Terceira Edição

Mãos à Obra: Aprendizado de Máquina com Scikit-Learn, Keras & TensorFlow

Conceitos, Ferramentas e Técnicas para a construção de sistemas inteligentes

Terceira Edição

Mãos à Obra: Aprendizado de Máquina com Scikit-Learn, Keras & TensorFlow

Conceitos, Ferramentas e Técnicas para a construção de sistemas inteligentes

Aurélien Géron

ALTA BOOKS
EDITORA
Rio de Janeiro, 2025

Mãos à obra: aprendizado de máquina com Scikit-Learn, Keras & TensorFlow - 3ª edição

ISBN: 978-85-508-2645-5

Translated from original Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 3ed. Copyright © 2023 by Aurélien Géron. *ISBN 978-19505084-02. This translation is published and sold by O'Reilly Media, Inc., the owner of all rights to publish and sell the same. PORTUGUESE language edition published by Starlin Alta Editora e Consultoria Eireli, Copyright © 2025 by Starlin Alta Editora e Consultoria Eireli.*

Impresso no Brasil — 2ª Edição, 2025 — Edição revisada conforme o Acordo Ortográfico da Língua Portuguesa de 2009.

Dados Internacionais de Catalogação na Publicação (CIP)

G377m
1.ed.- Géron, Aurélien.
Mãos à obra: aprendizado de máquina com Scikit-Learn, Keras & TensorFlow / Aurélien Géron; tradução Eveline Machado. – 3 ed. – Rio de Janeiro: Alta Books, 2025.
640 p.; 15,7 x 23 cm.
Título original: Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow
ISBN 978-85-508-2645-5

1. Aprendizado de máquina. 2. Inteligência artificial. 3. Redes neurais artificiais. 4. Scikit-learn. 5. TensorFlow. I. Título. II.

CDD 006.31

Índice para catálogo sistemático:
1. Aprendizado de máquina — 006.31

Grupo Editorial Alta Books

Diretor Editorial: Anderson Vieira.
Vendas ao Governo: Cristiane Mutüs.
Gerência Marketing: Viviane Paiva.

Produtora Editorial: Isabella Gibara.
Tradução e Copidesque: Eveline Machado.
Revisão Gramatical: Denise Himpel.
Diagramação: Natalia Curupana.

ALTA BOOKS
GRUPO EDITORIAL

Rua Viúva Cláudio, 291 — Bairro Industrial do Jacaré
CEP: 20.970-031 — Rio de Janeiro (RJ)
Tels.: (21) 3278-8069 / 3278-8419
www.altabooks.com.br — altabooks@altabooks.com.br
Ouvidoria: ouvidoria@altabooks.com.br

Editora **afiliada à:**

Sobre o Autor

Aurélien Géron é consultor de aprendizado de máquina e palestrante. Ex-funcionário do Google, liderou as classificações do YouTube de 2013 a 2016. Também foi fundador e diretor de tecnologia em diferentes empresas: Wifirst, o provedor de internet wireless líder na França; Polyconseil, uma empresa de consultoria com foco em telecomunicações, mídia e estratégia; e Kiwisoft, uma empresa de consultoria focada em aprendizado de máquina e privacidade de dados.

Antes disso, Aurélien trabalhou como engenheiro em diversos setores: finanças (JP Morgan e Societé Générale), segurança (Departamento de Defesa Nacional do Canadá) e assistência médica (transfusão de sangue). Publicou também alguns livros técnicos (sobre C++, Wi-Fi e arquitetura da internet) e lecionou Ciência da Computação em uma faculdade de engenharia francesa.

Curiosidades: ele ensinou seus três filhos a contar em binário com os dedos (até 1.023), estudou microbiologia e genética evolutiva antes de entrar para engenharia de software e, uma vez, seu paraquedas não abriu no segundo salto.

Sumário

Prefácio

O Tsunami do Aprendizado de Máquina

Em 2006, Geoffrey Hinton *et al.* publicou um artigo[1] demonstrando como treinar uma rede neural profunda capaz de reconhecer algarismos escritos à mão com uma precisão de ponta (> 98%). Chamou-se técnica de "aprendizado profundo" [*Deep Learning* ou DL]. Uma rede neural profunda é um modelo (bem) simplificado do nosso córtex cerebral, constituído por uma pilha de camadas de neurônios artificiais. Na época, treinar uma rede neural profunda era basicamente uma tarefa impossível[2] e a maioria dos pesquisadores havia desistido da ideia no fim dos anos 1990. Esse artigo reacendeu o interesse da comunidade científica e, em pouco tempo, muitos artigos novos comprovavam que o aprendizado profundo não só era possível como era capaz de resultados surpreendentes (com a ajuda de uma capacidade de processamento monstruosa e de quantidades de dados imensas), que não poderiam se igualar a nenhuma outra técnica de aprendizado de máquina (ML). Esse entusiasmo logo se estendeu a muitas outras áreas do aprendizado de máquina.

Cerca de uma década depois, o aprendizado de máquina conquistou todos os segmentos industriais: está no coração de grande parte dos produtos de alta tecnologia de hoje, classificando os resultados de pesquisa na web, viabilizando o reconhecimento de voz dos smartphones, fazendo recomendações de vídeos e talvez até esteja dirigindo seu carro.

O Aprendizado de Máquina em Seus Projetos

Claro que você está entusiasmado com o aprendizado de máquina e gostaria de participar da festa!

E se você tivesse um robô doméstico e, quem sabe, lhe desse um cérebro? E reconhecimento facial? Ou o ensinasse a andar?

Ou talvez sua empresa tenha toneladas de dados (logs de usuários, dados financeiros, de produção, de sensores de máquinas, estatísticas de atendimento ao cliente, relatórios de RH etc.), e muito provavelmente você poderia desenterrar algum tesouro escondido se soubesse onde procurar. Com o aprendizado de máquina, é possível fazer isso e muito mais:

- Segmentar os clientes e identificar a melhor estratégia de marketing para cada grupo.
- Recomendar produtos para cada cliente com base no que clientes similares compraram.
- Detectar as transações suscetíveis à fraude.
- Prever o faturamento do próximo ano.

Seja lá qual for a razão, você decidiu estudar o aprendizado de máquina e usá-lo em seus projetos. Ótima ideia!

1 Geoffrey E. Hinton *et al.*, "A Fast Learning Algorithm for Deep Belief Nets", *Neural Computation* 18 (2006): 1527–1554.

2 Apesar do fato de as redes neurais convolucionais de aprendizado profundo de Yann Lecun terem funcionado bem para o reconhecimento de imagens desde a década de 1990, elas não eram de uso geral.

Objetivo e Abordagem

Este livro parte do princípio de que você não sabe quase nada sobre aprendizado de máquina. O objetivo é apresentar os conceitos, as ferramentas e a intuição necessários para implementar programas capazes de *aprender com os dados*.

Abordaremos muitas técnicas, desde as mais simples e mais comumente utilizadas (como a regressão linear) até algumas das técnicas de aprendizado profundo que vencem competições com frequência. Para tanto, utilizaremos os frameworks executáveis do Python:

- Scikit-Learn é bem fácil de usar e implementa muitos algoritmos do ML de maneira eficiente, por isso é uma excelente porta de entrada para o aprendizado de máquina. Foi criada por David Cournapeau em 2007 e é agora liderada por uma equipe de pesquisadores do Instituto Francês de Pesquisa em Ciência da Computação e Automação (Inria).
- TensorFlow é uma biblioteca mais complexa para cálculo numérico distribuído. Possibilita treinar e executar grandes redes neurais de maneira eficiente, distribuindo os cálculos entre centenas de servidores com múltiplas GPUs (unidades de processamento gráfico). A TensorFlow (TF) foi criada no Google e suporta muitas das aplicações em larga escala do aprendizado de máquina. Em novembro de 2015, ela se tornou uma biblioteca de código aberto e a versão 2.0 foi lançada em setembro de 2019.
- Keras é uma API de aprendizado profundo de alto nível que facilita o treinamento e a execução de redes neurais. Ela roda com a TensorFlow e conta com ela para toda a computação intensiva.

O livro favorece uma abordagem prática, viabilizando uma compreensão intuitiva do aprendizado de máquina por meio de exemplos objetivos e concretos, e um pouco de teoria.

Embora você possa ler este livro sem usar o computador, recomendo muitíssimo que treine com os exemplos de código.

Exemplos de Código

Todos os exemplos de código neste livro são open source e estão disponíveis online em *https://github.com/ageron/handson-ml3*, como os notebooks Jupyter [links *https* com conteúdo em inglês]. São documentos interativos contendo texto, imagens e fragmentos de código executável (Python no nosso caso). A maneira mais fácil e rápida de começar é executar esses notebooks usando o Google Colab: é um serviço gratuito que permite executar qualquer notebook Jupyter diretamente online, sem instalar nada em sua máquina. Tudo o que você precisa é de um navegador e uma conta do Google.

Neste livro, presumirei que você está usando o Google Colab, mas também testei os notebooks em outras plataformas online, como Kaggle e Binder, então você pode usá-los se preferir. Uma alternativa é instalar as bibliotecas e as ferramentas necessárias (ou a imagem Docker deste livro), e executar os notebooks diretamente em sua própria máquina. Veja as instruções em *https://homl.info/install*.

