algumas características importantes (Haykin, 2008):

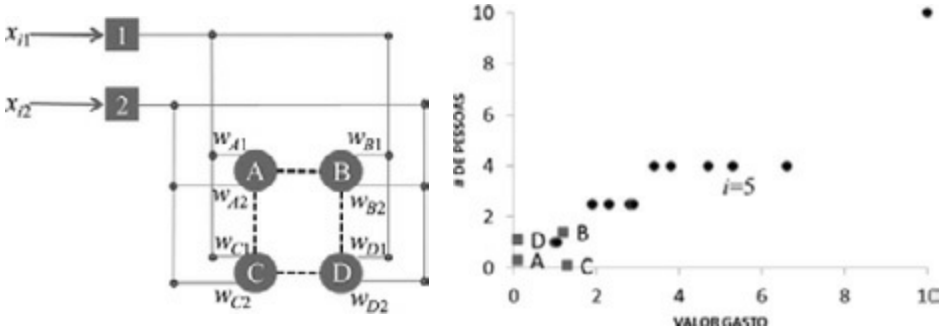
- Fornece uma boa aproximação do espaço de entrada, de forma que o conjunto de neurônios no mapa pode ser visto como um conjunto de protótipos dos exemplares do conjunto de dados. Assim, uma grande quantidade de exemplares presentes em um conjunto de dados pode ser representada pelos neurônios do mapa, geralmente em menor quantidade, fornecendo, portanto, uma forma de compactação do conjunto de dados, exercendo a função de um quantizador vetorial do espaço dos dados.
- Fornece um ordenamento topológico dos neurônios no mapa, de forma que a localização de um neurônio pode ser vista como a representação de um domínio em particular (uma característica ou um conjunto de características) dos exemplares do conjunto de dados. Portanto, quanto mais as relações topológicas dos exemplares do conjunto de dados forem mantidas durante o processo de construção do mapa, melhor será a interpretação do mapa gerado, no que diz respeito à representação de vizinhança existente entre os exemplares no seu espaço original.
- Reflete variações estatísticas da distribuição dos dados no seu espaço original por meio da distribuições de neurônios por regiões específicas do mapa. Regiões do espaço de entrada, nas quais existe maior densidade de exemplares, são projetadas em regiões do mapa em que uma quantidade também densa de neurônios se organiza. Já regiões do espaço de entrada nas quais existe baixa densidade de exemplares são projetadas em regiões do mapa em que existem poucos neurônios.

A análise do mapa de características gerado com o treinamento de um SOM, com respeito às características supracitadas, proporciona o conhecimento das propriedades de um conjunto de dados e, portanto, pode ser usada como solução para a tarefa de agrupamento de dados. Entretanto, tal análise precisa ser realizada por meio de procedimentos de pós-processamento. Entre eles, estão os procedimentos de visualização e

procedimentos de avaliação da qualidade do SOM (medidas de qualidade). Uma das formas mais didáticas de visualizar um SOM é por meio da *matriz U* (Ultsch, 2003). As medidas de qualidade mais comumente aplicadas são: avaliação do erro de quantização e avaliação do erro topológico (Rocha *et al*., 2012; Costa, 1999). Outras medidas são indicadas na literatura comentada na Seção 4.3.

4.1.4.1. Exemplo didático para Mapas Auto-Organizáveis

Para exemplificar de maneira didática o algoritmo de treinamento da rede neural SOM (Algoritmo 4-5), o conjunto de dados *Análise de Vendas* (Tabela 4-6) será utilizado. As seguintes decisões para inicialização dos parâmetros do SOM são consideradas: quatro neurônios na camada de saída (O = 4), espaço matricial bidimensional com *lattice* retangular 2 × 2, pesos $\mathbf{W}$ escolhidos aleatoriamente ($\mathbf{w}_{\rightarrow A}$ = (0,1; 0,3); $\mathbf{w}_{\rightarrow B}$ = (1,2; 1,4); $\mathbf{w}_{\rightarrow C}$ = (1,3; 0,1); $\mathbf{w}_{\rightarrow D}$ = (0,1; 1,1)), taxa de aprendizado inicial η = 0,5 e função de vizinhança v como função gaussiana com raio de vizinhança r = 1. A arquitetura do SOM e um gráfico de dispersão ilustrando as posições iniciais dos neurônios no espaço dos dados estão organizados na Figura 4-18. Observe na figura que, devido à inicialização aleatória dos vetores de pesos, a ordenação dos neurônios no espaço matricial não é refletida no espaço vetorial.



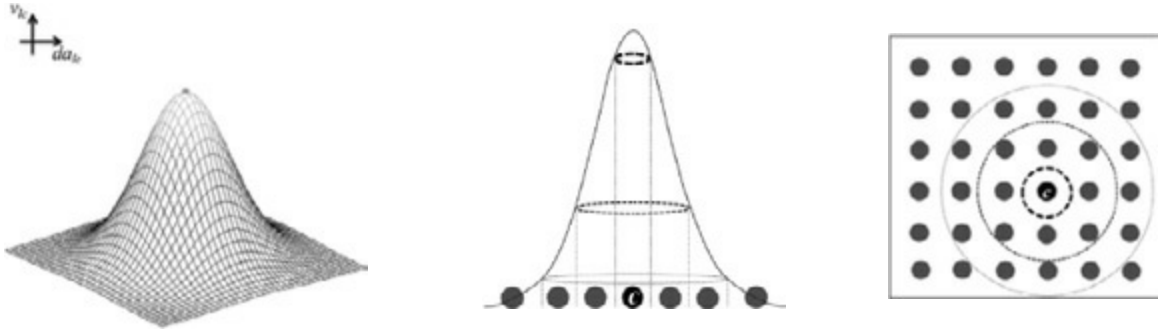
(a) Arquitetura do SOM

(b) Gráfico de dispersão contemplando os exemplares (círculos) e os neurônios (quadrados)

Figura 4-18: Representação da arquitetura do SOM e dados de entrada para treinamento da rede

Considere que o primeiro exemplar escolhido para ser apresentado à rede neural (*Passo 2.1*) é $\mathbf{x}_{\rightarrow 5}$ (marcado na Figura 4-18(b)). O cálculo da distância euclidiana (*Passo 2.1.1*) gera os seguintes valores: $dist(\mathbf{x}_{\rightarrow 5}, \mathbf{w}_{\rightarrow A}) = 6{,}4$; $dist(\mathbf{x}_{\rightarrow 5}, \mathbf{w}_{\rightarrow B}) = 4{,}9$; $dist(\mathbf{x}_{\rightarrow 5}, \mathbf{w}_{\rightarrow C}) = 5{,}6$; $dist(\mathbf{x}_{\rightarrow 5}, \mathbf{w}_{\rightarrow D}) = 6{,}0$. O neurônio que minimiza os valores de distâncias obtidos (*Passo 2.1.2*) é o neurônio B, portanto, ele é o BMU para o $\mathbf{x}_{\rightarrow 5}$ nesta época. Então, no *Passo 2.1.3*, os pesos desse neurônio e de seus vizinhos topológicos devem ser ajustados.

Para determinar os vizinhos topológicos do neurônio BMU e a intensidade com a qual o ajuste será feito nos pesos de cada neurônio, é preciso aplicar a função de vizinhança sobre os neurônios organizados no espaço de saída. Na Figura 4-19, algumas ilustrações sobre o relacionamento da função gaussiana com o espaço de saída do SOM são mostradas. Na visão (a), mostra-se a função gaussiana em perspectiva, sendo o gráfico da função formado por uma malha que alude à organização dos neurônios no espaço de saída, sob um *lattice* retangular. As visões (b) e (c) permitem verificar que cada neurônio BMU é o centro **c** da função gaussiana e que, quanto mais distante (da_{lc}) um segundo neurônio estiver do BMU, menor será o valor da função gaussiana (v_{lc}), remetendo à diminuição da alteração aplicada aos pesos deste segundo neurônio.



(a) visão em perspectiva (b) visão lateral (c) visão superior

Figura 4-19: Função de vizinhança gaussiana apresentada sob diferentes visões

Os ajustes de pesos do neurônio vencedor e de seus vizinhos é realizado seguindo o especificado no *Passo 2.1.3.* Visto que a vizinhança gaussiana está sendo considerada, todos os neurônios no espaço de saída, dentro da vizinhança topológica do neurônio vencedor, terão seus pesos alterados, no entanto, com intensidades ponderadas. Os cálculos dos ajustes de pesos são:

$$\vec{\mathbf{w}}_B = \begin{bmatrix}1{,}2\\1{,}4\end{bmatrix} + \exp(0) * 0{,}5 * \begin{bmatrix}5{,}3-1{,}2\\4{,}0-1{,}4\end{bmatrix} \approx \begin{bmatrix}3{,}3\\2{,}6\end{bmatrix}; \quad \vec{\mathbf{w}}_A = \begin{bmatrix}0{,}1\\0{,}3\end{bmatrix} + \exp(-0{,}5) * 0{,}5 * \begin{bmatrix}5{,}3-0{,}1\\4{,}0-0{,}3\end{bmatrix} \approx \begin{bmatrix}1{,}7\\1{,}4\end{bmatrix};$$

$$\vec{\mathbf{w}}_C = \begin{bmatrix}1{,}3\\0{,}1\end{bmatrix} + \exp(-0{,}5) * 0{,}5 * \begin{bmatrix}5{,}3-1{,}3\\4{,}0-0{,}1\end{bmatrix} \approx \begin{bmatrix}2{,}5\\1{,}3\end{bmatrix} \text{ e } \vec{\mathbf{w}}_D = \begin{bmatrix}0{,}1\\1{,}1\end{bmatrix} + \exp(-0{,}5) * 0{,}5 * \begin{bmatrix}5{,}3-0{,}1\\4{,}0-1{,}1\end{bmatrix} \approx \begin{bmatrix}1{,}7\\2{,}0\end{bmatrix}.$$

Para melhor entendimento em relação aos ajustes, considere a representação gráfica da Figura 4-20, na qual o reposicionamento dos neurônios, por conta do ajuste de pesos, é ilustrado. Na sequência a esse ajuste, os demais exemplares devem ser apresentados à rede neural para completar o laço FOR do *Passo 2.1.*

Os próximos passos do Algoritmo 4-5 executam a avaliação da movimentação dos neurônios, o incremento do contador de épocas e a atualização da taxa de aprendizado e do raio de vizinhança.

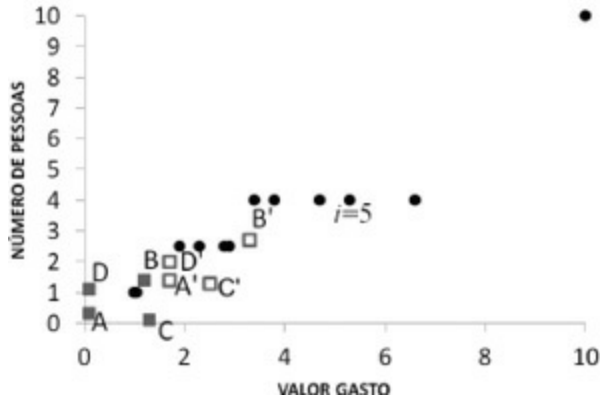


Figura 4-20: Repsresentação gráfica do ajuste de pesos dos neurônios: representação dos pesos atuais (A,B,C,D) e pesos ajustados (A’,B’,C’,D’) sobre o conjunto de dados

4.1.4.2. Exemplo prático para Mapas Auto-Organizáveis em R

Dentre os pacotes disponíveis sobre o SOM no CRAN, foi escolhido o ***kohonen*** (Wehrens e Buydens, 2007) para ilustrar este exemplo prático. Trata-se de um pacote com uma série de funções disponíveis que abrangem desde o treinamento de um SOM até opções de visualização do resultado e medidas de qualidade do SOM. A função que gera e treina um SOM é chamada `som( )`, e sua sintaxe e parâmetros podem ser vistos no Quadro 4-7.

Quadro 4-7: Sintaxe e parâmetros para a função SOM

Sintaxe para aplicação da função que implementa SOM
Usando a função `som( )` do pacote *kohonen*

Gerando o mapa

`som_model<- som(dados, grid, época, taxa_de_aprendizado)`

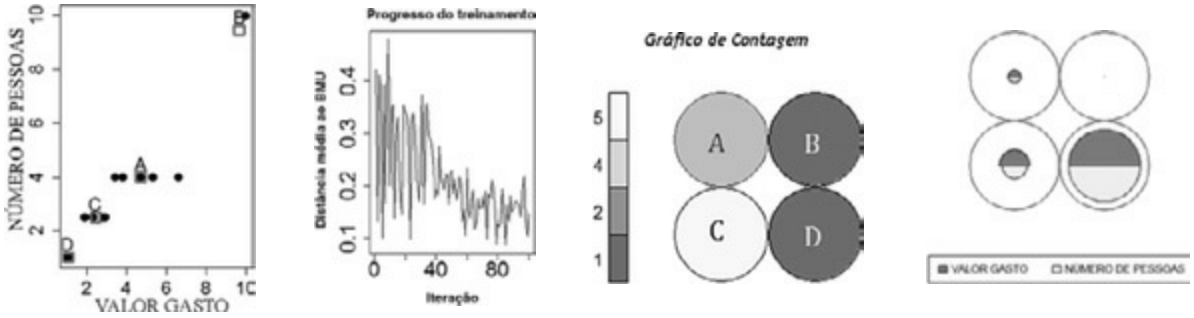
- `dados` é o conjunto de dados analisado pelo SOM, disponível em um formato de matriz em que cada linha representa um exemplar.
- `grid` é um objeto com a dimensão do mapa e seu *lattice* (`rectangular` ou `hexagonal`).
- `época` é número de vezes que o conjunto completo de dados é apresentado à rede.
- `taxa_de_aprendizado` é um vetor de dois números indicando o valor inicial para a taxa e o valor final. Por padrão, se os valores não forem alterados, a taxa iniciará em 0,05 e será reduzida linearmente até 0,01, a partir do número de épocas.

A função retorna um objeto com os valores dos parâmetros da arquitetura do SOM, incluindo os pesos dos neurônios ajustados e também os dados de treinamento.

Para a execução de um SOM, os parâmetros neste exemplo prático são: espaço de saída bidimensional 2 × 2, com *lattice* retangular, número de épocas = 100, taxa de aprendizado inicial $\eta = 0{,}05$ e final $\eta = 0{,}01$, função de vizinhança e raio de vizinhança foram mantidos conforme valores *default* para a topologia escolhida (retangular):

```
   > install.packages("kohonen")
   > library("kohonen")
   >                              vendas                                  <-
matrix(c(10,6.6,3.4,4.7,5.3,1,2.8,2.3,2.9,1.05,3.8,1.9,10,4,
   4,4,4,1,2.5,2.5,2.5,1,4,2.5), nrow=12)
   >                             som_model                                <-
som(vendas,grid=somgrid(2,2,"rectangular"),rlen=100,
   alpha=c(0.05,0.01))
```

O treinamento resulta em um objeto com diferentes informações sobre ele, o que possibilita construir gráficos que ajudam na interpretação dos resultados do SOM, por exemplo, como ilustrado na Figura 4-21: (a) visualização do posicionamento dos neurônios ajustados no espaço de entrada (quando esse espaço é de baixa dimensão, duas ou três, e pode ser representado em um gráfico – para dimensões mais altas, vários gráficos devem ser feitos, combinando os atributos descritivos dois a dois ou três a três); (b) variação da distância média de cada exemplar ao neurônio mais próximo (BMU) durante o treinamento, possibilitando acompanhar a evolução da convergência do treinamento (estabilização das mudanças ocorridas nos pesos dos neurônios da camada de saída); (c) contagem de exemplares representados por cada neurônio; e (d) influência de cada atributo descritivo em cada neurônio da camada de saída (em cinza, a influência do atributo "valor gasto", e, em branco, a influência do atributo "número de pessoas").



(a) (b) (c) (d)

Figura 4-21: Interpretações gráficas do SOM

Os gráficos mostrados na Figura 4-21(b-d) são gerados a partir da função `plot( )`, também do pacote ***kohonen***. O gráfico da Figura 4-21(a) é construído com a função `plot( )` tradicional. A sintaxe e parâmetros para

essa função são descritos no Quadro 4-8. Os comandos que geraram os gráficos discutidos são apresentados na sequência.

Quadro 4-8: Parâmetros da função PLOT do pacote KOHONEN

Sintaxe para plotagem do SOM
Usando a função `plot( )` do pacote *kohonen*
Função para visualização de resultados do SOM `plot(som_model, tipos)` • `som_model` é o objeto resultante do uso da função `som( )`. • `tipos` indica qual gráfico deve ser plotado. Há diferentes tipos, entre eles os quatro usados na Figura 4-21: changes (b), counts (c) e codes (d). A função retorna um gráfico.

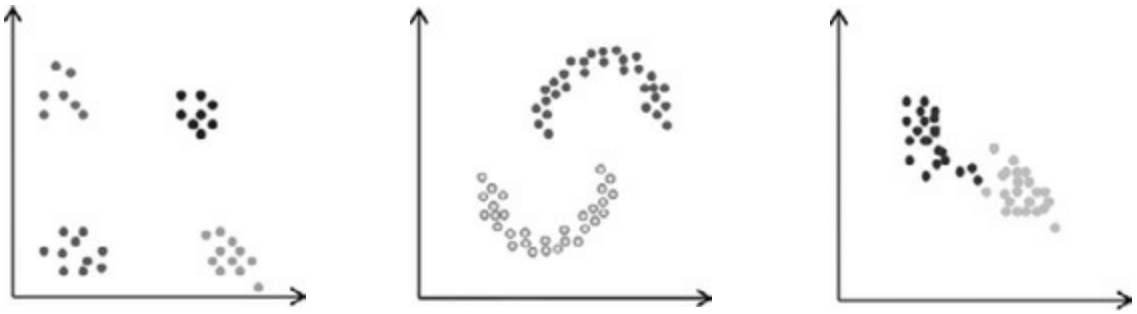
```
> plot(som_model, "changes")
> plot(som_model, "counts")
> plot(som_model, "codes")
```

Como discussão final sobre o algoritmo SOM, é importante ressaltar suas duas características principais, a quantização e a preservação topológica. Considerando os gráficos da Figura 4-21, a quantização pode ser percebida analisando inicialmente a figura (a), na qual, por exemplo, o neurônio rotulado como A está representando cinco exemplares (os cinco exemplares para os quais o neurônio A é o BMU) e pode ser entendido com a indicação de um grupo nos dados. O vetor de pesos (*codebook*) do neurônio pode ser visto como um representante para todos os exemplares para os quais ele é o BMU, e esse fato constitui a característica de quantização vetorial. Ainda, analisando conjuntamente as Figuras 4-18 e 4-21 (a), é possível notar a preservação de topologia. Note, na Figura 4-18(a), que a vizinhança topológica do neurônio A é composta pelos neurônios B e C; e, na Figura 4-21 (a), é possível verificar que os vizinhos vetoriais do neurônio A são também B e C. Isso mostra que o SOM foi capaz de manter a topologia do espaço de saída quando se adaptou sobre o espaço de entrada. Essa característica fica mais evidente quando se tem conjuntos de dados mais

complexos e também mais neurônios na camada de saída (veja, por exemplo, a Figura 4-15(d)).

4.2. Avaliação de modelos para análise de agrupamento

A **avaliação** do resultado obtido na análise de **agrupamento** é comumente chamada de validação. Na validação do modelo de grupos resultante de um algoritmo, tem-se o objetivo de avaliar se esse modelo de fato representa a organização dos exemplares no conjunto de dados sob análise. Contudo, essa expectativa de representação ideal pode ser bastante vaga e difícil de ser avaliada e, para amenizar essa dificuldade, dois critérios de avaliação podem ser considerados: compacidade e separabilidade (Halkidi, Batistakis e Vazirgiannis, 2001). O primeiro critério, compacidade, está diretamente relacionado com o objetivo de encontrar grupos com um subconjunto de exemplares que maximize a **similaridade intragrupos**, e uma forma bastante simples de medir a compacidade é usar a medida estatística de variância (Capítulo 2), de forma que, quanto menor a variância dos exemplares dentro de um grupo, maior a sua compacidade . O segundo critério, separabilidade , está diretamente relacionado com o objetivo de encontrar grupos com um subconjunto de exemplares que minimize a **similaridade intergrupos**. Ou seja, os grupos devem estar espaçados. Para medir a separabilidade, uma forma simples é analisar a distância entre dois grupos seguindo uma das seguintes abordagens (Seção 4.1.1): menor distância (*single-linkage*), distância média (*average-linkage*), maior distância (*complete-linkage*). Na Figura 4-22 (a,b,c), são ilustradas organizações de exemplares em grupos com alta e baixa compacidade e separabilidade.



(a) Alta compacidade e alta separabilidade

(b) Baixa compacidade e alta separabilidade

(c) Alta compacidade e baixa separabilidade

Figura 4-22: Representação gráfica de grupos com compacidades e separabilidades altas e baixas

A avaliação de um modelo de agrupamento, incluindo a quantificação de compacidade e separabilidade, é feita por índices de validação de agrupamento de dois tipos: índices baseados em critérios externos e índices baseados em critérios internos.

4.2.1. Índices baseados em critérios externos

A validação de agrupamento baseado em critérios externos depende da existência de algum conhecimento sobre como o conjunto de dados sob análise está estruturado, já que a validação estará baseada na comparação da estrutura de grupos gerada pelo algoritmo de agrupamento e na estrutura de grupos já conhecida. Para que esse tipo de índice de validação seja calculado, é necessário conhecer a partição do conjunto de dados, o que equivale a conhecer o atributo rótulo de cada exemplar. Entretanto, tal informação não é usada no treinamento do modelo, sendo apenas útil para avaliar o agrupamento gerado por um algoritmo.

Uma pergunta que pode surgir, nesse caso, é: se houver o conhecimento sobre as partições, qual é a vantagem em realizar uma análise de agrupamento em detrimento de uma análise preditiva? Na realidade, quando há a proposição de um novo algoritmo para realização de agrupamento, o uso de índices externos ajuda a avaliar o quão eficiente ele é ou para que tipo de organização em grupos ele pode ser mais adequado. Essa avaliação é feita por

meio da comparação dos resultados obtidos pelo novo algoritmo com os resultados obtidos a partir de um algoritmo de agrupamento notadamente adequado para tal situação ou diretamente comparando com partições conhecidas no conjunto de dados.

Existem vários índices que podem ser usados como medidas em uma comparação entre os grupos gerados por um algoritmo de agrupamento **G** e a organização de partições previamente conhecida **P**. Três índices são apresentados aqui, e outros podem ser consultados em Halkidi, Batistakis e Vazirgiannis (2001).

Para definir esses índices, é preciso definir como comparar ambas as organizações, **G** e **P**. Então, considere a organização em grupos **G** = $\{G_1, ..., G_i, ..., G_K\}$, em que K é o número de grupos; e a organização em partições **P** = $\{P_1, ..., P_j, ..., P_C\}$, em que C é o número de partições. Considere ainda todos os pares de exemplares $\{\mathbf{x}_{\rightarrow p}, \mathbf{x}_{\rightarrow q}\}$ agrupados em uma organização, sendo $p \neq q$ e $\{\mathbf{x}_{\rightarrow p}, \mathbf{x}_{\rightarrow q}\} = \{\mathbf{x}_{\rightarrow q}, \mathbf{x}_{\rightarrow p}\}$, para valorar as seguintes somas (por simplicidade, as somas serão nomeadas como A, B, C e D, constituindo-se em variáveis que assumem um número inteiro, estabelecendo notação baseada em Halkidi, Batistakis e Vazirgiannis (2001)):

- Soma A: quantidade de pares de exemplares que pertencem ao mesmo grupo em **G** e à mesma partição em **P**.
- Soma B: quantidade de pares de exemplares que pertencem ao mesmo grupo **G** e à partições **P** diferentes.
- Soma C: quantidade de pares de exemplares que pertencem a grupos **G** diferentes e à mesma partição **P**.
- Soma D: quantidade de pares de exemplares que pertencem a grupos **G** diferentes e à partições **P** diferentes.

A partir das considerações estabelecidas, alguns índices capazes de medir o grau de similaridade entre **G** e **P** podem ser considerados, como:

- *Índice de Rand* (Rand, 1971), dado por $I_{Rand} = (A + D) / (A + B + C + D)$.
- *Índice de Jaccard* (ou Coeficiente de *Jaccard*), dado por $I_{Jaccard} = A / (A + B + C)$.
- *Índice de Folkes e Mallows* , dado por $I_{FM} = \sqrt{\frac{A}{A+B} * \frac{A}{A+C}}$.

Para todos os índices citados, valores altos indicam alto grau de similaridade entre grupos e partições (máximo 1); e, quando assumem valores baixos, é indício de alto grau de dissimilaridade (mínimo 0).

4.2.2. Índices baseados em critérios internos

A validação de agrupamento baseado em critérios internos independe de qualquer tipo de informação extra sobre o conjunto de dados sob análise, sendo unicamente dependente da distribuição dos exemplares entre os grupos resultantes do algoritmo de agrupamento. Esses índices são construídos de forma a avaliar as relações entre compacidade e separabilidade dos grupos.

Dentre os índices internos mais populares, estão o Índice de Dunn (Dunn, 1973), o Índice de Davies-Bouldin (Davies e Bouldin, 1979) e o Índice Silhouette (Rousseeuw, 1987):

Índice Dunn :

$$I_{Dunn} = \min_{1 \le p \le K} \left\{ \min_{1 \le q \le K, p \ne q} \left\{ \frac{dist(G_p, G_q)}{\max disp(G_K)} \right\} \right\} \qquad \textbf{Equação 4-3}$$

em que $dist(G_p, G_q)$ é uma medida de distância entre grupos (*single-linkage*, *complete-linkage*, *average-linkage* ou distância entre centroides, como definido na Seção 4.1.1) $disp(G_K)$, é uma medida de dispersão em um grupo (maior distância dentro do grupo – *complete-linkage*, erro de quantização,[6] ou

outras medidas de dispersão), e $p, q = 1 \ldots K$ e K é a quantidade de grupos. A ideia básica desse índice é fazer uma comparação entre a distância existente entre os grupos e o tamanho do grupo mais disperso. Repare que o índice quer encontrar a relação mínima entre distâncias entre grupos e maior dispersão de um grupo. Quanto maior for essa relação, melhor será a separabilidade entre os grupos e a compacidade de grupos.

Índice Davies-Bouldin :

$$I_{DB} = \frac{1}{K}\sum_{p=1}^{K} R_p \qquad \textbf{Equação 4-4}$$

em que

$$R = max_{p \neq q} R_{pq},\ p, q = 1 .. K, \qquad \textbf{Equação 4-5}$$

e R_{pq} é uma medida de similaridade entre grupos (Halkidi, Batistakis e Vazirgiannis, 2001):

$$R_{pq} = (disp(C_p) + disp(C_q))/dist(C_p, C_q) \qquad \textbf{Equação 4-6}$$

O índice I_{DB} está interessado em encontrar a média das similaridades de cada grupo p e o grupo mais similar a ele. Nesse caso, portanto, valores baixos para o índice indicam as melhores formações de grupos, evidenciado por baixas medidas de dispersão e altas medidas de distância no cálculo de R_{pq}.

Índice Silhouette :

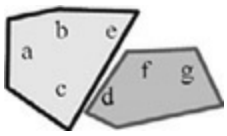
$$I_{SIL} = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}} \qquad \textbf{Equação 4-7}$$

em que $a(i)$ é a distância média do exemplar i a todos os demais exemplares de seu grupo, e $b(i)$ é a distância média do exemplar i a todos os demais exemplares do grupo mais próximo ao seu. Note que o I_{SIL} é calculado para cada exemplar, assim, o I_{SIL} de um grupo é a média dos I_{SIL} de todos os seus exemplares. E, finalmente, o I_{SIL} para um agrupamento é a média dos I_{SIL} de todos os exemplares do conjunto de dados. Esse índice resulta em um número entre –1 e 1, sendo que, quanto mais próximo de 1, mais bem definidos são os grupos, pois as distâncias $a(i)$ são bem menores que as distâncias $b(i)$, e, quanto mais próximo de –1, o contrário é observado, levando à conclusão de que a organização de grupos formada está equivocada. Esse índice foi inicialmente proposto como forma de visualizar o quão bem um exemplar se encaixa em um grupo. Detalhes sobre como construir a visualização e interpretá-la são apresentados em Rousseeuw (1986).

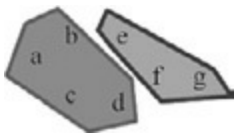
Esses índices podem ser considerados *critérios relativos*, uma vez que são comumente usados para comparar resultados de agrupamento provenientes de execuções diferentes, com condições iniciais e/ou parâmetros diferentes, de um mesmo algoritmo (Halkidi, Batistakis e Vazirgiannis, 2001).

4.2.3. Exemplo didático para avaliação de modelos para análise de agrupamento

Nesta seção, são apresentados exemplos didáticos para cálculo dos índices de avaliação externos e internos. Para estudar os índices externos, considere os grupos e partições da Figura 4-23, baseada no exposto em Rand (1971). A figura traz uma representação gráfica de exemplares (representados por letras) de um conjunto de dados. Na Figura 4-23(a), é mostrado o agrupamento (**G**) obtido a partir da aplicação de um algoritmo de agrupamento, e, na Figura 4-23(b), é mostrado o particionamento conhecido (**P**).

G: {a,b,c,e},{d,f,g} 

(a) Grupos (**G**)

P: {a,b,c,d},{e,f,g} 

(b) Partições (**P**)

Figura 4-23: Representação gráfica de agrupamentos (a) e partições (b) de um conjunto de dados (adaptada de Rand (1971))

Os exemplares devem ser analisados aos pares, a fim de quantificar a similaridade entre **G** e **P**. Esse resultado é apresentado na Tabela 4-9, em que se tem os exemplares comparados aos pares para as somas A, B, C e D (Seção 4.2.1), com marcações (**x**) quando o par atende à condição especificada na soma, e os resultados finais para cada soma.

Tabela 4-9: Indicação de exemplares combinados aos pares e quantificação das similaridades entre G e P

Pares	**ab**	**ac**	**ad**	**ae**	**af**	**ag**	**bc**	**bd**	**be**	**bf**	**bg**	**cd**	**ce**	**cf**	**cg**	**de**	**df**	**dg**	**ef**	**eg**	**fg**	**Total**
Soma A	x	x					x														x	4
Soma B				x					x				x				x	x				5
Soma C			x					x				x							x	x		5
Soma D					x	x				x	x			x	x	x						7
																						21

Note que a soma $A + B + C + D$ deve resultar em um valor igual ao número de pares de exemplares no conjunto de dados sob análise. Observe também que as somas A e D indicam similaridade entre as organizações de grupos e partições, enquanto as somas B e C indicam diferenças entre as organizações. Finalmente, com as variáveis quantificadas, calcula-se os índices. Os resultados dos índices baseados em critérios externos estudados nesta seção são:

- $I_{Rand} = (A + D) / (A + B + C + D) = (4 + 7) / (4 + 5 + 5 + 7) = 0{,}52$

- $I_{Jaccard} = A / (A + B + C) = 4 / (4 + 5 + 5) = 0,29$
- $I_{FM} = I_{FM} = \sqrt{\frac{A}{A+B} * \frac{A}{A+C}} = \sqrt{\frac{4}{4+5} * \frac{4}{4+5}} = 0,44$

Como forma de ilustrar o uso dos índices baseados em critérios internos , utilizou-se um conjunto de dados bastante simples e com um único atributo, conforme ilustrado na Figura 4-24, sendo que o resultado obtido pela análise de agrupamentos está ilustrado por meio das elipses pontilhadas (cada elipse diz respeito a um grupo). O objetivo aqui é validar o agrupamento obtido por meio dos índices *Dunn*, *Davies-Bouldin* e *Silhouette*. Supondo que a distância entre dois grupos seja dada pela distância entre seus centroides, é necessário tê-los calculados: *centroide*(G_1) = 3, *centroide*(G_2) = 8,5 e *centroide*(G_3) = 18,33. A distância euclidiana está sendo usada neste exemplo.

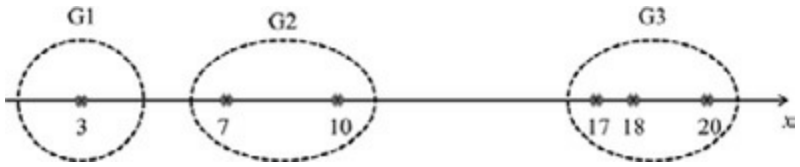


Figura 4-24: Conjunto de dados com único atributo descritivo

Índice Dunn

Para o cálculo do índice *Dunn*, é necessário conhecer as distâncias entre os grupos (aqui, a menor distância entre elementos dos grupos – *single-linkage*) e a dispersão (aqui, a maior distância entre elementos do grupo – *complete-linkage*) de cada grupo. A Tabela 4-10 apresenta tais cálculos e os resultados obtidos.

