"Pedro Domingos desmistifica o machine learning e mostra como o futuro será surpreendente e empolgante." – Walter Isaacson

O ALGORITMO MESTRE

COMO A BUSCA PELO ALGORITMO DE MACHINE LEARNING DEFINITIVO RECRIARÁ NOSSO MUNDO

novatec

Pedro Domingos

Elogios prévios a *O Algoritmo Mestre*

"Este é um livro muito importante e útil. O machine learning já é crucial em nossas vidas e no trabalho, e isso só aumentará. Pedro Domingos escreveu sobre o assunto de maneira clara e inteligível".

– *THOMAS H. DAVENPORT, professor ilustre da Babson College e autor de Competing on Analytics e Big data no trabalho*

"O machine learning, conhecido nos meios comerciais como análise preditiva, está mudando o mundo. Este livro empolgante, abrangente e inspirador introduz os conceitos científicos profundos até mesmo a leitores não técnicos, além de trazer para os especialistas uma perspectiva nova e intensa, que revela as direções mais promissoras da pesquisa. É realmente uma obra rara".

– *ERIC SIEGEL, fundador da Predictive Analytics World e autor de Predictive Analytics*

"O machine learning oferece um mundo fascinante nunca antes vislumbrado por leigos. Pedro Domingos o iniciará nas misteriosas linguagens faladas por suas cinco tribos e o convidará para se associar ao plano de uni-las, criando a tecnologia mais poderosa que nossa civilização já viu".

– *SEBASTIAN SEUNG, professor em Princeton e autor de Connectome*

"[Uma] introdução ao machine learning entusiasmada, de alto nível, lúcida e consistentemente informativa. Com destreza, visão e conhecimento, Domingos descreve como os cientistas estão criando programas que permitem que um computador ensine a si próprio. Os leitores terão insights fascinantes".

– *Kirkus Reviews*

Editor: Rubens PratesTradução: Aldir José Coelho Corrêa da Silva Revisão gramatical: Priscila A. Yoshimatsu Editoração eletrônica: Carolina Kuwabata Capa: Carolina Kuwabata
ISBN: 978-85-7522-542-4
Histórico de edições impressas:
Janeiro/2017 Primeira edição
Novatec Editora Ltda.
Rua Luís Antônio dos Santos 110
02460-000 – São Paulo, SP – Brasil Tel.: +55 11 2959-6529
E-mail: *novatec@novatec.com.br*
Site: *www.novatec.com.br*
Twitter: *twitter.com/novateceditora*
Facebook: *facebook.com/novatec*

O ALGORITMO MESTRE

(The Master Algorithm)

COMO A BUSCA PELO ALGORITMO DE MACHINE LEARNING DEFINITIVO RECRIARÁ NOSSO MUNDO

Pedro Domingos

Novatec

À memória de minha irmã Rita,
que perdeu a batalha contra o câncer enquanto eu escrevia este livro

A grande meta da ciência é abordar o maior número possível de fatos experimentais por dedução lógica a partir do menor número de hipóteses ou axiomas.

– Albert Einstein

A civilização avança ao expandir o número de operações importantes que podemos executar sem pensar.

– Alfred North Whitehead

Sumário

Prólogo

Talvez você não saiba, mas o machine learning está em todos os locais ao seu redor. Quando digitamos uma consulta em um mecanismo de busca, é dessa forma que o mecanismo define os resultados que deve exibir (e também os anúncios). Quando lemos emails, não vemos grande parte do spam, porque o machine learning desconsidera essas mensagens. Quando acessamos a Amazon.com para comprar um livro ou a Netflix para assistir a um vídeo, um sistema de machine learning recomenda outros que possam nos interessar. O Facebook usa o machine learning para decidir quais atualizações exibirá, e o Twitter faz o mesmo com os tuítes. Sempre que você usar um computador, provavelmente o machine learning estará envolvido em algum momento.

Tradicionalmente, a única maneira de fazer um computador executar uma operação – desde somar dois números a pilotar um avião – era escrever um algoritmo que explicasse como, com detalhes minuciosos. Porém, os algoritmos de machine learning, também conhecidos como aprendizes, são diferentes: eles descobrem tudo sozinhos, fazendo inferências a partir de dados. E quanto mais dados têm, melhor ficam. Atualmente, não precisamos programar os computadores; eles mesmos se programam.

Isso não ocorre apenas no ciberespaço: todo o seu dia, do momento em que você acorda até a hora em que vai dormir, tem a ajuda do machine learning.

Seu rádio-relógio o desperta às 7:00. Está tocando uma canção que você não conhece, mas que está gostando. Por cortesia da rádio personalizada Pandora, ele está aprendendo o que você aprecia em música, como se fosse seu DJ pessoal. Talvez a canção também tenha sido produzida com a ajuda de machine learning. Você toma o café da manhã e lê o jornal, que saiu da máquina de impressão há algumas horas, com o processo de impressão cuidadosamente ajustado por intermédio do machine learning para evitar riscos. A temperatura em sua casa está agradável e sua conta de energia é baixa, já que você instalou um termostato inteligente Nest.

Quando você dirige para o trabalho, seu carro ajusta continuamente a injeção de combustível e a recirculação de gases de escape para otimizar o uso da gasolina. Você emprega o Inrix, um sistema de informação de tráfego, para encurtar a viagem na hora do rush, o que também ajuda a baixar o nível de

estresse. No trabalho, o machine learning o ajuda a combater a sobrecarga de informações. Você usa um cubo de dados para resumir massas de informações, examiná-las de todos os ângulos e se aprofundar nas partes mais importantes. É preciso tomar uma decisão: seria o layout A ou o B a trazer mais negócios para o seu site? Um sistema de machine learning faz testes com os dois e retorna informações. Você precisa verificar o site de um possível fornecedor, mas ele está em um idioma estrangeiro. Sem problemas: o Google faz a tradução automaticamente. Seus emails são convenientemente classificados em pastas, restando apenas as mensagens mais importantes na caixa de entrada. Seu editor de texto verifica a gramática e a ortografia. Você encontra um voo para uma futura viagem, mas ainda não faz a compra da passagem porque o Bing Travel prevê que o preço cairá em breve. Sem perceber, você consegue fazer muito mais coisas todo os dias, muito mais do que faria sem a ajuda do machine learning.

Durante um intervalo, você verifica seus fundos mútuos. A maioria usa algoritmos de aprendizado para a seleção de ações, e um deles é totalmente gerenciado por um sistema de aprendizado. Na hora do almoço, você caminha pela rua, com o smartphone na mão, à procura de um lugar para almoçar. O sistema de aprendizado Yelp o ajuda a encontrá-lo. Seu celular está cheio de algoritmos de aprendizado. Eles são rigorosos na correção de seus erros de digitação, na compreensão de seus comandos de voz, na redução de erros de transmissão, no reconhecimento de códigos de barra e muito mais. Seu telefone pode até mesmo antecipar o que você vai fazer e aconselhá-lo de acordo. Por exemplo, quando você termina de almoçar, ele o avisa discretamente que sua reunião da tarde com um visitante de fora da cidade terá de começar mais tarde porque o voo atrasou.