Este livro existe para ajudá-lo a fazer o seu trabalho. Se quiser usar o conteúdo adicional além dos exemplos de código, e esse uso estiver fora do escopo das diretrizes de uso justas (como vender ou distribuir o conteúdo dos livros da O'Reilly, ou incorporar uma quantidade significativa de material deste livro na documentação do seu produto), entre em contato para pedir permissão.

Agradecemos, mas não exigimos, a atribuição. Uma atribuição costuma incluir título, autor, editor e ISBN. Por exemplo: "*Mãos à Obra: Aprendizado de Máquina com Scikit-Learn, Keras & TensorFlow* de Aurélien Géron. Rio de Janeiro: Alta Books, 2025".

Pré-requisitos

Este livro pressupõe que você tem alguma experiência de programação em Python. Se não conhece o Python ainda, *https://learnpython.org* é um ótimo começo. O tutorial oficial sobre Python.org (*https://docs.python.org/3/tutorial*) também é muito bom.

Este livro também presume que você está familiarizado com as principais bibliotecas científicas do Python — em particular, NumPy, Pandas e Matplotlib. Se nunca usou essas bibliotecas, não se preocupe; são fáceis de aprender, e eu criei um tutorial para cada uma delas. Você pode acessá-las online em *https://homl.info/tutorials*.

Além disso, se quiser entender bem como os algoritmos do aprendizado de máquina funcionam (não apenas como usá-los), então deve ter pelo menos uma compreensão básica de alguns conceitos de matemática, especialmente de álgebra linear. Especificamente, você deve saber o que são vetores e matrizes, e como executar algumas operações simples, como adicionar vetores, ou transpor e multiplicar matrizes. Se precisar de uma introdução rápida para álgebra linear (não é um bicho de sete cabeças!), forneço um tutorial em *https://homl.info/tutorials*. Você também encontrará um tutorial sobre cálculo diferencial, que pode ser útil para compreender como as redes neurais são treinadas, mas não é inteiramente essencial para entender os principais conceitos. Às vezes, este livro também usa outros conceitos matemáticos, tais como, exponenciais e logaritmos, um pouco de teoria da probabilidade e alguns conceitos básicos de estatística, mas nada muito avançado. Se precisar de ajuda em qualquer um dos tópicos, confira *https://khanacademy.org*, que oferece muitos cursos de matemática online excelentes e gratuitos.

Roteiro

Este livro foi organizado em duas partes. A Parte I, "Fundamentos do Aprendizado de Máquina", aborda os seguintes tópicos:

- O que é aprendizado de máquina, quais problemas ele tenta resolver, quais são as principais categorias e os conceitos fundamentais dos seus sistemas.
- Os passos de um típico projeto de aprendizado de máquina.
- Aprender a ajustar um modelo aos dados.
- Otimizar a função de custos.
- Manipular, limpar e preparar os dados.
- Selecionar e desenvolver funcionalidades.
- Selecionar um modelo e ajustar os hiperparâmetros usando a validação cruzada.
- Os desafios do aprendizado de máquina, especialmente o subajuste [underfitting] e o sobreajuste [overfitting] (equilíbrio entre viés/variância).
- Os algoritmos de aprendizado mais comuns: regressão linear e polinomial, regressão logística, *k*-ésimo vizinho mais próximo, máquinas de vetor de suporte, árvores de decisão, florestas aleatórias e métodos de ensemble (combinação).
- Reduzir a dimensionalidade dos dados de treinamento para combater a "maldição da dimensionalidade".
- Outras técnicas de aprendizado não supervisionado, incluindo clusterização, estimativa de densidade e detecção de anomalias.

A Parte II, "Redes Neurais e Aprendizado Profundo", aborda os seguintes tópicos:

- O que são redes neurais e para que servem.
- Construir e treinar redes neurais usando TensorFlow e Keras.
- As arquiteturas de redes neurais mais importantes: redes neurais do tipo feedforward para dados tabulares, redes convolucionais para visão computacional, redes recorrentes e redes de memória de longo e curto prazos (LSTM) para o processamento de sequências, encoders/decoders e transformadores para o processamento de linguagem natural, redes autoencoders e adversárias generativas (GANs), além de modelos de difusão para o aprendizado generativo.
- Técnicas para o treinamento de redes neurais profundas.
- Como criar um agente (por exemplo, um bot em um jogo) que pode aprender boas estratégias por tentativa e erro, usando o aprendizado por reforço.
- Carregar e pré-processar grandes quantidades de dados com eficiência.
- Treinar e implementar os modelos TensorFlow em grande escala.

A Parte I se baseia na Scikit-Learn, já a Parte II usa TensorFlow e Keras.

Não dê um passo maior que a perna: embora o aprendizado profundo seja, sem sombra de dúvida, uma das áreas mais interessantes do aprendizado de máquina, você deve dominar os princípios básicos primeiro. Além do mais, grande parte dos problemas pode ser resolvida com técnicas mais simples, como os métodos de floresta aleatória e de ensemble (discutidos na Parte I). O aprendizado profundo se adéqua melhor aos problemas complexos, como reconhecimento de imagem e de voz ou processamento de linguagem natural, e requer muitos dados, capacidade de processamento e paciência (a menos que você possa utilizar uma rede neural pré-treinada, como será visto).

Mudanças entre a Primeira e a Segunda Edições

Se você já leu a primeira edição, veja as principais mudanças entre a primeira e segunda:

- Todo o código foi migrado da TensorFlow 1.x para a TensorFlow 2.x, e eu substituí a maioria do código de baixo nível da TensorFlow (gráficos, sessões, colunas de características, estimadores etc.) pelo código Keras muito mais simples.
- A segunda edição introduziu a Data API para carregar e pré-processar grandes conjuntos de dados, a API de estratégias de distribuição para treinar e implantar modelos TF em escala, TF Serving e plataforma Google Cloud AI para produzir modelos e (em breve) TF Transform, TFLite, TF Addons/Seq2Seq, TensorFlow.js e TF Agents.
- Também foram introduzidos muitos outros tópicos de ML, incluindo um novo capítulo sobre aprendizado não supervisionado, técnicas de visão computacional para detecção de objetos e segmentação semântica, manipulação de sequências usando redes neurais convolutivas (CNNs), processamento de linguagem natural (NLP) usando redes neurais recorrentes (RNNs), CNNs e transformadores, GANs etc.

Acesse *https://homl.info/changes2* para mais detalhes.

Mudanças entre a Segunda e a Terceira Edições

Se você leu a segunda edição, veja as principais mudanças entre a segunda e a terceira:

- Todo o código foi atualizado com as versões mais recentes da biblioteca. Em particular, esta terceira edição introduz muitas novas adições à Scikit-Learn (ex.: rastreamento de nomes de características, gradient boosting baseado em histogramas, propagação de rótulos e mais). Ela também apresenta a biblioteca *Keras Tuner* para o ajuste de hiperparâmetros, a biblioteca *Transformers* da Hugging Face para o processamento da linguagem natural, o novo pré-processamento da Keras e camadas de aumento de dados.
- Vários modelos de visão foram adicionados (ResNeXt, DenseNet, MobileNet, CSPNet e EfficientNet), bem como diretrizes para escolher o caminho certo.
- O Capítulo 15 agora analisa os dados de passageiros de ônibus e trens de Chicago, em vez da série temporal gerada, e apresenta o modelo ARMA e suas variantes.
- O Capítulo 16 sobre processamento da linguagem natural agora constrói um modelo de tradução do inglês para o espanhol, primeiro usando um encoder/decoder RNN, depois usando um modelo de transformador. O capítulo também cobre os modelos de linguagem, como Switch Transformers, DistilBERT, T5 e PaLM (com a técnica Chain-of-Thought Prompting — CoT). Além disso, apresenta transformadores de visão (ViTs) e dá uma ideia geral de alguns modelos visuais baseados em transformadores, como transformadores de imagem com eficiência de dados (DeiTs), Perceiver e DINO, bem como uma rápida visão geral de alguns grandes modelos multimodais, incluindo CLIP, DALL E, Flamingo e GATO.
- O Capítulo 17 sobre aprendizagem generativa agora apresenta modelos de difusão e mostra como implementar um modelo probabilístico de difusão de ruído (DDPM) a partir do zero.
- O Capítulo 19 migrou da plataforma Google Cloud AI para a Google Vertex AI, e usa o Keras Tuner distribuído para pesquisar hiperparâmetros em grande escala. O capítulo agora inclui o código TensorFlow.js que você pode experimentar online. Também introduz técnicas de treinamento adicionais distribuídas, incluindo PipeDream e Pathways.
- Para permitir todo o novo conteúdo, algumas seções se tornaram online, incluindo as instruções de instalação, a análise do componente principal do kernel (PCA), os detalhes matemáticos das combinações bayesianas gaussianas, agentes TF e os antigos Apêndices A (soluções de exercícios), C (suporte à matemática de máquinas vetoriais) e E (arquiteturas de redes neurais extras).

Veja *https://homl.info/changes3* para ter mais detalhes. [Conteúdo em inglês.]

Outros Recursos

Há muitos recursos excelentes disponíveis a respeito do aprendizado de máquina. Por exemplo, o curso de ML de Andrew Ng no Coursera (*https://homl.info/ngcourse*) é incrível, embora exija um investimento de tempo significativo.