Tabela 4-10: Cálculos das distâncias entre grupos e da dispersão de cada grupo, considerando a organização em grupos ilustrada na Figura 4-24

***dist*(G_1, G_2)**	***dist*(G_1, G_3)**	***dist*(G_2, G_3)**

$dist(G_1,G_2)=$ $\min(\sqrt{(3-7)^2};\sqrt{(3-10)^2});$ $=\min(4;7)=4$	$dist(G_1,G_3)=$ $\min(\sqrt{(3-17)^2};\sqrt{(3-18)^2});\sqrt{(3-20)^2});$ $=\min(14;15;17)=14$	$dist(G_2,G_3)=$ $\min(\sqrt{(7-17)^2};\sqrt{(7-18)^2};$ $\sqrt{(7-20)^2};\sqrt{(10-17)^2};\sqrt{(10-18)^2};$ $\sqrt{(10-20)^2});$ $=\min(10;11;13;7;8;10)=7$
***disp*(G_1)**	***disp*(G_2)**	***disp*(G_3)**
$disp(G_1)=\sqrt{(3-3)^2}=0$	$disp(G_2)=\sqrt{(10-7)^2}=3$	$disp(G_3)=\sqrt{(17-20)^2}=3$

O índice de Dunn (I_{Dunn}) pode então ser calculado como:

$$I_{Dunn}=\min_{1\le p\le K}\left\{\min_{1\le q\le K,p\neq q}\left\{\frac{dist(G_p,G_q)}{\max disp(G_K)}\right\}\right\}=\min\left\{\frac{4}{3},\frac{4}{3},\frac{7}{3},\right\}=\frac{4}{3}=1{,}33$$ **Equação 4-8**

Índice Davies-Bouldin

O índice de *Davies-Bouldin*, assim como o de *Dunn*, considera a dispersão de cada grupo e a distância entre eles, porém, as relaciona de forma diferente – pela medida de similaridade dada na Equação 4-6. Os cálculos para essa medida são representados na Tabela 4-11. Também, na mesma tabela, são apresentados os resultados da comparação entre as medidas de similaridade obtidas, conforme Equação 4-4.

Tabela 4-11: Cálculos das similaridades entre grupos e comparação das similaridades, considerando a organização em grupos ilustrada na Figura 4-24

$R_{G1,G2}$	$R_{G1,G3}$	$R_{G2,G3}$
$R_{G1,G2}=\frac{0+3}{4}=0{,}75$	$R_{G1,G3}=\frac{0+3}{14}=0{,}21$	$R_{G2,G3}=\frac{3+3}{7}=0{,}86$
R_{G1}	R_{G2}	R_{G3}
$R_{G1} = max\ (R_{G1,G2}, R_{G1,G3}) = 0{,}75$	$R_{G2} = max\ (R_{G1,G2}, R_{G2,G3}) = 0{,}86$	$R_{G3} = max\ (R_{G1,G3}, R_{G2,G3}) = 0{,}86$

O índice de *Davies-Bouldin* pode então ser calculado como:

$$I_{DB} = \frac{1}{3}(0{,}75 + 0{,}86 + 0{,}86) = 0{,}82$$

Equação 4-9

Índice Silhouette é

O cálculo do Índice *Silhouette* é um pouco diferente dos anteriores. Primeiro, é preciso calcular o índice para cada grupo para, então, chegar ao índice para o agrupamento. Assim, os cálculos serão apresentados por grupo. Para o grupo G1, $I_{SIL(G1)}$= 0 por convenção (Rousseeuw, 1987), e, eventualmente, o elemento pode ser considerado um outlier ou um ruído. Nessa análise, assim como ocorre na implementação em R, esse grupo é desconsiderado do cálculo final I_{SIL}. Para o grupo G2 e G3, os cálculos são um pouco mais trabalhosos e seguem ilustrados na Tabela 4-12 (para facilitar a exposição dos resultados, está-se considerando *i* = valor do exemplar *i*). A partir dos resultados dessa tabela, tem-se: $I_{SIL(G2)}$ = (0,25 + 0,57)/2 = 0,41 e $I_{SIL(G3)}$ = (0,77 + 0,84 + 0,78)/3 = 0,80. E, finalmente, o Índice de *Silhouette* para todo o agrupamento é dado por:

$$I_{SIL} = \frac{0 + 0{,}25 + 0{,}57 + 0{,}77 + 0{,}84 + 0{,}78}{6} = 0{,}54$$

Equação 4-10

Tabela 4-12: Cálculos de a(i) e b(i) para o Índice *Silhouette*, considerando a organização em grupos ilustrada na Figura 4-24. A fórmula das distâncias no cálculo de b(i) foram omitidas para melhorar a visualização dos resultados

G2	a(i)	b(i)				$I_{SIL(G2)}$
	$a(7)=\sqrt{(7-10)^2}=3$	b(7)	3		*média*	4 – 3 / 4 = **0,25**
			4		4	

	$a(10)=\sqrt{(10-7)^2}=3$	b(10)	3		*média*	7 – 3 / 7 = **0,57**
			7		7	

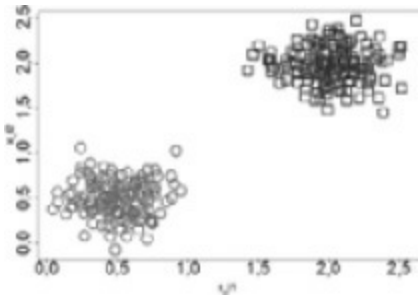
G3	a(i)	b(i)				$I_{SIL(G3)}$
	$a(17)=\frac{\sqrt{(17-18)^2}+\sqrt{(17-20)^2}}{2}=2$	b(17)	7	10	*média*	8,5 – 2 / 8,5 = **0,77**
			10	7	8,5	

	$a(18)=\frac{\sqrt{(18-17)^2}+\sqrt{(18-20)^2}}{2}=1,5$	b(18)	7	10	*média*	9,5 – 1,5 / 9,5 = **0,84**
			11	8	9,5	

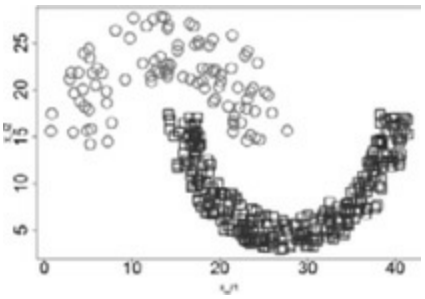
	$a(20)=\frac{\sqrt{(20-17)^2}+\sqrt{(20-18)^2}}{2}=2,5$	b(18)	7	10	*média*	11,5 – 2,5 / 11,5 = **0,78**
			13	10	11,5	

4.2.4. Exemplo prático para avaliação de modelos para análise de agrupamento em R

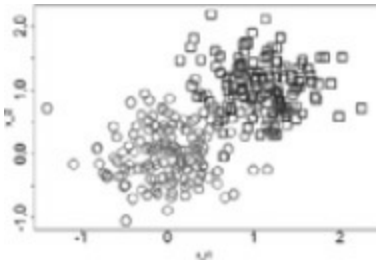
Com o objetivo de combinar todos os índices discutidos nesta seção e ilustrar de maneira prática a interpretação de cada um deles, foram gerados três conjuntos de dados representando as situações de compacidade e separabilidade discutidas antes, na Figura 4-22. Os três conjuntos de dados podem ser visualizados na Figura 4-25. Nas figuras, um grupo de exemplares está representado por círculos, e o outro, por quadrados.



(a) alta compacidade e alta separabilidade



(b) baixa compacidade



(c) alta compacidade e baixa separabilidade

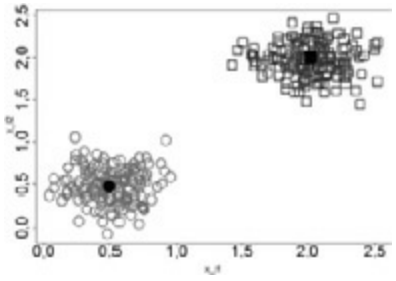
Figura 4-25: Conjunto de dados gerados para representar diferentes compacidades e separabilidades

Para gerar e visualizar esses conjuntos de dados, proceda da seguinte forma:

```
# alta compacidade e alta separabilidade
> x1 <- rbind(matrix(rnorm(300,mean=2,sd=0.2),ncol=2),
matrix(rnorm(300,mean=0.5,sd=0.2),ncol=2))
> plot(x1,type="n",
xlab="x_i1",ylab="x_i2",cex.lab=1.5,cex.axis=2.5)
> points(x1[1:150,1],x1[1:150,2],col="4",lwd=3.5,cex=3.5,pch=22)
>
points(x1[151:300,1],x1[151:300,2],col="2",lwd=3.5,cex=3.5,pch=21)
> label = as.integer(c(rep(1,150),rep(2,150)))

# baixa compacidade
# conjunto de dados disponível em
http://cs.joensuu.fi/sipu/datasets/
> url <- "C:\\Users\\LivroDM\\Agrupamento\\jain.csv"
> x2 <- read.table(file=url,header=FALSE,sep="\t")
> x2 <- as.matrix(x2)
> x2 <- cbind(x2[,1],x2[,2])
> plot(x2,type="n",
xlab="x_i1",ylab="x_i2",cex.lab=1.5,cex.axis=2.5)
> points(x2[1:97,1],x2[1:97,2],col="2",lwd=3.5,cex=3.5,pch=21)
```

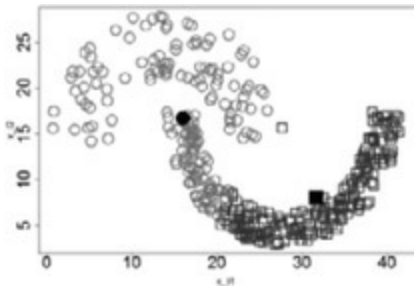
```
>
points(x2[98:373,1],x2[98:373,2],col=”4”,lwd=3.5,cex=3.5,pch=22)

# alta compacidade e baixa separabilidade
> x3 <- rbind(matrix(rnorm(300,sd=0.4),ncol=2),
matrix(rnorm(300,mean=1,sd=0.4),ncol=2))
> plot(x3,type=“n”,
xlab=“x_i1”,ylab=“x_i2”,cex.lab=1.5,cex.axis=2.5)
> points(x3[1:150,1],x3[1:150,2],col=“2”,lwd=3.5,cex=3.5,pch=21)
>
points(x3[151:300,1],x3[151:300,2],col=“4”,lwd=3.5,cex=3.5,pch=22)
> label = as.integer(c(rep(1,150),rep(2,150)))
```

Agora, os conjuntos de dados serão submetidos a um processo de agrupamento de dados pelo *k*-médias. Como já se sabe previamente o número de grupos, o parâmetro *k* foi definido como 2. O resultado dos agrupamentos obtidos em cada um dos conjuntos de dados é apresentado na Figura 4-26, e, na sequência, está o código em R que implementa esse processo para o primeiro conjunto de dados. Nessa figura, em complemento à interpretação visual dos agrupamentos gerados, estão também os índices de validação internos e externos estudados no capítulo.



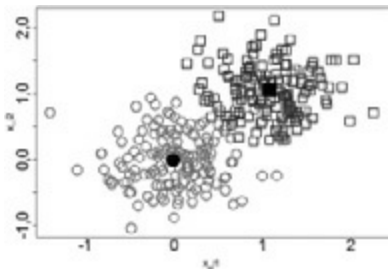
I_{Rand} =1,00 $I_{Jaccard}$ =1,00
I_{FM} =1,00 I_{Dunn} = 8,06
I_{DB} = 0,23 I_{SIL} = 0,83

(a) alta compacidade e alta separabilidade



I_{Rand} = 0,66 $I_{Jaccard}$ = 0,53
I_{FM} = 0,69 I_{Dunn} = 2,45
I_{DB} = 0,78 I_{SIL} = 0,49

(b) baixa compacidade



I_{Rand} = 0,93 $I_{Jaccard}$ = 0,87
I_{FM} = 0,93 I_{Dunn} = 2,74
I_{DB} = 0,71 I_{SIL} = 0,52

(c) alta compacidade e baixa separabilidade

Figura 4-26: Resultado do agrupamento de dados pelo *k*-médias

```
> library("stats")
> grupos <-kmeans(x1,2)
> plot(x1,type="n",
xlab="x_i1",ylab="x_i2",cex.lab=1.5,cex.axis=2.5)
> G1 <- x1[(grupos$cluster == 1),1:2]
> points(G1[,1],G1[,2],col="4",lwd=3.5,cex=3.5,pch=22)
> G2 <- x1[(grupos$cluster == 2),1:2]
> points(G2[,1],G2[,2],col="2",lwd=3.5,cex=3.5,pch=21)
> C1 <- grupos$centers[1,]
> points(C1[1],C1[2],col="1",lwd=4,cex=4,pch=15)
> C2 <- grupos$centers[2,]
> points(C2[1],C2[2],col="1",lwd=4,cex=4,pch=16)
```

A partir desse exemplo, é possível compreender, de forma mais detalhada, a relação entre os índices de validação e a qualidade dos agrupamentos obtidos. Os índices externos (I_{Rand}, $I_{Jaccard}$ e I_{FM}) tendem ao seu valor máximo (1,00) quando os grupos são identificados adequadamente. Os índices internos (I_{Dunn}, I_{DB}, I_{SIL}) são voltados à avaliação de características como compacidade e separabilidade. Enquanto para o I_{DB} os valores menores indicam melhor relação entre essas características, o I_{Dunn} e I_{SIL} indicam melhores resultados de agrupamento quando atingem valores altos. A análise de índices internos permite criar estratégias para explorar valores para os parâmetros dos algoritmos: por exemplo, pode-se executar várias vezes o *k*-

médias sobre o mesmo conjunto de dados e mesmas condições iniciais de centroides, porém, variando os valores de *k;* em seguida, aplica-se o índice para cada um dos agrupamentos obtidos em cada execução; a obtenção dos melhores valores para os índices indicará o melhor valor de *k* para aquele conjunto de dados.

A maioria dos índices apresentados são calculados com o uso do pacote ***clv*** (*Cluster Validation Techniques*) (Nieweglowski, 2013). O único que não está disponível no pacote ***clv*** é o I_{SIL} e que será calculado com o pacote ***fpc*** (Hennig, 2014), já utilizado para o DBSCAN.

Para os índices de validação externos, a função a ser aplicada é a `std.ext()`, do pacote **clv**. A sintaxe e parâmetros para essa função estão descritos no Quadro 4-9. A chamada dessa função está exemplificada aqui, a partir de um particionamento e um agrupamento de dados igual ao da Figura 4-23:

```
> install.packages("clv")
> library("clv")
> G <- as.integer(c(1,1,1,1,2,2,2))
> P <- as.integer(c(1,1,1,2,1,2,2))
> std <- std.ext(P,G)
```

Quadro 4-9: Sintaxe e parâmetros para a função STD.EXT

Sintaxe para aplicação da função que calcula os índices externos
Usando a função `std.ext( )` **do pacote *clv***
Calculando a quantidade de pares entre G e P `std <- std.ext(P,G)` • `P` é o conjunto de rótulos dos dados, representando o particionamento conhecido. • `G` é o conjunto de rótulos associado aos dados por um algoritmo de agrupamento de dados. A função retorna um objeto com as variáveis Soma A, Soma B, Soma C e Soma D (como descrito na Seção 4.2.1).

A partir do cálculo das variáveis Soma A, Soma B, Soma C e Soma D, os índices de validação externos são obtidos a partir dos seguintes comandos:

```
> rand <- clv.Rand(std)
> jaccard <- clv.Jaccard(std)
> folk.mal <- clv.Folkes.Mallows(std)
```

Para os índices de validação externos, a função a ser aplicada é a `cls.scatt.data()` também do pacote **clv**. A sintaxe e os parâmetros da função são apresentados no Quadro 4-10.

Quadro 4-10: Sintaxe e parâmetros para a função CLS.SCATT.DATA

Sintaxe para aplicação da função que calcula os índices internos
Usando a função `cls.scatt.data( )` **do pacote *clv***
Calculando a similaridade intra e intergrupos `scatt <- cls.scatt.data(dados, grupos, distância)` • `dados` é o conjunto de dados submetido ao agrupamento. • `grupos` é o conjunto de rótulos associado aos dados por um algoritmo de agrupamento. • `distância` é a métrica de distância para o cálculo da similaridade inter e intragrupos, por exemplo, `euclidean`, `manhattan`, `correlation`. A função retorna um objeto com as medidas inter e intragrupos.

Considerando que se tem um conjunto de dados (`x`), como ilustrado na Figura 4-24, e esses dados foram agrupados por um algoritmo, como o *k*-médias, e o resultado foi armazenado em uma variável (grupos), a função deve, então, ser executada:

```
> x <- c(3,7,10,17,18,20)
> G <- kmeans(x, 3)
> grupos <- as.integer(G$cluster)
> scatt <- cls.scatt.data(as.matrix(x), grupos, dist="euclidean")
```

Após calculadas as medidas inter e intragrupos, é possível calcular os índices a partir dos seguintes comandos:

```
> davies <- clv.Davies.Bouldin(scatt, "complete", "single")
> dunn <- clv.Dunn(scatt, "complete", "single")
```

Note que há dois parâmetros que devem ser previamente definidos: intercluster e intracluster. Para o caso do intercluster, as principais opções são as medidas discutidas no agrupamento hierárquico: `single`, `complete`, `average` e `centroid`. No caso do intracluster, as opções são `complete`, `average` e `centroid`. Para os exemplos aqui apresentados, foi usada a opção `centroid` para os dois casos.

O Índice do *Silhouette* pode ser gerado com a função `cluster.stats( )`, disponível no pacote ***fpc*** (Henning, 2014). A sintaxe e os parâmetros para essa função são apresentados no Quadro 4-11, e sua execução é:

```
> indices <- cluster.stats(dist(x), grupos)
```

O índice finalmente pode ser apresentado a partir do seguinte comando:

```
> silhouette <- indices$avg.silwidth
```

Quadro 4-11: Sintaxe e parâmetros para a função *CLUSTER.STATS*

Sintaxe para aplicação da função usada no cálculo do índice *Silhouette*
Usando a função `cluster.stats( )` do pacote *fpc*
Calculando índices de agrupamento, incluindo o *Silhouette* `stat <- cluster.stats(dist(dados), grupos)` • `dist(dados)` é a matriz de distâncias entre os dados. • `grupos` é o conjunto de rótulos associado aos dados por um algoritmo de agrupamento de dados. A função retorna um objeto com vários valores de índices, entre eles o valor do Índice *Silhouette*.

4.3. Leituras adicionais

Neste capítulo, foram apresentados quatro algoritmos para resolução da tarefa de agrupamento. Na literatura especializada, há uma série de outros algoritmos que podem ser úteis em diferentes situações. Em Han, Kamber e Pei (2011), são apresentados vários outros algoritmos, incluindo alguns baseados em estratégias não discutidas aqui, como estratégias baseadas em grade (do inglês, *grid*), estratégias específicas para lidar com análise de agrupamento baseada em restrições e estratégias apropriadas para lidar com o problema de outliers.

Em relação à estratégia de agrupamento hierárquico , métodos diferentes daqueles tratados neste livro podem ser de interesse para aprofundamento no tema. Em Zhang, Ramakrishnan e Livny (1996), é apresentado o BIRCH (do inglês, *Balanced Iterative Reducing and Clustering using Hierarchies*). Trata-se de um método interessante para aplicação em grandes conjuntos de dados, já que é capaz de construir os grupos, de forma incremental e dinâmica, avaliando o conjunto de dados apenas uma vez. Melhorias no resultado podem ser obtidas com algumas avaliações adicionais do conjunto de dados. O método ROCK (do inglês, *RObust Clustering using linKs*), proposto por Guha, Rastogi e Shim (2000), é uma alternativa para o objetivo de descobrir agrupamentos em conjuntos de dados nos quais os atributos são binários ou categóricos. Esse método aplica uma medida de similaridade não métrica, útil, principalmente, quando a similaridade entre exemplares advém do conhecimento de um especialista.

O *k*-médias foi o método escolhido para representar as estratégias de agrupamento por particionamento . O *k*-médias foi apresentado na sua forma básica, porém, várias alterações podem ser feitas no processo, de forma a adequá-lo a diferentes situações. As alterações mais comuns são: *k-modes* – usar *moda* em vez de *média* para adequá-lo à aplicação sobre conjuntos de dados com atributos categóricos (também a forma de medir a similaridade

entre exemplares deverá ser alterada, nesse caso) (Huang, 1998); *k*-protótipos – estratégia para unir *k*-médias e *k-modes* (Huang, 1998); *k*-medoides – usar a *mediana* (o exemplar mais central no grupo) em vez de a *média* para diminuir a sensibilidade a ruídos nos dados (Kaufman e Rousseeuw, 1990); *fuzzy-c-means* – relaxar as restrições que definem os grupos e permitir a pertinência parcial de um exemplar a um grupo (Bezdek e Dunn, 1974).

Em relação à estratégia de agrupamento por densidade , também é interessante estudar os algoritmos OPTICS (Ankerst *et al.*, 1999) e DENCLUE (Hinneburg e Keim, 1998). O primeiro apresenta uma proposta diferenciada que permite analisar graficamente a estrutura dos grupos, de forma que o algoritmo se torna especialmente interessante quando se tem conjuntos de dados localizados em espaços de alta dimensão. O segundo é caracterizado por representar grupos por meio de equações que modelam suas funções de densidade. Esse algoritmo é especialmente útil por lidar muito bem com ruído e grupos de formas bastante diversas e, por isso, tem bom desempenho para dados do tipo multimídia, incluindo conjuntos de dados de bases moleculares (bioinformática).

A rede neural SOM foi apresentada neste livro sob sua forma básica. É recomendado ao leitor que acesse referências bibliográficas especializadas sobre SOM, para que o entendimento de todas as suas nuances se torne mais sedimentado. Livros e artigos do Prof. Teuvo Kohonen e de seu grupo de pesquisa devem ser acessados. Um especial destaque é feito para os artigos (Kaski *et al.*, 1998) e (Oja *et al.*, 2002), que trazem um levantamento de outros artigos sobre o tema. Em relação a medidas de avaliação do SOM, é indicada a leitura de (Kiviluoto, 1996).

Ainda, para quem necessita escolher um algoritmo de agrupamento adequado para resolver um problema em especial, Halkidi, Batistakis e Vazirgiannis (2001) apresentam um estudo interessante sobre vantagens e desvantagens de vários algoritmos, discutindo para que tipos de organização

de grupos cada algoritmo é mais adequado, apresentando informações sobre complexidade computacional e sobre sensibilidade a ruído e outliers.

Como complemento ao exposto neste livro sobre medidas de avaliação da organização em grupos alcançada a partir do uso de um algoritmo de agrupamento, sugere-se a leitura de Halkidi, Batistakis e Vazirgiannis (2001), na qual vários índices de validação adicionais são apresentados, bem como estratégias para usar tais índices para determinar o número de grupos ideal para ser admitido a partir de um conjunto de dados sob análise. Complementarmente, a leitura de Pakhira, Bandyopadhyay e Maulik (2004) é interessante para se ter contato com um experimento sobre a aplicação de vários índices de validação. Índices de validação baseados na Teoria da Informação são o núcleo da discussão apresentada em Vinh, Epps e Bailey (2010).

Os autores Dalton, Ballarin e Brun (2009) e Brun *et al*. (2007) apresentam índices de validação de grupos baseados em critérios relativos, muito usados na área de Bioinformática, mas cuja interpretação pode ser adaptada para que os índices sejam usados em outras áreas de aplicação.

4.4. Exercícios

1. Uma empresa possui um website onde seus serviços são anunciados e promovidos. A partir de informações sobre os usuários que navegam pelo website (idade e tempo de navegação) a empresa deseja fazer um estudo de perfis. A Tabela 4-13 apresenta os dados que foram coletados sobre os usuários e a partir dela, pede-se:

a) Como cada atributo do conjunto de dados pertence a um domínio e escala diferentes, faça a normalização dos dados com a técnica min-max para os valores no intervalo [0,1];

b) Com os dados normalizados, execute os seguintes algoritmos de agrupamento: *k*-médias, AGNES e DBSCAN. Para todos os algoritmos utilize a distância Euclidiana.

c) Para fins de visualização, apresente gráficos de dispersão com os resultados obtidos para os diferentes algoritmos, considerando diferente valores para os parâmetros dos mesmos (use cores para diferenciar os grupos). Ainda, para a finalidade de análise, apresente os cálculos dos índices de avaliação baseados em critérios internos.

Tabela 4-13: Conjunto de dados USUÁRIOS-NAVEGAÇÃO

	TEMPO	**IDADE**
$\mathbf{x}_1$	10	70
$\mathbf{x}_2$	1	71
$\mathbf{x}_3$	2	65
$\mathbf{x}_4$	3	72
$\mathbf{x}_5$	2	75
$\mathbf{x}_6$	6	17
$\mathbf{x}_7$	7	20

$\mathbf{x}_8$	8	30
$\mathbf{x}_9$	10	23

2. A partir da aplicação dos algoritmos hierárquicos aglomerativo (AGNES e DIANA), apresente resultados de agrupamento na forma de dendrogramas, considerando três formas de aplicação da medida de distância: menor (*single-linkage*), maior (*complete-linkage*) e média (*average-linkage*). Compare os resultados obtidos. As distâncias entre os exemplares são dadas na Tabela 4-14.

Tabela 4-14: Matriz de distâncias

	$\mathbf{x}_1$	$\mathbf{x}_2$	$\mathbf{x}_3$	$\mathbf{x}_4$	$\mathbf{x}_5$	$\mathbf{x}_6$
$\mathbf{x}_1$	0,00					
$\mathbf{x}_2$	1,41	0,00				
$\mathbf{x}_3$	2,00	1,42	0,00			
$\mathbf{x}_4$	3,60	3,21	3,22	0,00		
$\mathbf{x}_5$	4,50	3,20	3,80	1,00	0,00	
$\mathbf{x}_6$	5,10	4,00	3,20	2,20	2,41	0,00

3. Na Figura 4-27, são apresentados um conjunto de dados (a), o gráfico de dispersão dos exemplares desse conjunto (b) e os espaços matricial e vetorial (inicial) de um SOM. A partir dessas informações e da escolha de um exemplar do conjunto de dados, responda:

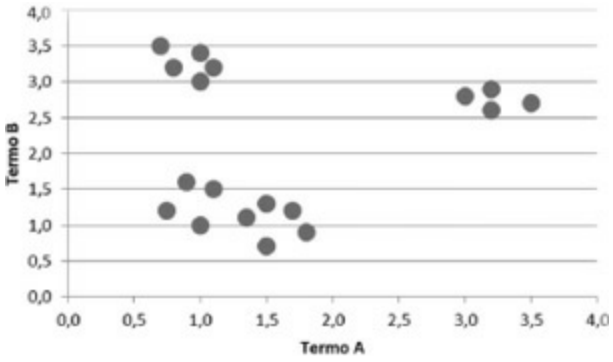
a) Qual é o neurônio vencedor? Mostre os cálculos que levam à descoberta do neurônio vencedor.

b) Apresente os cálculos para atualização do neurônio vencedor e de um de seus vizinhos topológicos, considerando uma função de vizinhança gaussiana (como realizado na Seção 4.1.4.1) e taxa de aprendizado igual a 1.

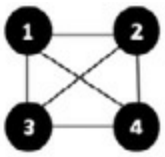
c) Apresente o gráfico de dispersão referente ao espaço vetorial, considerando os exemplares, os neurônios com seus pesos iniciais e, para o caso dos dois neurônios atualizados, com seus pesos atualizados – use cores diferentes na plotagem dos neurônios para indicar as mudanças de seus posicionamentos.

	Termo A	**Termo B**
$\mathbf{x}\rightarrow_1$	1,0	1,0
$\mathbf{x}\rightarrow_2$	0,8	1,2
$\mathbf{x}\rightarrow_3$	1,1	1,5
$\mathbf{x}\rightarrow_4$	0,9	1,6
$\mathbf{x}\rightarrow_5$	1,4	1,1
$\mathbf{x}\rightarrow_6$	1,5	0,7
$\mathbf{x}\rightarrow_7$	1,0	3,0
$\mathbf{x}\rightarrow_8$	0,8	3,2
$\mathbf{x}\rightarrow_9$	0,7	3,5
$\mathbf{x}\rightarrow_{10}$	1,0	3,4
$\mathbf{x}\rightarrow_{11}$	1,1	3,2
$\mathbf{x}\rightarrow_{12}$	3,0	2,8
$\mathbf{x}\rightarrow_{13}$	3,2	2,9
$\mathbf{x}\rightarrow_{14}$	3,5	2,7
$\mathbf{x}\rightarrow_{15}$	3,2	2,6
$\mathbf{x}\rightarrow_{16}$	1,5	1,3
$\mathbf{x}\rightarrow_{17}$	1,8	0,9
$\mathbf{x}\rightarrow_{18}$	1,7	1,2

(a) Conjunto de dados



b) gráfico de dispersão



$\vec{w}_1 = \{0{,}8, 3{,}4\}$
$\vec{w}_2 = \{3{,}0, 2{,}5\}$
$\vec{w}_3 = \{1{,}2, 1{,}3\}$
$\vec{w}_4 = \{1{,}5, 0{,}8\}$

c) Espaços matricial e vetorial (iniciais) de uma rede SOM

Figura 4-27: Informações sobre o conjunto de dados e sobre a rede SOM

4. O câncer de mama consiste no desenvolvimento anormal das células da mama, que se multiplicam repetidamente até formarem um tumor maligno. O diagnóstico do tumor, ainda no estágio inicial, permite uma eficiência de até 99% de cura. Esse diagnóstico é possível com a mamografia, exame muito simples que consiste em um raio-X da mama, como ilustrado na Figura 4-28. Um grupo de pesquisa em Computação desenvolveu um sistema capaz de extrair características da mama por meio da imagem de mamografia. Em um estudo inicial do sistema, foi realizada uma análise com o exame de seis pacientes. As características medidas encontram-se disponíveis no conjunto de dados apresentado na Tabela 4-15. A partir da aplicação do algoritmo *k*-médias, pede-se:

a) Considerando os exemplares $\mathbf{x}_{\rightarrow 2}$ e $\mathbf{x}_{\rightarrow 4}$ como centroides iniciais, e a distância euclidiana como a métrica de similaridade, apresente todos os cálculos envolvidos na solução do problema.

b) De forma experimental com uso do R, repita o trabalho de descoberta de agrupamentos; porém, faça uso também dos algoritmos AGNES e DBSCAN. Avalie os resultados fornecidos pelos algoritmos a partir do uso de diferentes parâmetros.

Figura 4-28: Imagem de raio-X obtido em um exame de mamografia

Tabela 4-15: Conjunto de dados MAMOGRAFIA

	TEXTURA	**PERÍMETRO**	**ÁREA**	**SUAVIDADE**
$\mathbf{x}\rightarrow_1$	5	1	4	2
$\mathbf{x}\rightarrow_2$	4	1	1	3
$\mathbf{x}\rightarrow_3$	6	6	6	3
$\mathbf{x}\rightarrow_4$	8	6	10	6
$\mathbf{x}\rightarrow_5$	10	10	8	8
$\mathbf{x}\rightarrow_6$	9	8	7	4

5. Para cada um dos conjuntos de dados citados a seguir, disponíveis no repositório UCI – *Machine Learning* (Lichman, 2013), e com o uso do R: (a) resolva o problema de descoberta de agrupamentos usando um dos algoritmos discutidos neste capítulo; (b) avalie os resultados obtidos a partir do uso de diferentes valores para os parâmetros do algoritmo escolhido. No processo de descoberta de agrupamentos, lembre-se de desconsiderar o

atributo de rótulo do conjunto de dados quando este estiver disponível. Para avaliação, aplique tanto os índices baseados em critérios internos quanto em critérios externos (se o atributo de rótulo estiver disponível). Ainda, verifique se o conjunto de dados precisa ser pré-processado.

a) Iris Data set (Fisher, 1950) **(http://archive.ics.uci.edu/ml/datasets/Iris)**

b) Letter Recognition Data Set (Frey e Slate, 1991) **(https://archive.ics.uci.edu/ml/datasets/Letter+Recognition)**

c) Libras Movement Data Set (Dias *et al*., 2009) **(http://archive.ics.uci.edu/ml/datasets/Libras+Movement)**

d) 20Newsgroup Data Set (Mitchell, 1997) (Joachims, 1996) **(https://archive.ics.uci.edu/ml/datasets/Twenty+Newsgroups)**

CAPÍTULO 5

REGRAS DE ASSOCIAÇÃO

Descoberta de regras de associação é o processo de analisar os relacionamentos existentes entre atributos de uma base de dados transacional, com o objetivo de encontrar associações ou correlações. Nesse caso, uma transação armazenada é um evento no domínio de análise (um jantar em um restaurante, por exemplo), e um atributo é um dos itens envolvidos no evento (como um prato qualquer do restaurante escolhido para fazer parte da refeição). A existência de associações ou a correlação entre os atributos implica que eles frequentemente aparecem juntos em uma transação ou que uma variação na frequência de observação de um atributo num conjunto de transações ocorre com uma variação na frequência de observação de um segundo atributo nesse mesmo conjunto de transações. Ainda nesse contexto, é pertinente a análise de observações ordenadas de itens que frequentemente ocorrem na base de dados transacional (clientes pedem uma “tábua de queijos” para, na sequência, solicitarem uma “porção de mandioca frita”) ou a análise de subconjuntos de itens que formam as associações ou correlações (clientes que compram “pão e leite” também compram “manteiga e patê”; clientes que compram “pão italiano e vinho” também compram “azeite de oliva e vinagre balsâmico”).

As regras de associação, independentemente de expressarem associações simples, ordenadas ou correlações, são comumente representadas por meio de afirmações do tipo SE ENTÃO, sendo também interpretadas como

implicações do antecedente da regra (ou premissa) para o seu consequente (ou conclusão). Assim, algoritmos usados para descobrir tais regras são baseados no pressuposto de que a presença de um atributo em um evento implica a presença de outro atributo no mesmo evento.

As áreas de aplicações para a tarefa de mineração “descoberta de regras de associação ” são diversas. As aplicações mais comuns são as relacionadas com análise de compras (*market basket analysis*), como sistemas de *e-Commerce* ou análise de perfis de clientes de um estabelecimento (ou segmento) comercial. Nessa área, as regras de associação geralmente são apresentadas da seguinte forma: SE o cliente compra determinado produto (ou um subconjunto de produtos), ENTÃO o cliente também compra outro produto (ou outro subconjunto de produtos). Entretanto, outros domínios podem se beneficiar da resolução da tarefa de regras de associação, como: análise de mercado de ações, no qual o comportamento das ações se associa a acontecimentos mundiais; análise de desempenho físico, quando resultados de treinamentos se associam a condições corporais; análise de comportamento eleitoral, em que resultados de pesquisas de opinião se associam a notícias veiculadas na mídia; serviços bancários; descoberta de preferências de clientes; previsão de doenças a partir da investigação de hábitos de pacientes etc.