Já é noite quando você sai do trabalho. O machine learning o ajuda a se sentir seguro enquanto vai até o carro, monitorando o vídeo da câmera de vigilância do estacionamento e alertando a equipe de segurança se detectar alguma atividade suspeita. No caminho para casa, você para no supermercado e caminha pelos corredores, que foram dispostos com a ajuda de algoritmos de aprendizado para decidir: quais mercadorias devem ser armazenadas, o que deve ser oferecido no fim dos corredores e se a salsa deve ser colocada na seção de temperos ou perto dos chips de tortilla. Você paga com cartão de crédito. Um algoritmo de aprendizado foi que decidiu te oferecer este cartão e aprovou sua solicitação.

Outro procura transações suspeitas e avisa se achar que o número de seu cartão foi roubado. Um terceiro algoritmo tenta estimar quanto você está satisfeito com o cartão. Se você for um bom cliente, mas não estiver satisfeito, receberá uma oferta atraente antes que mude para outro.

Você chega em casa e vai até a caixa de correio. Encontra uma carta enviada por um amigo, direcionada por um algoritmo de aprendizado que lê endereços escritos à mão. Também há o lixo usual, selecionado por outros algoritmos de aprendizado (nem tudo são flores). Você para por um momento para apreciar a fria brisa noturna. O crime na cidade chegou a um nível muito baixo desde que a polícia começou a usar o aprendizado estatístico para prever em que locais há maior probabilidade de ocorrer crimes e a concentrar rondas policiais nesses locais. Você janta com sua família. O prefeito está no noticiário. Você votou nele porque o próprio lhe telefonou no dia da eleição, após um algoritmo de aprendizado tê-lo detectado como um eleitor indeciso. Após o jantar, você assiste a um jogo de futebol. As duas equipes selecionaram seus jogadores com a ajuda do aprendizado estatístico. Ou talvez você jogue em seu Xbox com as crianças, e o algoritmo de aprendizado do Kinect descobrirá onde você está e o que está fazendo. Antes de dormir, você toma seu remédio, que foi projetado e testado com a ajuda de mais algoritmos de aprendizado. Seu médico também pode ter usado o machine learning no diagnóstico, seja para interpretar raios X ou descobrir um conjunto incomum de sintomas.

O machine learning está presente em cada estágio de sua vida. Se você estudou online para o exame SAT de admissão para a universidade, um algoritmo de aprendizado deu nota aos seus trabalhos. E se você tentou entrar para a escola de negócios e fez o exame GMAT recentemente, um de seus avaliadores foi um sistema de aprendizado. É possível que, quando se candidatou a um emprego, um algoritmo de aprendizado tenha selecionado seu currículo na pilha virtual e informado ao empregador: este é um forte candidato; dê uma olhada neste currículo. Seu último aumento pode ter sido cortesia de outro algoritmo de aprendizado. Se estiver pretendendo comprar uma casa, o site Zillow.com pode estimar qual vale a pena entre as que você está considerando. Quando tiver se decidido por uma, você tentará obter um empréstimo para a compra da casa, e um algoritmo de aprendizado estudará sua proposta e recomendará (ou não) sua aceitação. E o que talvez seja o mais importante, se você usou um serviço de namoro online, o machine learning pode tê-lo ajudado até mesmo a encontrar o

amor de sua vida.

A sociedade está mudando ao ritmo de cada algoritmo de aprendizado que é produzido. O machine learning está recriando a ciência, a tecnologia, os negócios, a política e a guerra. Satélites, sequenciadores de DNA e aceleradores de partículas sondam a natureza em detalhes cada vez menores, e os algoritmos de aprendizado transformam as torrentes de dados em novo conhecimento científico. As empresas conhecem seus clientes como jamais conheceram. O candidato com os melhores modelos de eleitores vence, como Obama contra Romney. Veículos não tripulados pilotam a si próprios na terra, no mar e no ar. Ninguém programou nossas preferências no sistema de recomendações da Amazon; um algoritmo de aprendizado as descobriu sozinho, tirando conclusões a partir de compras passadas. O carro autodirigível do Google aprendeu sozinho como permanecer na estrada; nenhum engenheiro escreveu um algoritmo para instruí-lo, passo a passo, como ir de A a B. Ninguém sabe como programar um carro para dirigir sozinho, e não precisamos saber, porque um carro equipado com um algoritmo de aprendizado aprende observando o que o motorista faz.

O machine learning é algo novo em nossas vidas: é uma tecnologia que constrói a si própria. Desde que nossos ancestrais começaram a afiar pedras para transformá-las em ferramentas, os humanos têm projetado artefatos, sejam construídos manualmente ou produzidos em massa. Porém, os algoritmos de aprendizado são artefatos que projetam outros artefatos. “Os computadores são inúteis”, dizia Picasso. “Eles só podem nos dar respostas”. Os computadores não deveriam ser criativos; deveriam fazer o que lhes disséssemos para fazer. Se lhes pedirmos para ser criativos, teremos o machine learning. Um algoritmo de aprendizado é como um mestre artesão: cada uma de suas produções é diferente e adaptada primorosamente às necessidades do cliente. No entanto, em vez de transformar pedras em alvenaria ou ouro em joias, os aprendizes transformam dados em algoritmos. E quanto mais dados eles têm, mais intrincados são os algoritmos.

O *Homo sapiens* é a espécie que adapta o mundo às suas necessidades e não uma espécie que se adapta ao mundo. O machine learning é o mais novo capítulo desta saga de milhões de anos: com ele, o mundo deduz o que queremos e muda de acordo, sem levantarmos um dedo sequer. Como uma floresta mágica, os seus arredores – virtuais hoje, físicos amanhã – se reorganizam quando passamos por eles. O caminho que escolhemos entre as árvores e os arbustos se transforma em

uma estrada. Sinais apontando a direção surgem nos locais em que nos perdemos.

Essas tecnologias aparentemente mágicas funcionam porque a essência do machine learning é a previsão: ele prevê o que queremos, os resultados de nossas ações, como atingir nossos objetivos, como o mundo mudará. No passado, essa tarefa era dos xamãs e adivinhadores, mas eles erravam muito. As previsões da ciência são mais confiáveis, porém são restritas ao que podemos observar sistematicamente e modelar com cautela. O big data e o machine learning expandem muito esse escopo. Algumas tarefas cotidianas podem ser impulsionadas pela mente sem ajuda externa, seja apanhar uma bola ou manter uma conversa. Já outras, por mais que tentemos, são imprevisíveis. Para o amplo espaço intermediário entre elas, temos o machine learning.

Paradoxalmente, até quando abrem novas janelas para a natureza e o comportamento humano, os algoritmos de aprendizado são envoltos em mistério. É difícil passarmos um dia sem uma história na mídia envolvendo o machine learning, seja o lançamento do assistente pessoal Siri da Apple, a vitória do computador Watson da IBM sobre o campeão humano de *Jeopardy!*, a descoberta de que uma adolescente estava grávida, feita pela loja Target antes de seus pais, ou a procura de conexões entre fatos pela NSA. Porém, em todos os casos o algoritmo de aprendizado que conduz a história é uma caixa preta. Até mesmo os livros sobre big data não explicam o que acontece quando o computador recebe todos esses terabytes e magicamente gera novos insights. Na melhor das hipóteses, somos deixados com a impressão de que os algoritmos apenas encontram correlações entre pares de eventos, como procurar no Google "remédio para a gripe" e estar gripado. No entanto, encontrar correlações é para o machine learning não mais que os tijolos são para as casas, e as pessoas não vivem em tijolos.