Há também muitos sites interessantes sobre aprendizado de máquina, incluindo, claro, o excepcional Guia do Usuário da Scikit-Learn (*https://homl.info/skdoc*). Você também pode gostar do Dataquest (*https://dataquest.io*), que oferece tutoriais interativos muito interessantes e blogs de ML, como aqueles listados no Quora (*https://homl.info/1*).

Há também muitos outros livros introdutórios sobre aprendizado de máquina:

- *Data Science do Zero*, 2ª ed., de Joel Grus (Alta Books). Apresenta os conceitos básicos do aprendizado de máquina e implementa alguns dos principais algoritmos em Python puro (do zero, como o nome sugere).

- *Machine Learning: An Algorithmic Perspective*, de Stephen Marsland. É uma ótima introdução ao aprendizado de máquina que aborda uma ampla gama de tópicos em profundidade, com exemplos de código em Python (também a partir do zero, mas utilizando o NumPy).
- *Python Machine Learning*, 3ª ed., de Sebastian Raschka. É também uma ótima introdução ao aprendizado de máquina e incentiva o uso das bibliotecas de código aberto do Python (Pylearn 2 e Theano).
- *Deep Learning with Python,* 2ª ed., de François Chollet. Livro muito prático que aborda uma grande variedade de tópicos de forma clara e concisa, como seria de esperar do autor da excelente biblioteca Keras. Prefere os exemplos de código à teoria matemática.
- *The Hundred-Page Machine Learning Book,* de Andriy Burkov. Livro pequeno que abrange uma variedade impressionante de tópicos, apresentando-os de forma bem acessível sem fugir das equações matemáticas.
- *Learning from Data*, de Yaser S. Abu-Mostafa, Malik Magdon-Ismail e Hsuan-Tien Lin. Com uma abordagem bastante teórica sobre ML, esta obra proporciona insights profundos, especialmente sobre o equilíbrio entre viés/variância (veja o Capítulo 4).
- *Inteligência Artificial* de Stuart Russell e Peter Norvig. É um livro ótimo (e gigante) que aborda uma quantidade incrível de tópicos, incluindo o aprendizado de máquina. Ajuda a esclarecer muitas coisas a respeito do ML.
- *Deep Learning for Coders with fastai and PyTorch* de Jeremy Howard e Sylvain Gugger fornece uma introdução maravilhosamente clara e prática para o aprendizado profundo usando as bibliotecas fastai e PyTorch.

Por último, uma ótima forma de aprender é entrar em sites de competição como o Kaggle.com, que lhe possibilita praticar suas habilidades usando problemas reais com a ajuda e os insights de alguns dos melhores profissionais de ML existentes.

Convenções Usadas Neste Livro

As seguintes convenções tipográficas são usadas neste livro:

Itálicos

Indica termos novos, URLs, endereços de e-mail, nomes de arquivos e extensões de arquivos.

`Fonte monoespaçada`

Usada para listagens de programas, bem como dentro de parágrafos para referenciar elementos do programa: nomes de variáveis ou funções, bancos de dados, tipos de dados, variáveis de ambiente, declarações e palavras-chave.

`Fonte monoespaçada em negrito`

Mostra comandos ou outro texto que deve ser digitado pelo usuário.

`Fonte monoespaçada em itálico`

Mostra o texto que deve ser substituído por valores fornecidos pelo usuário ou por valores determinados pelo contexto.

Pontuação

Para evitar confusão, aparece fora das aspas no livro. Peço desculpas aos puristas.

Este elemento significa uma dica ou uma sugestão.

Este significa uma nota geral.

Este indica alerta ou cautela.

Agradecimentos

Nunca, nem em sonho, imaginei que a primeira e a segunda edições deste livro atingissem um público tão grande. Recebi muitas mensagens dos leitores, muitas perguntas, algumas simpáticas indicando erratas e a maioria me enviando palavras de incentivo. Mal posso expressar o quanto sou grato a todos eles pelo imenso apoio. Muitíssimo obrigado a todos. Por favor, não pense duas vezes em registrar os problemas no GitHub (*https://homl.info/issues3*), caso encontre erros nos exemplos de código (ou queira perguntar algo) ou enviar errata (*https://homl.info/errata3*) se encontrar erros no texto. Alguns leitores também compartilharam como este livro os ajudou a conseguir seu primeiro emprego ou a solucionar um problema concreto em que estavam trabalhando. É o tipo de feedback extremamente motivador. Se você achar este livro proveitoso, adoraria que compartilhasse sua história comigo em particular (por exemplo, via LinkedIn, em *https://www.linkedin.com/in/aurelien-geron/*) ou publicamente (com um tuíte ou por meio de uma avaliação na Amazon).

Sou também imensamente grato a todas as pessoas incríveis que tiraram um tempo para revisar a terceira edição, corrigindo os erros e fazendo incontáveis sugestões. Esta edição é muito melhor graças a: Olzhas Akpambetov, George Bonner, François Chollet, Siddha Ganju, Sam Goodman, Matt Harrison, Sasha Sobran, Lewis Tunstall, Leandro von Werra e meu querido irmão Sylvain. Vocês são incríveis!

Também sou muito grato às pessoas que me apoiaram ao longo do caminho, respondendo às minhas perguntas, sugerindo melhorias e contribuindo com o código no GitHub: em particular, Yannick Assogba, Ian Beauregard, Ulf Bissbort, Rick Chao, Peretz Cohen, Kyle Gallatin, Hannes Hapke, Victor Khaustov, Soonson Kwon, Eric Lebigot, Jason Mayes, Laurence Moroney, Sara Robinson, Joaquín Ruales e Yuefeng Zhou.

Este livro não existiria sem a equipe incrível da O'Reilly, sobretudo Nicole Taché, que me deu um feedback elucidativo, sempre cheia de ânimo, encorajadora e prestativa: eu não poderia sequer sonhar com uma editora melhor. Muito obrigado a Michele Cronin também, que foi muito solícita nos capítulos finais e me fez cruzar a linha de chegada. Obrigado a toda a equipe de produção, em particular a Elizabeth Kelly e Kristen Brown. Obrigado a Kim Cofer pela minuciosa correção e a Johnny O'Toole, que cuidou do relacionamento com a Amazon e respondeu a muitas de minhas perguntas. Obrigado a Kate Dullea por melhorar muito minhas ilustrações. Obrigado a Marie Beaugureau, Ben Lorica, Mike Loukides e Laurel Ruma por acreditarem neste projeto e me ajudarem a definir seu escopo. Agradeço a Matt Hacker e toda a equipe da Atlas por responderem a todas as minhas perguntas técnicas sobre formatação, AsciiDoc,

MathML e LaTeX, e a Nick Adams, Rebecca Demarest, Rachel Head, Judith McConville, Helen Monroe, Karen Montgomery, Rachel Roumeliotis e todos da O'Reilly que contribuíram para este livro.

Jamais esquecerei todas as pessoas maravilhosas que me ajudaram com a primeira e segunda edições deste livro: amigos, colegas, especialistas, incluindo muitos membros da equipe TensorFlow. A lista é longa: Olzhas Akpambetov, Karmel Allison, Martin Andrews, David Andrzejewski, Paige Bailey, Lukas Biewald, Eugene Brevdo, William Chargin, François Chollet, Clément Courbet, Robert Crowe, Mark Daoust, Daniel "Wolff " Dobson, Julien Dubois, Mathias Kende, Daniel Kitachewsky, Nick Felt, Bruce Fontaine, Justin Francis, Goldie Gadde, Irene Giannoumis, Ingrid von Glehn, Vincent Guilbeau, Sandeep Gupta, Priya Gupta, Kevin Haas, Eddy Hung, Konstantinos Katsiapis, Viacheslav Kovalevskyi, Jon Krohn, Allen Lavoie, Karim Matrah, Grégoire Mesnil, Clemens Mewald, Dan Moldovan, Dominic Monn, Sean Morgan, Tom O'Malley, James Pack, Alexander Pak, Haesun Park, Alexandre Passos, Ankur Patel, Josh Patterson, André Susano Pinto, Anthony Platanios, Anosh Raj, Oscar Ramirez, Anna Revinskaya, Saurabh Saxena, Salim Sémaoune, Ryan Sepassi, Vitor Sessak, Jiri Simsa, Iain Smears, Xiaodan Song, Christina Sorokin, Michel Tessier, Wiktor Tomczak, Dustin Tran, Todd Wang, Pete Warden, Rich Washington, Martin Wicke, Edd Wilder-James, Sam Witteveen, Jason Zaman, Yuefeng Zhou e meu irmão Sylvain.

Por último, mas não menos importante, sou infinitamente grato à minha digníssima esposa, Emmanuelle, e aos nossos três filhos maravilhosos, Alexandre, Rémi e Gabrielle, por me incentivarem a trabalhar com afinco neste livro e pela curiosidade insaciável: explicar alguns dos conceitos mais difíceis deste livro para minha esposa e filhos me ajudou a desanuviar a mente e a aprimorar muitos aspectos da obra. E eles ainda me traziam café e cookies! O que mais eu poderia querer neste mundo?