Convém ressaltar que os resultados obtidos na descoberta de regras de associação são considerados de fácil interpretação, pois se trata de regras que podem ser expressas em “linguagem natural”, cuja semântica está explícita. Essas características não estão presentes na maioria dos resultados das tarefas de mineração de dados, nas quais se faz necessário um trabalho de pós-processamento para que uma interpretação semântica possa ser atribuída às descobertas realizadas.[1] No entanto, análises posteriores, motivadas pela necessidade de compreensão das regras de associação em relação ao contexto em que ocorrem, podem levar a conhecimentos interessantes sobre perfis de

clientes, comportamentos de processos, entre outros, ou podem levar à conclusão de que apenas um conhecimento inútil foi descoberto.

Uma vez mineradas as regras de associação, elas podem ser usadas para melhorar processos inerentes ao domínio estudado. Seguindo no exemplo de regras de associação relacionadas com a análise de compras, um restaurante self-service pode organizar os balcões expositores, distribuindo os pratos a partir de diretrizes extraídas das regras descobertas. Duas estratégias podem ser usadas: a primeira é aproximar, no balcão, os pratos que aparecem juntos em uma regra de associação, fazendo com que clientes que não costumavam saboreá-los juntos os vejam e decidam fazê-lo; a segunda é colocá-los em lugares distantes, de forma que os clientes que costumam escolhê-los necessitem procurar por eles nos balcões e, assim, passem os olhos por diversos outros pratos e decidam escolhê-los também, consumindo uma quantidade mais diversa e maior de comida. Nesse contexto, profissionais da área de marketing podem embasar suas estratégias de ação no conhecimento que as regras de associação fornecem, de forma a otimizar suas campanhas.

As estratégias algorítmicas aplicadas na resolução da tarefa de descoberta de regras de associação geralmente são compostas de duas etapas, sendo a primeira para verificar itens que frequentemente aparecem juntos, e a segunda para a geração das regras. São dois os principais algoritmos que frequentemente aparecerem na literatura: o Apriori (Agrawal e Srikant, 1994) e o FP-*Growth* (Han Pei e Yin, 2000; Zaki, 1997), que diferem pela forma como organizam os dados durante seu processo de execução, o que influencia no tempo necessário para finalizar as análises e gerar as regras, sendo o FP-*Growth* mais eficiente que o Apriori. Tais algoritmos operam, em sua forma-padrão, com atributos categóricos nominais e atributos binários. Para outros tipos de atributos, há a necessidade de pré-processamento ou adaptação dos algoritmos.

Outra característica interessante de tais algoritmos é que critérios de qualidade do resultado estão incorporados em seu processo de execução,

diferentemente de outros algoritmos usados em mineração de dados, em que é necessário aplicar critérios de qualidade de avaliação de resultados após sua execução.

Este capítulo discute vários conceitos, teóricos e práticos, necessários ao entendimento de como pode se dar a resolução da tarefa de descoberta de regras de associação. Os algoritmos Apriori e FP-*Growth* são apresentados, e a implementação do Apriori, usando R, é fornecida. Além disso, a aplicação desses algoritmos em um contexto como um exemplo didático é discutida. Ao final do capítulo, são sugeridos exercícios para fixação dos conteúdos e leituras adicionais que trazem informações mais aprofundadas sobre o assunto.

5.1. A tarefa de descoberta de regras de associação

Na resolução da tarefa de **descoberta de regras de associação** , algumas definições e estratégias são classicamente estabelecidas pela literatura especializada na área. Essa seção se destina primeiramente a apresentá-las com a formalização que se seguirá no capítulo, e, na sequência, os algoritmos Apriori e FP-*Growth* são introduzidos.

5.1.1. Conceitos básicos

Um conjunto de dados transacionais, é composto por vários exemplares de dados, como já discutido neste livro. Contudo, para o caso da tarefa de descoberta de regras de associação, um conjunto de dados deve ser, na realidade, um conjunto de dados transacionais, no sentido de que cada exemplar diz respeito a uma transação realizada (ou evento ocorrido) no domínio de aplicação. Além disso, cada uma das transações é composta por uma série de itens (ou elementos). Os itens que fazem parte da transação são, necessariamente, pertencentes a um conjunto de itens relacionado com o domínio de aplicação.

Formalmente, em um domínio de aplicação, existe um **conjunto de itens do domínio** $I = \{i_1,..., i_m\}$. Uma **transação T** é composta pela ocorrência de um subconjunto desses itens, ou seja, $T = \{i_1,..., i_l\}$, tal que $i_l \subset I$ e $l <= m$. Também é possível dizer que $T \subseteq I$. Um conjunto de transações, representado por *TID*, é então a base de dados transacional. Cada transação nesse conjunto é representada por T_{ID}. A Figura 5-1 ilustra uma base de dados transacional genérica sob duas formas de representação.

Base de dados transacional T_{ID}

T_{ID}	i_1	i_2	i_3	i_4

T_1= {i_1,i_3}	T_1	1	0	1	0
T_2 = {i_1,i_2, i_3}	T_2	1	1	1	0
T_3 = {i_1}	T_3	1	0	0	0
T_4 = {i_1,i_3,i_4}	T_4	1	0	1	1
T_5 = {i_3}	T_5	0	0	1	0
T_6 = {i_1,i_2}	T_6	1	1	0	0
T_7 = {i_1,i_2,i_3,i_4}	T_7	1	1	1	1
(a)	(b)				

Figura 5-1: Exemplo de base transacional genérica: (a) representação por conjuntos; (b) representação matricial

Por simplicidade, neste texto, em relação à representação matricial, o valor *0* na base de dados indica que o item não ocorre na transação, e o valor *1* indica que o item ocorre na transação. Na abordagem tratada neste livro, as regras de associação serão geradas a partir da ocorrência de valores 1.

Um conjunto de itens, ou um subconjunto de itens do domínio, é frequentemente denominado ***itemset*** . Um *itemset* composto por *k* itens é chamado de *k-itemset*. Assim, um *itemset* do tipo {i_1, i_3} é chamado de *2-itemset*; um *itemset* do tipo {i_1, i_3, i_4} é chamado de *3-itemset*; e assim por diante.

Uma **regra de associação** é do tipo $A \vee B$, sendo que A e B são, ambos, *itemsets* compostos de itens pertencentes a I, e $A \cap B = \varnothing$. Um exemplo de regra genérica a partir da base transacional da Figura 5.1 é: $\{i_2\} \vee \{i_1, i_3\}$, em que $A = \{i_2\}$ e $B = \{i_1, i_3\}$.

Dois conceitos importantes para a resolução da tarefa de regras de associação são as medidas **suporte** e **confiança** . A medida **suporte**, quando aplicada a um *itemset*, diz respeito à frequência desse *itemset* na base transacional sob análise. Ou seja, o suporte de um *itemset* é a frequência com que os itens que o compõem aparecem juntos em transações individuais da

base de dados transacional. Essa frequência é geralmente expressa em termos percentuais. Por exemplo, o *2-itemset* $\{i_1, i_3\}$ tem suporte igual a 57%, uma vez que os itens i_1 e i_3 aparecem juntos em quatro (T_1, T_2, T_4 e T_7) das sete transações existentes na base. Quando a medida **suporte** é aplicada a uma regra, ela diz respeito à frequência com que todos os itens dos dois *itemsets* envolvidos na regra aparecem juntos em transações individuais da base de dados transacional. Ou seja, o suporte de uma regra do tipo $A \vee B$ é o suporte do *itemset* formado por $A \cup B$, em que A e B são também *itemsets*. Assim, o suporte de uma regra dada por $\{i_2\} \vee \{i_1, i_3\}$, na base transacional da Figura 5.1, é o suporte do *itemset* $\{i_1, i_2, i_3\}$, que é 28% (2/7). Portanto, o suporte de um *itemset* ou de uma regra diz respeito a quão frequentes eles são no domínio de aplicação analisado. A medida de suporte para uma regra de associação pode ser dada por:

$$\text{suporte}_{regra}(A \cup B) = \frac{cont(A \cup B)}{cont(T)}$$

Equação 5-1

em que *cont* é uma operação de contagem; $\cup$ é uma operação derivada da Teoria de Conjuntos; $A \cup B$ indica a união dos *itemsets* da premissa (A) e da conclusão (B) da regra; e T é o conjunto completo de transações. Assim, o numerador é a contagem de transações nas quais tanto os elementos do *itemset* A quanto os elementos do *itemset* B aparecem juntos, e o denominador é a contagem de transações existentes.

A medida **confiança** se aplica apenas às regras e tem o objetivo de expressar uma noção da importância e da confiabilidade de uma regra, dada a possibilidade de sua ocorrência. A medida também é geralmente expressa por meio de percentual e dada pela razão entre o suporte da regra e o suporte da premissa da regra. Para a regra $\{i_2\} \vee \{i_1, i_3\}$, a confiança é 65% (28/43). O significado por trás dessa porcentagem é que “em 65% da vezes em que a

premissa da regra ocorre, a conclusão também ocorre". A medida de confiança para uma regra de associação pode ser dada por:

$$\text{confiança}_{regra}(A \cup B) = \frac{\text{suporte}_{regra}(A \cup B)}{\text{suporte}_{itemset}(A)}$$ **Equação 5-2**

de forma que a confiança pode também ser entendida como a probabilidade de a regra $A \vee B$ ocorrer, dado que sua premissa A ocorre.

5.1.2. Itens frequentes e Propriedade Apriori

A frequência de um item em uma base de dados transacional é elemento-chave para o desenvolvimento de algoritmos de descoberta de regras de associação. Assim, o estudo dos itens frequentes e da propriedade Apriori, associada à frequência de itens, é importante nesse contexto.

Um item ou um *itemset* é considerado frequente quando seu suporte satisfaz a um suporte mínimo (um número entre *0* e *100*, se percentuais estiverem sendo usados, e, caso contrário, um número entre *0* e *1*) pré-estabelecido. Quem determina o suporte mínimo é o usuário (pessoa que aplica algoritmos de descoberta de regras de associação em um domínio de aplicação), e essa determinação é dependente do contexto, não havendo uma regra pré-definida para seu estabelecimento. Trata-se, portanto, de um parâmetro que deve ser estudado para cada caso de aplicação.

É importante notar que encontrar os *itemsets* frequentes em uma base de dados transacional grande é uma tarefa custosa, principalmente se o suporte mínimo estabelecido for um número pequeno. Dado que os algoritmos necessitam saber quais são todos os *itemsets* frequentes para a base transacional em análise, estratégias eficientes para realizar essa tarefa devem ser estabelecidas. Para ter uma noção do volume de trabalho envolvido, note que, dado um *k-itemset* frequente, todos os (*k-1*)-*itemsets* derivados dele

serão também frequentes; assim, a existência de um *10-itemset* frequente implica a existência de 2^{10}-1 *itemsets* frequentes (1023 *itemsets*), formados por todas as combinações de tamanho 1 a 9 de itens pertencentes ao *10-itemset*, mais o *itemset* original. Tal princípio é conhecido como **Propriedade Apriori** : todo subconjunto não vazio de um *itemset* frequente é também frequente.

Como exemplo, considere novamente a base de dados transacional da Figura 5-1. O *3-itemset* $\{i_1, i_2, i_3\}$ tem suporte de 28%, ou seja, aparece em 28% das transações da base. Se um usuário estabelecer que o suporte mínimo deve ser 20%, ele será um *itemset* frequente. Analisando todos os *itemsets* formados pelos itens i_1, i_2 e i_3, de tamanho menor que 3 e seus suportes, obtêm-se os números apresentados na Tabela 5-1. Note que todos os valores de suporte são iguais ou maiores que o suporte do *3-itemset* em análise.

Tabela 5-1: *Itemsets* derivados a partir do *3-itemset* $\{i_1, i_2, i_3\}$ e os respectivos valores de suporte

Itemset	**Suporte**
$\{i_1,i_2\}$	43%
$\{i_1, i_3\}$	57%
$\{i_2, i_3\}$	28%
$\{i_1\}$	86%
$\{i_2\}$	43%
$\{i_3\}$	71%

Essa propriedade é derivada a partir da seguinte regra: se um *k-itemset* não for frequente, então a adição de qualquer novo item *i* a ele gerará um novo (*k+1*)-*itemset*, que, com certeza, será tão frequente quanto ele ou terá uma frequência ainda menor. Assim, uma vez observado que um *k-itemset* não é frequente, não é mais necessário avaliar nenhum super *itemset* dele.

Como exemplo, a base de dados transacional da Figura 5-1 e o *3-itemset* $\{i_1, i_2, i_3\}$ de suporte 28% são novamente analisados. Para criar o único super *itemset* possível nesse caso, o item i_4 é acrescentado ao *itemset* sob análise, resultando no *4-itemset* $\{i_1, i_2, i_3, i_4\}$. Note que apenas uma transação da base de dados envolve os quatro itens juntos, logo, o seu suporte é 14%, e ele não é considerado frequente (levando em conta o suporte mínimo de 20% definido anteriormente). É preciso notar que, se uma nova transação ocorrer e envolver os quatro itens, uma nova configuração da base de dados será criada, e todas as contagens deverão ser atualizadas. Esse caso é ilustrado na Figura 5-2, em que o suporte dos *itemsets* foi alterado, e a Propriedade Apriori continua válida.

Base de dados transacional TID
$T_1 = \{i_1, i_3\}$
$T_2 = \{i_1, i_2, i_3\}$
$T_3 = \{i_1\}$
$T_4 = \{i_1, i_3, i_4\}$
$T_5 = \{i_3\}$
$T_6 = \{i_1, i_2\}$
$T_7 = \{i_1, i_2, i_3, i_4\}$
$T_8 = \{i_1, i_2, i_3, i_4\}$

4-itemset	**Suporte**
$\{i_1, i_2, i_3, i_4\}$	25%

3-itemset	**Suporte**
$\{i_1, i_2, i_3\}$	37%
$\{i_1, i_3, i_4\}$	37%

2-itemset	**Suporte**
$\{i_1, i_2\}$	50%
$\{i_1, i_3\}$	62%
$\{i_1, i_4\}$	37%
$\{i_2, i_3\}$	37%
$\{i_2, i_4\}$	25%

1-itemset	**Suporte**
$\{i_1\}$	87%
$\{i_2\}$	50%
$\{i_3\}$	75%
$\{i_4\}$	37%

$\{i_1, i_2, i_4\}$	25%	$\{i_3, i_4\}$	37%
$\{i_2, i_3, i_4\}$	25%		

Figura 5-2: Novo contexto para a base de dados transacional; os *itemsets* e seus respectivos suportes. Note que transações diferentes podem envolver o mesmo subconjunto de itens

Essas constatações permitem criar uma estratégia simples, porém eficiente, para encontrar os *itemsets* frequentes em uma base de dados transacional, como será visto no estudo dos algoritmos, ainda neste capítulo.

5.1.3. Descoberta de regras de associação e medidas de avaliação

A partir da análise de itens frequentes, as regras de associação podem ser derivadas. De forma simplificada, a derivação dessas regras parte dos itens e *itemsets* frequentes, e, por meio de sua combinação, as associações presentes no domínio em análise são descobertas. Por exemplo, a partir do *2-itemset* frequente $\{i_1,i_3\}$, duas regras de associação podem ser geradas: $\{i_1\} \vee \{i_3\}$ e $\{i_3\} \vee \{i_1\}$. Na Figura 5-3, a transposição desse exemplo genérico para o domínio de aplicação de um restaurante é ilustrada, em que os itens são elementos do cardápio, e as transações são pedidos dos clientes.

2-itemset = {file à parmegiana, batatas fritas}
Regra 1: file à parmegiana batatas fritas
Regra 2: batatas fritas file à parmegiana

Figura 5-3: Exemplo de geração de regras de associação a partir de um *itemset* de tamanho 2

Embora parecidas, as regras têm significados diferentes. A primeira regra significa que um cliente que pede filé à parmegiana frequentemente também pede batatas fritas. Se essa regra de associação for válida, então grande parte dos clientes desse restaurante não apreciarão comer o prato file à parmegiana sem que o acompanhamento de batatas fritas lhe seja também servido. Se a segunda regra for considerada válida, haverá outra conotação, a de que

grande parte dos clientes acha que batatas fritas devem ser servidas com file à parmegiana. Note que, considerando o senso comum, poderíamos arriscar afirmar que a primeira regra é válida, mas a segunda não, uma vez que batatas fritas podem acompanhar uma variedade de outros pratos e também serem servidas sem acompanhamentos.

A forma de averiguar se uma regra é válida ou não em um domínio sob análise é por meio da análise das medidas de suporte e confiança. Espera-se que os valores obtidos para essas medidas na análise de cada regra de associação estejam acima de limiares mínimos estabelecidos pelo usuário (um limiar mínimo para o suporte e um limiar mínimo para a confiança). Diz-se que uma regra que possui suporte e confiança acima dos respectivos limiares mínimos é uma regra forte e, portanto, considerada válida.

Suponha agora que as regras apresentadas na Figura 5-3 apresentem o mesmo valor para suporte (50% – metade dos pedidos realizados no restaurante envolvem *filé à parmegiana* e *batatas fritas*) e que a primeira regra apresente confiança de 83%, e a segunda, de 71%. Suponha também que os limiares mínimos estabelecidos foram 40% para suporte e 80% para confiança. Esses limiares mínimos estão informando que, para que uma regra seja aceita, ela deve estar presente em pelo menos 40% das transações da base de dados transacional e que sua conclusão deve ocorrer em pelo menos 80% das vezes em que sua premissa ocorre. Nesse cenário:

- A primeira regra (*filé à parmegiana* $\vee$ *batatas fritas*) é considerada forte e será aceita como válida: *filé à parmegiana* e *batatas fritas* aparecem em metade dos pedidos dos restaurantes e, além disso, considerando todos os pedidos de *filé à parmegiana*, 83% contêm também *batatas fritas*. O suporte da regra é suficiente para que seja considerada frequente no domínio em análise, e a confiança da regra é suficiente para que seja aceita como frequente quando se considera a parte do domínio em análise no qual o *filé à parmegiana* aparece.

- A segunda regra (*batatas fritas* $\vee$ *filé à parmegiana*) não é considerada forte e não será aceita como válida: *batatas fritas* e *filé à parmegiana*, como visto, aparecem em metade dos pedidos dos restaurantes; entretanto, considerando todos os pedidos que envolvem *batatas fritas*, apenas 71% deles também envolvem *filé à parmegiana*. Embora o suporte da regra seja suficiente para ser considerada frequente no domínio em análise, quando se analisa um subconjunto de pedidos em que *batatas fritas* são pedidas com certeza, há uma frequência considerada insuficiente da observação do prato *filé à parmegiana*. Isso quer dizer que, com relevante frequência, *batatas fritas* aparecem também acompanhando outros pratos ou sozinhas.

Como consequência da análise dessas regras, o gerente do restaurante pode decidir criar um prato combinado de filé à parmegiana com batatas fritas que custe um pouco mais caro que o filé à parmegiana sozinho, mas que seja mais barato que solicitar o filé à parmegiana e batatas fritas separadamente. O gerente pode decidir ainda pela retirada do prato filé à parmegiana (sozinho) de seu cardápio. A venda do combinado pode, potencialmente, aumentar. Contudo, não é vantagem para ele retirar o prato batatas fritas (sozinho), haja vista que este é solicitado, com certa frequência, isoladamente ou com outros pratos.

Outros tipos de regras de associação podem ainda ser descobertas, sendo necessário para tal realizar a adequação dos algoritmos de geração e das medidas de avaliação. Alguns exemplos de regras de associação estabelecidas em formatos diferentes são (baseados em Han, Kamber e Pei (2011)):

- Regras de associação de múltiplos níveis de abstração: essas regras são obtidas a partir da estruturação dos itens em uma hierarquia (Figura 5-4). As regras de associação de múltiplos níveis podem envolver itens de diferentes níveis na hierarquia, por exemplo, *aperitivos frios* $\vee$ *cervejas*.

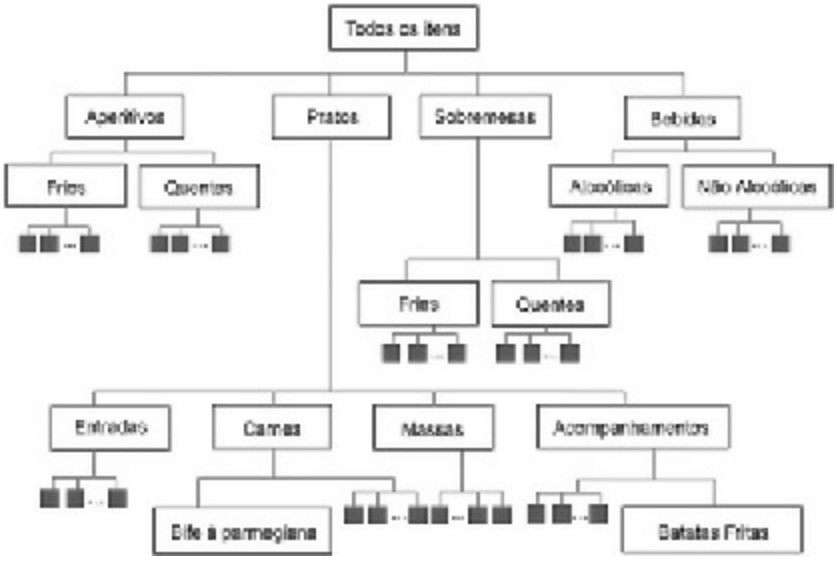


Figura 5-4: Exemplo de itens para o domínio de um restaurante, organizados em múltiplos níveis de abstração

- Regras de associação multidimensionais: envolvem itens de diferentes dimensões em suas premissas e/ou conclusões. Por exemplo, a regra *{grupo de amigos, bebidas não alcoólicas}* $\vee$ *{pagamento com vale-refeição}* é composta por três dimensões diferentes: perfil de cliente, pedido, forma de pagamento. Para que regras desse tipo sejam derivadas, a base de dados transacional precisa também ser multidimensional (veja o conceito de dimensões no Capítulo 1).
- Regras de associação quantitativas: trata-se de regras que envolvem itens quantitativos discretizados (Capítulo 2) antes de serem submetidos aos algoritmos de descoberta de regras de associação ou durante o processo de descoberta. Um exemplo deste tipo de regra é: {*clientes de "35 a 45 anos"*} $\vee$ {*vinhos do "tipo reserva"*}. Note que essa regra é também multidimensional.
- Regras de correlação: são regras de associação avaliadas não somente por meio das medidas de suporte e confiança, mas também pela correlação existente entre as ocorrências dos *itemsets* da premissa e da conclusão. Um exemplo de regras de correlação é: {*massas}* $\vee$ *{batatas fritas} (-)*, em que *(-)* significa uma correlação negativa. A interpretação

dessa regra deve ser: o pedido de *massas* como prato principal inibe o pedido de *batatas fritas*; ou, quando o pedido de *massas* como prato principal é frequente, o pedido de *batatas fritas* como acompanhamento diminui. Importante destacar que, para a correlação ser significativa, a regra precisa ser válida em relação às medidas de suporte e confiança.

A análise da correlação entre as ocorrências dos *itemsets* da premissa e da conclusão pode fornecer informações muito importantes, inclusive mostrando que, em alguns casos, o uso apenas das medidas de suporte e confiança pode levar a interpretações equivocadas sobre a validade de uma regra.

Considere a situação da venda de “frios” e “vinhos” sob as seguintes condições:[2] o suporte do 1-*itemset* {*frios*} é 0,6, indicando que, em 60% dos pedidos, pratos do tipo “frios” são solicitados; o suporte do 1-*itemset* {*vinhos*} é 0,8, indicando que os “vinhos” são muito requisitados no restaurante (em 80% dos pedidos, eles estão presentes); e o suporte do 2-*itemset* {*frios, vinhos*} é 0,45. Sendo assim, uma regra do tipo *frios* $\vee$ *vinhos* teria o suporte de 0,45 e a confiança de 0,75 (0,45/0,6). Isso indica que, em 75% das vezes que frios são solicitados, vinhos também são. A regra *vinhos* $\vee$ *frios* teria também o suporte de 0,45 e a confiança de 0,56 (0,45/0,8), significando que, em 56% dos pedidos em que se solicita vinhos, frios também são solicitados. Se os valores mínimos de suporte e confiança forem estabelecidos como 0,4 e 0,5, respectivamente, essas regras seriam consideradas interessantes e poderiam suportar decisões referentes a, por exemplo, promoções de venda conjunta dos dois itens. Ainda que o limite mínimo da confiança fosse um pouco mais restrito, 0,7, por exemplo, a primeira regra analisada ainda seria considerada válida. No entanto, note que pedidos incluindo vinhos (sem necessariamente incluir frios) são mais frequentes (80%) que os pedidos que incluem vinho e frios (45%), e uma relação parecida vale também para a venda isolada do item frios (60%). Isso indica que há, na realidade, uma inibição da venda dos itens quando

considerados juntos, e tomar a decisão por estimular a venda conjunta pode levar a uma redução de lucros.

Aplicar uma medida de correlação para os itens pode ajudar a melhorar a análise das regras. Como exemplo, considere a medida de correlação *lift*[3] = confiança$_{regra}$/suporte$_{conclusão}$, sendo que *lift > 1* indica correlação positiva, *lift = 1* indica descorrelação, e *lift < 1* indica correlação negativa. A medida de *lift* para a regra *frios* $\vee$ *vinhos* é 0,93, (0,75/0,8) indicando, de fato, uma correlação negativa.

5.1.4. Algoritmo Apriori

O Apriori é um algoritmo clássico usado em mineração de dados para resolução da tarefa de descoberta de regras de associação. Esse algoritmo é composto por duas fases. Na primeira (Algoritmo 5-1), é verificada a quantidade de vezes que cada item (ou *1-itemset*) ocorre na base de dados transacional (*TID*), ou seja, o suporte de cada item é calculado. Os itens com suporte maior que o mínimo estabelecido pelo usuário são selecionados e combinados, de forma a compor *2-itemsets*, e o suporte desses novos *itemsets* é calculado. Aqueles de suporte maior ou igual ao suporte mínimo são então selecionados. Na sequência, os *itemsets* selecionados (*2-itemsets*) são novamente combinados, formando os *3-itemsets*, e todo o processo é repetido até que, em uma iteração, não seja produzido nenhum *itemset* frequente. A cada iteração *k* desse processo, uma lista de *itemsets* frequentes (L_k) é produzida. Ao final, as listas $L_{k<>1}$[4] são concatenadas na lista *L* de *itemsets* frequentes.

Algoritmo 5-1: Primeira fase do algoritmo Apriori : identificação dos *itemsets* frequentes

Parâmetros de entrada:
- ***Tid***: um conjunto de transações;
- s_{min} : suporte mínimo;
- *k*: contador inicializado em 1;

Parâmetro de saída:

- ***L***: lista de *itemsets* frequentes.

Passo 1: calcule o suporte de cada *1-itemset* de ***Tid***;
Passo 2: selecione os *1-itemsets* com suporte maior ou igual a $\boldsymbol{s}_{\min}$ e os insira em L_k;
Passo 3: **enquanto** $L_{\boldsymbol{k}} <> \varnothing$ **faça**
Passo 3.1: combine os *itemsets* de L_k gerando o conjunto de $(k+1)$-*itemsets* candidatos;
Passo 3.2: calcule o suporte de cada novo *itemset*;
Passo 3.3: k++;
Passo 3.4: selecione os *itemsets* com suporte maior ou igual a $s_{\min}$ e os insira em L_k;
Passo 3.5: $\boldsymbol{L} \cup L_k$;

Há uma maneira eficiente para executar a combinação dos *itemsets* de uma lista, que se baseia na combinação dos itens de cada dois *itemsets* da lista, seguindo uma organização ordenada. Por exemplo, supondo a seguinte lista com todos os *2-itemsets* frequentes $\{\{i_1, i_2\}, \{i_1, i_4\}, \{i_2, i_3\}, \{i_2, i_4\}, \{i_2, i_5\}, \{i_3, i_5\}\}$. A seguinte lista com *3-itemsets* deve ser resultante $\{\{i_1, i_2, i_4\}, \{i_2, i_3, i_5\}\}$. Note que o 3-*itemset* $\{i_1, i_2, i_3\}$, por exemplo, não precisa ser gerado, pois o 2-*itemset* $\{i_1, i_3\}$ não é frequente, e, segundo a propriedade Apriori, qualquer super *itemset* que o contenha não será frequente.

Como exemplo da execução da primeira fase do algoritmo Apriori, considere a base de dados transacional *Tid* da Figura 5-1, um s_{mim}= 40% e os resultados listados na Tabela 5-2.

Tabela 5-2: Resultados obtidos na execução da primeira fase do algoritmo Apriori : cálculo do suporte dos *1-itemsets*

1-itemset (candidatos)	**Subconjunto de transações nas quais o *itemset* ocorre**	**Cálculo do $\text{suporte}_{\text{item}}$: *cont*(T) = 7**	**Suporte (%)**
i_1	$\{T_1, T_2, T_3, T_4, T_6, T_7\}$	suporte_{i1} = 6/7= 0,86	86%
i_2	$\{T_2, T_6, T_7\}$	suporte_{i2} = 3/7= 0,43	43%
i_3	$\{T_1, T_2, T_4, T_5, T_7\}$	suporte_{i3} = 5/7= 0,71	71%
i_4	$\{T_4, T_7\}$	suporte_{i4} = 2/7= 0,28	28%

Inicialmente, na execução do *Passo 1*, o suporte dos *1-itemsets* foram calculados. A lista de *1-itemsets* frequentes selecionados no *Passo 2*, convenientemente denominada de L_1, é: L_1={ $\{i_1\}$, $\{i_2\}$, $\{i_3\}$}.

Com os *itemsets* presentes na lista L_k, com k = 1, faz-se as combinações (*Passo 3.1*) para criar *itemsets* de tamanho maior (os *itemsets* candidatos), calcula-se o suporte de cada um e seleciona-se os *itemsets* frequentes (*Passos 3.2* a *3.4*). Os resultados obtidos com a execução desses passos são mostrados na Tabela 5-3. Nessa iteração do *Passo 3*, a lista L_2 = { $\{i_1, i_2\}$, $\{i_1, i_3\}$ } é criada (*Passo 3.4*) e concatenada à lista ***L*** (*Passo 3.5*). Na segunda iteração do *Passo 3*, a lista L_3 é vazia.

Tabela 5-3: Resultados obtidos na execução da primeira fase do algoritmo Apriori: cálculo do suporte dos *2-itemsets* e *3-itemsets*

2-itemset **(candidatos)**	**Subconjunto de transações nas quais o *itemset* ocorre**	**Cálculo do suporte$_{itemset}$: *cont*(T) = 7**	**Suporte (%)**
$\{i_1,i_2\}$	$\{T_2,T_6,T_7\}$	suporte$_{(\{i1,i2\})}$ = 3/7 = 0,43	43%
$\{i_1,i_3\}$	$\{T_1,T_2,T_4,T_7\}$	suporte$_{(\{i1,i3\})}$ = 4/7 = 0,57	57%
$\{i_2,i_3\}$	$\{T_2,T_7\}$	suporte$_{(\{i2,i3\})}$ = 2/7 = 0,28	28%
3-itemset **(candidatos)**	**Subconjunto de transações nas quais o *itemset* ocorre**	**Cálculo do suporte$_{itemset}$: *cont*(T) = 7**	**Suporte (%)**
$\{i_1,i_2,i_3\}$	$\{T_2,T_7\}$	suporte$_{(\{i1,i2,i3\})}$ = 2/7 = 0,28	28%

Note que o subconjunto de transações no qual os itens do $k_{ésimo}$-*itemset* ocorre é a interseção dos subconjuntos de transações em que ocorrem os itens dos *itemsets* que o formaram.

Como não há mais itens a combinar, a lista retornada pelo algoritmo é: L = { $\{i_1, i_2\}$, $\{i_1, i_3\}$ }. O resultado esperado, a partir da execução do algoritmo Apriori, no entanto, é um conjunto de regras do tipo SE-ENTÃO. Para chegar a esse resultado, a segunda fase do algoritmo precisa ser executada.

As regras geradas a partir do algoritmo Apriori devem satisfazer a dois requisitos principais: devem ocorrer com uma frequência mínima (suporte) e devem ser confiáveis (confiança); ou seja, as regras devem ser fortes.

A geração das regras de associação, feita por meio da execução da segunda fase do algoritmo, está descrita no Algoritmo 5-2, que recebe a lista de *itemsets* frequentes ***L*** e, a partir de cada *itemset* na lista, gera combinações de itens para formar premissas e conclusões de regras. Regras com confiança maior que a confiança mínima estabelecida pelo usuário são inseridas no conjunto de regras de associação a ser fornecido pelo algoritmo. Note que a satisfação da frequência mínima (suporte mínimo) para as regras vem da propriedade Apriori, pois, se uma regra for formada por *itemsets* frequentes, ela será frequente, ou seja, ocorrerá em uma quantidade mínima de transações seguindo o suporte mínimo estabelecido.

Algoritmo 5-2: Segunda fase do algoritmo Apriori : geração das regras de associação

Parâmetros de entrada:
- ***L***: lista de *itemsets* frequentes;
- c_{min}: confiança mínima;

Parâmetro de saída:
- ***C***: conjunto de regras de associação válidas (fortes);

Passo 1: **para** cada *itemset* frequente presente na lista ***L*** **faça**
Passo 1.1: gere todas as regras de associação possíveis ($A \quad B$) por meio da combinação de subconjuntos não vazios de itens do *itemset*, lembrando que $A \cap B = \varnothing$;
Passo 2: **para** cada regra gerada **faça**
*Passo 2.1:*calcule a confiança da regra;
Passo 2.2: se a confiança da regra for maior ou igual a c_{mim}, insira-a no conjunto ***C***.