Quando uma nova tecnologia é tão difusa e revolucionária como o machine learning, não é sábio deixá-la como uma caixa preta. A opacidade abre a porta para o erro e a utilização incorreta. O algoritmo da Amazon determina melhor que qualquer pessoa quais livros são lidos atualmente. Os algoritmos da NSA descobrem se você é um terrorista. Modelos climáticos decidem qual nível de dióxido de carbono é seguro na atmosfera. Modelos de seleção de ações conduzem mais a economia do que nós próprios. Você não pode controlar o que não entende e é por isso que precisa entender o machine learning – como

cidadão, profissional e ser humano engajado na busca da felicidade.

O primeiro objetivo deste livro é apresentá-lo aos segredos do machine learning. Só engenheiros e mecânicos precisam saber como o motor de um carro funciona, mas todos os motoristas têm de saber que virar o volante muda a direção do carro e pisar no freio o faz parar. Atualmente, poucas pessoas sabem quais são os elementos correspondentes de um aprendiz, imagine então quantas conseguem usá-los. O psicólogo Don Norman cunhou o termo *modelo conceitual* para se referir ao conhecimento preliminar de uma tecnologia necessário para a usarmos eficientemente. Este livro lhe dará um modelo conceitual do machine learning.

Nem todos os algoritmos de aprendizado funcionam da mesma forma, e as diferenças têm consequências. Vejamos, por exemplo, os algoritmos de recomendações da Amazon e Netflix. Se um deles o estivesse guiando por uma livraria física, tentando determinar o que é "interessante no seu caso", provavelmente o da Amazon o levaria às prateleiras que você consultou anteriormente; o da Netflix o levaria a seções da loja desconhecidas e aparentemente estranhas, mas com itens que você acabaria adorando. Neste livro, usaremos os diferentes tipos de algoritmos que empresas como a Amazon e a Netflix empregam. O algoritmo da Netflix tem um conhecimento melhor (mesmo sendo limitado) de nossos gostos que o da Amazon, mas ironicamente isso não significa que a Amazon se sairia melhor se o usasse. O modelo de negócios da Netflix depende do direcionamento da demanda para para filmes e programas de TV obscuros, que fazem parte da cauda longa e que não são caros para ela, e de seu distanciamento dos grandes sucessos, que nossa assinatura não pagaria. A Amazon não tem esse problema; embora esteja preparada para se beneficiar da cauda longa, fica igualmente satisfeita em nos vender itens populares mais caros, que também simplificam sua logística. E nós, como clientes, estaremos mais propensos a conhecer um item singular se tivermos uma assinatura em vez de ter de pagar por ele separadamente.

Novos algoritmos são inventados às centenas todo ano, porém são baseadas nas mesmas ideias. Este livro é sobre essas ideias, que são o que você precisa conhecer para entender como o machine learning está mudando o mundo. Longe de serem um passe de mágica, e bem diferentes até mesmo de seu uso em computadores, elas são respostas para perguntas que interessam a todos nós: Como aprendemos? Há uma maneira melhor? O que podemos prever? Podemos

confiar no que aprendemos? Escolas de pensamento rivais dentro da área do machine learning têm respostas muito diferentes para essas perguntas. As principais escolas se resumem a cinco, e dedicaremos um capítulo a cada uma. Os simbolistas consideram o aprendizado como o inverso da dedução e pegam ideias da filosofia, psicologia e lógica. Os conexionistas usam o cérebro em sua engenharia reversa e são inspirados pela neurociência e física. Os evolucionários simulam a evolução no computador e se baseiam na genética e na biologia evolutiva. Os bayesianos acreditam que o aprendizado é um tipo de inferência probabilística e têm suas raízes na estatística. Os analogistas aprendem fazendo extrapolações a partir de julgamentos de semelhanças e são influenciados pela psicologia e otimização matemática. Levados pelo objetivo de construir máquinas que aprendem, percorreremos boa parte da história intelectual dos últimos cem anos e a veremos por um novo prisma.

Cada uma das cinco tribos de machine learning tem seu próprio algoritmo mestre, um aprendiz de uso geral que, em princípio, podemos usar para obter conhecimento a partir de dados de qualquer área. O algoritmo mestre dos simbolistas é a dedução inversa, o dos conexionistas é a backpropagation (retropropagação), o dos evolucionários é a programação genética, o dos bayesianos é a inferência bayesiana e o dos analogistas é a máquina de vetores de suporte. Na prática, no entanto, cada um desses algoritmos é bom para algumas coisas, mas não para outras. O que queremos é um algoritmo único que combine os recursos-chave de todos eles: o algoritmo mestre definitivo. Para algumas pessoas, esse é um sonho inatingível, mas para muitos de nós, que trabalhamos com machine learning, é o que nos dá fôlego e nos faz trabalhar até tarde da noite.

Se existir, o Algoritmo Mestre poderá derivar de dados todo o conhecimento existente no mundo – passado, presente e futuro. Sua invenção seria um dos maiores avanços da história da ciência. Ele aceleraria o progresso do conhecimento em todos os sentidos e mudaria o mundo de maneiras difíceis de imaginar. O Algoritmo Mestre é para o machine learning o que o Modelo-Padrão é para a física de partículas ou o Dogma Central é para a biologia molecular: uma teoria unificada que explica tudo que conhecemos até o momento e constrói a base para décadas ou séculos de progresso futuro. O Algoritmo Mestre é o meio de resolução de alguns dos problemas mais difíceis que enfrentamos, desde a construção de robôs domésticos até a cura do câncer.

Consideremos o câncer. É difícil curá-lo porque o câncer não é uma única doença, mas sim muitas. Tumores podem ser disparados por um enorme conjunto de causas e eles mudam ao sofrer metástase. A melhor maneira de matar um tumor é sequenciando seu genoma, descobrindo quais drogas funcionarão contra ele – sem prejudicar o paciente, e considerando seu genoma e histórico médico – e talvez até projetando uma nova droga específica para o caso. Nenhum médico domina todo o conhecimento necessário. Parece uma tarefa perfeita para o machine learning: na verdade, é uma versão mais complicada e desafiadora das buscas que a Amazon e a Netflix fazem todo dia, exceto por procurar o tratamento em vez do livro ou o filme certo. Infelizmente, embora os algoritmos de aprendizado atuais consigam diagnosticar muitas doenças com precisão super-humana, a cura do câncer ultrapassa seu alcance. Se formos bem-sucedidos em nossa busca pelo Algoritmo Mestre, não ultrapassará mais.

Consequentemente, o segundo objetivo deste livro é *habilitá-lo* a inventar o Algoritmo Mestre. Você deve estar pensando que isso demandaria muita matemática e um trabalho teórico rigoroso. Pelo contrário, o que é necessário é ter um olhar que se distancie do raciocínio matemático para revelar o padrão definitivo dos fenômenos do aprendizado; e para essa tarefa, o leigo, aproximando-se da floresta com esse olhar distanciado, está em alguns aspectos em melhor posição que o especialista, já profundamente imerso no estudo de árvores específicas. Uma vez que tivermos a solução conceitual, poderemos inserir os detalhes matemáticos; mas isso não é assunto para este livro nem é a parte mais importante. Portanto, ao visitar cada tribo, nosso objetivo será pegar a peça que ela representa no quebra-cabeça e descobrir onde ela se encaixa, cientes de que nenhum dos cegos consegue ver o elefante inteiro[1]. Especificamente, veremos o que cada tribo pode trazer para a cura do câncer e também o que está faltando. Em seguida, reuniremos passo a passo todas as peças da solução – ou melhor, em *uma* solução que ainda não será o Algoritmo Mestre, mas o mais próximo que alguém já chegou, e que, espero, seja um bom ponto de partida para sua imaginação. Também examinaremos o uso desse algoritmo como uma arma na luta contra o câncer. À medida que for lendo, sinta-se à vontade para examinar apenas superficialmente ou pular qualquer parte que achar difícil; é o cenário maior que importa, e provavelmente você aproveitará melhor essas partes se voltar a elas depois que o quebra-cabeça

estiver montado.