PARTE I

Fundamentos do Aprendizado de Máquina

O Cenário do Aprendizado de Máquina

Não faz muito tempo, se você pegasse seu telefone e perguntasse sobre o caminho para casa, ele o teria ignorado — e as pessoas questionariam sua sanidade. No entanto, o aprendizado de máquina (ML) não é mais ficção científica: bilhões de pessoas o utilizam todos os dias. Na verdade, há algumas décadas ele foi introduzido como reconhecimento ótico de caracteres (OCR) em alguns aplicativos específicos, mas o primeiro aplicativo que realmente se popularizou e conquistou o mundo na década de 1990, melhorando a vida de centenas de milhões de pessoas, foi o *filtro de spam*. Não é exatamente um robô autoconsciente, mas pode ser tecnicamente classificado como aprendizado de máquina (uma máquina que aprendeu tão bem que raramente é necessário marcar um e-mail como spam). O filtro de spam foi seguido por centenas de aplicativos ML que agora, silenciosamente, integram centenas de produtos e funcionalidades usados com regularidade: mensagens de voz, tradução automática, pesquisa de imagens, recomendações de produtos e muitos mais.

Onde começa e onde termina o aprendizado de máquina? O que significa exatamente para uma máquina *aprender* algo? Se eu fizer download de uma cópia da Wikipédia, meu computador realmente aprenderá algo? De repente ficará mais inteligente? Neste capítulo, começaremos esclarecendo o que é o aprendizado de máquina e por que você desejará utilizá-lo.

Antes de iniciarmos a exploração do mundo do aprendizado de máquina, analisaremos seu mapa e conheceremos suas principais regiões e os cenários mais conhecidos: aprendizado supervisionado vs. não supervisionado, aprendizado online vs. em batch, aprendizado baseado em instância (IBL) vs. baseado em modelo. Em seguida, analisaremos o fluxo de trabalho de um típico projeto de ML, discutiremos os principais desafios enfrentados e mostraremos como avaliar e aperfeiçoar um sistema de aprendizado de máquina.

Este capítulo apresenta muitos conceitos básicos (e jargões) que todo cientista de dados deve saber de cor. Será uma visão geral de alto nível (o único capítulo sem muito código), tudo simplificado, mas você deve garantir que entendeu tudo perfeitamente antes de continuar. Então, pegue um café e mãos à obra!

Se você já conhece todos os conceitos básicos do aprendizado de máquina, vá direto para o Capítulo 2. Se não tem certeza, tente responder a todas as perguntas listadas no fim do capítulo antes de continuar.

O que É Aprendizado de Máquina?

Aprendizado de máquina é a ciência (e a arte) da programação de computadores de modo que eles possam *aprender com os dados*.

Veja uma definição mais generalizada:

> [Aprendizado de máquina é o] campo de estudo que possibilita aos computadores a habilidade de aprender sem explicitamente programá-los.
>
> — Arthur Samuel, 1959

Agora uma definição da engenharia:

> Alega-se que um programa de computador aprende pela experiência *E* em relação a alguma tarefa *T* e alguma medida de performance *P* se sua performance em *T*, conforme medida por P, melhora com a experiência *E*.
>
> — Tom Mitchell, 1997

Seu filtro de spam é um programa de aprendizado de máquina que, dados exemplos de spam (marcados pelos usuários) e exemplos de e-mails normais (não spam, também chamados de "ham"), pode aprender a marcar o spam. Os exemplos utilizados pelo sistema para o aprendizado se chamam *conjunto de treinamento*. Cada exemplo de treinamento se chama *instância de treinamento* (ou *amostra*). A parte de um sistema de aprendizado de máquina que aprende e faz predições se chama *modelo*. As redes neurais e as florestas aleatórias são exemplos de modelos.

Nesse caso, a tarefa *T* sinaliza os e-mails novos como spam, a experiência *E* é o *dado de treinamento* e a medida de performance *P* precisa ser definida; por exemplo, você pode utilizar a proporção de e-mails sinalizados corretamente. Essa medida de performance em particular se chama *acurácia* e é usada frequentemente em tarefas de classificação.

Se você baixar uma cópia da Wikipédia, seu computador terá bastante dados, mas não terá um melhor desempenho repentino em nenhuma tarefa. Isso não é aprendizado de máquina.

Por que Usar o Aprendizado de Máquina?

Pense em como você desenvolveria um filtro de spam usando as técnicas de programação tradicionais (Figura 1-1):

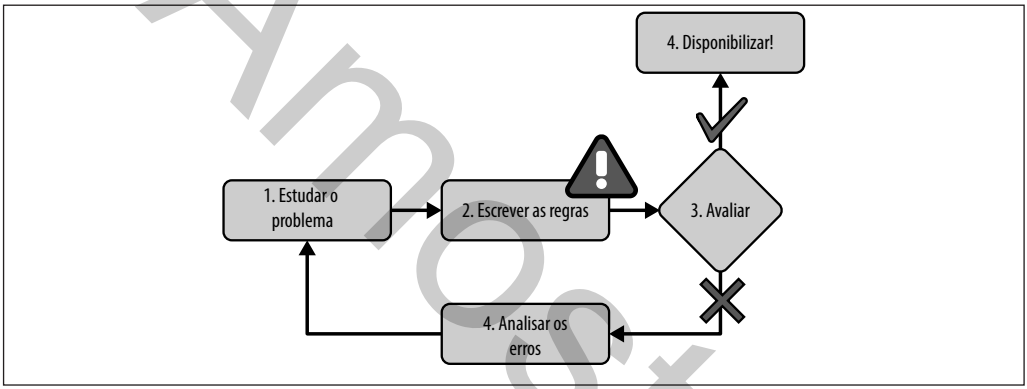

Figura 1-1. Abordagem tradicional

1. Primeiro, você examinaria as características do spam. Talvez perceba que algumas palavras ou frases ("Para você", "cartão de crédito", "de graça" e "oferta imperdível") costumam aparecer muito no campo do assunto. Talvez você repare em outros padrões no nome do remetente, no corpo do e-mail e nas outras partes do e-mail.
2. Escreveria um algoritmo de detecção para cada um dos padrões observados e, se fossem detectados, seu programa sinalizaria esses e-mails como spam.
3. Testaria seu programa e repetiria os passos 1 e 2 até que ele estivesse bom o suficiente para ser disponibilizado.

Se o problema for difícil, seu programa acabará se tornando uma extensa lista de regras complexas — a manutenção não será nada fácil.

Em contrapartida, um filtro de spam baseado em técnicas de aprendizado de máquina aprende automaticamente quais palavras e frases são bons indicadores de spam ao detectar os padrões de palavras inusitadamente frequentes em exemplos de spam quando comparados aos exemplos dos e-mails "ham" (Figura 1-2). O programa é bem menor, de fácil manutenção e, certamente, mais preciso.

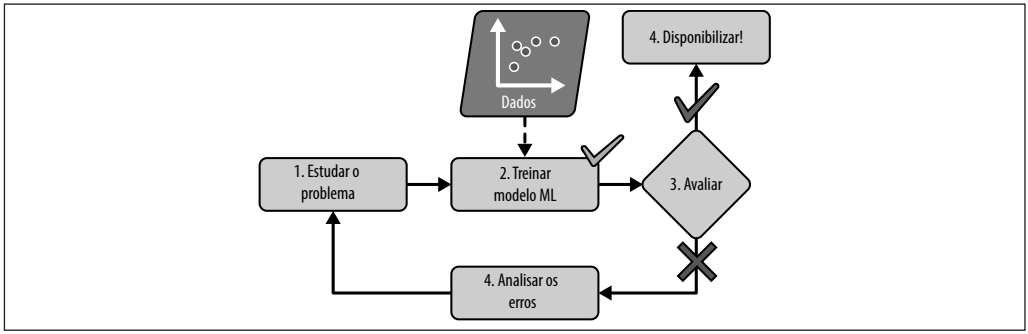

Figura 1-2. Abordagem do aprendizado de máquina

E se os spammers perceberem que todos os seus e-mails com "Para você" estão sendo bloqueados? Eles poderão começar a escrever "Só para você". Um filtro de spam que utiliza técnicas de programação tradicionais precisaria ser atualizado para sinalizar os e-mails "Só para você". Se os spammers continuarem burlando seu filtro de spam, será preciso escrever novas regras interminavelmente.

Por outro lado, um filtro de spam baseado em técnicas de aprendizado de máquina percebe automaticamente que "Só para você" se tornou frequente no spam sinalizado pelos usuários e, assim, começa a marcá-los sem a sua intervenção (Figura 1-3).

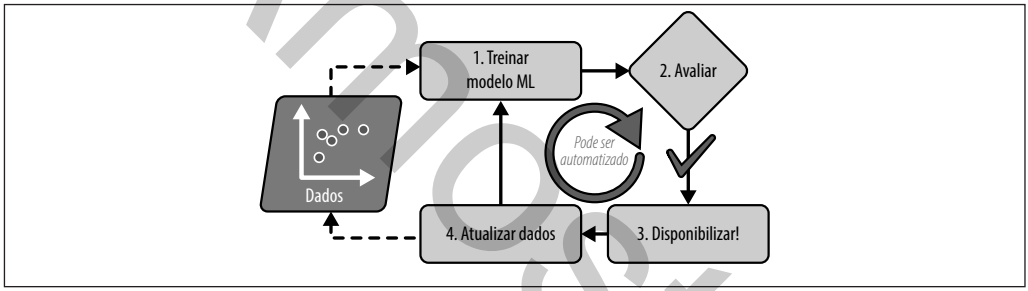

Figura 1-3. Adaptando-se automaticamente à mudança

Outro campo em que o aprendizado de máquina se destaca é o de problemas muito complexos para as abordagens tradicionais ou que não têm um algoritmo conhecido. Por exemplo, considere o reconhecimento de voz. Digamos que você deseja começar com o básico e escreva um programa capaz de distinguir as palavras "one" e "two". Você deve perceber que a palavra "two" começa com um som chiado e agudo ("T"), então apela para a codificação rígida e programa um algoritmo que calcula a intensidade do som e o utiliza para distinguir "one" e "two" — obviamente essa técnica não se aplicará a milhares de palavras faladas por milhões de pessoas diferentes em ambientes confusos e em centenas de idiomas. A melhor solução (pelo menos hoje) seria escrever um algoritmo que aprenda sozinho por meio de muitas gravações de exemplos para cada palavra.