Como exemplo ilustrativo da execução da segunda fase do algoritmo Apriori, considere a base de dados transacional *Tid* da Figura 5-1, um c_{mim} = 70%, e a lista ***L*** resultante da execução da primeira fase do algoritmo. Dado que há dois *itemsets* na lista *L*, o *Passo 1* do Algoritmo 5-2 é executado duas vezes, e as regras geradas são: $i_1 i_2$, $i_2\ i_1$, $i_1 i_3$ e $i_3 i_1$. O cálculo da medida de confiança de cada regra, assim como a decisão sobre inseri-la ou não no

conjunto *C*, são executados no *Passo 2* desse algoritmo. O resultado desse passo está listado na Tabela 5-4. O conjunto *C* será composto por duas regras: $i_2 i_1$ e $i_3 i_1$.

Tabela 5-4: Resultados obtidos na execução da segunda fase do algoritmo Apriori: geração das regras de associação

Premissa (A)	**Conclusão (B)**	**SUPORTE$_{regra}$**	**SUPORTE$_{premissa}$**	**CONFIANÇA$_{regra}$**	**Inserção no conjunto *C***
i1	i_2	0,43	0,86	0,50	Não
i2	i_1	0,43	0,43	1	Sim
i1	i_3	0,57	0,86	0,66	Não
i3	i_1	0,57	0,71	0,80	Sim

Desse resultado, a seguinte interpretação é gerada: a descoberta de conhecimento em questão é desenvolvida em uma base de dados transacionais com sete transações; nesse universo de transações, duas regras são fortes:

- $i_2 i_1$ com suporte de 43% e confiança de 100%, significando que i_1 está presente em todo o universo de transações em que i_2 aparece.
- $i_3 i_1$ com suporte de 57% (mais frequente que a regra anterior) e confiança de 80%, significando que i_1 aparece em 80% das transações em que i_3 aparece.

O exemplo anterior não apresentou um *itemset* frequente de tamanho > 2; contudo, é importante observar como as regras seriam formadas caso isso ocorresse. Considerando um *3-itemset* frequente hipotético $\{i_2, i_3, i_4\}$, as seguintes regras de associação deveriam ser geradas e analisadas quanto à

medida de confiança: $\{i_2, i_3\} \quad i_4, i_4 \{ \quad i_2, i_3\}, \{i_3, i_4\} \quad i_2, i_2 \{ \quad i_3, i_4\}, \{i_2, i_4\} \quad i_3$ e $i_3 \{ \quad i_2, i_4\}$.

5.1.5. Algoritmo FP-*Growth*

A geração de regras de associação no algoritmo Apriori depende de um procedimento que, em sua implementação básica, analisa a totalidade da base de dados transacional várias vezes, a fim de encontrar os *itemsets* frequentes. Portanto, esse processo de encontrar os *itemsets* frequentes é computacionalmente caro, sobretudo quando o conjunto total de transações envolve um número alto de itens e os *itemsets* frequentes de tamanhos pequenos são numerosos.

Como alternativa à estratégia executada no algoritmo Apriori, o algoritmo FP-*Growth* (ou *frequent-pattern growth*) permite que os *itemsets* frequentes sejam descobertos sem a necessidade da criação de listas de *itemsets* candidatos. Isso é feito a partir de um processo no qual a base de dados transacional é consultada uma vez para cálculo da frequência de cada *1-itemset*. Os *1-itemsets* que aparecem poucas vezes (frequência menor que o f_{min}) são removidos do processo, e os demais são ordenados de forma decrescente em uma lista (L_{ord}), seguindo o valor das respectivas frequências. Os *1-itemsets* pertencentes a L_{ord} são organizados em uma estrutura hierárquica por meio da qual uma série de procedimentos são realizados a partir de apenas mais uma leitura da base transacional. O algoritmo FP-*Growth* está descrito no Algoritmo 5-3. Embora a saída do FP-*Growth* seja comumente denominada “lista de padrões frequentes”, neste texto, por conveniência, a nomenclatura “lista de *itemsets* frequentes” será adotada.

Algoritmo 5-3: Algoritmo FP-*Growth*: geração da lista de *itemsets* frequentes

Parâmetros de entrada:
- ***Tid***: um conjunto de transações;
- f_{min}: suporte mínimo;

Parâmetro de saída:
- ***L***: lista de *itemsets* frequentes;

Passo 1: calcule a frequência de cada *1-itemset* de ***Tid***;
Passo 2: selecione os *1-itemsets* com frequência maior ou igual a f_{min};
Passo 3: organize os *1-itemsets* selecionados na lista L_{ord}, em uma ordem decrescente de frequências;
Passo 4: inicialize uma estrutura hierárquica – uma árvore com raiz *A*;
Passo 5: **para** cada transação *t* de ***Tid*** **faça**
Passo 5.1: raiz atual = A;
Passo 5.2: **para** cada item *i* da lista L_{ord} presente na transação *t*, seguindo a ordem da lista **faça**
Passo 5.2.1: **se** o item *i* está presente como nó filho da *raiz atual*
então
Passo 5.2.1.1: visite o nó filho;
Passo 5.2.1.2: incremente o contador do rótulo do nó filho;
Passo 5.2.1.3: raiz atual = nó filho;
senão
Passo 5.2.1.4: crie um novo ramo na árvore com um nó para o item *i*;
Passo 5.2.1.5: inicie seu contador em 1;
Passo 5.2.1.6: raiz atual = nó do novo ramo;
Passo 6: **para** cada item *i* da lista L_{ord}, em ordem inversa **faça**
Passo 6.1: encontre os nós do item *i* na árvore e **para** cada nó *j* **faça**
Passo 6.1.1: extraia os ramos da árvore que precedem o nó *j; % caminhos que finalizam no nó j*
Passo 6.1.2: associe aos nós do ramo a frequência associada ao nó *j*;
Passo 6.1.3: combine os ramos associados ao item *i* somando as frequências associadas aos nós, criando subárvores associadas ao item *i*;
Passo 6.1.4: exclua das subárvores os nós com frequência associada menor que f_{min};
Passo 6.1.5: combine o item *i* com cada nó das subárvores associadas a ele criando os *itemsets;*
Passo 6.1.6: insira os *itemsets* criados na lista ***L***.

Como exemplo da execução do algoritmo FP-*Growth*, considere a base de dados transacional *TID* da Figura 5-1 e um f_{min} = 3. No *Passo 1*, todas as transações são visitadas, e as frequências dos *1-itemsets*, calculadas. *1-itemsets* com frequência menor que 3 são descartados (*Passo 2*), e os remanescentes, ordenados na lista L_{ord} (*Passo 3*). O resultado desse passo do algoritmo é: L_{ord} = { {i_1:6},{i_3:5},{i_2:3} }. No *Passo 4*, uma estrutura de dados hierárquica deve ser criada. Essa estrutura será instanciada na execução do *Passo 5*.

No *Passo 5*, cada uma das transações da base de dados transacional é novamente consultada. Na primeira iteração do laço no *Passo* 5, não há

nenhum item na árvore. A análise da transação T_1, que possui os itens i_1 e i_3, resulta na estrutura hierárquica ilustrada na Figura 5-5(a). Para essa transação, foi criado um novo ramo na árvore, e os nós desse ramo foram rotulados com o identificador de cada item da transação e um contador de frequência inicializado com 1. É importante observar que a ordem dos itens no ramo da árvore segue a ordem em que tais itens aparecem na L_{ord}. A próxima transação a ser analisada (T_2) possui três itens i_1, i_2 e i_3, e a ordem de tais itens em L_{ord} é i_1, i_3 e i_2. O item i_1 já está presente na árvore, então ele terá seu contador incrementado, e o nó atual (no caso o nó de i_1) passa a ser considerado a raiz da estrutura processada na sequência. Sendo assim, ao final da análise de todos os itens da transação T_2, a árvore estará como ilustrado na Figura 5-5(b). As árvores resultantes do processamento de cada uma das sete transações da base *Tid* são mostradas na Figura 5-5. Importante notar que, na análise das transações T_3 e T_7, um dos itens nela presente, o i_4, não é introduzido na árvore, pois não faz parte da L_{ord}.

Uma vez construída a árvore, é necessário analisá-la para descobrir todos os *itemsets* frequentes. Essa análise (*Passo 6*) constitui-se pela especificação de caminhos da árvore que finalizam em cada um dos *1-itemsets* presentes na L_{ord}. A segunda coluna da Tabela 5-5 lista os caminhos encontrados acompanhados de suas frequências (*Passos 6.1.1* e *6.1.2*). Então, nos *Passos 6.1.3 e 6.1.4*, são criadas as subárvores de nós frequentes para os itens da L_{ord}. Note nos resultados obtidos da execução desse passo (Tabela 5-5 – terceira coluna) que o item i_3, na subárvore associada a i_2, tem frequência 2 ($< f_{min}$) e, por isso, não aparece no resultado final. Finalmente, os *itemsets* frequentes são gerados no *Passo 6.1.5* por meio da combinação dos *1-itemsets* e os respectivos itens frequentes presentes nas suas subárvores (Tabela 5-5– quarta coluna).

Tabela 5-5: Análise da árvore (*Passo 6* do Algoritmo 5-3)

1-*itemset* em	**Ramos (caminhos e frequências**	**Subárvores de nós**	***Itemsets***

L_{ord}	associadas)	frequentes	frequentes
i_2	{ {i_1, i_3 : 2},{ i_1: 1} }	{i_1: 3}	{i_1, i_2 }
i_3	{ {i_1 : 4} , {} }	{i_1: 4}	{i_1, i_3 }
i_1	{ {} }	{}	

A partir dos *itemsets* frequentes encontrados, regras de associação devem ser geradas e avaliadas quanto à sua medida de confiança, como já explicado para a segunda fase do algoritmo Apriori. Assim, as seguintes regras são criadas: $i_1\, i_2$, $i_2 i_1$, $i_1 i_3$ e $i_3 i_{1;}$ e os cálculos mostrados na Tabela 5-4 se aplicam.

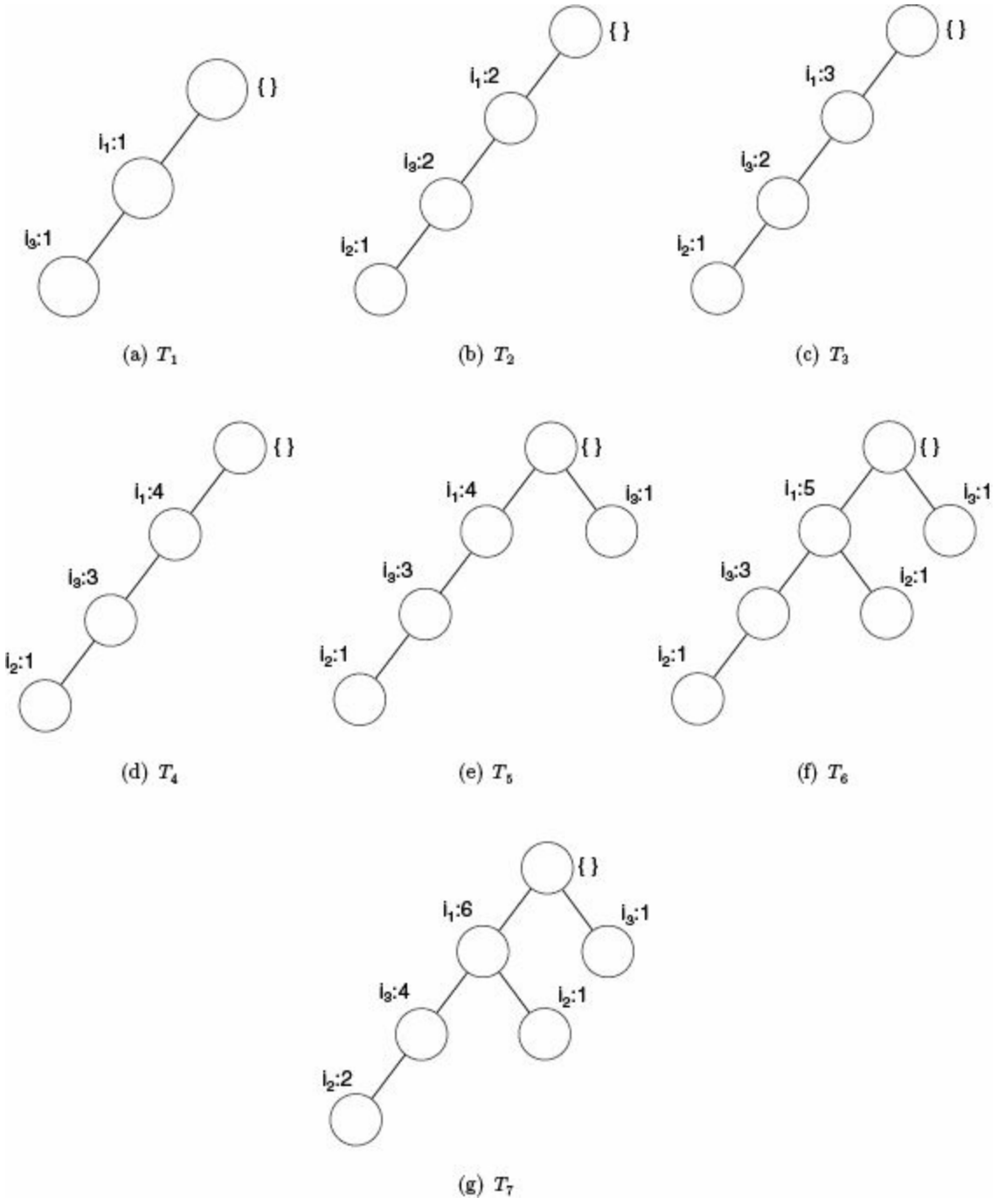


Figura 5-5: Árvores criadas na execução do *Passo 5* do Algoritmo 5-3

5.2. Exemplo didático para descoberta de regras de associação

Esta seção apresenta um caso prático para reforçar os conceitos apresentados neste capítulo e também o entendimento do processo executado pelo algoritmo Apriori. O mesmo caso pode ser executado sob o processo implementado pelo algoritmo FP-*Growth*, e essa execução é deixada como exercício para o leitor. Considere a base de dados transacional hipotética chamada *Registro de Pedidos*, apresentada na Tabela 5-6. Nela, estão representados 10 pedidos de clientes, os quais englobam apenas algumas opções de pratos do restaurante. O problema de mineração relacionado com esse caso é descobrir regras de associação que indiquem quais itens aparecem frequentemente juntos nos pedidos dos clientes e qual é a relação entre eles.

Tabela 5-6: Base de dados transacional *REGISTRO DE PEDIDOS*

	feijão	***salada***	***filé à parmegiana***	***batatas fritas***	***arroz***
T1	0	1	0	0	1
T2	0	0	0	1	0
T3	0	0	1	1	0
T4	0	0	1	0	1
T5	0	0	1	1	0
T6	1	0	1	1	1
T7	0	0	0	1	0
T8	1	0	1	1	1
T9	0	0	0	0	1
T10	0	0	1	1	0

Seguindo o processo executado pelo algoritmo Apriori, a primeira ação é encontrar a lista de itens frequentes na base de dados transacional, *Registro de Pedidos*. Para isso, considere o valor de 40% para o suporte mínimo. Então, os suportes para cada um dos itens (os *1-itemsets*) são calculados, e o resultado desse passo é mostrado na Tabela 5-7.

Tabela 5-7: Resultados obtidos para cálculo do suporte de *1-itemsets* – primeira fase do algoritmo Apriori

1-*itemset* (candidatos)	**Subconjunto de transações nas quais e o *itemset* ocorre**	**Cálculo do suporte$_{item}$: *cont*(T) = 10**	**Suporte (%)**
{*feijão*}	{T_6, T_8}	suporte$_{feijão}$ = 2/10 = 0,2	20%
{*salada*}	{T_1}	suporte$_{salada}$ = 1/10 = 0,1	10%
{*filé à parmegiana*}	{T_3, T_4, T_5, T_6, T_8, T_{10}}	suporte$_{parmegiana}$ = 6/10 = 0,6	60%
{*batatas fritas*}	{T_2, T_3, T_5, T_6, T_7, T_8, T_{10}}	suporte$_{fritas}$ = 7/10 = 0,7	70%
{*arroz*}	{ T_1, T_4, T_6, T_8, T_9}	suporte$_{arroz}$ = 5/10 = 0,5	50%

Diante dos resultados obtidos no cálculo do suporte dos *1-itemsets*, a lista L_1 deve ser composta por: {{*filé à parmegiana*}, {*batatas fritas*}, {*arroz*}}. As próximas ações envolvem combinar esses *itemsets* para encontrar os *2-itemsets* frequentes (L_2), e assim por diante, para os *itemsets* de tamanhos maiores. O resultado referente à análise dos *2-itemsets* é mostrado na Tabela 5-8.

Tabela 5-8: Resultados obtidos para cálculo do suporte de *2-itemsets* – primeira fase do algoritmo Apriori

***2-itemset* (candidatos)**	**Subconjunto de transações nas quais o *itemset* ocorre**	**Cálculo do suporte$_{itemset}$: *cont*(T) = 10**	**Suporte (%)**
{*filé* à*parmegiana, batatas fritas*}	{T_3, T_5, T_6, T_8, T_{10}}	suporte$_{(parmegiana,fritas)}$ = 5/10 = 0,5	50%
{*filé* à*parmegiana, arroz*}	{T_4, T_6, T_8}	suporte$_{(parmegiana,arroz)}$ = 3/10 = 0,3	30%

{*batatas fritas, arroz*}	{ T_6, T_8}	suporte$_{(frita,arroz)}$ = 2/10 = 0,2	20%

Com os resultados obtidos, chega-se à lista L_2 = {{*filé à parmegiana*, *batatas fritas*}}. Note que não há itens a serem combinados na L_2, e, portanto, a L_3 é vazia, e o processo de encontrar a lista de itens frequentes resulta na lista ***L*** = {{*filé à parmegiana*, *batatas fritas*}}.

A próxima ação é gerar e avaliar as regras de associação a partir da lista ***L*** de *itemsets* frequentes. Para isso, considere 80% o limite mínimo para a medida de confiança. As regras geradas, bem como suas avaliações, são mostradas na Tabela 5-9. Note que apenas a regra *filé* à*parmegiana batatas fritas* é forte e pode fazer parte do resultado do algoritmo Apriori.

Tabela 5-9: Resultados obtidos na geração das regras de associação – segunda fase do algoritmo Apriori

Premissa	**Conclusão**	***SUPORTE*$_{regra}$**	***SUPORTE*$_{premissa}$**	***CONFIANÇA*$_{regra}$**	**Regra aceita**
filé à parmegiana	*batatas fritas*	0,50	0,60	0,83	Sim
batatas fritas	*filé à parmegiana*	0,50	0,70	0,71	Não

5.3. Exemplo prático para descoberta de regras de associação em R

Esta seção apresenta uma implementação para o algoritmo Apriori em R. Essa implementação ilustra a resolução do problema apresentado na Seção 5. 2. No ambiente R, existe uma função que implementa o algoritmo Apriori, a função `apriori( )`. Ela faz parte do pacote ***arules*** (Hahsler, 2005; Hahsler, 2014). A função `apriori( )` exige que a base de dados esteja em um formato de transações próprio do ambiente R. Caso a base de dados esteja representada no formato matricial (muito comum principalmente em bases de dados disponibilizadas para uso público), similar ao da Figura 5-1(a), será necessário um trabalho de pré-processamento, que também pode ser implementado usando as próprias funções do ambiente R.

Para exemplificar essa necessidade de pré-processamento, considere que a base de dados transacionais esteja disponibilizada como mostra o Quadro 5-1, ou seja, os dados estão dispostos como uma matriz, na qual as transações estão em linhas, e os itens, nas colunas, separados por “;”, e a existência de um item em uma transação é expressa pelo valor “SIM”. Portanto, os dados estão disponíveis em um formato *.csv*, em que a base de dados transacional se encontra, inicialmente, representada como matriz. A escolha de usar a leitura de um arquivo do tipo *.csv* se deve ao fato de ser comum encontrar bases de dados disponibilizadas em tal formato; além disso, também é comum usar como fonte de dados para resolução da tarefa de descoberta de regras de associação dados originalmente preparados para outros contextos de análise.

Quadro 5-1: Base de dados *REGISTRO DE PEDIDOS* [5]

Base de dados transacional de entrada

```
Feijao;Salada;Parmegiana;Fritas;Arroz
NAO;SIM;NAO;NAO;SIM
NAO;NAO;NAO;SIM;NAO
NAO;NAO;SIM;SIM;NAO
```

```
NAO;NAO;SIM;NAO;SIM
NAO;NAO;SIM;SIM;NAO
SIM;NAO;SIM;SIM;SIM
NAO;NAO;NAO;SIM;NAO
SIM;NAO;SIM;SIM;SIM
NAO;NAO;NAO;NAO;SIM
NAO;NAO;SIM;SIM;NAO
```

A solução a partir de R se iniciacom a leitura da base de dados, ou seja, é preciso importar o conteúdo do arquivo para dentro de variáveis do ambiente R. A importação desta base de dados pode ser feita como explicado no Apêndice, por meio do uso da função `read.table( )`. A instrução completa para a leitura da base *Registro de Pedidos* é:

```
> base_dados <- read.table
("C:\\Users\\LivroDM\\Associacao\\registro_pedidos.csv",
header=TRUE,sep=";")
```

Essa solução de leitura de arquivos gera uma estrutura de dados do tipo *data.frame*, e, então, pode-se proceder com a descoberta das regras. Outro tipo de organização dos dados possível para processamento na função `apriori( )` é o tipo `transactions`, que organiza os dados de forma que cada linha contenha os itens que pertencem à uma transação (Quadro 5-2).

Quadro 5-2: Formato de dados transacionais aceito na função *apriori*( ), considerando as três primeiras transações da base de dados transacional REGISTRO DE PEDIDOS

```
SALADA;ARROZ
FRITAS
PARMEGIANA;FRITAS
...
```

Para obter uma base de dados estruturada como `transactions`, a conversão pode ser feita com o uso da função `as( )` (também do pacote

arules), cujos sintaxe de uso e conjunto de parâmetros são apresentados no Quadro 5-3.

Quadro 5-3: Sintaxe e parâmetros da função AS

Sintaxe para conversão de dados (*data_frame* para *transactions*)
Usando a função `as( )` do pacote *arules*

`base_convertida <- as(base de dados, formato)`

- `base de dados` armazenada em uma variável instanciada por meio da leitura de um arquivo.
- `formato` final da conversão desejada (*transactions* ou *data.frame*).

Assim, no exemplo em desenvolvimento, pode-se realizar a conversão:

```
> install.packages("arules")
> library("arules")
> base_dados_transacional <- as(base_dados, "transactions")

> inspect(base_dados_transacional)
items transactionID
1 {Feijao=NAO,
Salada=SIM,
Parmegiana=NAO,
Fritas=NAO,
Arroz=SIM} 1
2 {Feijao=NAO,
Salada=NAO,
Parmegiana=NAO,
Fritas=SIM,
Arroz=NAO} 2
...
```

Com a base de dados armazenada em uma variável no formato *transactions* (ou mesmo no formato *data.frame*), já é possível usar a função que implementa o algoritmo *Apriori* para a descoberta das regras de associação. Os principais parâmetros que essa função recebe são apresentados no Quadro 5-4.

Quadro 5-4: Sintaxe e parâmetros da função APRIORI

Sintaxe para aplicação da função que implementa o algoritmo Apriori
Usando a função `apriori()` **do pacote** ***arules***

Encontrando as regras de associação

`regras <-apriori(base de dados, parâmetros, itens na transação)`

- `base de dados` é um ***data.frame*** ou um ***transactions*** contendo dados transacionais para associação.
- `parâmetros` permitem uma lista com o número mínimo de itens para associação, valor de suporte mínimo e de confiança mínima.
- `itens na transação` é uma lista que indica o que deve ser analisado pelo algoritmo como um *item*. Podem ser indicadas restrições ao que deve ser considerado premissa da regra (lista `rhs`), ao que deve ser considerado conclusão da regra (lista `lhs`), valores que não devem ser considerados (lista `none`) ou valores que devem ser considerados (lista `itens`).

A função retorna uma objeto contendo informações sobre as associações geradas.

Como a base de dados transacional em uso apresenta valores booleanos expressos como *strings* {SIM,NAO}, a indicação do valor que informa que um item foi consumido na transação precisa ser fornecida à função; caso contrário, como a função não é capaz de automaticamente decidir sobre a diferença semântica das palavras SIM e NAO, ela executaria um processamento que procuraria padrões frequentes considerando ambos os valores. O parâmetro `itens na transação` resolve esse problema, pois tem o objetivo de restringir o valor analisado, ou seja, a função `apriori( )`encontrará regras de associação a partir dos itens que tenham apenas os valores indicados pelo usuário. Se esse parâmetro não for informado explicitamente na chamada da função, a lista de opções se torna irrestrita.

Na solução do exemplo didático (Seção 5.2), os valores mínimos para suporte e confiança foram 40% e 80%, respectivamente. E, para que seja possível reproduzir o mesmo experimento, a lista mínima de itens foi parametrizada com 2. Além disso, é interessante analisar apenas a presença dos itens nas transações. Seguindo essas diretrizes, o resultado completo para aplicação da função `apriori( )` é:

```
> regras<- apriori(data = base_dados_transacional, parameter =
list(minlen=2, supp=0.4, conf=0.8), appearance =
```

```
list(none=c("Feijao=NAO","Salada=NAO",
"Parmegiana=NAO","Fritas=NAO","Arroz=NAO")))
```

As regras geradas podem ser visualizadas a partir do seguinte comando em R:

```
> inspect(regras)
```

E o resultado é apresentado como:

```
lhs rhs support confidence lift
1{Parmegiana=SIM} => {Fritas=SIM} 0.5 0.8333333 1.190476
```

sendo que o `lhs` é o antecedente (premissa), e `rhs,` o consequente (conclusão) da regra de associação gerada.

A descoberta de regras de associação implementada na função `apriori( )` está baseada em um processo de contagem de frequências mínimas, que não diferencia, originalmente, se o padrão positivo é SIM, NÃO, FALSO, VERDADEIRO etc., de forma que o importante é a ocorrência frequente de pares <*item, valor*>. Isso significa dizer que a análise da base de dados transacional *Registro de Pedidos* poderia envolver tanto produtos vendidos em um pedido quanto produtos não vendidos em um pedido, analisando, por exemplo, com que frequência uma refeição que envolve o item *filé à parmegiana* não envolve o item *salada*.

Como não poderia ser diferente, o leitor já deve ter percebido que a implementação do algoritmo Apriori em R está relacionada com uma série de parâmetros, e que os valores estabelecidos em cada um deles devem, certamente, influenciar os resultados obtidos. Por isso, é interessante desenvolver análises extras para explorar possibilidades e verificar os efeitos de mudanças nos valores determinados para os parâmetros da função. A discussão então se inicia com um experimento no qual não se restringem quais itens devem ou não ser analisados no cômputo das frequências

mínimas. Portanto, a função `apriori( )` é executada sem o parâmetro *appearance*:

```
> regras <- apriori(data= base_dados_transacional,
parameter=list(minlen=2,supp=0.4, conf=0.8)),
```

e a visualização das 24 regras geradas é como segue:

```
> inspect(regras)
lhs rhs support confidence lift
1 {Parmegiana=NAO} => {Feijao=NAO} 0.4 1.0000000 1.2500000
2 {Arroz=SIM} => {Salada=NAO} 0.4 0.8000000 0.8888889
3 {Arroz=NAO} => {Fritas=SIM} 0.5 1.0000000 1.4285714
4 {Arroz=NAO} => {Feijao=NAO} 0.5 1.0000000 1.2500000
5 {Arroz=NAO} => {Salada=NAO} 0.5 1.0000000 1.1111111
6 {Parmegiana=SIM} => {Fritas=SIM} 0.5 0.8333333 1.1904762
7 {Parmegiana=SIM} => {Salada=NAO} 0.6 1.0000000 1.1111111
8 {Fritas=SIM} => {Salada=NAO} 0.7 1.0000000 1.1111111
9 {Feijao=NAO} => {Salada=NAO} 0.7 0.8750000 0.9722222
10 {Fritas=SIM,
Arroz=NAO} => {Feijao=NAO} 0.5 1.0000000 1.2500000
11 {Feijao=NAO,
Arroz=NAO} => {Fritas=SIM} 0.5 1.0000000 1.4285714
12 {Feijao=NAO,
Fritas=SIM} => {Arroz=NAO} 0.5 1.0000000 2.0000000
13 {Fritas=SIM,
Arroz=NAO} => {Salada=NAO} 0.5 1.0000000 1.1111111
14 {Salada=NAO,
Arroz=NAO} => {Fritas=SIM} 0.5 1.0000000 1.4285714
15 {Feijao=NAO,
Arroz=NAO} => {Salada=NAO} 0.5 1.0000000 1.1111111
16 {Salada=NAO,
Arroz=NAO} => {Feijao=NAO} 0.5 1.0000000 1.2500000
17 {Parmegiana=SIM,
Fritas=SIM} => {Salada=NAO} 0.5 1.0000000 1.1111111
18 {Salada=NAO,
Parmegiana=SIM} => {Fritas=SIM} 0.5 0.8333333 1.1904762
19 {Feijao=NAO,
Parmegiana=SIM} => {Salada=NAO} 0.4 1.0000000 1.1111111
20 {Feijao=NAO,
Fritas=SIM} => {Salada=NAO} 0.5 1.0000000 1.1111111
21 {Feijao=NAO,
Fritas=SIM,
Arroz=NAO} => {Salada=NAO} 0.5 1.0000000 1.1111111
```

```
22 {Salada=NAO,
Fritas=SIM,
Arroz=NAO} => {Feijao=NAO} 0.5 1.0000000 1.2500000
23 {Feijao=NAO,
Salada=NAO,
Arroz=NAO} => {Fritas=SIM} 0.5 1.0000000 1.4285714
24 {Feijao=NAO,
Salada=NAO,
Fritas=SIM} => {Arroz=NAO} 0.5 1.0000000 2.0000000
```

Alternativamente,

```
>inspect(head(regras))

lhs rhs support confidence lift

1 {Parmegiana=NAO} => {Feijao=NAO} 0.4 1.0000000 1.2500000
2 {Arroz=SIM} => {Salada=NAO} 0.4 0.8000000 0.8888889
3 {Arroz=NAO} => {Fritas=SIM} 0.5 1.0000000 1.4285714
4 {Arroz=NAO} => {Feijao=NAO} 0.5 1.0000000 1.2500000
5 {Arroz=NAO} => {Salada=NAO} 0.5 1.0000000 1.1111111
6 {Parmegiana=SIM} => {Fritas=SIM} 0.5 0.8333333 1.1904762
```

A *regra 6* é a gerada no exemplo didático e na primeira execução do Apriori na presente seção. Nessa execução, foram geradas outras regras que consideram também o valor NAO, referente à ausência de um item em um pedido. Na prática, isso pode ser interessante para fomentar uma campanha publicitária ou no direcionamento ao garçom para melhorar o atendimento. Por exemplo, a *regra 2* poderia motivar uma campanha de forma que “quem pedisse arroz ganharia um desconto na salada”, por exemplo. Ou então, uma regra como a de número 3 poderia evidenciar um comportamento assumido na “correção de sugestões presentes em um cardápio”, melhorando a adequabilidade do cardápio atual às preferências gerais dos clientes.

Ainda nesse sentido, é possível parametrizar o que deve ser considerado em cada lado da regra, por exemplo, solicitando que, para a premissa, apenas itens que não entraram em pedidos sejam considerados, e, para o consequente, apenas itens que entraram nos pedidos sejam considerados. Tais possibilidades são deixadas para o leitor explorar em seus experimentos.

Contudo, é preciso chamar a atenção ao fato de que regras que não trazem informação interessante podem ser geradas, e a análise sobre o quão interessante é uma regra é de responsabilidade do usuário, que as interpreta. Por exemplo, a *regra 1* traz um tipo de informação de difícil análise, pois abrange apenas ausências de itens: “quando não se pede parmegiana, não se pede feijão”. Conhecimento baseado no complemento do conjunto que representa um fato é de difícil interpretação e, em muitos casos, inútil.

Outro aspecto que deve ser abordado na experimentação de descobertas de regras de associação é o número de regras que podem ser geradas e como é possível restringir esse conjunto de forma eficiente, sem correr o risco de perder conhecimento interessante. Note que, para uma pequena base de dados transacional, com apenas 10 transações, uma mudança em um dos parâmetros leva a um aumento no número de regras. Extrapolando isso para uma base real, com centenas ou milhares de transações, o aumento de regras geradas por ocasião da alteração de um parâmetro pode ser consideravelmente grande, podendo chegar a ser intratável. Uma maneira de minimizar os efeitos danosos da geração de um conjunto de regras muito grande é usar comandos de ordenação (Quadro 5-5), filtros implementados por meio da quantificação das medidas de avaliação da qualidade das regras (Quadro 5-6) e recursos de visualização gráfica.

Quadro 5-5: Sintaxe e parâmetros da função SORT

Sintaxe para aplicação da função que implementa a ordenação das regras de associação descobertas
Usando a função `sort( )` **do pacote *arules***
Encontrando as regras de associação: `regras_ordenadas <-sort(regras, indicador)` • `regras` é o resultado da função `apriori()`. • `indicador` é a medida de qualidade que deve ser usada para guiar a ordenação das regras. Os valores possíveis são `by="support"`, `by="confidence"` e `by="lift"`. A função retorna uma variável contendo o conjunto ordenado de regras de associação.