Faço pesquisas com machine learning há mais de vinte anos. Meu interesse na área foi despertado por um livro com um título estranho que vi em uma livraria na época de veterano na faculdade: *Artificial Intelligence*. Ele tinha apenas um capítulo curto sobre machine learning, mas após lê-lo fiquei imediatamente convencido de que o aprendizado era a chave da inteligência artificial e de que o estado de excelência era tão primitivo que talvez eu pudesse contribuir com algo. Fazendo planos para obter um MBA, entrei no programa de PhD da Universidade da Califórnia, em Irvine. Na época, o machine learning era uma pequena área obscura e a UCI tinha um dos poucos grupos de pesquisa significativos. Alguns de meus colegas desistiram por não verem muito futuro na área, mas persisti. Para mim, nada poderia ter mais impacto que ensinar os computadores a aprender: se pudéssemos fazê-lo, nos beneficiaríamos em todas as outras áreas. Quando me formei cinco anos depois, estava ocorrendo a explosão da mineração de dados, assim como o que me levou a escrever este livro. Em minha dissertação de doutorado, unifiquei o aprendizado simbólico e o analógico. Passei grande parte dos últimos dez anos unificando o simbolismo e o bayesianismo, e, mais recentemente, os dois com o conexionismo. É hora de passar para a próxima etapa e tentar estabelecer uma síntese dos cinco paradigmas.

Eu tinha vários públicos-alvo diferentes, mas sobrepostos, em mente quando escrevi o livro.

Se você está curioso para saber o que é todo esse tumulto ao redor do big data e do machine learning e suspeita que haja algo mais profundo ocorrendo além do que vemos nas publicações, está correto! Este livro é seu guia para a revolução.

Se está interessado principalmente no uso profissional do machine learning, o livro pode ajudá-lo em pelo menos seis aspectos: a se tornar um consumidor de análises mais experiente; a tirar o máximo de seus cientistas de dados; a evitar as armadilhas que frustram tantos projetos de mineração de dados; a descobrir o que você pode automatizar sem precisar de um software codificado manualmente; a reduzir a rigidez de seus sistemas de informação; e a antecipar algumas das novas tecnologias com as quais se deparará. Vi muito tempo e dinheiro serem desperdiçados em tentativas de resolver um problema com o algoritmo de aprendizado errado ou na interpretação incorreta do que o algoritmo dizia. Não é difícil evitar esses fiascos. Na verdade, só é preciso ler

este livro.

Se você é um cidadão ou um legislador preocupado com as questões sociais e políticas trazidas pelo big data e o machine learning, o livro será um guia sobre a tecnologia – o que ela abrange, para onde está nos levando, o que torna ou não possível – sem incomodá-lo com os detalhes. Da privacidade ao futuro do trabalho e à ética da guerra robotizada, veremos onde estão os problemas e como resolvê-los.

Se é um cientista ou engenheiro, o machine learning será uma arma poderosa que você vai querer ter em mãos. As velhas e testadas ferramentas estatísticas não o levarão longe na era do big data (ou até mesmo do medium data). Você precisa das habilidades não lineares do machine learning para modelar a maioria dos fenômenos de maneira precisa, pois ele traz uma nova visão científica do mundo. Atualmente, a expressão *mudança de paradigma* é usada de maneira muito casual, mas acredito que não seja exagero dizer que é ela que este livro descreve.

Se é um especialista em machine learning, você já conhece grande parte do que o livro aborda, mas também encontrará nele muitas ideias recentes, novidades históricas e exemplos e analogias úteis. Espero que o livro forneça principalmente uma nova perspectiva do machine learning e talvez até o faça começar a pensar em novas direções. As oportunidades estão por todos os lados e cabe a nós aproveitá-las, mas não devemos deixar de lado as recompensas maiores que se encontram mais além. (A propósito, espero que me desculpe pela licença poética de usar o termo *algoritmo mestre* para me referir a um aprendiz de uso geral.)

Se é um aluno de qualquer idade – um estudante do ensino médio avaliando o que irá cursar, um universitário decidindo se deve ingressar na pesquisa ou um profissional experiente considerando uma mudança de carreira –, espero que o livro desperte em você o interesse por essa fascinante área. O mundo tem poucos especialistas em machine learning, e se você decidir se juntar a nós terá não só momentos empolgantes e recompensas materiais, mas também uma oportunidade única de servir à sociedade. E se já estiver estudando machine learning, espero que o livro o ajude a conhecer o caminho a percorrer; se em sua saga você encontrar o Algoritmo Mestre, ele terá cumprido seu papel.

Por fim, mas não menos importante, se você gosta de se surpreender, o machine learning é um banquete intelectual; sinta-se convidado – RSVP!

1 N.T.: Alusão ao conto religioso sobre três cegos que não conseguiam enxergar um elefante inteiro, fazendo uma analogia com a grandeza de Deus.

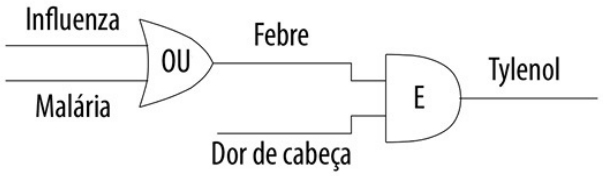
capítulo 1

A revolução do machine learning[1]

Vivemos na era dos algoritmos. Há apenas uma ou duas gerações, a simples menção da palavra *algoritmo* não significava nada para a maioria das pessoas. Atualmente, os algoritmos integram tudo que se faz no mundo civilizado. Eles fazem parte da trama que compõe nossa vida diária. Não estão apenas nos celulares ou laptops, mas nos carros, em nossa casa, nos utensílios domésticos e em brinquedos. As instituições bancárias são um imenso quebra-cabeça de algoritmos, com pessoas apertando botões do outro lado. Os algoritmos programam voos e também pilotam aeronaves. Eles gerenciam fábricas, comercializam e entregam mercadorias, calculam os lucros e mantêm registros. Se todos os algoritmos parassem de funcionar inesperadamente, o mundo que conhecemos chegaria ao fim.