Por fim, o aprendizado de máquina pode ajudar os seres humanos a aprender (Figura 1-4). Os modelos ML podem ser inspecionados para saber o que eles aprenderam (ainda que para alguns modelos isso possa ser complicado). Por exemplo, uma vez que o filtro foi treinado para o spam, ele pode ser simplesmente inspecionado e evidenciar uma lista de palavras e combinações previstas que ele acredita serem as mais prováveis. Às vezes, isso revelará correlações inesperadas ou tendências novas, resultando em uma melhor compreensão do problema. Investigar grandes quantidades de dados para descobrir padrões ocultos se chama *mineração de dados*, e o aprendizado de máquina é excelente.

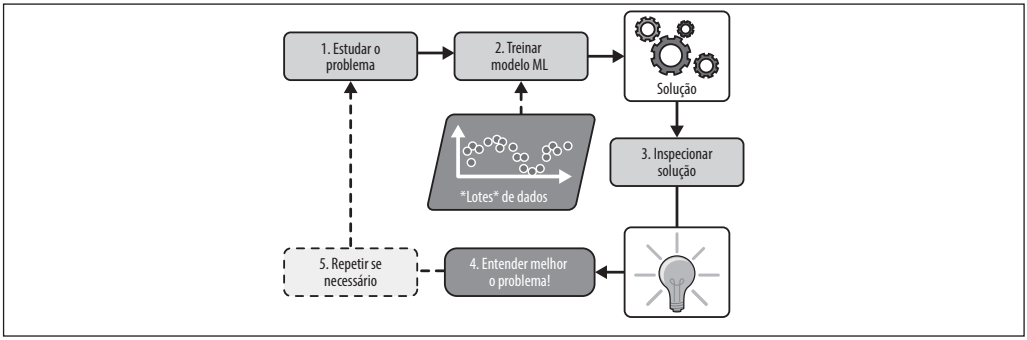

Figura 1-4. O aprendizado de máquina pode ajudar no ensino de humanos

Resumindo, o aprendizado de máquina é ótimo para:

- Problemas para os quais as soluções atuais exigem muitos ajustes finos ou extensas listas de regras: um modelo de aprendizado de máquina geralmente simplifica e tem um desempenho melhor do que a abordagem tradicional.
- Problemas complexos para os quais não existe uma boa solução quando utilizamos uma abordagem tradicional: talvez as melhores técnicas de aprendizado de máquina encontrem uma solução.
- Flutuação de ambientes: um sistema de aprendizado de máquina pode ser treinado com facilidade em novos dados, sempre mantendo-os atualizados.
- Entendimento de problemas complexos e grandes quantidades de dados.

Exemplos de Aplicações

Vejamos alguns exemplos reais de tarefas ML e suas respectivas técnicas:

Análise de imagens de produtos em uma linha de produção a fim de classificá-los automaticamente

Classificação de imagens normalmente realizada usando redes neurais convolucionais (CNNs; veja o Capítulo 14) ou, às vezes, transformadores (Capítulo 16).

Detecção de tumores em exames de imagens cerebrais

Segmentação da imagem semântica, em que cada pixel da imagem é classificado (quando queremos determinar a localização exata e a forma dos tumores), geralmente usando também CNNs ou transformadores.

Classificação automática de artigos de notícias

Processamento de linguagem natural (PNL) e, mais especificamente, classificação de texto, que pode ser abordada utilizando redes neurais recorrentes (RNNs) e CNNs, mas os transformadores são ainda melhores (Capítulo 16).

Sinalização automática de comentários ofensivos em fóruns de discussão

Classificação de texto, usando as mesmas ferramentas de PNL.

Resumo automático de documentos extensos

Ramo da PNL chamado resumo de texto, novamente usando as mesmas ferramentas.

Criação de um chatbot ou um assistente pessoal

Envolve muitos componentes da PNL, dentre eles a compreensão de linguagem natural (NLU) e módulos de resposta a perguntas.

Previsão do faturamento da empresa no próximo ano, com base em muitas métricas de desempenho

Tarefa de regressão (ou seja, predição de valores) que pode ser abordada usando qualquer modelo de regressão: modelo de regressão linear ou regressão polinomial (Capítulo 4), modelo SVM de regressão (Capítulo 5), floresta aleatória de regressão (Capítulo 7) ou rede neural artificial (Capítulo 10). Se quiser englobar as sequências de métricas de desempenho anteriores, use RNNs, CNNs ou transformadores (Capítulos 15 e 16).

Fazer seu app responder a comandos de voz

Reconhecimento de voz que exige o processamento de amostras de áudio: como são sequências longas e complexas, normalmente são processadas usando RNNs, CNNs ou transformadores (Capítulos 15 e 16).

Detecção de fraudes com cartão de crédito

Detecção de anomalia, que pode ser abordada usando florestas de isolamento, modelos de mistura de gaussianas (Capítulo 9) e autoencoders (Capítulo 17).

Segmentação de clientes com base em suas compras, para que você possa elaborar uma estratégia de marketing diferente para cada segmento

Uso da clusterização, que pode ser feita usando *k*-means, DBSCAN e outros (Capítulo 9).

Representação de um conjunto de dados complexos e de alta dimensão em um diagrama claro e criterioso

Visualização de dados, em geral, engloba técnicas de redução de dimensionalidade (Capítulo 8).

Recomendação de um produto no qual um cliente possa se interessar, com base em compras anteriores

Sistema de recomendação. Uma das abordagens é fornecer os dados das compras anteriores (e outras informações sobre o cliente) a uma rede neural artificial (Capítulo 10) e fazer com que ela evidencie a próxima compra mais provável. Essa rede neural normalmente seria treinada em sequências anteriores de compras de todos os clientes.

Criar bot inteligente para um jogo

Via de regra, usa-se o aprendizado por reforço (RL; veja o Capítulo 18), que é um ramo do aprendizado de máquina que treina agentes (como bots) a fim de escolher as ações que maximizarão suas recompensas ao longo do tempo (por exemplo, um bot pode receber uma recompensa sempre que o jogador perde alguns pontos de vida) em determinado ambiente (como o jogo). O famoso programa AlphaGo, que derrotou o campeão mundial no jogo Go, foi desenvolvido usando RL.

Essa lista é praticamente infinita, mas espero que você tenha uma ideia do alcance e da complexidade impressionante das tarefas que o aprendizado de máquina pode abordar e as técnicas usadas para cada tarefa.

Tipos de Sistemas do Aprendizado de Máquina

Existem tantos tipos diferentes de sistemas de aprendizado de máquina que ajuda e muito classificá-los em categorias amplas, com base nos seguintes critérios:

- Serem ou não treinados com supervisão humana (aprendizado supervisionado, não supervisionado, semissupervisionado, autossupervisionado e outros).
- Se podem ou não aprender gradativamente em tempo real (aprendizado online vs. em batch).
- Se funcionam simplesmente comparando novos pontos de dados com pontos de dados conhecidos ou se detectam padrões em dados de treinamento e criam um modelo preditivo, como os cientistas (aprendizado baseado em instância vs. baseado em modelo).

Esses critérios não são restritivos; você pode combiná-los como quiser. Por exemplo, um filtro de spam de última geração pode aprender instantaneamente com a utilização de um modelo de rede neural profundo treinado a partir de exemplos de spam e ham, o que faz dele um sistema de aprendizado supervisionado online, baseado em modelos.

Analisaremos cada um desses critérios com um pouco mais de atenção.

Supervisão com Treinamento

Os sistemas ML podem ser classificados de acordo com a quantidade e o tipo de supervisão que recebem durante o treinamento. Existem muitas categorias, mas veremos as principais: supervisionado, não supervisionado, autossupervisionado, semissupervisionado e aprendizado por reforço.

Aprendizado supervisionado

No *aprendizado supervisionado*, o conjunto de treinamento que você fornece ao algoritmo inclui as soluções desejadas, chamadas de *rótulos* ou *labels* (Figura 1-5).

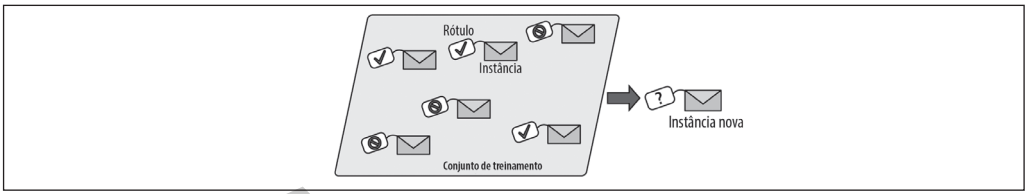

Figura 1-5. Um conjunto de treinamento rotulado para classificação de spam (um exemplo de aprendizado supervisionado)

A *classificação* é uma típica tarefa de aprendizado supervisionado. O filtro de spam é um bom exemplo: ele é treinado com muitos exemplos de e-mails junto com suas *classes* (spam ou ham) e deve aprender a classificar os e-mails novos.