Quadro 5-6: Sintaxe e parâmetros da função QUALITY

Sintaxe para aplicação de filtros com base em medidas de qualidade

Usando a função `quality( )` do pacote *arules*

Encontrando as regras de associação:

`sub_regras <-regras_apriori[quality(rules)$indicador condição-valor]`

- `indicador` é a medida de qualidade que deve ser usada como base para o filtro. Os valores possíveis são `SUPPORT`, `CONFIDENCE` e `LIFT`.
- `condição-valor` representa o operador usado no filtro (maior que, menor que etc.) e o valor de comparação.

A função retorna o subconjunto de regras de associação que atendem à condição de filtragem.

O comando de ordenação (apresentado no Quadro 5-5), considerando ordenação por meio do valor de confiança da regra, e a visualização dos resultados produzidos são implementados como:

```
> regras_ordenadas <- sort(regras, by="confidence")
> inspect(head(regras_ordenadas))

lhs rhs support confidence lift
1 {Parmegiana=NAO} => {Feijao=NAO} 0.4 1 1.250000
2 {Arroz=NAO} => {Fritas=SIM} 0.5 1 1.428571
3 {Arroz=NAO} => {Feijao=NAO} 0.5 1 1.250000
4 {Arroz=NAO} => {Salada=NAO} 0.5 1 1.111111
5 {Parmegiana=SIM} => {Salada=NAO} 0.6 1 1.111111
6 {Fritas=SIM} => {Salada=NAO} 0.7 1 1.111111
```

A instrução para aplicar um filtro é:

```
> sub_regras = regras[quality(regras)$confidence > 0.8];
> inspect(head(sub_regras))

lhs rhs support confidence lift
1 {Parmegiana=NAO} => {Feijao=NAO} 0.4 1.0000000 1.250000
2 {Arroz=NAO} => {Fritas=SIM} 0.5 1.0000000 1.428571
3 {Arroz=NAO} => {Feijao=NAO} 0.5 1.0000000 1.250000
4 {Arroz=NAO} => {Salada=NAO} 0.5 1.0000000 1.111111
5 {Parmegiana=SIM} => {Fritas=SIM} 0.5 0.8333333 1.190476
6 {Parmegiana=SIM} => {Salada=NAO} 0.6 1.0000000 1.111111
```

Ainda, o usuário deve experimentar diferentes valores mínimos para suporte e para confiança, de forma que tenha uma visão mais abrangente dos padrões de associação escondidos na base de dados transacional.

Finalmente, há algumas opções de visualização gráfica dos resultados, disponíveis no pacote para visualização de regras ***arulesViz*** (Hahsler e Chelluboina, 2015). A visualização gráfica ajuda na interpretação das regras geradas. O pacote ***arulesViz*** disponibiliza a função `plot( )`,que permite a interpretação de resultados sob diferentes perspectivas. A sintaxe para uso da função `plot( )` é apresentada no Quadro 5-7.

Quadro 5-7: Sintaxe e parâmetros da função PLOT

Sintaxe para geração de visualizações gráficas sobre as regras de associação
Usando a função `plot( )` do pacote *arulesViz*
Plotando visualizações das regras de associação: `plot(regras, método, medida, sombreamento)` • `regras` constitui-se do resultado da função `apriori()`. • `método` é uma *string* com o tipo do gráfico gerado, por exemplo, "`scatterplot`", "`matrix`", "`grouped`" etc. • `medida` é um conjunto de dois parâmetros que será usado para comparação, ou seja, "`suport`", "`confidence`" e "`lift`". • `sombreamento` é a medida de interesse usada para colorir os pontos dos gráficos.

O gráfico do tipo *scatter plot* permite visualizar as regras geradas em termos de seus valores de suporte, confiança e *lift*. Essa visualização fornece algumas pistas para definição de valores a serem aplicados em filtragem de regras. A sintaxe do comando que gera um gráfico *scatter plot* é:

```
> install.packages("arulesViz")
> library("arulesViz")
> plot(regras, method="scatter",measure=c("support","confidence"),
shading="lift");
```

e o resultado gráfico para o conjunto de regras geradas (com parâmetros `minlen=2, supp=0.4, conf=0.8`) é mostrado na Figura 5-6. Nesse caso, cada

regra se torna um ponto em um gráfico “confiança *versus* suporte”, e a tonalidade dos pontos é distribuída em uma escala que representa a variação da medida *lift*.

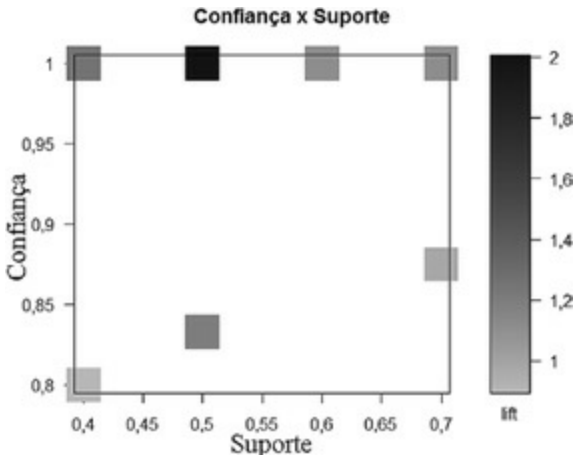


Figura 5-6: Gráfico *scatter plot* gerado na ferramenta R para as regras geradas a partir da base de dados transacional *REGISTRO DE PEDIDOS*. As 24 regras geradas estão plotadas no gráfico; contudo, muitas regras possuem os mesmos valores para suporte e confiança e se sobrepõem na visualização. Nesta função de impressão, as legendas são impressas em inglês e foram traduzidas para esta publicação

O gráfico do tipo matriz permite uma análise visual da quantidade de itens presentes em cada lado das regras. Nesse caso, o parâmetro referente à medida informa qual será a usada para guiar a escala de cores na visualização. Na sintaxe usada em:

```
> plot(regras, method="matrix",measure="confidence"),
```

a medida confiança guia a escala de tonalidades, e o gráfico resultante é mostrado na Figura 5-7. Para o entendimento dessa figura, é importante analisar os índices dos eixos a partir dos valores retornados após execução da chamada da função de plotagem:

```
Itemsets in Antecedent (LHS)
[1]"{Parmegiana=NAO}" "{Arroz=SIM}" "{Arroz=NAO}"
[4]"{Parmegiana=SIM}" "{Fritas=SIM}" "{Feijao=NAO}"
[7]"{Fritas=SIM,Arroz=NAO}""{Feijao=NAO,Arroz=NAO}""{Feijao=NAO,Fr
```

```
[10]"{Salada=NAO,Arroz=NAO}""{Parmegiana=SIM,Fritas=SIM}""{Salada=
[13]"{Feijao=NAO,Parmegiana=SIM}""{Feijao=NAO,Fritas=SIM,Arroz=NAO
Fritas=SIM,Arroz=NAO}"
[16]"{Feijao=NAO,Salada=NAO,Arroz=NAO}""{Feijao=NAO,Salada=NAO,Fri

Itemsets in Consequent (RHS)
[1] "{Feijao=NAO}" "{Salada=NAO}" "{Fritas=SIM}" "{Arroz=NAO}"
```

São 17 *itemsets* diferentes usados nos antecedentes e 4 *itemsets* diferentes usados nos consequentes das regras. No gráfico, cada elemento do eixo LHS relacionado com cada elemento do eixo RHS representa uma regra, e a tonalidade usada para colorir a célula correspondente representa a confiança da regra.

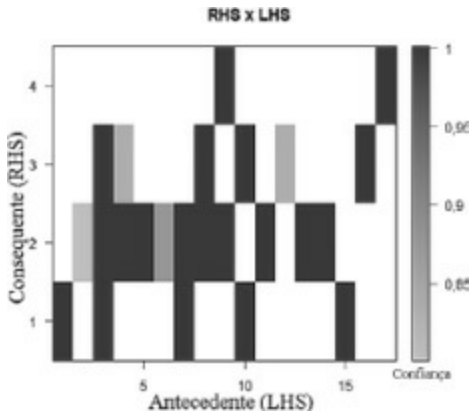


Figura 5-7: Gráfico do tipo matriz para as regras geradas a partir da base de dados transacional *REGISTROS DE PEDIDOS*. Nesta função de impressão, as legendas são impressas em inglês e foram traduzidas para esta publicação

O gráfico do tipo *grouped* propicia uma visualização, em termos de cor e tamanho dos elementos do gráfico (círculos), que combina informações sobre os itens em cada lado das regras e os valores das medidas de avaliação indicadas nos parâmetros da função. A sintaxe

```
> plot(regras, method="grouped", measure=c("support"),
shading="confidence")
```

resulta no gráfico ilustrado na Figura 5-8. Observe no gráfico que é possível analisar cada item que aparece na premissa da regra (`LHS`) em relação aos que

aparecem na conclusão da regra (`RHS`) e verificar o quão forte é cada regra (resultante dessa relação). Por exemplo, a presença do item “fritas” em um pedido ocorre com a ausência do item “salada”, com um suporte alto (círculo grande) e confiança alta (círculo escuro).

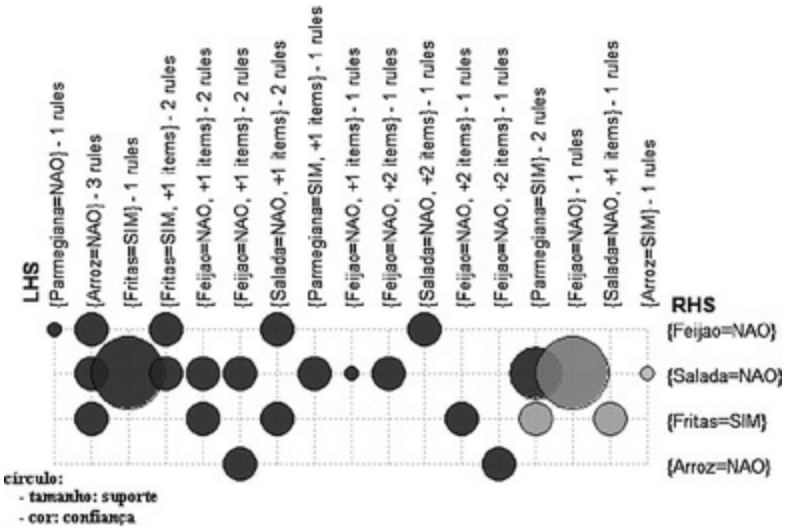


Figura 5-8: Gráfico do tipo *grouped* para as regras geradas a partir da base de dados transacional *REGISTRO DE PEDIDOS*. Valores altos para as medidas são representados por cículos grandes e escuros. Valores baixos são representados por círculos pequenos e claros. Nesta função de impressão, as legendas são em inglês, mas foram traduzidas para exemplificação nesta publicação.

5.4. Leituras adicionais

Estudos sobre descoberta de regras de associação tiveram início nos anos 1990. Já quando os estudos se iniciaram, o interesse pela área foi muito grande, já que se trata realmente de uma possibilidade de descoberta de conhecimento muito interessante e que leva a resultados de interpretação factível para agentes de diferentes níveis e setores organizacionais: desde o setor técnico de informática até os setores gerenciais administrativos. Para conhecer a origem da área de descoberta de regras de associação, é indicada a leitura dos artigos dos autores Rakesh Agrawal, Tomasz Imielinski, Arun Swami, Ramakrishnan Srikant e John C. Shafer. Algumas sugestões mais específicas são: Agrawal, Imielinski e Swami (1993), Agrawal e Srikant (1994), Srikant e Agrawal (1995), Agrawal e Srikant, (1995) e Agrawal e Shafer (1996).

Descoberta de regras de associação pode ser entendida como sinônimo de *Market Basket Analysis* por pesquisadores da área de Administração. Esse ponto de vista é discutido por Aguinis, Forcum e Joo (2013), em um artigo no qual fazem um resumo do assunto sob um ponto de vista um pouco diferente do adotado neste livro. Eles também discutem aspectos referentes a vantagens e desvantagens da análise de dados via tarefa de descoberta de regras de associação e trazem alguns exemplos da aplicação de regras em diferentes contextos.

O tópico referente ao estudo de medidas de avaliação, do quão interessantes e úteis podem ser as regras de associação descobertas no processo de mineração de dados, não se resume apenas a medidas de suporte, confiança e correlação, como discutido neste livro. Muitas outras medidas podem ser usadas. Uma discussão sobre tais medidas é apresentada em Bong *et al*. (2014) e Tan, Steinbach e Kumar (2006). Nesses artigos, os autores apresentam diferentes medidas para avaliação de regras de associação e também discutem formas de escolher quando aplicá-las.

Como já comentado neste capítulo, o algoritmo mais comum para implementação da descoberta de regras de associação é o algoritmo Apriori. No entanto, usá-lo sob uma implementação básica pode ser computacionalmente custoso. Porém, implementações mais sofisticadas podem ser construídas para otimizá-lo, fazendo uso de *hashing* (Han, Kamber e Pei, 2011; Tan, Steinback e Kumar, 2006), paralelismo (Sarawagi, Thomas e Agrawal, 1998), uso do formato vertical para organização e análise da base de dados transacional (Zaki, 2000). Ainda, sugere-se que o leitor se aprofunde na formalização associada à definição da Propriedade Apriori (Han, Kamber e Pei, 2011; Tan, Steinback e Kumar, 2006).

Também sobre estratégias de implementação, alterações básicas podem ser feitas no algoritmo *FP-Growth*. De fato, a implementação apresentada neste capítulo, por questões de simplificação e para entendimento da lógica básica do algoritmo, não inclui, na estrutura de lista final de itens frequentes, a menção às frequências desses itens e tampouco faz uso de uma tabela de ponteiros que melhore a eficiência da implementação do algoritmo. Pequenas alterações na lógica apresentada neste capítulo podem ser feitas para que as frequências possam ser calculadas e também para o uso da tabela de ponteiros. Para entender como isso pode ser feito, sugere-se o estudo dos exemplos fornecidos por Han, Kamber e Pei (2011) e por Witten, Frank e Hall (2011).

Ultimamente, algumas áreas de aplicação de mineração de dados têm se destacado na orientação das atenções dos esforços de pesquisa de desenvolvimento em mineração. A análise de dados não estruturados, em especial no formato textual, é uma das que vêm recebendo atenção especial e na qual se tem conseguido avanços interessantes. Especificamente, considerando a tarefa de descoberta de associações, sugere-se a leitura de dois artigos científicos para iniciar os estudos na área: Lopes *et al*. (2007) e Cherfi, Napoli e Toussaing (2006). Outras áreas de aplicação de descoberta de regras de associação que precisam de um estudo especializado são análise

de dados *streaming*, análise de séries temporais e bioinformática. Para esses exemplos, Han, Kamber e Pei (2011) fornecem uma série de conceitos e algoritmos que formam uma base inicial de estudos bastante abrangente.

Existem alguns conjuntos de dados, reais ou sintéticos, disponíveis publicamente ou sob requisição especial, que podem ser usados como referência para desenvolvimento e teste de algoritmos ou mesmo para a prática dos conceitos referentes à descoberta de regras de associação. O conjunto de dados TA-*Feng Grocery Shopping Datasets* (Brijs *et al*., 1999) contém cerca de 27 mil transações (vendas) ocorridas em uma loja de conveniência durante cinco meses e meio.[6]

5.5. Exercícios

1. Em alguns dos exemplos de mineração discutidos neste capítulo, a base de dados transacional *Registro de Pedidos* foi usada para descoberta de regras de associação, considerando itens solicitados juntos em uma refeição. Essa análise, portanto, se referiu a exemplos positivos (o item FOI solicitado no pedido). Entretanto, existe a possibilidade de analisar os itens não solicitados nos pedidos e, então, implementar uma análise de exemplos negativos. Considerando essa possibilidade e o contexto em que a base de dados transacional *Registro de Pedidos* é construída:

- Construa alguns exemplos de regras de associação contextualizadas no mundo real, que façam referência **apenas** à análise de exemplos negativos (itens não solicitados nos pedidos).
- Apresente a solução completa para a análise de apenas exemplos negativos, seguindo os passos que constituem os algoritmos Apriori e FP-*Growth*, considerando suporte mínimo de 40% e confiança mínima de 80% (lembre-se de que os algoritmos são equivalentes, embora diferentes, ou seja, as mesmas regras devem ser geradas ao final do processo se limiares iguais para as medidas de suporte/frequência e para a medida de confiança forem usados em ambos os algoritmos).
- Discuta a utilidade das regras de associação geradas no item anterior.

2. Na tarefa de descoberta de regras de associação, ainda é possível realizar análises sofisticadas. Um exemplo é considerar, no processo de descoberta, tanto os exemplos positivos quanto os negativos. Por exemplo: pessoas que pedem *filé à parmegiana* não pedem *feijão*. Muitas das ferramentas para automação da tarefa de descoberta de regras de associação implementam essa versão da análise. Discuta razões para que esse tipo de análise seja realizada em contextos reais e exercite a sua compreensão dos algoritmos

Apriori e FP-*Growth*, considerando apenas dois itens, por exemplo, *filé à parmegiana* e *feijão*. Varie os parâmetros de suporte mínimo e confiança mínima e analise como diferentes valores influenciam na resposta dos algoritmos. Lembre que agora é preciso considerar uma representação em que o item seja um par < item, valor >.

3. Você recebeu a base de dados $\mathbf{TID}_{ex}$ (Tabela 5-10) em um projeto de mineração de dados. O objetivo desse projeto é encontrar associações entre quatro disciplinas de um curso de Computação:

- BD – Banco de Dados
- IA – Inteligência Artificial
- MD – Mineração de Dados
- PP – Programação Paralela

Para encontrar as associações, o desempenho de cinco diferentes alunos ($\mathbf{TID}_{ex}$) em cada uma das disciplinas será analisado. Considere 1, se o aluno foi **aprovado**, e 0, se ele foi **reprovado**. Como decisão de projeto, opta-se por associar as disciplinas a partir dos alunos aprovados. Como parâmetros para resolução, considere o limite de 40% o mínimo para o suporte (se o processo do FP-*Growth* for utilizado, a frequência mínima equivalente deve ser aplicada, $f_{min} = 2$) e o limite de 60% o mínimo para a confiança. Aplicando as fases de um dos algoritmos apresentados neste capítulo:

- Mostre os resultados da execução passo a passo do processo de descoberta de regras de associação e apresente as regras obtidas.
- Como você pretende apresentar as regras geradas ao solicitante do projeto de mineração? Quais interpretações são possíveis? Você sugere algum tipo de ação para melhorar o contexto de *disciplinas X desempenho dos alunos*, mediante as regras obtidas?

Tabela 5-10: Base de dados transacional $\mathbf{TID}_{ex}$

$\mathbf{TID}_{ex}$	BD	IA	MD	PP
1	1	0	1	0
2	0	1	1	0
3	1	0	1	1
4	1	0	0	0
5	1	0	0	1

4. [Exercício baseado em discussão apresentada por Han e Kamber (2011)] – Suponha que o Sr. Abdul Samir, gerente de um sofisticado e refinado restaurante de comida árabe e mediterrânea, esteja preocupado com o consumo de pratos árabes e vinhos brancos no restaurante pelo qual é responsável. Então, ele solicitou a você uma análise sobre o comportamento das vendas desses produtos no restaurante, desde que foi inaugurado, há quatro anos. Nesses quatro anos, o restaurante vendeu 2.688 refeições em que *pratos árabes* e *vinhos brancos* foram consumidos. A soma de vendas de refeições no restaurante, durante este mesmo intervalo de tempo, foi 6.720. Os dados que o gerente lhe passou ainda apontam que, em 4.032 pedidos de refeições, um *prato árabe* está presente; e, em 5.040, o *vinho branco* está presente. Diante dos dados que lhe foram fornecidos, você descobriu a seguinte regra:

se o cliente consome um prato árabe, também consome vinho branco

sendo que o suporte da regra foi de ٤٠٪, e a confiança foi de ٦٦٪. A regra foi considerada forte porque o suporte mínimo que você estabeleceu foi de ٤٠٪, e a confiança mínima foi de ٦٠٪. Diante desse contexto, analise se a regra gerada é de fato interessante para o Sr. Abdul Samir e em que termos deve ser interpretada, a fim de que decisões corretas sejam tomadas a partir dela.

5. No repositório de bases de dados *UCI Machine Learning* (Lichman, 2003), há uma série de conjuntos de dados sintéticos e conjuntos de dados reais que podem ser analisados como uma base de dados transacional e, portanto, podem ser fonte de descoberta de regras de associação. Para cada um dos conjuntos de dados a seguir, analise a aplicabilidade da descoberta de regras de associação e, sendo aplicável, implemente uma solução completa usando R (considere realizar algum pré-processamento sobre os conjuntos de dados de forma a adequá-los à tarefa de descoberta de regras de associação):

a) Contraceptive Method Choice Data Set (Lim, Loh e Shih, 2000) (**http://archive.ics.uci.edu/ml/datasets/Contraceptive+Method+Choic**

b) Adult Data Set (Kohavi,1996) (**https://archive.ics.uci.edu/ml/datasets/Adult)**

c) Mushroom Data Set (Schlimmer,1987) (**http://archive.ics.uci.edu/ml/datasets/Mushroom)**

d) Congressional Voting Records Data Set (Schlimmer, 1987) (**https://archive.ics.uci.edu/ml/datasets/Congressional+Voting+Recor**

APÊNDICE

INICIANDO EM R

O R é uma linguagem de programação e um ambiente para desenvolvimento de ferramentas computacionais que envolvam conceitos de estatística e apresentação de resultados na forma de gráficos (Hornik, 2015; R Core Team, 2015). O ambiente é distribuído como parte do projeto *Free Software Foundation's* GNU, cujo regulamento tem objetivos similares aos de outros softwares livres.[1]

O R está disponível para download,[2] com versões para os sistemas operacionais Linux, OS e Windows. Desde 2011, o R conta com uma versão chamada R-Studio,[3] cujo principal diferencial é a interface gráfica. Alternativamente, ainda é possível encontrar disponíveis na internet aplicações em nuvem que permitem a simulação de códigos R sem a necessidade de instalação local do seu ambiente.

Este apêndice tem o objetivo de introduzir o leitor à prática de uso de funções desenvolvidas em R. Para isso, são fornecidas explicações básicas sobre como manipular estruturas de dados, arquivos, pacotes, funções para construção de gráficos e algumas outras funções úteis para o desenvolvimento dos programas que ilustram os capítulos deste livro.[4]

A.1. Descobrindo o R

A criação e execução de códigos em R podem ser feitas de duas maneiras. A primeira é por meio da interface em linha de comando. A linha de comando constitui-se de um terminal identificado pelo cursor `>`, a partir do qual são executadas instruções em R, como definição de variáveis, criação de vetores e matrizes, chamadas das funções etc. A execução de uma instrução na linha de comando é realizada quando a tecla *ENTER* é pressionada. Por exemplo, operações matemáticas simples na linha de comando são executadas da seguinte maneira: digite `1+2+3` em frente ao cursor **>** e então tecle *ENTER*. O efeito dessa execução será:

```
> 1+2+3
[1] 6
```

Outros exemplos são:

```
> 1+2*3
[1] 7
>(1+2)*3
[1] 9
```

Observe, nos exemplos, que as instruções definidas após o cursor `>` e executadas a partir do acionamento da tecla *ENTER* geraram respostas apresentadas após um índice `[1]`. Esse índice indica o primeiro elemento da linha que contém a resposta de cada instrução. Se a resposta de uma instrução ocupa mais de uma linha, esse índice servirá como suporte para compreensão da resposta. Como ilustração, considere uma instrução que cria uma sequência numérica com valores entre 1 e 60:

```
> 1:60
```

cuja resposta é exibida da seguinte maneira:

```
[1] 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
25
[26]26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46
47 48 49 50
[51]51 52 53 54 55 56 57 58 59 60.
```

A marcação `[1]` é o índice do primeiro elemento da linha, `[26]`, o primeiro da segunda linha, e `[51]`, o primeiro da terceira linha. Note que o número dentro dos colchetes é referente ao posicionamento do elemento na sequência, e não ao seu valor. Para se certificar dessa informação, crie uma sequência de 60 números usando: `> 2:61`. É claro que a quantidade de valores exibidos em uma linha depende do espaço disponível na janela de execução do R. Para sair do ambiente R, digite `q( )`. Para limpar a tela da linha de comando, digite *CTRL+L* (no sistema operacional Windows) ou *COMMAND+OPTION+L* (no sistema operacional OS).

A segunda maneira de criar e executar códigos em R é por meio de um editor de scripts. No editor, é possível reunir todos os comandos desejados em uma implementação de uma nova função. Na Figura A-1(a) está ilustrada a interface do editor no ambiente R. Note que, nessa figura, há um menu de opções. Na opção Arquivo, vê-se a subopção para criação de um novo script (e outras, como abrir script, imprimir, sair do ambiente etc.). Ao escolher um novo script, abre-se o editor, conforme ilustrado na Figura A-1(b). Nesse exemplo, há um comentário (texto após o caractere "`#`") e uma variável (`x`) recebendo (`<-`) o resultado da operação de soma (`1+2+3`). Para executar o comando, deve-se escolher, na opção de menu, Editar, e então a subopção Executar (executar todo o script ou então fazer a execução passo a passo).

Figura A-1: Interface do ambiente R (sistema operacional Windows)

A.2. Pacotes

As funções em R estão organizadas nos chamados pacotes (*packages*) e estarão disponíveis a partir do momento em que o pacote for carregado. Esse é um artifício usado para otimizar memória e para lidar com conflitos em nomes de funções criadas por diferentes desenvolvedores. Existem muitos pacotes disponíveis para download, e aqueles recomendados pelo comitê do R são acessíveis por meio do repositório CRAN.[5]

A partir de uma conexão ativa à internet, a instalação de um pacote pode ser feita pela interface do ambiente (opção Pacotes do menu, veja Figura A-1) ou então digitando na linha de comando:

```
> install.packages("nome_pacote")
```

A partir de uma instalação do R, é possível visualizar os pacotes nela disponíveis com o uso da função `library( )`, a partir da linha de comando, da seguinte forma:

```
> library( )
```

Essa mesma função pode ser usada para carregar um pacote, por meio da passagem do nome do pacote como parâmetro da função. Por exemplo, o carregamento do pacote de funções gráficas *lattice* é feito da seguinte maneira:

```
> library("lattice")
```

Então, para saber quais pacotes estão disponíveis para uso, a função `search( )` deve ser usada:

```
> search()
```

Caso seja necessário buscar ajuda para a compreensão de alguma função disponível em um pacote já carregado, a função `help.search( )` deve ser usada, adicionando-se como parâmetro o nome da função sobre a qual se deseja obter mais detalhes. Por exemplo, para saber mais detalhes da função logarítmica, carregada dentro do pacote *default* do R, proceda com:

```
> help.search("logarithm")
```

Várias das funções descritas no decorrer deste apêndice vêm de pacotes geralmente carregados na inicialização do R. Porém, eventualmente, se alguma função exigir um pacote ainda não carregado pelo ambiente, basta carregá-lo como explicado nesta seção. Nos exemplos práticos apresentados no decorrer deste livro, cada algoritmo pode estar disponível em um particular pacote. Sendo assim, scripts implementados para realização de testes dos algoritmos deverão conter, nas primeiras linhas, as funções de instalação e carregamento do pacote, ou seja, `install.packages( )` e `library( )`.

A.3. Variáveis

Em R, uma variável é definida com o uso do operador `<-`, o qual significa "gets" e, em português, poderia ser traduzido como "atribuição". Por exemplo, a atribuição do valor 1 à variável `x`, em R, é realizada da seguinte forma:

```
> x <- 1
```

Para verificar o valor atribuído à variável, proceda da seguinte forma:

```
> x
```

E, após o acionamento da tecla *ENTER*, o valor da variável será mostrado na tela:

```
[1] 1
```

Essa notação para atribuição de variável e comumente usada em pseudocódigos e é bastante comum em livros que apresentam discussões sobre algoritmos. Perceba também que não é necessário definir o tipo da variável, a própria linguagem fará essa definição no momento de execução da atribuição. Veja outros exemplos de atribuições, considerando diferentes tipos de dados:

```
# Atribuindo um conjunto de caracteres (string) à variável X
> x <- "Olá mundo!"
> x
[1] "Olá mundo!"

# Atribuindo um valor numérico do tipo real à variável X
> x <- 3.46
> x
```

```
[1] 3.46
```

A.4. Funções matemáticas

Assim como todas as linguagens de programação, o R possui uma série de funções nativas. Entre elas, estão as funções matemáticas. O uso de algumas delas segue exemplificado a seguir:

```
>#raiz quadrada
> sqrt(4)
[1] 2

>#exponencial
> exp(1)
[1] 2.718282

>#cosseno
> cos(90)
[1] -0.4480736

#logaritmo. O primeiro argumento diz respeito ao valor sobre o
qual será #aplicada a função logaritmo; o segundo argumento indica
a base do #logaritmo
> log(1,2)
[1] 0
```

Nos exemplos anteriores, não foram realizadas atribuições. Dessa forma, embora as funções tenham sido executadas e seus resultados tenham sido ecoados na tela, esses valores não estão acessíveis por meio de variáveis. Obviamente, tais funções podem ser usadas em atribuições, como em outros exemplos deste apêndice.

A.5. Tipos de dado

Embora não seja necessário definir o tipo de variável antes de lhe atribuir um valor, existem situações em que se faz necessário saber o tipo de variável. Em R, essa consulta pode ser feita com o uso da função `class( )`.

Os tipos de dado retornados pela função `class` são numérico (`numeric`) e caractere (`character`). Um tipo numérico pode ser tanto um valor real (*float* ou *double*) quanto um valor inteiro (*integer*), ou seja, não se faz a distinção entre esses dois tipos de valor. A linguagem faz a conversão de valor real para valor inteiro de forma transparente ao usuário. Veja como conhecer o tipo de uma variável por meio das seguintes instruções:

```
> x <- 3.46
> class(x)
[1] "numeric"

> x <- 10
> class(x)
[1] "numeric"
```

Em situações em que se faz necessário definir uma variável como um valor do tipo inteiro, é preciso realizar a atribuição do valor à variável acompanhado de **`L`**, ou seja:

```
> x <- 10L
> class(x)
[1] "integer"
```

Note que, independentemente da natureza numérica atribuída, a linguagem entende a variável como sendo do mesmo tipo, `numeric`. Entretanto, caso seja necessário verificar o tipo exato da variável, as funções `is.numeric( )` ou `is.integer( )` devem ser executadas.[6] Essas funções

retornam como resultado os valores lógicos **TRUE** ou **FALSE** para verdadeiro ou falso, respectivamente. Veja o seguinte exemplo:

```
x <- 3.46
> is.numeric(x)
[1] TRUE
> is.integer(x)
[1] FALSE
```

Para variáveis às quais foi atribuído um conjunto de caracteres, a consulta sobre seu tipo terá como resposta o valor *char*. Para descobrir a quantidade de caracteres na variável, a função `nchar( )` pode ser usada. Veja o exemplo a seguir:

```
> x <- "Olá mundo!"
> class(x)
[1] "character"
> nchar(x)
[1] 10
```

A.5.1. Vetores

Vetores são variáveis com um ou mais valores do mesmo tipo: inteiro, real, lógico, caractere (ou *string*) etc. Em R, mesmo uma variável com um único valor (um escalar) é considerada um vetor, nesse caso, um vetor de comprimento 1.