Um algoritmo é uma sequência de instruções que informa ao computador o que ele deve fazer. Os computadores são compostos por bilhões de minúsculas chaves chamadas transistores, e os algoritmos ligam e desligam essas chaves bilhões de vezes por segundo. O algoritmo mais simples é: gire a chave. O estado de um transistor contém um único bit de informação: um, se o transistor estiver ativado, e zero, se estiver desativado. Um único bit em algum local dos computadores de um banco informa se nossa conta tem ou não saldo. Outro bit dos computadores da administração da previdência social informa se estamos vivos ou mortos. O segundo algoritmo mais simples é: combine dois bits. Claude Shannon, conhecido como o pai da teoria da informação, foi a primeira pessoa a entender que o que os transistores fazem, quando ligam e desligam em resposta a outros transistores, chama-se raciocínio. (Essa foi sua tese de mestrado no MIT – a mais importante tese de mestrado de todos os tempos.) Se o transistor A só liga quando os transistores B e C estão ligados, ele está envolvido em um pequeno

esforço de raciocínio lógico. Se A liga quando B ou C está ligado, essa é outra minúscula operação lógica. E se A liga sempre que B está desligado, e vice-versa, é uma terceira operação. Acredite ou não, todos os algoritmos, não importando sua complexidade, podem ser reduzidos a apenas três operações: E, OU e NÃO. Algoritmos simples podem ser representados por diagramas, com o uso de diferentes símbolos para as operações E, OU e NÃO. Por exemplo, se influenza ou malária causam febre, e Tylenol cura febre ou dor de cabeça, isso pode ser expresso assim:



Combinando várias dessas operações, podemos executar cadeias complexas de raciocínio lógico. Frequentemente as pessoas acham que os computadores só lidam com números, mas não é isso que ocorre. Os computadores são pura lógica. Os números e a aritmética são feitos de lógica, assim como tudo o mais que existe em um computador. Deseja somar dois números? Há uma combinação de transistores que faz a soma. Quer vencer o campeão do programa *Jeopardy!*? Também há uma combinação de transistores para isso (porém muito maior).

No entanto, seria proibitivamente caro se tivéssemos de construir um novo computador para cada tarefa diferente que quiséssemos executar. Em vez disso, um computador moderno é um vasto conjunto de transistores que pode fazer várias coisas, dependendo dos transistores que forem ativados. Michelangelo dizia que ele apenas via a estátua dentro do bloco de mármore e removia o excesso até ela ser revelada. Da mesma forma, um algoritmo desativa os transistores excedentes no computador até a função pretendida ser executada, seja o piloto automático de uma aeronave ou um novo filme da Pixar.

Um algoritmo não é apenas qualquer conjunto de instruções: elas têm de ser suficientemente precisas e não ambíguas para serem executadas por um computador. Por exemplo, uma receita culinária não é um algoritmo porque não especifica exatamente em que ordem as tarefas devem ser executadas ou cada etapa envolvida. Qual seria a medida exata de uma colher de açúcar? Como qualquer pessoa que já tentou preparar uma nova receita sabe, segui-la pode resultar em algo delicioso ou em uma bagunça. Por outro lado, um algoritmo sempre produz o mesmo resultado. Mesmo se uma receita especificar

precisamente meio grama de açúcar, isso ainda não resolverá o problema, porque o computador não sabe o que é açúcar ou um grama. Se quiséssemos programar um robô cozinheiro para fazer um bolo, teríamos de instruí-lo a reconhecer o açúcar a partir de um vídeo, a pegar uma colher, e assim por diante. (Ainda estamos trabalhando nisso.) O computador tem de saber como executar o algoritmo até o nível de ativar e desativar transistores específicos. Logo, uma receita culinária está longe de ser um algoritmo.

Já as instruções a seguir são o algoritmo do jogo da velha:

Se você ou seu oponente tiver duas marcações em sequência, marque o quadrado restante.
Caso contrário, se houver uma jogada que crie duas linhas com duas marcações em sequência, use-a.
Caso contrário, se o quadrado central estiver livre, marque-o.
Caso contrário, se seu oponente tiver marcado um dos cantos, marque o canto oposto.
Caso contrário, se houver um canto vazio, marque-o.
Como última alternativa, marque qualquer quadrado vazio.

Esse algoritmo tem a vantagem de nunca perder o jogo! É claro que estão faltando muitos detalhes, como a maneira pela qual o tabuleiro é representado na memória do computador e como essa representação é alterada por uma jogada. Por exemplo, poderíamos ter dois bits para cada quadrado, com o valor 00 se o quadrado estiver vazio, que é alterado para 01 quando o quadrado tem um círculo e para 10 quando tem um xis. No entanto, ele tem precisão e ausência de ambiguidade suficientes para que qualquer computador apto possa preencher os quadrados vazios. O fato de não precisarmos especificar um algoritmo até o nível dos transistores individuais também ajuda; podemos usar algoritmos existentes como ponto de partida, e há muitos deles à disposição.

Os algoritmos são um padrão minucioso. Você já deve ter ouvido falar que não entendemos realmente algo até conseguirmos expressá-lo na forma de um algoritmo. (Como Richard Feynman certa vez disse, “Não posso criar o que não entendo”.) As equações, os ingredientes básicos dos físicos e engenheiros, são na verdade apenas um tipo de algoritmo. Por exemplo, a segunda lei de Newton, talvez a mais importante de todos os tempos, diz que podemos calcular a força resultante aplicada sobre um objeto multiplicando massa e aceleração. Ela também diz implicitamente que a aceleração é a força dividida pela massa, mas tornar isso explícito é, por si só, uma etapa algorítmica. Em qualquer área da ciência, quando uma teoria não pode ser expressa como algoritmo, ela não é totalmente rigorosa. (Sem mencionar que você não poderia usar um computador para comprová-la, limitando o que seria possível fazer com ela.) Cientistas criam

teorias e engenheiros criam dispositivos. Os cientistas da computação criam algoritmos, que são ao mesmo tempo teorias e dispositivos.

Não é fácil projetar um algoritmo. Existem muitas armadilhas e nada pode ser pressuposto como certo. Algumas de nossas intuições se mostram erradas e temos de encontrar outro caminho. Além de projetar o algoritmo, temos de escrevê-lo em uma linguagem que os computadores consigam entender, como Java ou Python (momento em que ele passa a se chamar programa). Em seguida, temos de depurá-lo: encontrar cada erro e corrigi-lo até o computador executar o programa sem problemas. Porém, quando chegamos a um programa que faz o que queremos, não precisamos mais nos preocupar. Os computadores farão o que devem milhões de vezes, a uma velocidade muita alta, sem reclamar. Qualquer pessoa em qualquer lugar do mundo poderá usar nossa criação. O custo pode ser zero, se assim quisermos, ou ser o suficiente para nos deixar bilionários, se o problema resolvido for importante. Um programador – alguém que cria algoritmos e os codifica – é um pequeno deus, criando universos quando deseja. Poderíamos dizer até que o próprio Deus da Bíblia é um programador: linguagem, e não manipulação, é sua ferramenta de criação. Palavras tornam-se mundos. Nos dias de hoje, você também pode ser um deus, sentado no sofá com seu laptop. Imaginar um universo e torná-lo real. As leis da física são opcionais.

Conforme o tempo passa, os cientistas da computação se baseiam nos trabalhos uns dos outros e inventam algoritmos para novas tarefas. Esses algoritmos se unem a outros para usar seus resultados, produzindo por sua vez resultados para outros algoritmos. A cada segundo, bilhões de transistores em bilhões de computadores são ligados bilhões de vezes. Os algoritmos formam um novo tipo de ecossistema – em crescimento contínuo, só comparável em riqueza à própria vida.