Outra tarefa típica é prever um valor numérico *alvo* [*target*], como o preço de um carro, dado um conjunto de *características* (quilometragem, tempo de uso, marca etc.). Esse tipo de tarefa se chama *regressão* (Figura 1-6).[1] Para treinar o sistema, é necessário fornecer muitos exemplos de carros, incluindo seus preditores e rótulos (ou seja, seus preços).

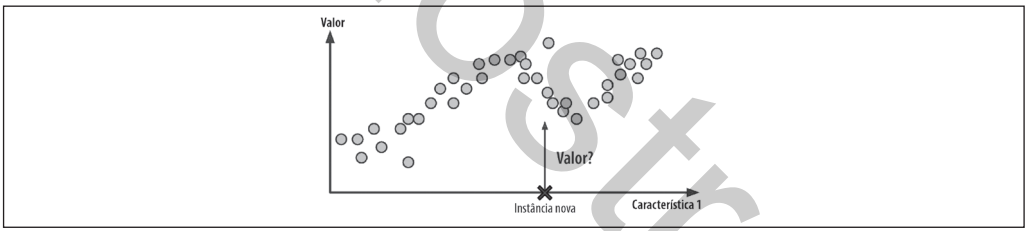

Figura 1-6. Um problema de regressão: prever um valor, dada uma característica (geralmente existem diversas entradas de característica e, às vezes, diversos valores de saída)

As palavras *alvo* e *rótulo* costumam ser tratadas como sinônimas no aprendizado supervisionado, porém *alvo* é mais comum nas tarefas de regressão e *rótulo* é mais usual nas tarefas de classificação. Além disso, *características* por vezes são chamadas de *preditores* ou *atributos*. Esses termos podem ser referir a amostras individuais (ex.: "quilometragem do carro = 15.000") ou a todas as amostras (ex.: "quilometragem está fortemente correlacionada a preço").

Observe que alguns algoritmos de regressão também podem ser utilizados para a classificação e vice-versa. Por exemplo, a *regressão logística* é comumente utilizada para a classificação, pois consegue gerar um valor correspondente à probabilidade de pertencer a determinada classe (por exemplo, 20% de chance de ser spam).

1 Curiosidade: esse nome esquisito é um termo estatístico cunhado por Francis Galton, ao estudar o fato de que os filhos de pessoas altas costumam ter a estatura mais baixa que os pais. Como a estatura das crianças era menor, ele chamou de *regressão à média*, nome dado aos métodos usados para analisar as correlações entre as variáveis.

Aprendizado não supervisionado

No *aprendizado não supervisionado*, como você pode imaginar, os dados de treinamento não são rotulados (Figura 1-7). O sistema tenta aprender sem um professor.

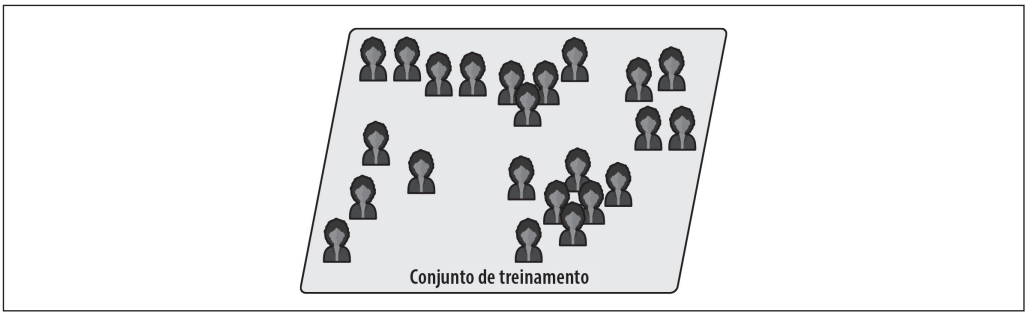

Figura 1-7. Conjunto de treinamento não rotulado para o aprendizado não supervisionado

Por exemplo, digamos que você tenha muitos dados sobre os visitantes do seu blog. Você quer executar um algoritmo de *clusterização* com o objetivo de detectar grupos de visitantes semelhantes (Figura 1-8). Em nenhum momento você informa ao algoritmo a qual grupo o visitante pertence: ele encontrará essas relações sem sua ajuda. Por exemplo, ele pode observar que 40% dos visitantes são adolescentes que adoram histórias em quadrinhos e em geral leem seu blog depois da escola, enquanto 20% são adultos amantes de ficção científica e o visitam nos finais de semana. Se você utilizar um algoritmo de *clusterização hierárquica*, ele também poderá subdividir cada grupo em grupos menores. Isso pode ajudá-lo a direcionar suas postagens para cada um desses grupos.

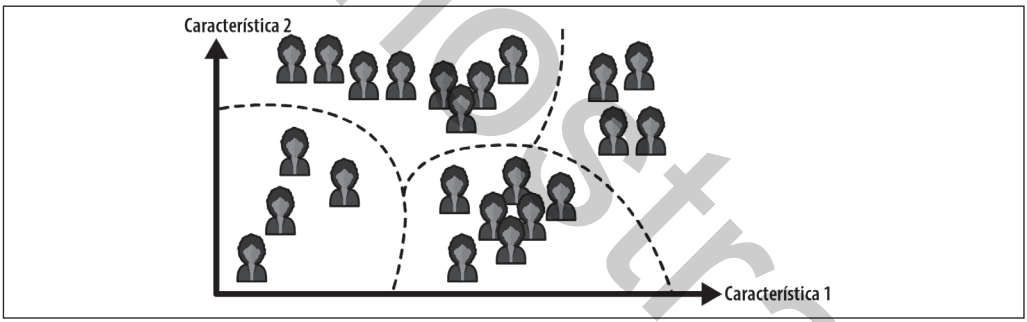

Figura 1-8. Clusterização

Os algoritmos de *visualização* também são bons exemplos de algoritmos de aprendizado não supervisionado: você lhes fornece muitos dados complexos e não rotulados, e eles geram uma representação 2D ou 3D de seus dados que podem ser facilmente plotados (Figura 1-9). Esses algoritmos tentam preservar o máximo da estrutura (por exemplo, tentam impedir que clusters separados no espaço de entrada se sobreponham na visualização), a fim de que você possa entender como os dados estão organizados e talvez identificar padrões inesperados.

A *redução da dimensionalidade* é uma tarefa relacionada cujo objetivo é simplificar os dados sem perder muita informação. Para tal, você pode mesclar diversas características correlacionadas em uma. Por exemplo, a quilometragem de um carro pode estar bastante correlacionada com seu tempo de uso, de modo que o algoritmo da redução de dimensionalidade fará a mesclagem em uma característica que representa o desgaste do carro. Isso se chama *extração de características*.

Não raro, é uma boa ideia tentar reduzir a dimensão dos dados de treinamento usando um algoritmo de redução de dimensionalidade antes de fornecê-lo a outro algoritmo ML (como um algoritmo de aprendizado supervisionado). Esse algoritmo será executado mais rapidamente, os dados ocuparão menos espaço em disco e na memória e, em alguns casos, poderão ter um melhor desempenho também.

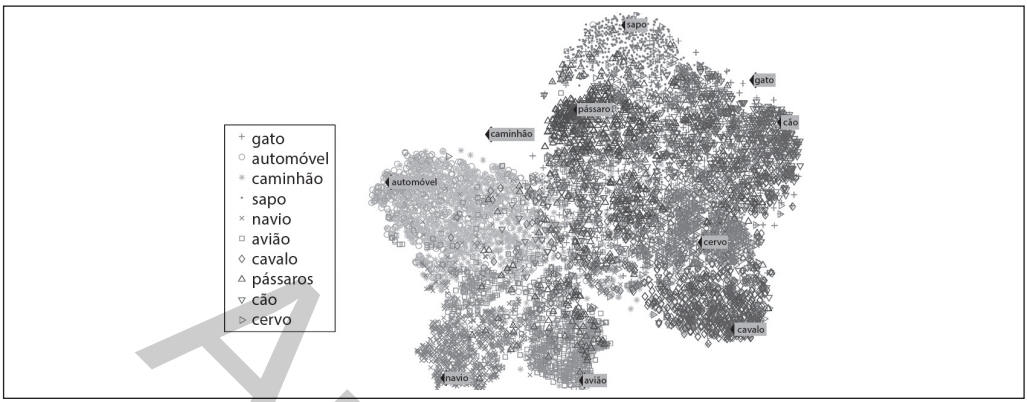

Figura 1-9. Exemplo de uma visualização t-SNE destacando clusters semânticos[2]

Outra tarefa fundamental não supervisionada é a *detecção de anomalias* — por exemplo, detectar transações incomuns em cartões de crédito para evitar fraudes, identificar defeitos de fabricação ou remover automaticamente os valores atípicos de um conjunto de dados antes de fornecê-lo a outro algoritmo de aprendizado. Na maioria das vezes, exibe-se o sistema em instâncias normais durante o treinamento, por isso ele aprende a reconhecê-las e, quando vê uma instância nova, é capaz de afirmar se ela parece normal ou é uma provável anomalia (veja a Figura 1-10). Uma tarefa bem semelhante é a *detecção de novidade*: visa detectar instâncias novas que parecem diferentes de todas as instâncias no conjunto de treinamento. Isso exige um conjunto de treinamento muito "limpo", desprovido de qualquer instância que você gostaria que o algoritmo detectasse. Por exemplo, se você tem milhares de fotos de cães e 1% delas representa Chihuahuas, um algoritmo de detecção de novidade não deve tratar as novas fotos de Chihuahuas como novidades. Em contrapartida, os algoritmos de detecção de anomalias podem considerar esses cães tão raros e diferentes de outros que provavelmente os classificariam como anomalias (nada contra os Chihuahuas).