A criação de um vetor em R pode ser feita de diferentes maneiras, dependendo do uso que se pretende para a variável. Se o propósito de criação do vetor é a representação de uma sequência numérica, então é esperada a definição do valor inicial e do valor final do vetor, separados pelo caractere **:** (dois-pontos). Por exemplo:

```
> x <- 1:10
> x
[1] 1 2 3 4 5 6 7 8 9 10
```

Também é possível criar uma sequência na qual o intervalo entre os valores não seja unitário. A função `seq( )` deve ser usada para esse propósito. Ela recebe três parâmetros: o valor inicial da sequência, o valor final da sequência e o intervalo desejado entre os valores dessa sequência. Caso o terceiro parâmetro não seja definido, a função assume o intervalo unitário. O exemplo a seguir gera uma sequência de 1 até 10 com intervalo de 2.

```
> seq(1,10,2)
[1] 1 3 5 7 9
```

Caso seja necessário criar uma sequência com valores repetidos, a função `rep( )` deve ser usada. Ela permite repetir um valor ou uma sequência de valores. Veja exemplos da aplicação dessa função a seguir:

```
> rep(1,5)
[1] 1 1 1 1 1

> rep(1:3,2)
[1] 1 2 3 1 2 3
```

Se a necessidade for gerar uma sequência aleatória, o R disponibiliza duas possibilidades, sendo uma delas a criação da sequência a partir de uma distribuição uniforme, e a segunda, a partir de uma distribuição normal. A função `runif( )` cria a sequência a partir da distribuição uniforme e necessita de três parâmetros: o número de elementos na sequência, o valor mínimo da sequência e o valor máximo da sequência. A sintaxe e o resultado do uso dessa função são ilustrados a seguir.

```
> x <- runif(5,1,10)
> x
[1] 6.044645 5.041281 7.988245 1.673179 7.553612
```

Para o caso da geração da sequência aleatória a partir de uma distribuição normal, a função `rnorm( )` se aplica. Ela recebe como parâmetros o número

de elementos desejado na sequência, o valor médio e o desvio da distribuição normal. Veja o exemplo:

```
> x <- rnorm(5,4,1)
> x
[1] 3.419219 4.753763 5.289175 3.800758 2.349655
```

Alternativamente, caso seja necessário definir um vetor com uma sequência numérica específica, há a possibilidade do uso da função concatenação `c( )`, como descrito a seguir:

```
> x <- c(30,10,40,20,80,50,70,100,90)
> x
[1] 30 10 40 20 80 50 70 100 90
```

A quantidade de operações possíveis em um vetor é bastante extensa. Para apresentar as principais, num contexto de utilidade para este livro, um resumo com os nomes, descrições breves e resultados de aplicação (em `x`, considerando o último exemplo) de algumas funções é apresentado na Tabela A-1. É importante saber também que, em R, o primeiro elemento do vetor é o elemento de índice 1 e, portanto:

```
> x[1]
[1] 30
```

Tabela A-1: Principais funções que podem ser aplicadas a vetores

Função	Descrição	Resultado
`min(x)`	Encontra o valor mínimo em `x`.	`[1] 10`
`max(x)`	Encontra o valor máximo em `x`.	`[1] 100`
`range(x)`	Gera um vetor com os valores de `min(x)` e `max(x)`.	`[1] 10 100`
`sort(x)`	Gera uma versão ordenada de `x`.	`[1] 10 20 30 40 50 70 80 90 100`

`rank(x)`	Gera um vetor de inteiros contendo os índices dos elementos de `x`, de acordo com a posição que o elemento assume em uma sequência ordenada. Veja que 100 é o maior valor e assume a última posição em uma sequência ordenada e, por isso, ele recebe o maior índice, 9.	`[1] 3 1 4 2 7 5 6 9 8`
`order(x)`	Um vetor de inteiros contendo os índices dos elementos de `x`, considerando a ordem crescente dos valores desses elementos. O primeiro inteiro no vetor resultante é 2, pois o menor elemento no vetor `x` (o número 10) está na segunda posição. Para usar a ordem decrescente, faça: `order(x, decreasing=TRUE).`	`[1] 2 4 1 3 6 7 5 9 8`
`length(x)`	Fornece o total de elementos do vetor `x`.	`[1] 9`
`sum(x)`	Calcula a soma de todos os elementos de `x`.	`[1] 490`
`mean(x)`	Calcula a média aritmética dos elementos de `x`.	`[1] 54.444444`
`median(x)`	Calcula a mediana de `x`.	`[1] 50`
`var(x)`	Calcula a variância de `x`.	`[1] 1027.778`
`quantile(x)`	Gera um vetor contendo o valor mínimo, quartil inferior, mediana, quartil superior e valor máximo de `x`.	`0% 25% 50% 75% 100%` `10 30 50 80 100`

Ainda é possível criar vetores com conjuntos de caracteres. Nesse caso, cada elemento deve estar entre aspas **“ ”** e separados por vírgula. Por exemplo:

```
> cardapio <- c(“Filé à parmegiana”, “Feijoada”, “Batatas Fritas”, “Lasanha à Bolonhesa”, “Salada Caprese”)
> cardapio
[1] “Filé à parmegiana” “Feijoada” “Batatas Fritas” “Lasanha à Bolonhesa”
[5] “Salada Caprese”
```

O acesso a cada elemento de um vetor com conjuntos de caracteres pode ser feito de maneira semelhante ao de vetores numéricos. Ou seja, para acessar o elemento de número 5, tem-se:

```
> cardapio[5]
```

```
[1] "Salada Caprese"
```

É possível fazer algumas manipulações em um vetor de conjuntos de caracteres. Por exemplo, para descobrir o número de caracteres em cada elemento do vetor, faça:

```
> nchar(cardapio)
[1] 17 8 14 19 14
```

Algumas das funções apresentadas na Tabela A-1 podem ser aplicadas a vetores de conjuntos de caracteres, como é o caso da função `length( )`, para descobrir o tamanho do vetor, a função `sort( )`, para obter uma ordenação em ordem alfabética, ou mesmo a função `order( )`, para ter a posição dos elementos em ordem alfabética. Veja a execução desses três exemplos a seguir:

```
> length(cardapio)
[1] 5

> sort(cardapio)
[1] "Batatas Fritas" "Feijoada" "Filé à parmegiana"
[4] "Lasanha à Bolonhesa" "Salada Caprese"

> order(cardapio)
[1] 3 2 1 4 5
```

A.5.2. Data frames

A função `data.frame( )`,que resulta em um tipo de dado especial chamado *data.frame*, permite estruturar dados em um formato semelhante ao que se encontra em planilhas eletrônicas. Esse recurso permite que vetores de diferentes tipos sejam arranjados em uma estrutura de linhas e colunas, na qual cada coluna pode ser nomeada. Por exemplo, considere que, para o exemplo anterior (o cardápio), se deseja acrescentar os preços de cada um dos pratos. Um vetor de preços é criado como especificado a seguir:

```
> precos <- c(32.50, 44.00, 12.00, 35.50, 27.00)
> precos
[1] 32.5 44.0 12.0 35.5 27.0
```

A função `data.frame( )` permite relacionar o vetor *cardapio* com o vetor *precos* em uma estrutura como uma tabela. Assim, é possível ter dois vetores de tipos diferentes em uma mesma estrutura de dados, mas que, sobretudo, referem-se à mesma unidade observacional, o cardápio do restaurante. Em um *data.frame*, ainda é possível especificar um nome para cada coluna da tabela, passando parâmetros para a função formados por pares *nome = vetor*, como mostrado a seguir.

```
> data.frame(Cardápio = cardapio, Preços = precos)
Cardápio Preços
1 Filé à parmegiana 32.5
2 Feijoada 44.0
3 Batatas Fritas 12.0
4 Lasanha à Bolonhesa 35.5
5 Salada Caprese 27.0
```

A.5.3. Listas

Uma lista em R, definida pela função `list( )`, é um recurso que tem características semelhantes às do *data.frame*. A lista também possibilita o relacionamento de vetores de diferentes tipos; entretanto, a principal diferença está na possibilidade de concatenar elementos de diferentes tamanhos. Por exemplo, considere que seja necessário armazenar a quantidade de vendas de cada um dos pratos do cardápio nos últimos três dias. Ou seja:

```
> dia1 <- c(5, 3, 7, 4, 2)
> dia2 <- c(4, 0, 4, 4, 0)
> dia3 <- c(5, 7, 3, 2, 5)

> dias <- list(dia1,dia2,dia3)
> dias
[[1]]
```

```
[1] 5 3 7 4 2

[[2]]
[1] 4 0 4 4 0

[[3]]
[1] 5 7 3 2 5

> list(cardapio, precos, dias)
[[1]]
[1] "Filé à parmegiana" "Feijoada" "Batatas Fritas"
[4] "Lasanha à Bolonhesa" "Salada Caprese"

[[2]]
[1] 32.5 44.0 12.0 35.5 27.0

[[3]]
[[3]][[1]]
[1] 5 3 7 4 2

[[3]][[2]]
[1] 4 0 4 4 0

[[3]][[3]]
[1] 5 7 3 2 5
```

A.5.4. Matrizes

Matriz é uma estrutura de dados essencial para a resolução de tarefas de análise de dados. A organização em linhas e colunas dessa estrutura de dados se assemelha ao *data.frame*, diferenciando-se pelo fato de que o tipo dos dados nela inseridos deve ser sempre o mesmo. A criação da matriz é feita com uso da função `matrix( )`, passando dois parâmetros: uma sequência numérica e o número de linhas desejado na matriz. O número de colunas será derivado da combinação desses parâmetros, no entanto, ele ainda pode ser definido com uso do parâmetro `col`. Veja a criação de uma matriz com 5 linhas e 2 colunas:

```
> X <- matrix(1:10, nrow = 5)
> X
```

```
[,1] [,2]
[1,] 1 6
[2,] 2 7
[3,] 3 8
[4,] 4 9
[5,] 5 10
```

Note que, nessa matriz, a linha é designada por um número seguido por vírgula **“,”** e a coluna é designada por um número precedido de uma vírgula **“,”**. Sendo assim, para ler o número 8 da matriz, deve-se trabalhar com um ajuste de índices da seguinte forma:

```
> X[3,2]
[1] 8
```

A.5.5. Arrays

Array é um vetor multidimensional, o que basicamente difere essa estrutura de dados de uma matriz. A função `array( )` recebe dois parâmetros: o(s) valor(es) que deve(m) ser armazenado(s) e um vetor contendo o número de linhas, número de colunas e número de dimensões desejado. Por exemplo, para guardar uma sequência de 12 números em um *array* bidimensional, sendo que cada dimensão deve contar com duas linhas e três colunas, a sintaxe é:

```
> X <- array(1:12,dim=c(2,3,2))
, , 1

[,1] [,2] [,3]
[1,] 1 3 5
[2,] 2 4 6

, , 2

[,1] [,2] [,3]
[1,] 7 9 11
[2,] 8 10 12
```

O acesso ao número 8 é feito indicando a linha, a coluna e a dimensão do *array*:

```
> X[2,1,2]
[1] 8
```

A.6. Trabalhando com vetores

A discussão, com um pouco mais de detalhes, de algumas possibilidades para trabalhar com vetores em R se justifica pelo fato de que esse tipo de dado pode estar organizado em *data.frames*, listas, matrizes ou *arrays*.

Um conceito muito útil para a implementação de algoritmos de análise de dados é o cálculo do produto interno. Ele é realizado pela somatória da multiplicação dos elementos de dois vetores quaisquer, porém de mesmo tipo e tamanho. Para implementar essa operação, a primeira solução que vem à mente é a criação de um laço de repetição para percorrer os elementos dos vetores e, a cada passo, multiplicar os elementos e somar o resultado em uma variável auxiliar. Considere, portanto, o código em R que realiza o cálculo do produto interno seguindo o procedimento descrito para dois vetores `x` e `y` quaisquer.

```
> x <- c(1,2,3)
> y <- c(3,2,1)
# variável auxiliar
> res <- 0
> for(i in 1:3){
res <- res + x[i]*y[i]
}
> print(res)
[1] 10
```

Em termos matemáticos, o resultado está correto. No entanto, a criação de um laço para percorrer os vetores exige um custo computacional que pode ser caro, dependendo do comprimento dos vetores. Uma vantagem no uso do R é a possibilidade de fazer cálculos vetoriais usando recursos apropriados e disponíveis na linguagem. Para uma apresentação mais ampla de como o cálculo vetorial é realizado, considere que, formalmente, a solução anterior pode ser descrita pela equação:

$$res = \sum_{i=1}^{3} (x_i \times y_i)$$ **Equação A-1**

em que × define a multiplicação dos elementos dos vetores. Em R, como as variáveis `x` e `y` foram atribuídas a um vetor, a operação da equação pode ser escrita usando a função `sum( )`, que recebe uma multiplicação como parâmetro, como especificado a seguir. Note que, nessa solução, não foi necessário especificar uma varredura nos vetores.

```
> res <- sum(x*y)
> res
[1] 10
```

Observe que é possível otimizar ainda mais a operação. Formalmente, reescrevendo a Equação A-1 em notação vetorial, tem-se que:

$$\text{res} = \boldsymbol{x}\rightarrow * \boldsymbol{y}\rightarrow^{T}$$ **Equação A-2**

Em R, a implementação seguindo essa formalização é dada por:

```
> x%*%y
[,1]
[1,] 10
```

E, nesse caso, o operador de multiplicação aparece entre os símbolos "`%%`", sendo suficiente para a linguagem entender que a Equação A-1 está sendo implementada. O símbolo "`%`" também permite que operações com *data.frames*, listas, matrizes e *arrays* também sejam realizadas. Veja o exemplo a seguir, que ilustra a multiplicação de duas matrizes `X` e `Y`:

```
> X <- matrix(c(2,8,3,2),nrow=2)
> X
[,1] [,2]
[1,] 2 3
```

```
[2,] 8 2

> Y <- matrix(c(9,7,5,10,1,3),nrow=2)
> Y
[,1] [,2] [,3]
[1,] 9 5 1
[2,] 7 10 3

> X%*%Y
[,1] [,2] [,3]
[1,] 39 40 11
[2,] 86 60 14
```

Ou, ainda, é possível repetir a operação para multiplicar um vetor por uma matriz. Veja o exemplo a seguir, em que o vetor é determinado pela leitura da primeira linha da matriz `X`.

```
> X[1,]%*%Y
[,1] [,2] [,3]
[1,] 39 40 11
```

Nos exemplos anteriores, não foi preciso transpor um vetor ou uma matriz. Caso seja necessário, em R, essa operação é feita com uso da função `t( )`. Portanto, a transposta da matriz `X` pode ser feita da seguinte forma:

```
> t(X)
[,1] [,2]
[1,] 2 8
[2,] 3 2
```

O R ainda disponibiliza uma série de funções para cálculo vetorial. As mais importantes para o contexto deste livro estão resumidas na Tabela A-2, considerando os vetores:

```
> x
[1] 1 2 3
> y
[1] 3 2 1
```

e a transformação dos vetores em matriz com uso da função `rbind()`:[7]

```
> X <- rbind(x,y)
> X
[,1] [,2] [,3]
x 1 2 3
y 3 2 1 .
```

Tabela A-2: Exemplos de funções para cálculo vetorial

Função	Descrição	Resultado
`cor(x,y)`	Calcula a correlação entre os vetores `x` e `y`.	[1] -1
`dist(X)`	Calcula a distância euclidiana entre os vetores `x` e `y`. A entrada deve ser uma matriz que pode ser gerada com auxílio da função **rbind( )**, a qual concatena os vetores como linhas da matriz. Ainda pode ser definido um segundo atributo chamado *method*, indicando outras medidas de distância a serem aplicadas pela função.	x y 2.828427

A.7. Importando dados de arquivos

Até este ponto, todos os exemplos discutidos fizeram uso de dados gerados na própria linha de comando. Entretanto, na prática, é comum a necessidade de usar dados armazenados em planilhas, arquivos textos ou em bases de dados. A importação de arquivos em R pode ser feita basicamente de duas maneiras: a partir da leitura de arquivo ou a partir da leitura de bases de dados. A linguagem R permite, inclusive, a chamada de consultas expressas em SQL (*Structured Query Language*). Neste apêndice, apenas a leitura de dados armazenados em arquivos será tratada, já que esse é o tipo de leitura usada nos algoritmos descritos neste livro.

O procedimento mais comum é a leitura de arquivos com valores separados por um identificador, como é o caso de arquivos do tipo CSV (*comma separated values*). O CSV é um formato de exportação frequentemente presente em ambientes que armazenam ou exploram dados. Isso significa que a explicação aqui apresentada permite o trabalho com dados que estejam em um Sistema de Gerenciamento de Banco de Dados ou que foram gerados em um software de edição de planilhas eletrônicas. Em ambos, a geração de um arquivo CSV é feita por meio de funcionalidades de exportação, nas quais ainda é possível escolher o tipo de separador a ser usado no arquivo. Essa possibilidade é bastante útil, pois há situações em que o separador “vírgula” é usado na notação numérica para separação da parte inteira e parte decimal em um valor, e, portanto, a escolha de outro separador ajuda a evitar erros de importação ou de leitura do arquivo.

A leitura de um arquivo CSV em R é feita pela função `read.table( )`. Ela exige pelo menos três parâmetros de entrada: o *caminho do arquivo,* que pode ser um caminho para uma pasta local ou um endereço da internet; um indicador (*header*) de que a primeira linha é um cabeçalho (TRUE) ou já é conteúdo de dados (FALSE); e o tipo de separador (*sep*), que diz respeito ao caractere usado para separar os valores contidos no arquivo.

Para finalidade de ilustração, será utilizado o conjunto de dados Iris (Fisher e Marshall, 1988), do repositório de conjuntos de dados da Universidade de Irvine – Califórnia[8] (Lichman, 2013). O conjunto de dados Iris consiste em 150 exemplares, descritos por quatro medidas numéricas de largura e altura de sépalas e pétalas de uma planta chamada Iris. O conjunto ainda tem um atributo de rótulo (de caracteres) com a informação sobre a espécie da planta. Este é um conjunto amplamente usado na literatura de análise de dados, pois reúne amostras de dados linearmente separáveis e não separáveis linearmente. A leitura do conjunto é feita como segue, considerando que o conjunto de dados foi organizado em um arquivo csv com cabeçalho na primeira linha:

```
> iris <- read.table("C:\\Users\\livroMD\\Apendice\\iris.csv",
header=TRUE,sep=",")
```

Nesse exemplo, foi confirmado que o cabeçalho (`header`) está presente na base (`TRUE`) e, como último parâmetro, foi informado que os valores estão separados (`sep`) pelo caractere "vírgula" (","). Uma amostra do conjunto de dados pode ser vista com o uso da função `head( )`, como a seguir:

```
> head(iris)
Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1 5.1 3.5 1.4 0.2 setosa
2 4.9 3.0 1.4 0.2 setosa
3 4.7 3.2 1.3 0.2 setosa
4 4.6 3.1 1.5 0.2 setosa
5 5.0 3.6 1.4 0.2 setosa
6 5.4 3.9 1.7 0.4 setosa
```

Note que a leitura transforma os dados em um *data.frame*. Consequentemente, toda explicação anterior para leitura de linha, coluna, multiplicação etc. é válida.

```
> class(iris)
[1] "data.frame"
```

Alternativamente, a leitura pode ser feita a partir de um arquivo sem cabeçalho. Nesse caso, o parâmetro `header` recebe o valor `FALSE`, e o R atribui rótulos genéricos para identificar os atributos.

```
> caminho <- "C:\\Users\\livroMD\\Apendice\\iris.csv"
> caminho
[1] "C:\\Users\\livroMD\\Apendice\\iris.data"
> iris2 <- read.table(file=caminho,header=FALSE,sep=",")
> iris2
V1 V2 V3 V4 V5
1 5.1 3.5 1.4 0.2 Iris-setosa
2 4.9 3.0 1.4 0.2 Iris-setosa
3 4.7 3.2 1.3 0.2 Iris-setosa
4 4.6 3.1 1.5 0.2 Iris-setosa
5 5.0 3.6 1.4 0.2 Iris-setosa
6 5.4 3.9 1.7 0.4 Iris-setosa
...
```

Por fim, o arquivo pode estar disponível em formato de uma planilha eletrônica (.xlsx, por exemplo). Para a leitura dos dados nesse formato, é preciso instalar o pacote **xlsx** (Dragulescu, 2014) para o uso da função `read.xlsx( )`. Após sua instalação e inicialização, a leitura do arquivo no mesmo caminho do exemplo anterior, com a diferença do formato do arquivo, será feita como:

```
> iris3 <- read.xlsx(caminho,"dados",header=TRUE)
```

sendo "dados" o nome da folha da planilha que deverá ser considerada na importação dos dados.

A.8. Gráficos

R inclui funções para a criação de vários de tipos de gráficos, como gráfico de barras, de setores, de linhas, de dispersão etc. Essas funções estão disponíveis no pacote *graphics*, nativo da linguagem. Há ainda outros tipos de gráficos possíveis, mais sofisticados, cujas funções para criação estão disponíveis no pacote *lattice*, por exemplo. Nesta seção, são apresentados os principais gráficos úteis no entendimento dos conceitos apresentados neste livro. A Tabela A-3 apresenta um resumo das funções para criação dos gráficos apresentadas neste apêndice.

Tabela A-3: Exemplos de funções para construção de gráficos

Função	Descrição
`plot( )`	Cria um gráfico de dispersão ou um gráfico de linhas
`barplot( )`	Cria um gráfico de colunas ou de barras
`pie( )`	Cria um gráfico de setores ou do tipo pizza
`boxplot( )`	Cria um gráfico de caixas

A.8.1. Gráfico de dispersão

Um gráfico de dispersão (*scatter plot*), ou espalhamento, permite a visualização da distribuição dos exemplares de um conjunto de dados. A função `plot( )` constrói esse gráfico a partir dos valores atribuídos a uma série de parâmetros (os principais estão listados na Tabela A-4). De maneira genérica, considerando que dois vetores serão usados na plotagem dos gráficos, a função é definida como:

```
> plot(x, y = NULL, type = "p", xlim = NULL, ylim = NULL, log =
"",
main = NULL, sub = NULL, xlab = NULL, ylab = NULL, col="blue")
```

Tabela A-4: Parâmetros da função PLOT

Parâmetros	Descrição
`x,y`	Os dados a serem plotados, que devem estar separados em dois vetores.
`type`	Tipo de marcador do gráfico: `p,` para pontos, `l,` para linhas, `o,` para pontos e linhas.
`xlim`	Um vetor com os limites inferior e superior do eixo *x*.
`ylim`	Um vetor com os limites inferior e superior do eixo *y*.
`log`	Uma sequência de caracteres deve conter “x” caso se queira que o eixo x esteja em escala logarítmica, “y” para o eixo y se quiser que ele esteja em escala logarítmica ou, no caso em que ambos os eixos devam estar nessa escala, usar a sequência “xy” ou “yx”.
`main`	O título principal do gráfico.
`sub`	O subtítulo do gráfico.
`xlab`	O rótulo do eixo *x*.
`ylab`	O rótulo para o eixo *y*.
`col`	Cor dos marcadores: `red`, `blue` ou `purple`.

Para exemplificar o uso da função `plot( )` na construção de um gráfico de dispersão, considere dois atributos descritivos do conjunto de dados Iris: `Sepal.Width` e `Petal.Width`. Inicialmente, esses atributos devem ser colocados em duas variáveis do tipo vetor (`x` e `y`), e, em seguida, a função pode ser executada.

```
> x <- iris[,”Sepal.Width”]
> y <- iris[,”Petal.Width”]
> plot(x,y,type=”p”,main = “PLANTA IRIS”,xlab = “Sepal
Width”,ylab= “Petal Width”)
```

O gráfico gerado a partir dessa última sequência de comandos está apresentado na Figura A-2. Nesse exemplo, nem todos os parâmetros apresentados na Tabela A-4 foram utilizados.

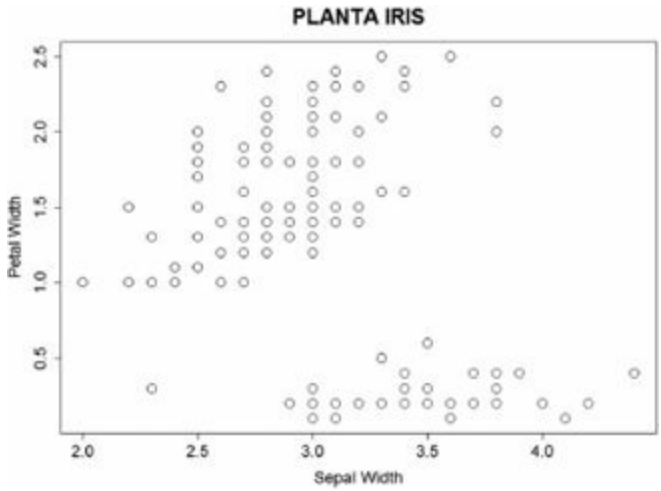


Figura A-2: Gráfico de dispersão para dois atributos descritivos do conjunto de dados Iris

No caso do conjunto de dados Iris, a informação referente ao rótulo de cada exemplar está disponível, e pode ser interessante usar essa informação na criação dos gráficos. Duas estratégias para uso dessa informação são apresentadas aqui. A primeira delas é com o uso da função `text( )`, que permite atribuir a informação de rótulo a cada ponto no gráfico (cada ponto no gráfico corresponde a um exemplar do conjunto de dados). Os principais parâmetros dessa função estão detalhados na Tabela A-5. A sequência de comandos necessários para a plotagem é apresentada a seguir, e o gráfico resultante é apresentado na Figura A-3.

Tabela A-5: Parâmetros da função TEXT

Parâmetros	Descrição
`x,y`	Os dados a serem plotados, que devem estar separados em dois vetores.
`labels`	Uma variável com as informações de rótulo de cada exemplar do conjunto de dados.
`cex`	Tamanho do texto que será plotado.
`adj`	Posicionamento do texto (para que fique posicionado de forma adjacente ao marcador).

```
> x <- iris[,"Sepal.Width"]
> y <- iris[,"Petal.Width"]
> especie <- iris[,"Species"]
> plot(x,y,type="p",main = "PLANTA IRIS",xlab = "Sepal
Width",ylab= "Petal Width")
```

```
> text(x,y,labels=especie,cex=0.8,adj=c(0,-1))
```

A segunda estratégia para explorar a informação sobre rótulos é por meio do uso da função `points( )`. Com essa função, é possível explorar recursos referentes a cores e marcadores. A Tabela A-6 apresenta algumas possibilidades de valores para o parâmetro cor da função `points( )`.

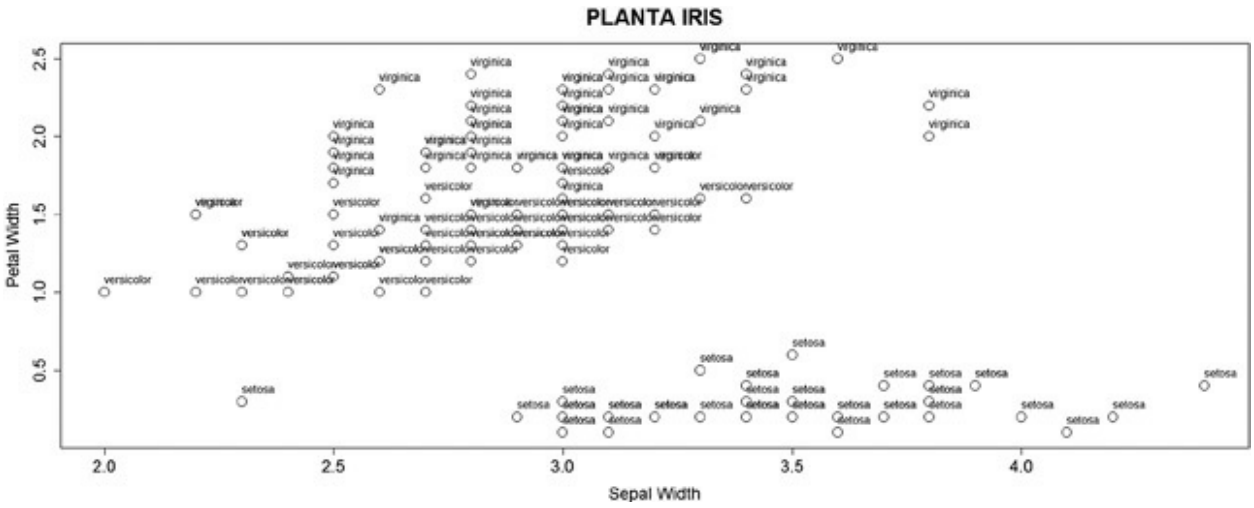


Figura A-3: Gráfico de espalhamento com os pontos rotulados pela função `TEXT`

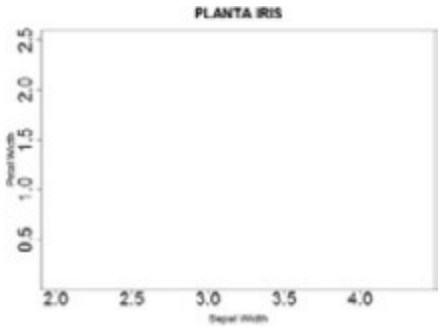
Tabela A-6: Valores para os parâmetros que usam cores, neste caso, exemplificando o `col` da função `plot`

Cor	Nome	Inteiro	Hexadecimal
Preta	`black`	`1`	`00000`
Vermelha	`red`	`2`	`FF0000`
Verde	`green`	`3`	`00FF00`
Azul	`blue`	`4`	`0000FF`

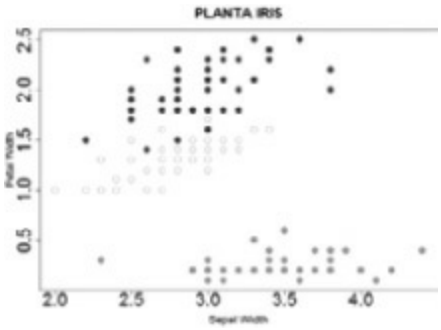
Para explorar o uso da função `plot( )`, algumas informações adicionais sobre o conjunto de dados Iris precisam ser consideradas. Esse conjunto de dados contém 150 exemplares, distribuídos uniformemente em três classes: 50 exemplares da classe setosa, 50 exemplares da classe versicolor e 50 exemplares da classe virgínica. Além disso, será necessário fazer uma

plotagem inicial, sem exibição dos marcadores dos exemplares (`type = n`) e, em seguida, a plotagem será preenchida adequadamente com a adição opcional de uma legenda (função `legend( )` – veja um resumo dos parâmetros dessa função na Tabela A-7 e opções de marcadores na Figura A-5). As três sequências de comandos a seguir se referem a esse procedimento, e os gráficos resultantes de cada uma são ilustrados pela Figura A-4.

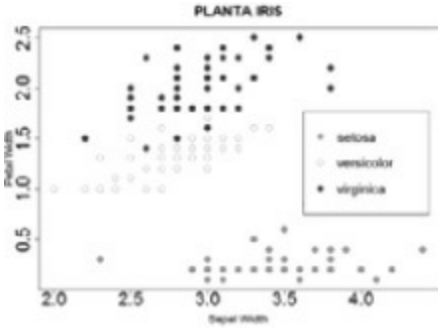
```
> plot(x,y, type="n", main = "PLANTA IRIS",xlab = "Sepal
Width",ylab= "Petal Width")
> points(x[1:50],y[1:50],col="2",cex=2,pch=19)
> points(x[51:100],y[51:100],col="3",cex=2,pch=1)
> points(x[101:150],y[101:150],col="4",cex=2,pch=19)
>
legend(locator(1),c("setosa","versicolor","virginica"),pch=c(19,1,
```



(a)



(b)



(c)

Figura A-4: Gráfico sem a impressão dos pontos (a), com a impressão de pontos de acordo com a distribuição das classes do conjunto de dados Iris (b), e com a impressão da legenda (c). Na execução do comando, os pontos serão coloridos

Tabela A-7: Parâmetros da função LEGEND

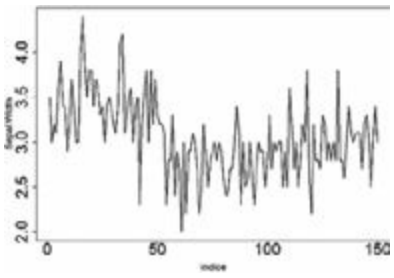
Parâmetros	Descrição
`pos` ou `locator`	A posição pode ser um vetor de coordenadas (`x`,`y`) ou então pode ser definida pelo usuário (`locator(1)`) com um clique de mouse no local desejado da legenda.
`names`	Os nomes que aparecerão na legenda. Podem ser definidos dentro de um vetor.
`pch`	Parâmetro que indica os tipos de marcador em uso. Uma indicação por rótulo utilizado.
`col`	Parâmetro que indica a cor escolhida para cada marcador. Uma indicação por rótulo utilizado.

Figura A-5: Códigos para marcadores disponíveis para o parâmetro `pch` da função `legend`

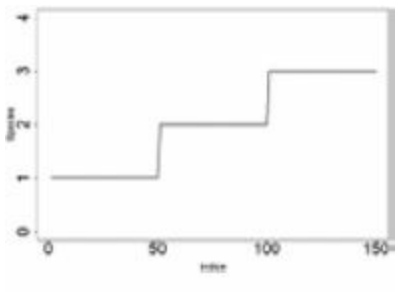
A.8.2. Gráfico de linhas

Com a função `plot( )`, também é possível imprimir gráficos de linhas . Um gráfico de linhas pode ser usado para suportar a análise de atributos individualmente. Em geral, no eixo *x* de um gráfico de linhas, estão os índices dos exemplares do conjunto de dados ou os tempos em que foram coletados, caso o conjunto de dados venha de uma série temporal de valores. Para exemplificar um gráfico de linhas, o atributo *Sepal Width* será utilizado. Assim como no gráfico de dispersão, inicialmente, é necessário executar a função `plot( )` com `type` = `n`, e, então, imprimir com a função `lines( )` a série de valores que darão origem à linha no gráfico. Veja a sequência de comandos necessários e a Figura A-6(a), que apresenta o gráfico.

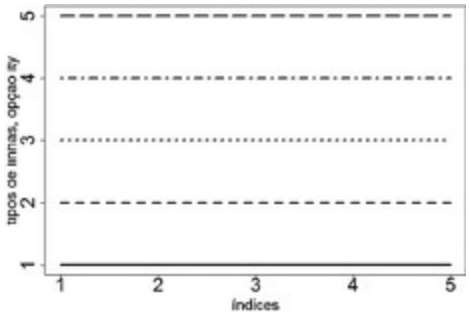
```
> plot(x,type="n",xlab="indice",ylab="Sepal.Width")
> lines(iris["Sepal.Width"])
```



(a)



(b)



(c)

Figura A-6: Gráfico de linhas: para o atributo descritivo numérico *Sepal Width* do conjunto de dados Iris (a), para o atributo de rótulo *Species*. Em (c), os diferentes tipos de linhas possíveis

A plotagem do comportamento assumido por valores de atributos categóricos também é possível. O gráfico apresentado na Figura A-6 ilustra a impressão do atributo de rótulo (*Species*) do conjunto de dados Iris. No gráfico, os nomes das espécies foram transformados em números apenas para fins de diferenciação (não há, porém, nenhuma relação de ordem entre as espécies). Com essa visualização, é possível perceber a distribuição uniforme das classes, como discutido anteriormente neste texto. Os comandos necessários para obter esse gráfico são:

```
> plot(x,type="n",xlab="indice",ylab="Species",ylim=c(0,4))
> lines(iris["Species"])
```

Na chamada da função `plot( )` desse exemplo, foi utilizado um novo parâmetro, o `ylim`. Esse parâmetro permite definir os valores mínimos e máximos do eixo *y*. Para a mesma parametrização do eixo *x*, o parâmetro `xlim` deve ser usado.

Ainda há outros tipos de linhas que podem ser usados nesses gráficos. A configuração do tipo da linha deve ser realizada por meio do parâmetro `lty` (*line type*); veja alguns dos tipos disponíveis na Figura A-6. Adicionalmente ao tipo de linha, é possível parametrizar sua espessura por meio do parâmetro `lwd` (*line weight*).