Inevitavelmente, no entanto, há uma serpente neste Éden. Ela se chama monstro da complexidade. Como a Hidra, o monstro da complexidade tem muitas cabeças. Uma delas é a complexidade do espaço: o número de bits de informação que um algoritmo precisa armazenar na memória do computador. Quando o algoritmo precisa de mais memória que o computador pode fornecer, ele é inútil e deve ser descartado. Há, então, a irmã diabólica, a complexidade do tempo: quanto tempo o algoritmo leva para ser executado, isto é, por quantas etapas de uso e reuso dos transistores ele têm de passar antes de produzir os resultados desejados. Se demorar mais do que pudermos esperar, o algoritmo

também será inútil. Porém, a face mais assustadora do monstro é a complexidade humana. Quando os algoritmos ficam complicados demais para nossos pobres cérebros humanos entenderem, quando as interações entre as diferentes partes do algoritmo se dão em número muito grande e são muito complexas, erros começam a surgir, não conseguimos encontrá-los e corrigi-los e o algoritmo não faz o que queremos. Mesmo se conseguirmos fazê-lo funcionar, ele acabará ficando desnecessariamente complicado para as pessoas usarem e não se adaptará bem a outros algoritmos, gerando problemas posteriores.

Todos os cientistas da computação enfrentam o monstro da complexidade diariamente. Quando perdem a batalha, a complexidade infiltra-se em nossas vidas. Você deve ter notado que muitas batalhas foram perdidas. Mesmo assim, continuamos a construir nossa torre de algoritmos, com cada vez mais dificuldade. Cada nova geração de algoritmos tem de se basear nas gerações anteriores e lidar com suas complexidades, além de lidar com as suas próprias. A torre vai ficando cada vez mais alta e abrange o mundo todo, mas também se torna cada vez mais frágil, como um castelo de cartas esperando cair. Um minúsculo erro em um algoritmo faz um foguete de bilhões de dólares explodir ou milhões de pessoas ficarem sem energia elétrica. Quando os algoritmos interagem de maneiras inesperadas, o mercado de ações desaba.

Se os programadores são pequenos deuses, o monstro da complexidade é o próprio demônio. Pouco a pouco, ele está vencendo a guerra.

Tem de haver um caminho melhor.

Surge o aprendiz

Todo algoritmo tem uma entrada e uma saída: os dados entram no computador, o algoritmo faz o que precisa com eles, e um resultado é produzido. O machine learning faz o contrário: entram os dados e o resultado desejado, e é produzido o algoritmo que transforma um no outro. Os algoritmos de aprendizado – também conhecidos como aprendizes – são aqueles que criam outros algoritmos. Com o machine learning, os computadores escrevem seus próprios programas, logo não precisamos mais fazê-lo.

Magnífico!

Os computadores escrevem seus próprios programas. Essa é uma ideia poderosa, talvez até um pouco assustadora. Se os computadores começarem a

programar a si próprios, como os controlaremos? Na verdade, será fácil controlá-los, como veremos. Uma objeção mais imediata seria que parece bom demais para ser verdade. É claro que escrever algoritmos requer inteligência, criatividade, aptidões de resolução de problemas – coisas que os computadores não têm? Em que o machine learning difere da mágica? Na verdade, atualmente as pessoas escrevem muitos programas que os computadores não conseguem aprender. Porém, o mais surpreendente é que os computadores aprendem habilidades que as pessoas não podem escrever. Sabemos como dirigir automóveis e decifrar uma escrita feita à mão, mas essas habilidades são subconscientes; não podemos explicar para um computador como executá-las. No entanto, se fornecermos a um aprendiz um número suficiente de exemplos dessas tarefas, ele aprenderá facilmente como executá-las; neste ponto podemos deixá-lo por conta própria. É assim que o correio lê códigos postais e é como os carros autodirigíveis estão surgindo.

Talvez o poder do machine learning possa ser explicado melhor com uma analogia de baixa tecnologia: a agricultura. Em uma sociedade industrial, as mercadorias são criadas em fábricas, o que significa que engenheiros têm de descobrir exatamente como montá-las a partir de suas peças, como fabricar essas peças, e assim por diante – chegando ao nível da matéria-prima. É muito trabalho. Os computadores são as mercadorias mais complexas já inventadas, e, se considerarmos como são projetados, as fábricas responsáveis por sua criação e os programas executados neles, teremos uma extensa cadeia de produção. Mas há outra maneira muito mais antiga pela qual podemos conseguir o que precisamos: deixar que a natureza crie. Na agricultura, plantamos as sementes, nos certificamos de que elas recebam a água e os nutrientes necessários e colhemos os frutos. Por que a tecnologia não pode ser assim? Ela pode, e é o que o machine learning promete. Os algoritmos de aprendizado são as sementes, os dados são o solo e os programas de aprendizado são as plantas crescidas. O especialista em machine learning é como um fazendeiro, plantando as sementes, irrigando e fertilizando o solo e dando atenção à integridade dos frutos, mas deixando o processo fluir em outros aspectos.

Quando consideramos o machine learning dessa forma, duas coisas se destacam imediatamente. A primeira é que quanto mais dados temos, mais aprendemos. Se não houver dados? Não há nada a aprender. Muitos dados? Muito a aprender. É por isso que o machine learning está surgindo em todos os

lugares, conduzido pela quantidade exponencialmente crescente de dados. Se fosse algo que pudesse ser comprado em um supermercado, sua embalagem diria: “Basta adicionar dados”.

A segunda é que o machine learning é uma arma com a qual podemos derrotar o monstro da complexidade. Se receber dados suficientes, um programa de aprendizado contendo apenas algumas centenas de linhas pode gerar com facilidade um programa com milhões de linhas, e pode fazê-lo repetidamente para diferentes problemas. A redução da complexidade para o programador é assombrosa. É claro que, como a Hidra, o monstro da complexidade gerará novas cabeças assim que cortarmos as anteriores, mas inicialmente elas serão menores e demorarão um pouco para crescer, logo ainda teremos uma grande vantagem.

Podemos considerar o machine learning como o inverso da programação, assim como a raiz quadrada é o inverso do quadrado de um número ou a integração é o inverso da diferenciação. Da mesma forma que poderíamos perguntar “Qual número tem como raiz quadrada 16?” ou “Qual função tem como resultado x + 1?”, podemos perguntar “Qual algoritmo produz esta saída?”. Veremos em breve como transformar essa ideia em algoritmos de aprendizado concretos.

Alguns aprendizes obtêm conhecimento e outros adquirem aptidões. “Todos os humanos são mortais” é conhecimento. Andar de bicicleta é aptidão. No machine learning, com frequência o conhecimento assume a forma de modelos estatísticos, porque em grande parte ele é estatístico: todos os humanos são mortais, mas apenas 4% são americanos. As aptidões costumam assumir a forma de procedimentos: se a estrada virar à esquerda, gire a direção nesse sentido; se um cervo pular na sua frente, pise no freio. (Infelizmente, quando este texto foi escrito, os carros autodirigíveis do Google ainda confundiam sacos plásticos inflados com cervos.) Em geral, os procedimentos são bem simples e o conhecimento é que é complexo. Se você conseguir distinguir quais emails são spam, saberá quais excluir. Se souber qual posição é boa em um tabuleiro de xadrez, saberá qual jogada fazer (a que leve à melhor posição).

O machine learning assume muitas formas e é conhecido por muitos nomes: reconhecimento de padrões, modelagem estatística, mineração de dados, descoberta de conhecimento, análise preditiva, ciência de dados, sistemas adaptativos, sistemas auto-organizados *etc.* Cada um desses títulos é usado por

comunidades diferentes e tem associações distintas. Alguns terão vida longa, outros durarão menos. Neste livro, estou usando o termo *machine learning* para me referir amplamente a todos eles.