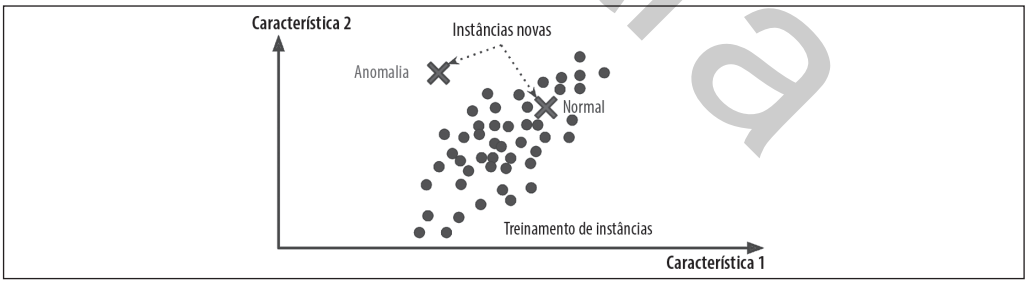

Figura 1-10. Detecção de anomalia

Finalmente, outra tarefa comum não supervisionada é o *aprendizado de regras por associação*, cujo objetivo é investigar grandes quantidades de dados e descobrir relações interessantes entre os atributos. Por exemplo,

2 Observe como os animais estão bem separados dos veículos, como os cavalos estão próximos dos cervos, mas longe dos pássaros, e assim por diante. Imagem reproduzida com permissão de Richard Socher *et al.*, "Zero-Shot Learning Through CrossModal Transfer", *Proceedings of the 26th International Conference on Neural Information Processing Systems* 1 (2013): 935–943.

suponha que você seja dono de um supermercado. Executar uma regra de associação em seus registros de vendas pode revelar que as pessoas que compram molho barbecue e batatas fritas também estão propensas a comprar carnes. Assim sendo, talvez você queira colocar esses itens próximos.

Aprendizado semissupervisionado

Como rotular os dados geralmente consome tempo e dinheiro, você terá uma grande quantidade de instâncias não rotuladas e poucas rotuladas. Alguns algoritmos podem lidar com dados parcialmente rotulados. Isso se chama *aprendizado semissupervisionado* (Figura 1-11).

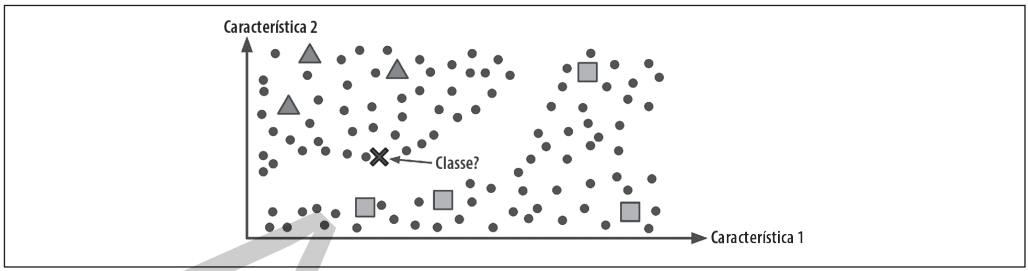

Figura 1-11. Aprendizado semissupervisionado com duas classes (triângulos e quadrados): os exemplos não rotulados (círculos) ajudam a classificar uma instância nova (cruz) na classe triângulo em vez de na classe quadrado, ainda que esteja mais próxima dos quadrados rotulados

Alguns serviços de hospedagem de fotos, como o Google Fotos, são bons exemplos. Ao fazer o upload de todas as fotos de família, o aplicativo reconhecerá automaticamente que a mesma pessoa A aparece nas fotos 1, 5 e 11, enquanto outra pessoa B aparece nas fotos 2, 5 e 7. Essa é a parte não supervisionada do algoritmo (clusterização). Agora, o sistema apenas precisa que você informe quem são essas pessoas. Acrescente somente um rótulo por pessoa[3] e ele conseguirá nomear todas, o que é útil para pesquisar fotos.

A maior parte dos algoritmos de aprendizado semissupervisionado são combinações de algoritmos supervisionados e não supervisionados. Por exemplo, um algoritmo de clusterização pode ser usado para agrupar instâncias semelhantes, então cada instância sem rótulo pode ser rotulada com o rótulo mais comum em seu cluster. Uma vez que todo o conjunto de dados é rotulado, é possível usar qualquer algoritmo de aprendizado supervisionado.

Aprendizado autossupervisionado

Outra abordagem para o aprendizado de máquina envolve realmente gerar um conjunto de dados a partir de um conjunto de dados totalmente rotulado. De novo, uma vez que todo o conjunto de dados é rotulado, o algoritmo de aprendizado supervisionado pode ser usado. Essa abordagem é chamada de *aprendizado autossupervisionado*.

Por exemplo, se você tem um grande conjunto de dados de imagens sem rótulo, pode mascarar aleatoriamente uma pequena parte de cada imagem e, em seguida, treinar um modelo para recuperar a imagem original (Figura 1-12). Durante o treinamento, as imagens mascaradas são usadas como entradas para o modelo e as imagens originais são usadas como os rótulos.

3 Isso quando o sistema funciona perfeitamente. Na prática, muitas vezes ele cria alguns clusters por pessoa e, outras, confunde duas pessoas parecidas; é preciso fornecer alguns rótulos por pessoa e limpar manualmente alguns clusters.

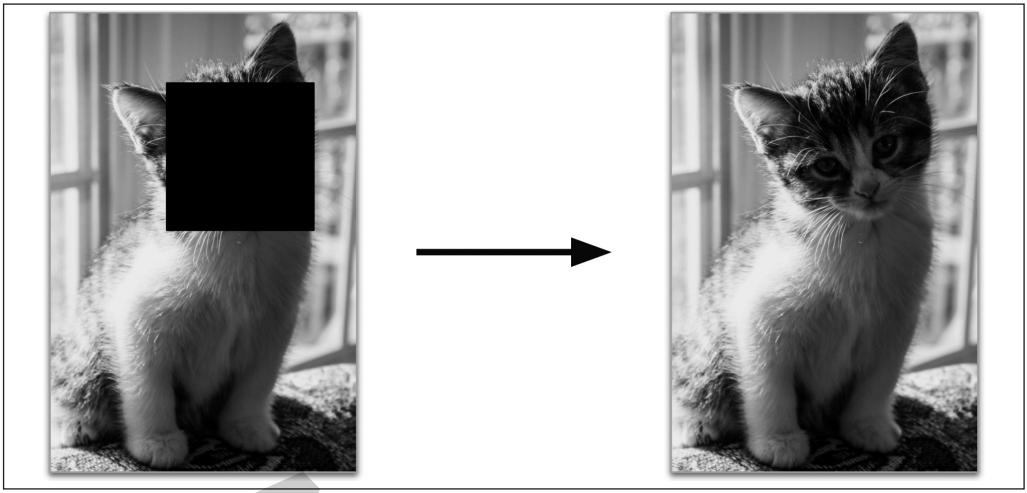

Figura 1-12. Exemplo de aprendizado autossupervisionado: entrada (esquerda) e alvo (direita).

O modelo resultante pode ser bem útil por si só — por exemplo, reparar danos em imagens ou apagar objetos indesejados nelas. Mas muitas vezes, um modelo treinado usando o aprendizado autossupervisionado não é o objetivo final. Você geralmente desejará ajustar e aprimorar o modelo para uma tarefa um pouco diferente, uma realmente importante na sua opinião.

Por exemplo, suponha que o que você realmente quer é ter um modelo de classificação de pets: dada uma foto de qualquer pet, ele dirá a espécie. Se você tem um grande conjunto de dados de fotos de pets não rotuladas, pode começar treinando um modelo de reparação de imagem usando o aprendizado autossupervisionado. Assim que funcionar bem, deve ser capaz de distinguir as diferentes espécies: quando repara a imagem de um gato cuja cara está mascarada, ele não deve adicionar a cara de um cão. Supondo que a arquitetura do seu modelo permite isso (e a maioria das arquiteturas de redes neurais permite), é possível então ajustar o modelo para prever espécies de pets em vez de reparar imagens. A etapa final consiste em ajustar o modelo em um conjunto de dados rotulado: o modelo já sabe como são gatos, cães e outras espécies, então essa etapa é necessária apenas para o modelo aprender o mapeamento entre as espécies que já conhece e os rótulos esperados.

Transferir conhecimento de uma tarefa para outra é chamado de *aprendizado de transferência*, e é uma das técnicas mais importantes no aprendizado de máquina hoje, sobretudo ao utilizar *redes neurais profundas* (ou redes neurais compostas de muitas camadas de neurônios). Detalharemos isso na Parte II.