A.8.3. Gráfico de barras

Alternativamente ao uso do gráfico de linhas, a observação da quantidade de exemplares que possuem determinado valor para um atributo descritivo categórico pode ser feita usando um gráfico de barras . A utilidade de ambos, gráfico de linhas e gráfico de barras, é similar, mas, no caso do gráfico de barras, as quantidades são agregadas em uma contagem. Para criação desse gráfico em R, é preciso que os valores estejam organizados em uma matriz, sendo que, em uma das colunas, estão os índices, e, em outra, estão as quantidades agregadas.

Em geral, os valores a serem observados por meio de um gráfico de barras devem passar por algum tipo de transformação, como uma agregação. Agregações podem ser feitas com uso de funções de soma, média, mediana, contagem etc. Por exemplo, considere a contagem de exemplares pertencentes a cada categoria do atributo de rótulo *Species*. A etapa de transformação dos valores pode ser realizada com a aplicação da função `tapply( )`. Os principais parâmetros dessa função estão definidos na Tabela A-8.

Tabela A-8: Parâmetros da função TAPPLY

Parâmetros	Descrição
`index`	Índices da série de valores.
`x`	Vetor com os valores.
`function`	Agregação a ser aplicada nos valores do vetor, como `sum`, `mean`, `max` e `min`.

Para criar o gráfico, a seguinte sequência de comandos deve ser executada. O gráfico resultante está ilustrado na Figura A-7.

```
> quantidade <- tapply(rep(1,150),iris["Species"],sum)
> barplot(quantidade,xlab="Espécies",ylab="Frequência")
```

Figura A-7: Gráfico de barras com a contagem de exemplares por classe do conjunto de dados Iris

A.8.4. Gráfico de setores

O gráfico de setores , ou gráfico de pizza, como é popularmente chamado, tem o mesmo princípio do gráfico de barras, porém, a apresentação gráfica é diferente. O gráfico de setores para a variável `quantidade`, criada para o exemplo anterior, pode ser impresso com o uso da função `pie( )`, por meio da execução do comando:

```
> pie(quantidade)
```

O gráfico resultante está apresentado na Figura A-8. Note que ambos os tipos de gráfico, gráfico de barras e gráfico de setores, permitem que sejam adicionadas informações como nomes, cores, legenda etc., usando as mesmas opções já discutidas anteriormente para a função `plot( )`.

Figura A-8: Gráfico de setores que ilustra a contagem de exemplares por classe do conjunto de dados Iris

A.8.5. Gráfico de caixas

O gráfico de caixas, ou *boxplot*, é um instrumento que permite identificar a variação dos valores de uma variável, apresentando como resultados a localização dos 50% valores centrais, mediana e valores extremos (mínimo e máximo). Esses resultados podem ser vistos em R pelo uso da função `summary( )` e por meio da plotagem de um gráfico via função `boxplot( )`. Veja, a seguir, um exemplo da execução da função `summary( )` sobre o conjunto de dados Iris.

```
> summary(iris)
Sepal.Length Sepal.Width Petal.Length Petal.Width Species
Min. :4.300 Min. :2.000 Min. :1.000 Min. :0.100 setosa :50
1st Qu.:5.100 1st Qu.:2.800 1st Qu.:1.600 1st Qu.:0.300
versicolor:50
Median :5.800 Median :3.000 Median :4.350 Median :1.300 virginica
:50
Mean :5.843 Mean :3.057 Mean :3.758 Mean :1.199
3rd Qu.:6.400 3rd Qu.:3.300 3rd Qu.:5.100 3rd Qu.:1.800
Max. :7.900 Max. :4.400 Max. :6.900 Max. :2.500
```

A plotagem desses dados via um gráfico de caixas gera, para cada variável (ou atributo do conjunto de dados), uma representação dividida em quatro partes: a primeira parte (inferior), representada por uma linha pontilhada, diz respeito aos primeiros 25% dos valores assumidos pela variável e é limitada pelo valor mínimo e pelo valor do 1º Quartil ou quartil inferior (Q1); a quarta parte (superior), também representada por uma linha

pontilhada, diz respeito aos últimos 25% dos valores assumidos pela variável e é limitada pelo valor do 3º Quartil ou quartil superior (Q3) e pelo valor máximo; no centro, representando os 50% valores centrais assumidos pela variável, há uma caixa, dividida por uma linha contínua que representa a mediana dos valores assumidos pela variável, também chamada de 2º quartil (Q2). A caixa, portanto, representa 50% de todos os valores observados, concentrados na tendência central dos valores, eliminando os 25% menores valores e 25% maiores valores (75% – 25% = 50%). Valores muito díspares que podem aparecer em alguma variável são considerados outliers e representados no gráfico por círculos. Na Figura A-9 é apresentado o gráfico de caixas, ou *boxplot*, para os atributos descritivos do conjunto de dados Iris. A função `bloxplot( )` foi usada para a construção do gráfico, da seguinte maneira:

```
> boxplot(iris[1:4])
```

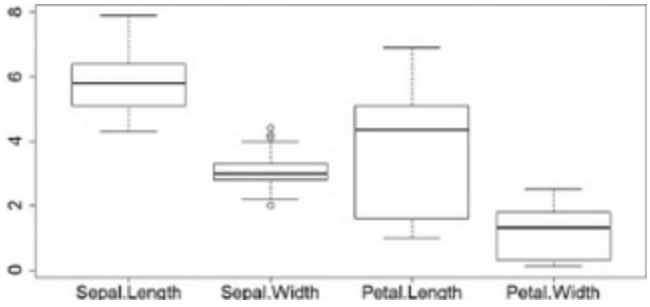


Figura A-9: Gráfico de caixas os atributos descritivos do conjunto de dados Iris

A.9. Histogramas

Um histograma , também conhecido como distribuição de frequências, pode ser apresentado em uma forma gráfica composta por colunas (ou barras), na qual as colunas dizem respeito à divisão em faixas (geralmente uniforme) dos valores de uma variável de um conjunto de dados. A função em R que gera um histograma é a `hist( )`, cujos principais parâmetros estão definidos na Tabela A-9.

Tabela A-9: Parâmetros da função HIST para plotagem do histograma

Parâmetros	Descrição
`x`	Conjunto de valores a serem organizados no histograma, geralmente um vetor.
`bin`	Quantidade de faixas nas quais se deseja dividir os valores de `x`.

Por exemplo, assumindo que se deseja separar os valores do atributo descritivo *Petal Width* em cinco faixas, tem-se:

```
> hist(t(iris[“Petal.Width”]),main=”Histograma de Petal.Width
separado
em faixas”,xlab=”Petal.Width”,5,ylab=”Frequência”)
```

Perceba que foi necessário aplicar a transposição do conjunto de valores da variável *Petal.Width* com o uso da função `t( )`. O gráfico do histograma está ilustrado na Figura A-10.

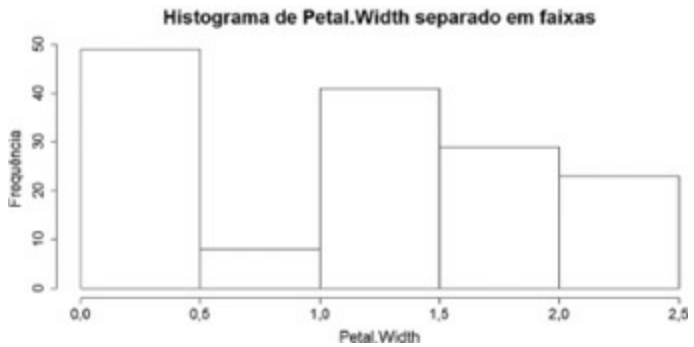


Figura A-10: Histograma para o atributo descritivo *Petal.Width* do conjunto de dados Iris, separado em cinco faixas

A.10. Acessando dados do Twitter

O acesso aos dados do Twitter, e de grande parte das redes sociais, só é permitido para aplicações cadastradas. O endereço para esse cadastro é https://apps.twitter.com/, e para o caso de usuários do Twitter, as mesmas credenciais de cadastro já em uso podem ser reaproveitadas. O cadastro da aplicação permite o recebimento de uma chave, um segredo, um *token* e uma senha que devem ser fornecidos no momento do acesso aos dados do Twitter. O acesso é feito por meio do *oAuth open authorization*, método-padrão para autorizar usuários a acessar aplicações de terceiros usando suas credenciais em uma rede social (http://oauth.net/). O uso desse recurso desobriga uma aplicação de manter, em seu sistema, o gerenciamento de usuários, permitindo que o credenciamento seja realizado pelas próprias credenciais do usuário cadastradas no Twitter. Esse é um recurso vastamente usado, já que é comum, hoje em dia, que diferentes aplicações façam uso do sistema de gerenciamento de usuários das redes sociais, como Facebook, LinkdIn e o próprio Twitter. Como aqui o interesse é acessar os dados do Twitter, no cadastro da aplicação, já devem ser solicitadas essas credenciais temporárias.

Para realizar a captura dos dados do Twitter via R, é necessário instalar e carregar os pacotes ***twitteR***[9] (Gentry, 2015), ***ROAuth***[10] (Gentry e Lang, 2015) e ***RCurl*** [11] (Lang, 2015). Para isso, os seguintes comandos devem ser executados:

```
> install.packages("twitteR")
> library("twitteR")
> install.packages("ROAuth")
> library("ROAuth")
> install.packages("RCurl")
> library("RCurl")
```

Após inicialização dos pacotes, a configuração da autenticação com todas as credenciais necessárias precisa ser realizada com uso da função

`setup_twitter_oauth( )`, ou seja:

```
setup_twitter_oauth (
consumer_key = 'digite aqui a chave da aplicação',
consumer_secret = 'digite aqui o segredo da
aplicação',
access_token = 'digite aqui o token de acesso',
access_secret = 'digite aqui a senha de acesso'
)
```

Após execução de autenticação, aparecerá, no console do R, a seguinte mensagem, para que seja feita a escolha entre armazenar as credenciais de acesso em um arquivo local ou em uma variável de sessão (nesse exemplo, escolha 2).

```
[1] "Using direct authentication"
Use a local file to cache OAuth access credentials between R
sessions?
1: Yes
2: No
```

Após a execução desses comandos, a conexão com o Twitter está estabelecida, e é possível então realizar pesquisas por palavras-chave, *hashtags* (caractere reservado "#", usado com uma *tag* para rotular postagens) e *tweets* de um usuário. A consulta por palavras-chave e *hashtags* é feita pela função `searchTwitter( )`, cuja sintaxe e principais parâmetros são explicados no Quadro A-1.

Quadro A-1: Sintaxe e parâmetros para a função SEARCHTWITTER

Sintaxe para a consulta por texto (palavra-chave ou *hashtag*) no Twitter
Usando a função `searchTwitter( )` do pacote *TwitteR*
Acessando os dados do Twitter: `S <- searchTwitter(texto de pesquisa, número de tweets, idioma, desde a data, geolocalização)` • `texto de pesquisa` é a palavra-chave ou *hashtag* que se deseja encontrar. Para a pesquisa de mais de um

termo, use o operador “+”.

- `número de tweets` é o total de resultados que se deseja recuperar.
- `idioma` restringe a pesquisa para um idioma específico, dado por um código ISO 639-1.
- `desde a data` é a data inicial da pesquisa, definida no formato ano - mês - dia (aaaa-mm-dd).
- `geolocalização` restringe a consulta para determinada latitude, longitude e raio.

A função retorna uma variável `S` que contém os *tweets* ou publicações recuperadas.

Por exemplo, imagine que se deseja capturar os *tweets* que tenham o termo *restaurante* em um total máximo de *100 postagens*, publicados em idioma *português*, desde *fevereiro de 2014* e na região da *Pompeia*, em *São Paulo*. A instrução para essa captura é:

```
> S <- searchTwitter("restaurante",n=100, lang="pt", since='2011-
01-01', geocode='-23.5372923,-46.6731866,50km')
```

Alternativamente, quando se deseja pesquisar as postagens feitas por um usuário específico, a captura pode ser feita com a função `userTimeline( )`, cujos sintaxe e parâmetros são descritos no Quadro A-2.

Quadro A-2: Sintaxe e parâmetros para a função USERTIMELINE

Sintaxe para a consulta por um usuário específico
Usando a função `userTimeline( )` **do pacote *TwitteR***
Acessando os dados de um usuário específico: `T <- userTimeline(usuário de pesquisa, número de tweets)` • `usuário de pesquisa` pode ser um caractere ou um objeto (nome ou *hashtag* do usuário). • `número de tweets` é o total de resultados que se deseja recuperar. A função retorna uma variável *T* com as postagens realizadas por `usuário de pesquisa`.

Por exemplo, considere que se deseja saber o que o usuário *Restaurant Week SP*, ou *@SaoPauloRW*, publicou nos primeiros 300 *tweets*. A instrução para essa captura é:

```
> T <- userTimeline('@SaoPauloRW',n=300)
```

Os *tweets* são recuperados em um formato que contempla a postagem propriamente dita e várias outras informações, como data de criação, número de *retweets*, latitude e longitude. Esse formato consiste em uma sequência de dados, separados por seus identificadores. Isso significa que, antes de realizar qualquer análise, é preciso organizar esses dados para selecionar os de interesse. Tal organização consiste em converter os dados da variável `T` (os *tweets* recuperados) em um *data.frame* (`as.data.frame( ))`, sendo cada *tweet* uma linha, e cada valor, uma coluna. A instrução para realizar esse procedimento é:

```
> df <- do.call("rbind", lapply(T, as.data.frame))
```
[12]

Agora, com os dados organizados, é possível analisar apenas dados do tipo texto, identificados no *data.frame* por `text`. Para separar esses dados, colocando-os em um vetor, use:

```
> tweets <- VectorSource(df$text)
```

A análise desses *tweets* pode ser executada como ilustrado no exemplo prático do Capítulo 2. A seguir, apresenta-se um recurso que pode ser usado para a visualização dos dados contidos nas postagens.

A.11. Construindo nuvens de palavras

Uma nuvem de palavras , ou *word cloud*, é uma representação visual para dados no formato de texto (geralmente palavras). As palavras, ou termos, podem ser visualizadas com diferentes tamanhos ou cores. O tamanho representa a frequência com que cada termo aparece em um documento (ou em um contexto qualquer, representado por uma unidade frequentemente denominada vocabulário ou *corpus*). A cor, por outro lado, pode significar o resultado de uma categorização, por exemplo, positividade ou negatividade de cada termo.

A geração de uma nuvens de palavras em R é bastante simples. Um dos pacotes disponíveis para esta visualização chama-se ***wordcloud*** (Fellows, 2014). A geração da nuvem é feita pela função `wordcloud()`, cujos sintaxe e parâmetros são apresentados no Quadro A-3.

Quadro A-3: Sintaxe e parâmetros para a função WORDCLOUD

Sintaxe para a impressão da nuvem de palavras
Usando a função `wordcloud( )` **do pacote *wordcloud***
Criando uma nuvem de palavras: `wordcloud(termos, frequência, frequência mínima)` • `termos` são os termos (ou palavras) presentes no *corpus* e organizados em uma matriz de frequência de termos. • `frequência` é a ocorrência de cada termo na matriz de frequência. • `frequência mínima` é um filtro que permite selecionar um limite inferior para imprimir o termo na nuvem. A função gera um gráfico referente à nuvem de palavras.

Como é possível perceber, a função espera que os dados estejam armazenados em uma matriz, que deve ter a estrutura ilustrada na Figura A-11, na qual os documentos são as postagens de *tweets* (para execução desse exemplo, use a variável `corpus_tf`, criada no exemplo prático do Capítulo 2).

Logo, se os dados não estão em uma matriz, é preciso transformá-los da seguinte maneira:

```
> m <- as.matrix(corpus_tf)
```

		DOCUMENTOS			
		D1	D2	D3	D4
TERMOS	restaurante	1	1	0	0
	aniversário	1	1	0	1
	saideira	0	1	1	0
	bistrot	0	1	1	1

Figura A-11: Estrutura esperada para uso na função *wordcloud( )*

Com os dados em uma matriz, as frequências das palavras precisam ser calculadas, e o resultado pode ser ordenado para facilitar a visualização. Esse procedimento pode ser executado por meio da seguinte instrução:

```
> v <- sort(rowSums(m), decreasing=TRUE)
```

E, finalmente, a função `wordcloud( )` pode ser executada como apresentado na instrução a seguir, e o resultado pode ser verificado no Capítulo 2.

```
> install.packages("wordcloud")
> library("wordcloud")
> wordcloud(names(v),v,min.freq=10)
```

REFERÊNCIAS

Agirre E.; Edmonds, P. Word Sense Disambiguation: Algorithms and Applications. 1 ed., Springer Netherlands, 2007, 366p.

Agrawal, R; Imielinski, T; Swami, A. “Mining Association Rules between Sets of Items in Large Databases”. Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data (SIGMOD’93), Washington, DC, maio de 1993, 207-216p.

Agrawal, R.; Srikant, R. “Fast Algorithm for Mining Association Rules in Large Databases”. Proceedings of the 20th International Conference on Very Large Data Bases. Morgan Kaufmann Publishers Inc., San Francisco, CA, 1994, 487-499p.

Agrawal, R.; Srikant, R. “Mining Sequential Patterns”. Proceedings of the 11th International Conference on Data Engineering (ICDE’95), Philip S. Yu e Arbee L. P. Chen (orgs.), 1995, Taipei, Taiwan. Washington, DC: IEEE Computer Society, 1995, 3-14p.

Agrawal, R.; Shafer, J. C. “Parallel Mining of Association Rules”. IEEE Transactions on Knowledge and Data Engineering, v. 8, no 6, dez de 1996, 962-969p.

Aguinis, H.; Forcum, L. E.; Joo, H. “Using Market Basket Analysis in Management Research”, Journal of Management, v. 39, no 7, nov de 2013,

1799-1824p.

Ankerst et al. “OPTICS: Ordering Points to Identify the Clustering Structure”. Proceedings of 1999 ACM-SIGMOD International Conference on Management of Data (SIGMOD’99), Philadelphia, PA, jun de 1999, 49-60p.

Alcaim, A.; Oliveira, C. A. S. Fundamentos do processamento de sinais de voz e imagem. 1. ed. Rio de Janeiro: Interciência, 2011, 302p.

Alfons, A. “cvTools: Cross-Validation Tools for Regression Models”. R package version 0.3.2., 2012. Disponível em: http://CRAN.R-project.org/package=cvTools.

Arel, I.; Rose, D. C.; Karnowski, T. P.“Deep Machine Learning – A New Frontier in Artificial Intelligence Research [Research Frontier]”. IEEE Computational Intelligence Magazine, v. 5, no 4, nov de 2010, 13-18p.

Arthur, D; Vassilvitskii, S. “k-means++: the Advantages of Careful Seeding”. Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA’07). Society for Industrial and Applied Mathematics, Philadelphia, PA, 2007, 1027-1035p.

Attneave, F. Applications of Information Theory to Psychology: A Summary of Basic Concepts, Methods, and Results. 1 ed. Nova York: Holt, Rinehart & Winston, 1959, 120p.

Bach, F. R. “Sparse Methods for Machine Learning: Theory and Algorithms”. Disponível em <http://videolectures.net/nips09_bach_smm/>.

Bellhouse, D. R. “The Reverend Thomas Bayes, FRS: A Biography to Celebrate the Tercentenary of his Birth”. Statistical Science, v. 19, no 1, fev de 2004, 3-43p.

Bergmeir, C.; Benitez, J. M. “Neural Networks in R Using the Stuttgart

Neural Network Simulator: RSNNS". In: Journal of Statistical Software, v. 46, no 7, jan de 2012, 1-26p.

Beyan, C.; Fisher, R. "Classifying Imbalanced Data Sets using Similarity Based Hierarchical Decomposition". Pattern Recognition, v. 48, no 5, maio de 2015, 1653-1672p.

Bezdek, J. C.; Dunn, J. C. "Optimal Fuzzy Partitions: A Heuristic for Estimating the Parameters in a Mixture of Normal Distributions". IEEE Transactions on Computers, v. C-24, no 8, ago de 1975, 835-838p.

Brijs et al."Using Association Rules for Product Assortment Decisions: A Case Study". Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'99). ACM, San Diego, CA, 1999, 254-260p.

Bong et al. "Selection and Aggregation of Interestingness Measures: A review". Journal of Theoretical and Applied Information Technology, v. 59, no 1, jan de 2014, 146-166p.

Breiman et al. Classification and Regression Trees. 1 ed. Belmont, CA: Wadsworth Group, 1984, 368p.

Brun et al. "Model-Based Evaluation of Clustering Validation Measures". Pattern Recognition, v. 40, no 3, mar de 2007, 807-824p.

Buneman et al. "Adding Structure to Unstructured Data". Proceedings of the 6th International Conference on Database Theory (ICDT'97), Lecture Notes in Computer Science, v. 1186, Springer, 1997, 336–350p.

Chambers, J. M. "Linear Models". Chapter 4. In: Chambers, J. M.; Hastie, T. J (orgs.). Statistical Models in S. Pacific Grove. 1 ed. Califórnia: Wadsworth & Brooks/Cole, 1992, 95-144p.

Cherfi, H.; Napoli, A.; Toussaint, Y. "Towards a Text Mining Methodology

Using Association Rule Extraction". Soft Computing, v. 10, no 5, mar de 2006, 431-441p.

Copi, I. M. Introdução à Lógica. 3 ed. São Paulo: Mestre Jou. 1981.

Costa, J. A. F. Classificação automática e análise de dados por redes neurais auto-organizáveis. Tese (Doutorado em Engenharia Elétrica) – Faculdade de Engenharia Elétrica e de Computação, Universidade Estadual de Campinas, Campinas, 1999.

Cristianini, N.; Shawe-Taylor, J. An Introduction to Support Vector Machines and Other Kernel-based Learning Methods.1ed., Cambridge, UK: Cambridge University Press, 2000, 189p.

Dai et al. "Boosting for Transfer Learning". In: Ghahramani, Z. (org.) Proceedings of the 24th International Conference on Machine learning (ICML'07). ACM, Nova York, 2007, 193-200p.

Dalton, L.; Ballarin, V.; Brun, M. Clustering Algorithms: "On Learning, Validation, Performance, and Applications to Genomics". Current Genomics, v. 10, no 6, set de 2009, 430-445p.

Damásio, A. R. O erro de Descartes: emoção, razão e o cérebro humano. 3 ed. São Paulo: Companhia da Letras, 2012, 264p.

Davies, D. L.; Bouldin, D. W. "A Cluster Separation Measure". IEEE Transactions on Pattern Analysis and Machine Intelligence, v.1, no 2, abr de 1979, 224–227p.

Desgraupes, B. "Clustering Indices. Package cluster Crit for R". University Paris Ouest – Lab Modal'X, 2013.

Devlin, K. Infoscience: Turning Information into Knowledge. Nova York: W. H. Freeman, 1999, 240p.

Deza, M. M.; Deza, E. Encyclopedia of Distances. Berlin, DE: Springer, 2009, 583p.

Dias et al. “Hand Movement Recognition for Brazilian Sign Language: A Study Using Distance-Based Neural Networks”. Proceedings of the 2009 International Joint Conference on Neural Networks (IJCNN’09), Atlanta, GA: IEEE Press, Piscataway, NJ: IEEE Press 2009,697-704p.

Dietterich, T.; Kearns, M.; Mansour, Y. “Applying the Weak Learning Framework to Understand and Improve C4.5”. In: Proceedings of the 13th International Conference on Machine Learning, San Francisco: Morgan Kaufmann, 1996, 96-104p.

Dragulescu, A. A. “xlsx: Read, write, format Excel 2007 and Excel 97/2000/XP/2003 files, R package version 0.5.7. 2014”. Disponível em: https://cran.r-project.org/web/packages/xlsx/index.html

Duncan T. L. and the CRAN team.“RCurl: General Network (HTTP/FTP/...) Client Interface for R. R package version 1.95-4.6. 2015”. Disponível em:http://CRAN.R-project.org/package=RCurl.

Dunn, J. C. “A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters”. Journal of Cybernetics, v. 3, no 3, jan de 1973, 32-57p.

Ester et al. “A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise”. Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining (KDD’96), Portland, Oregon: AAAI Press, 1996, 226-231p.

Fausett, L. Fundamentals of Neural Networks: Architectures, Algorithms and Applications. 1 ed. Englewood Cliffs, NJ: Pearson, 1993, 461p.

Fawcett, T. “An Introduction to ROC Analysis”. Pattern Recognition Letters,

v. 27, no 8, jun de 2006, 861-874p.

Fayyad, U.; Irani, D. B. “The Attribute Selection Problem in Decision Tree Generation”. Proceedings of 10th National Conference on Artificial Intelligence, Cambridge, MA: AAAI Press/ MIT Press, 1992, 104-110p.

Fayyad et al. Advances in Knowledge Discovery and Data Mining. Menlo Park: AAAI Press, 1996. 560p.

Feinerer, I.; Hornik, K.“tm: Text Mining Package. R package version 0.6-1. 2015”. Disponível em: http://CRAN.R-project.org/package=tm

Feinerer, I.; Hornik, K.; Meyer, D.“Text Mining Infrastructure in R”. Journal of Statistical Software, v. 25, no 5, 2015, 1-54p.

Feldman, R.; Sanger, J. The Text Mining Handbook: Advanced Approaches in Analyzing Unstructured Data. Cambridge: Cambridge University Press, 2007, 410 p.

Fellows, I. “Wordcloud: Word Clouds. R package version 2.5, 2014”. Disponível em: http://CRAN.R-project.org/package=wordcloud.

Ferri et al. “Learning Decision Tress Using the Area under the ROC Curve”. Proceedings of the 19th International Conference on Machine Learning (ICML’02). San Francisco, CA: Morgan Kaufmann, 2002,139-146p.

Fisher, R. A. The Use of Multiple Measurements in Taxonomic Problems. Contributions to Mathematical Statistics. Nova York: John Wiley, 1950.

Fisher et al. “Hypermedia Image Processing Reference, 2000”. Disponível em: http://homepages.inf.ed.ac.uk/rbf/HIPR2/index.htm

Freund, J. E. Estatística aplicada: Economia, Administração e Contabilidade. 11 ed. Porto Alegre, RS: Bookman, 2006, 536p.

Frey, P. W.; Slate, D. J. “Letter Recognition Using Holland-style Adaptive

Classifiers”. Machine Learning, v. 6, no 2, mar de 1991.

Friedman J. H. “A Recursive Partitioning Decision Rule for Nonparametric Classifiers”. IEEE Transactions on Computers, v. C-26, no 4, abr de 1977, 404-408p.

Fritsch, S.; Guenther, F.; following earlier work by Suling, M. “neuralnet: Training of neural networks. R package version 1.32, 2012”. Disponível em: http://CRAN.R-project.org/package=neuralnet

Gelfand et al. “An Iterative Growing and Pruning Algorithm for Classification Tree Design”. IEEE Transaction on Pattern Analysis and Machine Intelligence, v. 13, no 2, fev 1991, 163-174p.

Gentry, J. “twitteR: R Based Twitter Client. R package version 1.1.8. 2015”. Disponível em: http://CRAN.R-project.org/package=twitteR.

Gentry, J; Lang, D. T. “ROAuth: R Interface for OAuth. R package version 0.9.6. 2015”. Disponível em: http://CRAN.R-project.org/package=ROAuth.

Goldman, R.; McHugh, J.; Widom, J. “From Semistructured Data to XML: Migrating the Lore Data Model and Query Language”. Proceedings of the 2nd International Workshop on the Web and Databases (WebDB’99), Philadelphia, Pennsylvania, jun de 1999.

Gonzalez, R. C.; Woods, R. E. Processamento digital de imagens. 3 ed. São Paulo: Addison Wesley BRA, 2010, 624p.

Guha, S.; Rastogi, R.; Shim, K. ROCK: “A Robust Clustering Algorithm for Categorical Attributes”. Information Systems, v. 25, no 5, 2000, 345-366p.

Hahsler, M.; Chelluboina, S.“arulesViz: Visualizing Association Rules and Frequent Itemsets. R package version 1.0-0, 2014”. Disponível em: http://CRAN.R-project.org/package=arulesViz.

Hahsler, M.; Grün, B.; Hornik, K. “arules – A Computational Environment for Mining Association Rules and Frequent Item Sets”. Journal of Statistical Software, v. 14, no 15, 2005, 1–25p.

Hahsler et al.“arules: Mining Association Rules and Frequent Itemsets. R package version 1.1-6, 2014”. Disponível em: http://CRAN.R-project.org/package=arules.

Hair et al. Multivariate Data Analysis. 4ed., New Jersey: Prentice-Hall Inc., 1995.

Halkidi, M.; Batistakis, Y.; Vazirgiannis, M. “On Clustering Validation Techniques”. Journal of Intelligent Information Systems, v. 17, no 2-3, dez de 2001, 107-145p.

Han, J.; Kamber, M.; Pei, J. Data Mining: Concepts and Techniques. 3 ed. Waltham, MA: Morgan Kaufmann Series in Data Management Systems, 2011, 744p.

Han, J.; Pei, J.; Yin, Y. “Mining Frequent Patterns without Candidate Generation”. Proceedings of the ACM-SIGMOD International Conference on Management of Data. Dallas: ACM Press, 2000, 1–12p.

Hansen, L. K.; Salamon, P. “Neural Network Ensembles”. IEEE Transactions Pattern Analysis and Machine Intelligence, v. 12, no 10, out de 1990, 993-1001p.

Haykin, S. O. Neural Networks and Learning Machines. 3 ed. Upper Saddle River, New Jersey: Prentice-Hall, 2008, 906p.

Hennig, C. “fpc: Flexible procedures for clustering. R package version 2.1-9. 2014”. Disponível em: http://CRAN.R-project.org/package=fpc.

Hinneburg, A.; Keim, D. A. “An Efficient Approach to Clustering in Large Multimedia Databases with Noise”. Proceedings of 1998 International

Conference on Knowledge Discovery and Data Mining (KDD'98), Nova York, NY, ago de 1998, 58-65p.

Huang, Z. "Extensions to the K-Means Algorithm for Clustering Large Data Sets with Categorical Values". Data Mining and Knowledge Discovery, v. 2, no 3, 1998, 283-304p.

Inmon, W. H. Building the Data Warehouse, 4 ed. New York, NY: Wiley India Pvt. Limited, 2005, 576p.

Jacobs et al. "Adaptive Mixtures of Local Experts". Neural Computation, v. 3, no 1, mar de 1991, 79-87p.

Joachims, T. "A Probabilistic Analysis of the Rocchio Algorithm with TFIDF for Text Categorization". Computer Science Technical Report CMU-CS-96-118, Carnegie Mellon University, 1996.

Jolliffe, I. T. Principal Component Analysis, 2 ed. Nova York, NY: Springer, 2002, 519p.

Jordan, M. I.; Jacobs, R. A. "Hierarchical Mixtures of Experts and the EM Algorithm". Neural Computation, vol. 6, no 2, mar de 1994, 181-214p.

Kaynak, C. "Methods of Combining Multiple Classifiers and Their Applications to Handwritten Digit Recognition". (MSc Thesis). Institute of Graduate Studies in Science and Engineering, Bogazici University, 1995.

Kaski, S.; Kangas, J.; Kohonen, T. "Bibliography of Self-Organizing Map (SOM) Papers: 1981-1997".Neural Computing Surveys, v. 1, 1998, 102-350p.

Kaufman, L.; Rousseeuw, P. J. Finding Groups in Data: An Introduction to Cluster Analysis. Nova York: John Wiley & Sons, 1990.

Kearns, M.; Mansour Y. "On the Boosting Ability of to-down Decision Tree

Learning Algorithms". Journal of Computer and Systems Sciences, v. 58, no 1, fev de 1999, 109-128p.

Kimball, R.; Margy R. The Data Warehouse Tookit: The Complete Guide to Dimensional Modeling. 3 ed. Indianopolis, Indiana: John Wiley & Sons Inc., 2002, 464p.

Kiviluoto, K."Topology Preservation in Self-Organizing Maps". Proceedings of IEEE International Conference on Neural Networks, v.1, jun de 1996, 294-299p.

Koch, R. Princípio 80/20. Rio de Janeiro: Rocco, 2000. 270p.

Kohavi, R. "Scaling Up the Accuracy of Naive-Bayes Classifiers: A Decision-Tree Hybrid". Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (SIGKDD'96), 1996, 202-207p.

Kohonen, T. "Self-Organized Formation of Topologically Correct Feature Maps". Biological Cybernetics, v. 43, no 1, jan de 1982, 59-69p.

Kohonen, T. Self-organizing and Associative Memory. 2 ed. Nova York, NY: Springer-Verlag, 1997, 312p.

Kohavi, R.; John, G. H. "Wrappers for Feature Subset Selection". Artificial Intelligence, vol. 97, no 1-2, 1997, 273-324p.

Laudon, K. C.; Laudon, J. P. Sistemas de informações gerenciais. 11 ed. São Paulo: Pearson Brasil, 2001, 504p.

Lichman, M. "UCI Machine Learning Repository". Irvine, CA: University of California, School of Information and Computer Science, 2013. Disponível em: http://archive.ics.uci.edu/ml.

Lima, C. A. M. "Comitê de máquinas: uma abordagem unificada empregando

máquinas de vetores-suporte”. 2004. 342f. Tese (Doutorado em Engenharia Elétrica) – Faculdade de Engenharia Elétrica e de Computação, Universidade Estadual de Campinas, Campinas, 2004.

Liu, H.; Motoda, H. Instance Selection and Construction for Data Mining. Series: The Springer International Series in Engineering and Computer Science. Norwell, MA: Springer, 2001, 416p.