Às vezes, o machine learning é confundido com inteligência artificial (ou IA, na abreviação). Tecnicamente, ele é um subcampo da IA, mas cresceu tanto e foi tão bem-sucedido que ofuscou sua orgulhosa mãe. O objetivo da IA é ensinar os computadores a fazer o que atualmente os humanos fazem melhor, e aprender é sem dúvida a mais importante dessas tarefas; sem ela, nenhum computador pode se equiparar a um humano por muito tempo; com ela, o resto vem a reboque.

No ecossistema do processamento de informações, os aprendizes são os superpredadores. Os bancos de dados, crawlers, indexadores *etc.* são os herbívoros, pastando pacientemente em intermináveis campos de dados. Os algoritmos estatísticos, o processamento analítico online, e assim por diante, são os predadores. Os herbívoros são necessários, já que sem eles os outros não existiriam, mas os superpredadores levam uma vida mais agitada. Um crawler é como uma vaca, a web é seu pasto de abrangência mundial, cada página é uma folha de grama. Quando o crawler termina de pastar, uma cópia da web passa a existir em seus discos rígidos. Então, um indexador faz uma lista das páginas em que cada palavra aparece, semelhante ao índice no fim de um livro. Os bancos de dados, como os elefantes, são grandes e pesados e nunca esquecem. Entre essas pacientes bestas correm os algoritmos estatísticos e analíticos, compactando e selecionando, transformando dados em informações. Os aprendizes consomem essas informações, as digerem e as transformam em conhecimento.

Os especialistas em machine learning (também conhecidos como machine learners) são uma agremiação de elite até mesmo entre os cientistas da computação. Muitos cientistas da computação, principalmente os de gerações mais antigas, não entendem o machine learning tão bem quanto gostariam. Isso ocorre porque, tradicionalmente, a ciência da computação tem pensado deterministicamente, mas o machine learning requer pensamento estatístico. Se uma regra para, digamos, a rotulação de emails como spam tiver 99% de precisão, isso não significa que está incorreta; talvez seja o melhor a que pudemos chegar e suficientemente boa para ser útil. Essa diferença de raciocínio explica em grande parte porque a Microsoft teve mais problemas para acompanhar o Google do que teve para acompanhar o Netscape. No fim das contas, um navegador é apenas um software, mas um mecanismo de busca

requer uma mentalidade diferente.

Uma outra razão pela qual machine learners são os supergeeks é o fato de eles existirem em número bem menor que o necessário, até mesmo para os já altos padrões da ciência da computação. De acordo com o guru da tecnologia Tim O'Reilly, "cientista de dados" é a profissão mais cobiçada no Vale do Silício. O McKinsey Global Institute estima que, por volta de 2018, somente os Estados Unidos precisarão de 140.000 a 190.000 especialistas em machine learning a mais do que os que estarão disponíveis e mais 1,5 milhão de gerentes especializados em dados. As aplicações do machine learning tiveram um crescimento repentino muito grande para que o seu ensino conseguisse acompanhar e têm reputação de ser um tema difícil. A matemática contida nos livros pode até causar enjoo. No entanto, essa dificuldade é mais aparente que real. Todas as ideias importantes do machine learning podem ser expressas sem matemática. À medida que você for lendo este livro, talvez chegue até a inventar seus próprios algoritmos de aprendizado, sem o uso de qualquer equação.

A Revolução Industrial automatizou o trabalho manual e a Revolução da Informação fez o mesmo com o trabalho mental, mas o machine learning automatiza a si próprio. Sem ele, os programadores passam a ser o gargalo que detém o avanço. Com ele, o ritmo do avanço é acelerado. Se você for um cientista da computação lento e pouco brilhante, o machine learning é a ocupação ideal, porque os algoritmos de aprendizado farão seu trabalho, deixando-o levar o crédito. Por outro lado, os algoritmos podem deixá-lo sem emprego, o que nesse caso seria justo.

Por levar a automação a novos patamares, a revolução do machine learning causará extensas mudanças econômicas e sociais, como a internet, o computador pessoal, o automóvel e o motor a vapor causaram em sua época. Uma área em que essas mudanças já podem ser sentidas é nos negócios.

Por que as empresas adotam o machine learning

Por que o Google vale muito mais que o Yahoo? Ambos obtêm suas receitas da exibição de propagandas na web e são destinos muito procurados. Ambos usam leilões para vender espaço de propaganda e machine learning para prever com que frequência um usuário clicará em um anúncio (quanto maior a probabilidade, mais caro o anúncio). Porém, os algoritmos de aprendizado do Google são muito melhores que os do Yahoo. É claro que essa não é a única

razão para a diferença em seu valor de mercado, mas é uma razão importante. Cada clique previsto que não ocorre é uma oportunidade perdida para o anunciante e renda perdida para o site. Com a receita anual de 50 bilhões de dólares do Google, cada melhoria de 1% na previsão de cliques significa potencialmente mais meio bilhão de dólares anuais no banco para a empresa. Não é à toa que o Google é um grande fã do machine learning, e o Yahoo e outros sites estão arduamente tentando alcançá-lo.

A propaganda na web é apenas uma das manifestações de um fenômeno muito maior. Em qualquer mercado, os produtores e consumidores precisam entrar em contato antes que uma transação possa ocorrer. Na época pré-internet, os principais obstáculos para esse contato eram físicos. Só podíamos comprar livros na livraria local e esta tinha um espaço limitado na prateleira. Porém, ao podermos baixar qualquer livro no leitor eletrônico a qualquer momento, o problema passa a ser o grande número de opções. Como navegar nas prateleiras de uma livraria que tem milhões de títulos à venda? O mesmo ocorre com outros bens de informação: vídeos, músicas, notícias, tuítes, blogs e as simples páginas web. Também ocorre com todos os produtos e serviços que podem ser acessados remotamente: calçados, flores, equipamentos, vagas em hotéis, aulas, investimentos. É aplicável até mesmo à busca de emprego ou namoro. Como um encontra o outro? Esse é o problema que define a Era da Informação, e o machine learning é grande parte da solução.

À medida que as empresas crescem, passam por três fases. Primeiro, fazem tudo manualmente: os proprietários de um negócio familiar conhecem seus clientes pessoalmente e encomendam, exibem e recomendam itens conforme o caso. Isso é bom, mas não permite um aumento nas vendas. Na segunda e menos auspiciosa fase, a empresa cresceu tanto que precisa usar computadores. Surgem, então, os programadores, consultores e gerentes de bancos de dados, e milhões de linhas de código são escritas para automatizar todas as funções da empresa que puderem ser automatizadas. Um número muito maior de pessoas é atendido, mas não tão bem: as decisões são tomadas de acordo com categorias demográficas elementares, e os programas de computador são rígidos demais para satisfazer a infinita versatilidade humana.

Após um determinado ponto, não são encontrados programadores e consultores suficientes para fazer tudo que é necessário, e inevitavelmente a empresa adota o machine learning. A Amazon não consegue codificar precisamente os gostos de

todos os seus clientes em um programa de computador, e o Facebook não sabe como escrever um programa que selecione as melhores atualizações a serem exibidas para cada usuário. O Walmart vende milhões de produtos e tem bilhões de decisões para tomar todo dia; se seus programadores tentassem escrever um programa para tomar todas as decisões, nunca terminariam. Em vez disso, essas empresas aplicam algoritmos de aprendizado às montanhas de dados acumulados e deixam que eles adivinhem o que os clientes querem.