Algumas pessoas consideram o aprendizado autossupervisionado uma parte do aprendizado não supervisionado, uma vez que trata de conjuntos de dados totalmente não rotulados. Mas o aprendizado autossupervisionado usa rótulos (gerados) durante o treinamento, e nesse sentido é mais semelhante ao aprendizado supervisionado. E o termo "aprendizado não supervisionado" é geralmente usado quando lidamos com tarefas como clusterização, redução da dimensionalidade ou detecção de anomalias, já o aprendizado autossupervisionado foca as mesmas tarefas do aprendizado supervisionado: principalmente a classificação e a regressão. Em suma, é melhor tratar o aprendizado autossupervisionado como uma categoria própria.

Aprendizado por reforço

O *aprendizado por reforço* é bem diferente. O sistema de aprendizado, chamado de *agente* nesse contexto, pode observar o ambiente, selecionar e executar ações, e obter *recompensas* em troca (ou *penalidades* na forma de recompensas negativas, como na Figura 1-13). Ele deve aprender sozinho qual é a melhor estratégia, chamada de *política*, para obter o maior número de recompensas ao longo do tempo. Uma política define qual ação o agente deve escolher quando está em determinada situação.

Por exemplo, muitos robôs implementam algoritmos de aprendizado por reforço para aprender a andar. O programa AlphaGo da DeepMind também é um bom exemplo de aprendizado por reforço: ele apareceu nas manchetes em maio de 2017 quando venceu o campeão mundial Ke Jie no jogo Go. Ele aprendeu sua política de vitória analisando milhões de jogos e depois jogando contra si mesmo. Repare que o aprendizado foi desativado durante os jogos contra o campeão. O AlphaGo usou somente a política que havia aprendido. Como você verá na próxima seção, isso se chama *aprendizado offline*.

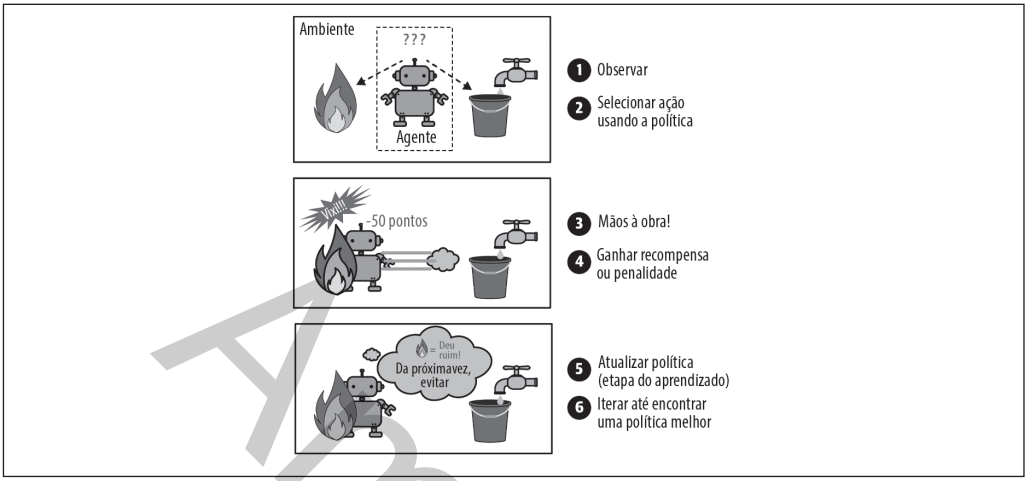

Figura 1-13. Aprendizado por reforço

Aprendizado em Batch versus Online

Outro critério utilizado para classificar os sistemas de aprendizado de máquina é se o sistema pode ou não aprender de forma incremental a partir da entrada de um fluxo de dados.

Aprendizado em batch

No *aprendizado em batch*, o sistema é incapaz de aprender de forma incremental: ele deve ser treinado usando todos os dados disponíveis. Via de regra, isso demandará muito tempo e recursos computacionais, portanto normalmente é realizado offline. Primeiro, o sistema é treinado, depois é implementado em produção e roda sem aprender mais nada, somente aplicando o que aprendeu. Isso se chama *aprendizado offline*.

Infelizmente, o desempenho de um modelo tende a cair aos poucos com o tempo, simplesmente porque o mundo continua a evoluir enquanto o modelo permanece inalterado. Esse fenômeno é muitas vezes chamado de *degradação do modelo* ou *deriva de dados*. A solução é retreinar regularmente o modelo com dados atualizados. Com que frequência você precisa fazer isso depende do caso de uso: se o modelo classifica imagens de gatos e cães, seu desempenho decairá muito lentamente, mas se o modelo lida com sistemas em rápida evolução, por exemplo, fazendo previsões no mercado financeiro, então é provável que decairá com mais rapidez.

Mesmo um modelo treinado para classificar imagens de cães e gatos pode precisar ser retreinado regularmente, não porque esses animais sofrerão mutações da noite para o dia, mas porque as câmeras continuam mudando, junto com os formatos de imagem, a nitidez, o brilho e as proporções de tamanho. Além disso, as pessoas podem gostar de raças diferentes no ano seguinte ou decidir vestir seus pets com chapéus minúsculos — vai saber!

Caso deseje que um sistema de aprendizado em batch tenha acesso a dados novos (como um novo tipo de spam), você precisa treinar uma nova versão do sistema a partir do zero no conjunto completo de dados (e não apenas com os dados novos, mas também com os antigos), em seguida, descontinuar o sistema antigo e substituí-lo pelo novo. Por sorte, todo o processo de treinamento, avaliação e disponibilização de um sistema de aprendizado de máquina pode ser facilmente automatizado (como na Figura 1-3). Assim sendo, mesmo um sistema de aprendizado em batch pode se adaptar às mudanças. Basta atualizar os dados e treinar uma nova versão do sistema a partir do zero sempre que necessário.

É uma solução simples que geralmente funciona bem, mas treinar usando o conjunto completo de dados pode levar horas; portanto, você normalmente treina um novo sistema apenas a cada 24 horas ou semanalmente. Caso seu sistema precise se adaptar a dados que mudam rapidamente (por exemplo, prever os preços das ações), é preciso uma solução mais responsiva.

Além disso, o treinamento no conjunto completo de dados exige mais recursos computacionais (CPU, espaço de memória, espaço em disco, E/S do disco, E/S de rede etc.). Se você tem muitos dados e seu sistema é automatizado para treinar todos os dias a partir do zero, esse processo custará uma fortuna. Se a quantidade de dados for gigantesca, talvez seja impossível utilizar um algoritmo de aprendizado em batch.

Por fim, caso o sistema precise aprender de forma autônoma e tenha recursos limitados (por exemplo, um aplicativo de smartphone ou um veículo do tipo rover explorando a superfície de Marte), carregar grandes quantidades de dados de treinamento e usar muitos recursos para treinar por dias e horas a fio provocaria um bug do tipo showstopper.

Uma opção melhor em todos esses casos seria utilizar algoritmos capazes de aprender de forma incremental.

Aprendizado online

No *aprendizado online*, é possível treinar o sistema incrementalmente, fornecendo as instâncias de dados de forma sequencial, individual ou em pequenos grupos, chamados de *minibatches*. Cada etapa do aprendizado é rápida e tem um custo baixo, assim o sistema pode aprender instantaneamente os dados novos em tempo real, assim que eles entram (veja a Figura 1-14).

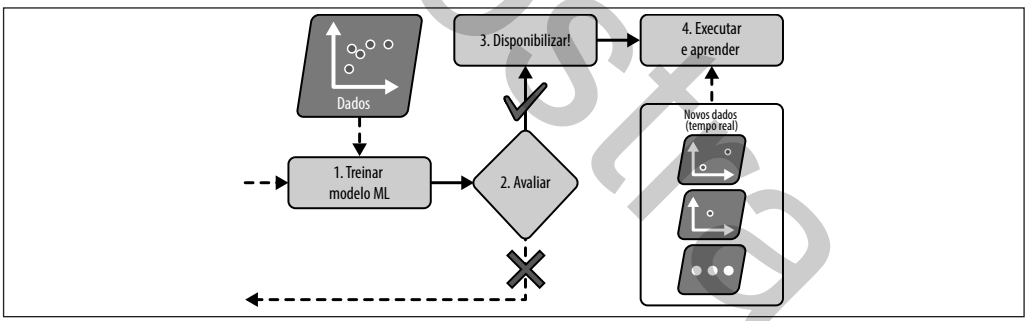

Figura 1-14. No aprendizado online, um modelo é treinado e colocado em produção, em seguida, continua aprendendo à medida que novos dados são recebidos

O aprendizado online é útil para sistemas que precisam se adaptar a mudanças rápidas e extremas (ex.: detectar novos padrões no mercado de ações). Também é uma boa opção se seus recursos computacionais são limitados; por exemplo, se o modelo é treinado em um dispositivo móvel.

E mais, os algoritmos de aprendizado online também podem ser utilizados para treinar modelos em grandes conjuntos de dados que não cabem na memória principal de uma máquina (isso se chama aprendizado *out-of-core*). O algoritmo carrega parte dos dados, executa uma etapa do treinamento nesses dados e repete o processo até ser executado em todos os dados (veja a Figura 1-15).