Lim, T-S.; Loh, W.-Y.; Shih, Y-S. “A Comparison of Prediction Accuracy, Complexity, and Training Time of Thirty-three Old and New Classification Algorithms”. Machine Learning Journal, v. 40, no 3, set de 2000, 203-228p.

Lloyd, S. P. “Least Squares Quantization in PCM”. IEEE Transaction on Information Theory, v. 28, no 2, mar de 1982, 128-137p.

Lohr, S. L. Sampling: Design and Analysis. 2. ed. Boston, MA: Cengage Learning, 2010, 608p.

Lopes et al. “Visual Text Mining using Association Rules”. Computers and Graphics, v. 31, no 3, jun de 2007, 316-326p.

MacQueen, J. “Some Methods for Classification and Analysis of Multivariate Observations”. Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability. Berkeley: University of California Press, v. 1, 281-297p., 1967.

Menezes, P. B. Linguagens formais e autômatos. 6ed., v. 3, série Livros Didáticos Informática da UFRGS, Porto Alegre: Bookman, 2011.

Meyer et al. “Misc Functions of the Department of Statistics (e1071), TU Wien. R package version 1.6-4, 2014”. Disponível em: http://CRAN.R-project.org/package=e1071

Milborrow, S. “rpart.plot: Plot rpart Models. An Enhanced Version of plot.rpart. R package version 1.5.1, 2014”. Disponível em: http://CRAN.R-

project.org/package=rpart.plot.

Moro, S. Cortez, P.; Rita, P. “A Data-Driven Approach to Predict the Success of Bank Telemarketing”. Decision Support Systems, Elsevier, v. 62, jun de 2014, 22-31p.

Nieweglowski, L. “clv: Cluster Validation Techniques. R package version 0.3-2.1”, 2013. Disponível em: http://CRAN.R-project.org/package=clv

Oja, M.; Kaski, S.; Kohonen T. “Bibliography of Self-Organizing Map (SOM) Papers: 1998-2001 Addendum”. Neural Computing Surveys, v. 3, 2002, 1-156p.

Ostrovsky et al. “The Effectiveness of Lloyd-Type Methods for the k-Means Problem”. Proceedings of 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS’06), out de 2006, 165-176p.

Pakhira, M. K.; Bandyopadhyay, S.; Maulik, U. “Validity Index for Crisp and Fuzzy Clusters”. Pattern Recognition, v. 37, no 3, mar de 2004, 487-501p.

Pan, S. J.; Yang, Q. “A Survey on Transfer Learning”. IEEE Transactions on Knowledge and Data Engineering, v. 22, no 10, out de 2010, 1345-1359p.

Piatetsky-Shapiro.“Knowledge Discovery in Real Databases: A Report on the IJCAI-89 Workshop”. AI Magazine, v. 11, no 5, jan de 1991, 68-70p. Disponível em: http://www.kdnuggets.com/meetings/kdd89/kdd-89-report-aimag.html.

Porter, M. “An Algorithm for Suffix Stripping”. Program, v. 14, no 3, 1980, 130-137p.

Prajapati, V. Big Data Analytics with R and Hadoop. Birmingham, UK: Packt Publishing Ltd., 2013, 238p.

Prandoni, P.; Vetterli, M. Signal Processing for Communications. Boca

Raton: EPFL Press, 2008, 371p.

Prati, R. C.; Batista, G. E. A. P. A.; Monard, M. C. “Curvas ROC para avaliação de classificadores”. IEEE Latin America Transactions, v. 6, no 2, jun de 2008, 215-222p.

Quinlan, J. R. C4.5: Programs for Machine Learning. San Francisco: Morgan Kaufmann, 1993, 302p.

_____. “Induction of Decision Trees”. Machine Learning, v. 1, no 1, mar de 1986, 81-106p.

_____“Simplifying Decision Trees”. International Journal of Man-Machine Studies, v. 27, no 3, set de 1987, 221-234p.

Quinlan, J. R. “Decision Trees and Multivalued Attributes”. In: Hayes, J. E.; Michie, D.; Richards, J. (orgs.). Machine Intelligence, v. 11. Oxford: Oxford Univ. Press, 1988, 305-318p.

R Core Team.“R: A Language and Environment for Statistical. R Foundation for Statistical”, Viena, 2015. Disponível em: https://www.R-project.org.

Rand, W. M. “Objective Criteria for the Evaluation of Clustering Methods”. Journal of the American Statistical Association, v. 66, no 336, dez de 1971, 846 – 850p.

Rijsbergen, C. J. Van. Information Retrieval. 2 ed., Londres: Butterworths, 1979, 208p.

Rocha et al. “Tutorial sobre Fuzzy-C-Means e Fuzzy Learning Vector Quantization: abordagens híbridas para tarefas de agrupamento e classificação”. Revista de Informática Teórica e Aplicada, v. 19, no 1, 2012, 120-163p.

Rokach, L.; Maimon, O. “Data Mining with Decision Trees: Theory and

Applications.” Series in Machine Perception and Artificial Intelligence. v. 69, Cingapura: World Scientific, 2008, 264 p.

Rousseeuw, P. J. Silhouettes: “A Graphical Aid to the Interpretation and Validation of Cluster Analysis”. Journal of Computational and Applied Mathematics, v. 20, no 1, nov de 1986, 53-65 p.

Rounds, E. “A Combined Non-parametric Approach to Feature Selection and Binary Decision Tree Design”. Pattern Recognition, v. 12, no 5, 1980, 313-317p.

Rumelhart, D. E; Hinton, G. E.; Williams, R. J. “Learning Representations by Back-propagation Erros”. Nature, v. 323. no 6088, out de 1986, 533-536p.

Sarawagi, S.; Thomas, S.; Agrawal, R. “Integrating Mining with Relational Database Systems: Alternatives and Implications”. Proceedings of the 1998 ACMSIGMOD International Conference on Management of Data (SIGMOD’98), Seattle, WA, 1998, 343-354p.

Schlimmer, J. S. “Concept Acquisition through Representational Adjustment (Technical Report 87-19)”. Doctoral Dissertation, Department of Information and Computer Science, Irvine: University of California, 1987.

Settles, B. “Active Learning Literature Survey”. Computer Sciences Technical Report 1648. University of Wisconsin–Madison, 2010, 65p.

Setzer, V. W. Os meios eletrônicos e a educação: uma visão alternativa. São Paulo: Editora Escrituras, Coleção Ensaios Transversais, v. 10, 2001, 288p.

Srikant, R.; Agrawal, R. “Mining Generalized Association Rules”. Proceedings of 21th International Conference on Very Large Data Bases (VLDB’95). Zurich, Switzerland. San Francisco, CA: Morgan Kaufmann Publishers Inc. 1995, 407-419p.

Tamhane A. C.; Dunlop, D. D. Statistics and Data Analysis: From

Elementary to Intermediate. 1 ed. Upper Saddle River, NJ: Pearson, 1999, 722p.

Tan, P-N; Steinbach, M. Kumar, V. Introduction to Data Mining. 1 ed. Boston, MA: Pearson, 2006, 769p.

Therneau, T.; Atkinson, B.; Ripley, B. “rpart: Recursive Partitioning and Regression Trees, R package version 4.1-8” Disponível em: http://CRAN.R-project.org/package=rpart, 2014.

Triola, M. F. Introdução à Estatística. 10 ed. Rio de Janeiro: LTC, 2008, 696p.

Tsoumakas, G.; Katakis, I. “Multi-label Classification: An Overview”. International Journal of Data Warehouse and Mining, v. 3, no 3, 2007, 1-13p.

Turban et al. Business Intelligence: um enfoque gerencial para a inteligência do negócio. Porto Alegre: Bookman, 2009, 254p.

Ultsch, A. “U*-matrix: A Tool to Visualize Clusters in High Dimensional Data”. University of Marburg, Department of Computer Science, Technical Report, no 36, dez de 2003.

Utgoff, P. E.; Clouse, J. A. “A Kolmogorov-Smirnoff Metric for Decision Tree Induction”. Technical Report, 96-3, Amherst, MA: University of Massachusetts, Department of Computer Science, 1996.

Venables, W. N.; Ripley, B. D. Modern Applied Statistics with S. 4 ed. Nova York: Springer, 2002, 498p.

Vinh, N. X.; Epps, J.; Bailey, J. “Information Theoretic Measures for Clusterings Comparison: Variants, Properties, Normalization and Correction for Chance”. Journal of Machine Learning Research, v. 11, out de 2010, 2837-2854p.

Wehrens, R.; Buydens, L.M.C. “Self-and Super-organizing Maps in R: The Kohonen Package”. Journal of Statistical Software, v. 21, no 5, out de 2007.

White. Hadoop: The Definitive Guide: Storage and Analysis at Internet Scale. 4 ed., 1005 Gravenstein Highway North: O’Reilly Media, 2015, 756p.

Wickham, H. “Scales. Scale Functions for Visualization. R package version 0.3.0. 2-15, 2015”. Disponível em: http://CRAN.R-project.org/package=scales.

Wickham, H. “Reshaping Data with the Reshape Package”. Journal of Statistical Software. v. 21, no 12, nov de 2007, 1-20p.

Wickham, H.; Chang, W. Devtools: “Tools to Make Developing R Packages Easier. R Package version 1.9.1, 2015”. Disponível em: http://CRAN.R-project.org/package=devtools.

Witten, I. H.; Frank, E.; Hall, M. A. Data Mining: Practical Machine Learning Tools and Techniques. 3 ed., Burlington, MA: Morgan Kaufmann Publishers, 2011, 629p.

Wilkinson, G. N.; Rogers, C. E. “Symbolic Descriptions of Factorial Models for Analysis of Variance”. Journal of the Royal Statistical Society. Series C (Applied Statistics), v. 22, no 3, 1973, 392–399p.

Zaki et al.. “New Algorithms for Fast Discovery of Association Rules”. Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining. Newport Beach, CA. Menlo Park, CA: AAAI Press, 1997, 283-286p.

Zaki, M. J. “Scalable Algorithms for Association Mining”. IEEE Transactions on Knowledge and Data Engineering, v. 12, no 3, 2000, 372-390p.

Zhang, T; Ramakrishnan, R; e Livny, M. “BIRCH: An Efficient Data

Clustering Method for Very Large Databases". In: Proceedings of 1996 ACM-SIGMOD International Conference on Management of Data, Montreal, jun de 1996, 103-114p.

Zhang, Z.; Zhang, R. Multimedia Data Mining: A Systematic Introduction to Concepts and Theory. 1 ed., Boca Raton, FL: Chapman and Hall/CRC, 2008, 320p.

NOTAS

Capítulo 1

[1] Há ainda o tipo de dado chamado “semiestruturado”. Nesse caso, algum nível de organização é imposto ao conteúdo não estruturado, como um texto, de forma a representá-lo e a facilitar seu processamento por meio de comparações, buscas e alterações. Estruturas de dados do tipo grafos podem ser usadas para este fim (Buneman *et al.*, 1997). A linguagem de marcação XML (*eXtensible Markup Language*), por exemplo, pode ser considerada um avanço na melhoria da organização de dados semiestruturados (Goldman, McHugh e Widom, 1999). Dados que apresentam algum nível de estrutura também são considerados semiestruturados. Como exemplo, há os conteúdos disponibilizados na Web acompanhados de *tags* (marcações que trazem algum tipo de informação descritiva – metadado – sobre o conteúdo).

[2] Procedimentos indutivos são aqueles decorrentes do raciocínio indutivo. O raciocínio indutivo parte de fatos particulares para conclusões gerais e é capaz de gerar conhecimento novo – ou seja, extrair conhecimento implícito dos fatos, não ainda conhecido. Este tipo de raciocínio gera conclusões que podem ser questionadas, embora seja possível conferir sua plausibilidade.

[3] Para ter ideia da importância que a área de Inteligência de Negócios representa hoje no mercado mundial, atente-se ao fato de que grandes empresas, como Microsoft, Oracle, SAS, IBM e SAP, apresentam soluções proprietárias para este tipo de aplicação.

[4] http://www.gartner.com/technology/about.jsp

[5] http://www.gartner.com/technology/research/methodologies/hype-cycles.jsp

[6] http://www.gartner.com/technology/research/methodologies/hype-cycle.jsp

[7] http://www.gartner.com/newsroom/id/1447613

[8] http://www.gartner.com/newsroom/id/1763814

[9] http://www.gartner.com/newsroom/id/2124315

[10] http://www.gartner.com/newsroom/id/2575515

[11] http://www.gartner.com/newsroom/id/2819918

[12] Eventualmente, os termos usados nos gráfico Hype Cycle podem variar embora estejam altamente relacionados, o que faz com que um tópico possa aparecer com diferentes nomenclaturas.

Capítulo 2

[1] Uma ressalva deve ser feita para o coeficiente de dispersão, que pode ser usado para comparar dois conjuntos de valores.

[2] A escolha da medida estatística a ser usada deve ser realizada com base no estudo do atributo. A média pode ser influenciada por valores *outliers;* a mediana num conjunto de valores contínuos não é, geralmente, recomendada. Como ilustração, a mediana e a média foram aplicadas, nesse exemplo, considerando o contexto dos valores já presentes no conjunto de dados.

[3] Essa decisão foi tomada com base apenas na análise dos valores do atributo “salário”. Alternativamente, se a análise de todos os demais atributos fosse considerada, poder-se-ia obter uma precisão maior no preenchimento do valor ausente.

[4] Novamente, foi considerado apenas o contexto do atributo para tomar a decisão. Se o contexto de toda a base de dados fosse usado, o valor poderia ser alterado para 100.000,00, já que o restaurante em questão parece ser maior e melhor que os demais em vários quesitos.

[5] A distância euclidiana é usada nesse exemplo para ilustrar uma operação que mensura a relação existente entre exemplares. Algoritmos de classificação e de agrupamento frequentemente aplicam medidas de distância para mensurar a relação existente entre os exemplares.

[6] Na construção do arquivo .csv para leitura no R, os valores faltantes devem ser representados como espaços vazios, e não por meio do caractere “?”.

[7] A média e a mediana, aqui, diferem do apresentado no resultado da função `summary( )`, pois essa função considera os valores inconsistentes. Para desconsiderar o valor inconsistente, no caso do atributo “lucratividade”, execute: `d$lucratividade[d$lucratividade==150000] <- mean(c(d$lucratividade[1],d$lucratividade[3:11]))`. Um procedimento semelhante deve ser executado para o atributo “capacidade”.

[8] Dados de outra natureza, como imagem, vídeo ou som, estão fora do escopo de discussão deste livro.

[9] A representação de um vetor, nesta seção, está sendo realizada por meio do “negrito” e “itálico”.

[10] Há algumas iniciativas em mineração de textos que não executam todas as fases do processo aqui descrito. Há também a discussão sobre a utilidade de cada uma dessas fases, que devem ser analisadas no contexto de cada aplicação desenvolvida.

[11] Há autores que consideram que o processo de geração de *tokens* não faz eliminações, ou seja, pontuação e dígitos são considerados *tokens*. Geralmente, tal abordagem é usada em análises que objetivam estudar elementos mais complexos de gramática e discurso.

[12] http://snowball.tartarus.org/texts/introduction.html

[13] http://snowball.tartarus.org/algorithms/portuguese/stemmer.html

[14] http://archive.ics.uci.edu/ml/datasets/Optical+Recognition+of+Handwritten+Digits – UCI – Machine Learning (Lichman, 2013) – um conjunto de dígitos manuscritos por 43 pessoas.

Capítulo 3

[1] Diz-se se tratar de uma abstração, pois não é possível usar mais de três eixos combinados em uma representação gráfica. Assim, o uso de vários eixos combinados 2 a 2 e dispostos em sequência indica que, na realidade, os exemplares se constituem como vetores localizados em um espaço de alta dimensão (> 3).

[2] Para conhecer mais sobre essas funções, veja CRAN – *Comprehensive R Archive Network*, disponível em http://cran.r-project.org.

[3] No contexto de redes neurais artificiais, cada exemplar do conjunto de dados usado no treinamento é convencionalmente chamado de padrão de treinamento. Cada padrão de treinamento é formado pelos atributos descritivos (representados nesta seção por x) e pelo rótulo associado a ele, quando tal rótulo existe.

[4] Para o caso de arquitetura com múltiplos neurônios na saída, a rotulação dos exemplares deve ser codificada em um vetor de saídas desejadas, em que cada elemento do vetor diz respeito à saída desejada de cada neurônio da camada de saída.

[5] A rede neural *Multilayer Perceptron* retorna um valor estimado para a saída y dentro de um domínio contínuo definido pela função de ativação dos neurônios. Esse valor precisa ser discretizado para atender às necessidades de uma tarefa de classificação, por exemplo, aproximando o valor estimado resultante para o inteiro mais próximo no domínio da função de ativação e associando esse inteiro a uma das classes envolvidas no problema. Ainda, por essa característica da saída da rede, ela pode ser aplicada para resolver o problema de predição numérica (regressão). Nesse último caso, a resposta esperada na regressão deve ser normalizada para o intervalo de operação da função de ativação.

[6] A representação em grafos de fluxo é baseada em Haykin (2008).

[7] O pacote e1071 é um dos pacotes disponíveis em R que implementa o *Naïve Bayes*. O pacote foi desenvolvido pelo TU Wien (*Vienna University of Technology*) e disponibiliza uma série de algoritmos de aprendizado de máquina, úteis para mineração de dados (Meyer *et al*., 2014).

[8] Veja como construir esse gráfico, usando R, no Apêndice.

[9] Esses subconjuntos são também conhecidos como *folds*. A estratégia de validação cruzada é comumente encontrada sob a nomenclatura de *k-fold cross validation.*

[10] Veja uma breve explicação sobre essa afirmativa em Han, Kamber e Pei (2011).

Capítulo 4

[1] Note que as distâncias entre pares de grupos são calculadas a partir da distância entre exemplares, portanto, a matriz de similaridade pode ser usada para o caso de “maior distância”, “menor distância” e

“distância média”. No caso da distância de centroides, é necessário encontrar um vetor médio (média dos exemplares) para cada grupo e calcular a distância entre cada par de vetores médios.

[2] A dispersão pode ser dada, por exemplo, pela maior distância entre quaisquer dois exemplares de um grupo.

[3] Diz-se que um exemplar B pertence à vizinhança de um exemplar A, se B está contido em uma região de tamanho T ao redor de A.

[4] Neste texto, o exemplar referencial é considerado no cômputo da densidade de sua vizinhança. Mas há, na literatura, definições que o excluem do cômputo. Essa decisão deve estar clara no início de execução do algoritmo de agrupamento.

[5] O símbolo *na* indica que o estado do exemplar é “não agrupado”, e é também o valor inicial da lista.

[6] O erro de quantização pode ser calculado como a distância média entre os vetores de um grupo e o seu centroide.

Capítulo 5

[1] Uma exceção a essa maioria são as árvores de decisão, a partir das quais também é simples a extração de regras SE-ENTÃO.

[2] Exemplo inspirado em Han, Kamber e Pei (2011) e adaptado para o contexto do restaurante.

[3] A escolha da medida *lift* nesse contexto é estimulada pelo seu uso nas funções em R, apresentadas na Seção 5.3.

[4] Não é necessário incluir L_1 em L, uma vez que *1-itemsets* frequentes não serão úteis na segunda fase do algoritmo.

[5] No caso de o arquivo *.csv* estar valorado com números {0,1}, por exemplo, seria necessário transformá-los em valores SIM e NAO antes de submetê-lo à função citada neste exemplo.

[6] Conjuntos de dados sintéticos são disponibilizados em http://goo.gl/FTGdhC.

Apêndice

[1] https://www.gnu.org/doc/doc.html

[2] http://cran.r-project.org/

[3] http://www.rstudio.com/

[4] Os comandos apresentados neste livro foram executados no sistema operacional Windows e na versão R 3.1.3 for Windows. Para outros sistemas operacionais ou outras versões do R, correções e procedimentos alternativos podem ser necessários.

[5] http://CRAN.R-project.org/

[6] Para outros tipos de dados, existem outras funções similares, como: `is.vector( )` ou `is.list( )`.

[7] A função `rbind()` permite combinar os vetores em linha. Para a combinação em coluna usa-se a função `cbind()`.

[8] UCI, http://archive.ics.uci.edu/ml/

[9] A versão utilizada neste livro foi a 1.1.9.

[10] A versão utilizada neste livro foi a 0.9.6.

[11] A versao utilizada neste livro foi a 1.95-4.5.

[12] `do.call` permite construir uma função que chama outra função. Por exemplo, nesta sintaxe, a função `lapply` é chamada para organizar os *tweets* em um vetor de *data.frame*, e a função `rbind` é depois executada para organizar o vetor em linhas.

ÍNDICE

A

Á

B

C

D

E

F

G

H

Í

I

K

L

M

O

P

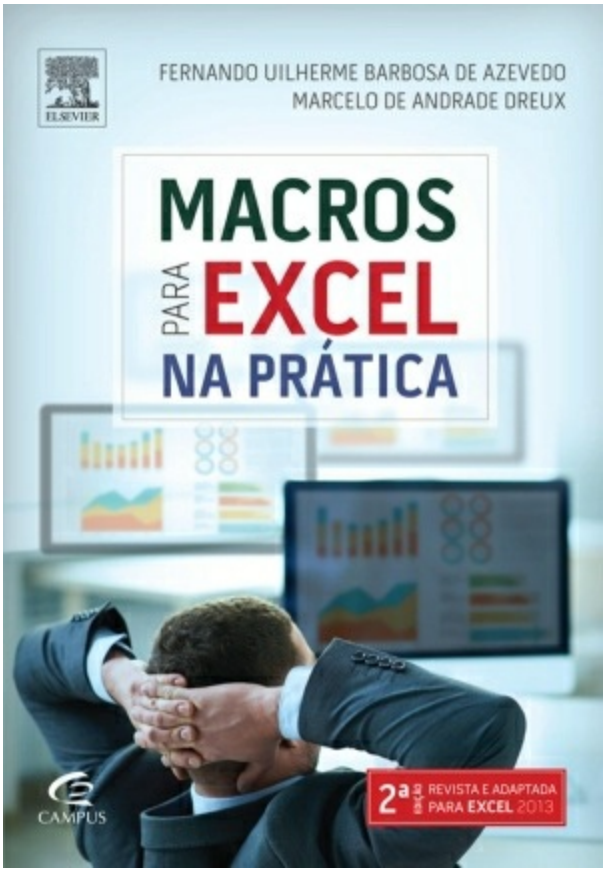
Q

R

S

T

V


ELSEVIER
FERNANDO UILHERME BARBOSA DE AZEVEDO
MARCELO DE ANDRADE DREUX
MACROS
PARA EXCEL
NA PRÁTICA
CAMPUS
2ª ed. REVISTA E ADAPTADA PARA EXCEL 2013

Macros para excel na prática

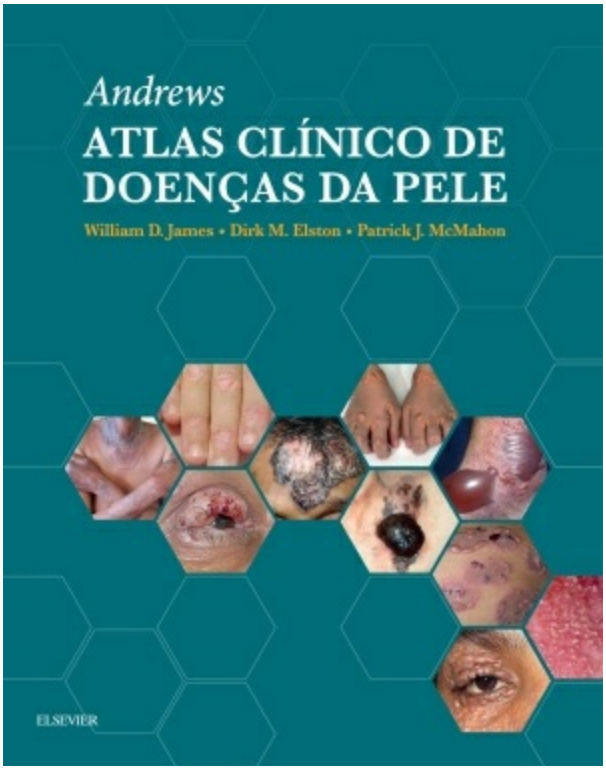
Azevedo, Fernando

9788535272659

240 páginas

O objetivo desta obra é ensinar ao leitor como desenvolver Macros para Excel. A cada capítulo devem ser apresentados os conceitos da linguagem VBA enquanto é desenvolvido um Sistema de Controle de Vendas de uma concessionária de automóveis fictícia. A apresentação dos comandos e funções para a criação do sistema proposto deve permitir ao leitor explorar as demais possibilidades de se trabalhar com Macros.

Compre agora e leia


Andrews
ATLAS CLÍNICO DE DOENÇAS DA PELE
William D. James • Dirk M. Elston • Patrick J. McMahon
ELSEVIER

Andrews atlas clínico de doenças da pele

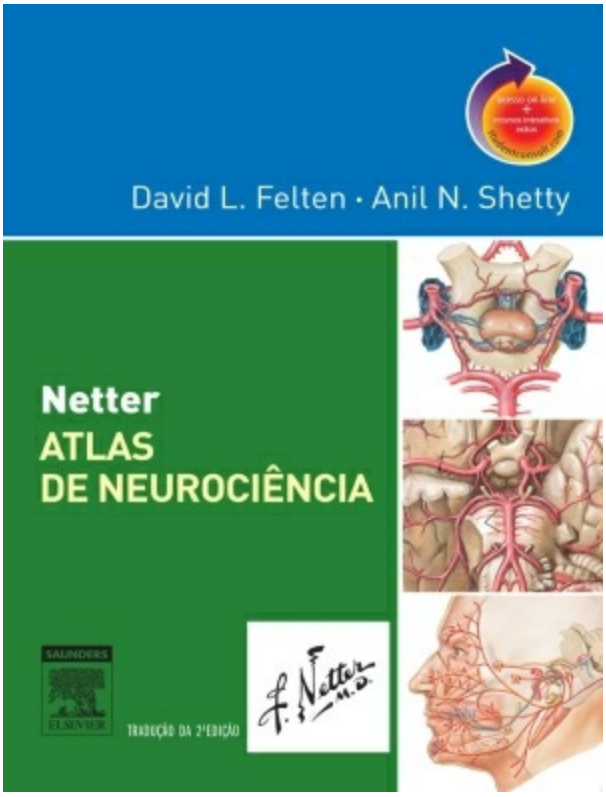
James, William D.

9788535290295

624 páginas

Este livro apresenta uma coleção de mais de 3.000 imagens de alta qualidade, incluindo novas doenças e condições raras, além de cabelo, unha e descobertas na membrana mucosa para atender as necessidades dos dermatologistas no momento do diagnóstico. Apresenta ainda um texto introdutório conciso para cada capítulo com uma visão geral e compreensiva para auxiliar no diagnóstico.Veja mais de 3.000 fotografias coloridas de alta qualidade que retratam o aspecto completo de doenças da pele em todos os tipos de pele em adultos, crianças e recém-nascidos; Destaca uma grande variedade de subtipos de condições comuns, tais como lichen planus, granuloma annulare e psoríase; Novas descobertas de doenças em cabelo, unha e membrana mucosa são apresentados no livro; Inclui representações de importantes condições sistêmicas como sarcoidose, lúpus

eritematoso e doenças infecciosas; Apresenta imagens nunca antes publicadas contribuídas por 54 líderes globais em dermatologia; Texto introdutório conciso em cada capítulo fornece aos leitores uma visão geral rápida e compreensiva da doença abordada.


David L. Felten · Anil N. Shetty
Netter
ATLAS
DE NEUROCIÊNCIA
SAUNDERS
ELSEVIER
TRADUÇÃO DA 2ª EDIÇÃO
F. Netter M.D.

Netter atlas de neurociência

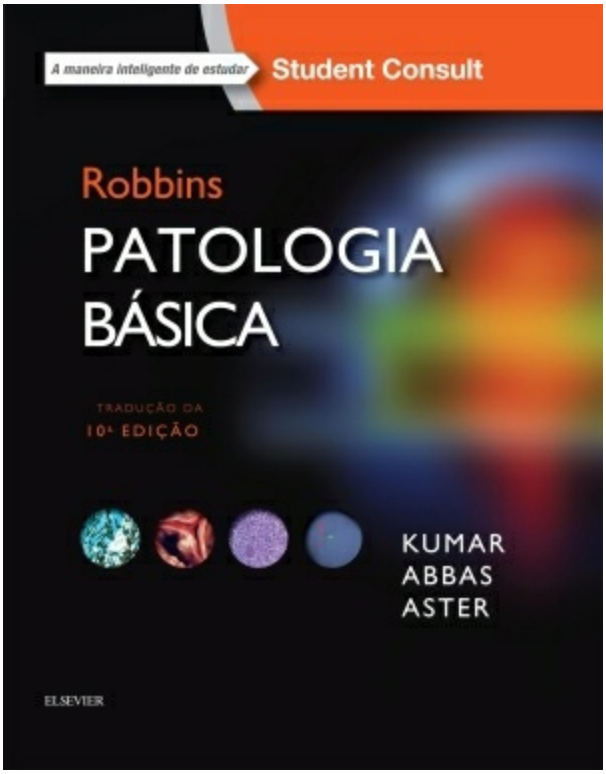
Felten, David L.

9788535246261

464 páginas

A nova edição do Netter Atlas de Neurociência oferece rica orientação visual, combinada a um texto conciso para ajudar você a dominar os princípios complexos, porém importantes, da neurociência.Uma cobertura de fácil entendimento dividida em três partes — uma visão geral do sistema nervoso, neurociência regional e neurociência sistêmica — permite o estudo das estruturas e sistemas neurais em múltiplos contextos. No conteúdo, você encontrará:• Informações atualizadas e novas figuras que refletem os atuais conhecimentos dos componentes neurais e de tecido conjuntivo, regiões e sistemas do cérebro, medula espinal e periferia para garantir que você conheça os avanços mais recentes.• Novas imagens coloridas em 3D de vias comissurais, de associação e de projeção do cérebro.• Quase 400 ilustrações com a excelência e o estilo Netter que destacam os conceitos-chave da neurociência e as

correlações clínicas, proporcionando uma visão geral rápida e fácil de memorizar da anatomia, da função e da relevância clínica.• Imagens de alta qualidade — Imagens de Ressonância Magnética (RM) de alta resolução nos planos coronal e axial (horizontal), além de cortes transversais do tronco encefálico — bem como angiografia e venografia por RM e arteriografia clássica — o que oferece uma melhor perspectiva da complexidade do sistema nervoso.• Anatomia esquemática transversa do tronco encefálico e anatomia cerebral axial e coronal — com RM — para melhor ilustrar a correlação entre neuroanatomia e neurologia.• Uma organização regional do sistema nervoso periférico, da medula espinal, do tronco encefálico, cerebelo e prosencéfalo — e uma organização sistêmica dos sistemas sensitivos, sistemas motores (incluindo o cerebelo e os núcleos da base) e dos sistemas límbicos/hipotalâmicos/autonômicos — que torna as referências mais fáceis e mais eficientes.• Novos quadros de correlação clínica que enfatizam a aplicação clínica das neurociências fundamentais.A compra deste livro permite acesso gratuito ao site studentconsult.com, um site com ilustrações para download para uso pessoal, links para material de referência adicional e conteúdo original do livro em inglês.


A maneira inteligente de estudar
Student Consult
Robbins
PATOLOGIA
BÁSICA
TRADUÇÃO DA
10ª EDIÇÃO
KUMAR
ABBAS
ASTER
ELSEVIER

Robbins Patologia Básica

Kumar, Vinay

9788535288551

952 páginas

Parte da confiável família Robbins e Cotran, Robbins Patologia Básica 10ª edição proporciona uma visão geral bem ilustrada, concisa e de leitura agradável dos princípios de patologia humana, ideal para os atarefados estudantes de hoje em dia. Esta edição cuidadosamente atualizada continua a ter forte ênfase na patogênese e nas características clínicas da doença, acrescentando novas ilustrações e diagramas mais esquemáticos para ajudar ainda mais no resumo dos principais processos patológicos e expandir o já impressionante programa de ilustrações.

Compre agora e leia


FUNDAMENTOS DE DIAGNÓSTICO POR
Imagem em Pediatria
Lane F. Donnelly
Tradução da
2ª EDIÇÃO
ELSEVIER

Fundamentos de Diagnóstico por Imagem em Pediatria

Donnelly, Lane F.

9788535289121

376 páginas

Realize com segurança e interprete com precisão os estudos de imagem pediátrica com este recurso conciso e altamente ilustrado! Fundamentos de Diagnóstico por Imagem em Pediatria, 2ª edição, abrange os conceitos essenciais que residentes e profissionais precisam dominar, estabelecendo uma base sólida para a compreensão do básico e a realização de diagnósticos radiológicos precisos. Este título, fácil de usar na série Fundamentos de Radiologia, enfatiza técnicas avançadas de imagem, incluindo aplicações neurológicas, ao mesmo tempo em que destaca a anatomia básica necessária para entender essa complexa especialidade.• Novas informações revisadas sobre temas de qualidade e de segurança, neuroimagem, ultrassonografia em imagens pediátricas e muito mais. • Pela primeira vez especialistas adicionais fornecem atualizações

em suas áreas: imagens neurológicas, musculoesqueléticas, cardíacas, torácicas e genitourinárias. • Cerca de 650 imagens digitais clinicamente relevantes e de alta qualidade demonstram, claramente, conceitos, técnicas e habilidades de interpretação essenciais. • Temas avançados de RM, como a enterografia por RM, a urografia por RM, a TC e a RM cardíacas, são discutidos minuciosamente. • Textos, tabelas e imagens acessíveis ao leitor facilitam e simplificam consultas a referências. • Editado por Lane F. Donnelly, MD, agraciado com o Prêmio 2009 Singleton-Taybi da Sociedade de Radiologia Pediátrica pela dedicação profissional à educação médica.