Os algoritmos de aprendizado são os conciliadores: eles unem produtores e consumidores, rompendo a sobrecarga de informações. Se forem suficientemente inteligentes, você terá o melhor de dois mundos: a ampla gama de opções e o baixo custo da larga escala com o toque personalizado da pequena escala. Os aprendizes não são perfeitos, e geralmente a última etapa da decisão continua sendo uma tarefa para os humanos, mas eles reduzem de maneira inteligente as opções a algo que uma pessoa possa gerenciar.

Olhando em retrospecto, é possível perceber que a progressão dos computadores para a internet e para o machine learning era inevitável: os computadores permitem o acesso à internet, que gera uma inundação de dados e o problema do número ilimitado de opções; o machine learning usa então a inundação de dados para resolver esse problema. A internet sozinha não é suficiente para mover a demanda de tipo “um tamanho para tudo” para a cauda longa[2] de variedade infinita. A Netflix pode ter mil filmes em estoque, mas, se os clientes não souberem como encontrar os filmes em que estão interessados, selecionarão os conhecidos. Só quando ela tiver um algoritmo de aprendizado para descobrir os gostos e recomendar filmes é que a cauda longa será realmente atendida.

Quando o inevitável ocorre e os algoritmos de aprendizado se tornam o intermediário, o poder se concentra neles. Os algoritmos do Google determinam em grande parte quais informações uma pessoa deve encontrar; os da Amazon, quais produtos ela deve comprar; e os do site Match.com, com quem ela deve sair. A última etapa é sempre nossa – escolher entre as opções apresentadas pelos algoritmos –, mas 99,9% da seleção foi feita por eles. O sucesso ou o fracasso de uma empresa agora depende de quantos aprendizes gostam de seus produtos, e o sucesso de toda a economia – com as pessoas obtendo os melhores produtos para suas necessidades pelo melhor preço – depende da excelência dos aprendizes.

A melhor maneira de uma empresa assegurar que os aprendizes gostem de seus produtos é elas mesmas os executarem. Quem tiver os melhores algoritmos e o maior número de dados vence. Um novo tipo de efeito de rede entra em ação: quem tiver mais clientes acumula mais dados, aprende os melhores modelos, atrai mais clientes novos, e assim por diante, em um círculo virtuoso (ou vicioso, se você for o rival). Mudar do Google para o Bing pode ser mais fácil que mudar do Windows para o Mac, mas na prática não o fazemos porque o Google, com a vantagem de ter sido lançado antes e sua parcela de mercado maior, sabe melhor o que desejamos, mesmo que a tecnologia do Bing seja equivalente. Os iniciantes no negócio de mecanismos de busca, começando sem qualquer dado e competindo com mecanismos com experiência de mais de uma década de aprendizado, terão que se esforçar.

Você pode pensar que após um tempo, mais dados seriam apenas mais do mesmo, porém não há sinal desse ponto de saturação A cauda longa não termina. Quando olhamos as recomendações que a Amazon ou a Netflix fazem, fica claro que esses serviços precisam melhorar muito, e os resultados das pesquisas do Google ainda deixam muito a desejar. Cada característica de um produto e cada ponto de um site podem ser melhorados com o uso do machine learning. O link da parte inferior de uma página deve ser vermelho ou azul? Faça testes com as duas cores e veja qual recebe mais cliques. Ou melhor, mantenha os aprendizes em execução e ajuste continuamente todos os aspectos do site.

A mesma dinâmica ocorre em qualquer mercado em que haja muitas opções e dados. A corrida começa e quem aprende mais rápido vence. Ela não para quando conhecemos melhor os clientes: as empresas podem aplicar o machine learning em todos os aspectos de suas operações, contanto que dados estejam disponíveis e cheguem de computadores, dispositivos de comunicação e sensores cada vez mais baratos e presentes. “Os dados são o novo petróleo” é um provérbio popular, e, assim como aconteceu com o petróleo, refiná-los é um grande negócio. A IBM, mais conectada ao mundo corporativo que qualquer outra empresa, organizou sua estratégia de crescimento a partir do fornecimento de análise para as corporações. As empresas consideram os dados um bem estratégico: quais dados tenho que meus rivais não têm? Como posso me beneficiar disso? Quais dados meus rivais têm que eu não tenho?

Da mesma forma que uma instituição bancária sem bancos de dados não pode competir com outra em que eles estejam presentes, uma empresa sem machine

learning não consegue acompanhar outra que faça uso dele. Enquanto os especialistas da primeira escrevem milhares de regras para prever o que os clientes desejam, os algoritmos da segunda aprendem bilhões de regras, um conjunto inteiro delas para cada cliente. É tão justo quanto lanças contra metralhadoras. O machine learning é uma tecnologia nova e arrojada, mas não é por isso que as empresas o adotam, e sim porque não têm escolha.

Reforçando o método científico

O machine learning é o método científico usando esteroides. Ele segue o mesmo processo de geração, teste e descarte ou refinamento de hipóteses. Porém, enquanto um cientista talvez passe sua vida inteira criando e testando algumas centenas de hipóteses, um sistema de machine learning pode fazer o mesmo em uma fração de segundo. O machine learning automatiza a descoberta. Logo, não é de surpreender que esteja revolucionando a ciência assim como os negócios.

Para progredir, todas as áreas da ciência precisam ter dados proporcionais à complexidade do fenômeno que elas estudam. É por isso que a física foi a primeira ciência a se destacar: os registros de Tycho Brahe referentes às posições dos planetas e as observações de Galileo sobre pêndulos e planos inclinados foram suficientes para a dedução das leis de Newton. Também é por isso que a biologia molecular, apesar de ser mais nova que a neurociência, a superou: os microarranjos de DNA e o sequenciamento de alto desempenho fornecem um volume de dados que os neurocientistas não têm como obter. Também é a razão para a pesquisa em ciências sociais ser uma batalha tão penosa: quando tudo que temos é uma amostragem de cem pessoas, com uma dúzia de medições para cada uma, só podemos modelar algum fenômeno muito restrito. Porém, até mesmo esse fenômeno restrito não existe isoladamente; ele é afetado por vários outros, o que significa que ainda estamos longe de entendê-lo.

A vantagem hoje é que as ciências que antes tinham poucos dados agora se valem de muitos. Em vez de pagar cinquenta universitários com a vista cansada para executar alguma tarefa no laboratório, os psicólogos podem acessar quantos assuntos quiserem postando a tarefa no Mechanical Turk da Amazon. (O que também fornece uma amostragem mais diversificada.) Já não é tão fácil lembrar, mas, há pouco mais de uma década, sociólogos que estudavam redes sociais lamentavam não conseguir acessar uma rede com mais de algumas centenas de membros. Atualmente há o Facebook, com mais de um bilhão. Boa parte desses

membros também posta relatos quase minuciosos de suas vidas; é como ter uma transmissão ao vivo da vida social no planeta Terra. Em neurociência, a conectômica e a geração de imagens funcionais por ressonância magnética abriram uma janela extraordinariamente detalhada para o cérebro. Em biologia molecular, os bancos de dados de genes e proteínas crescem exponencialmente. Até mesmo em ciências "mais antigas", como a física e a astronomia, o progresso continua devido à inundação de dados proveniente de aceleradores de partículas e pesquisas celestes digitais.

No entanto, não adianta ter big data[3] se você não puder transformá-lo em conhecimento, e não há cientistas suficientes no mundo para realizar a tarefa. Edwin Hubble descobriu novas galáxias se debruçando sobre chapas fotográficas, mas é claro que o meio bilhão de objetos celestes da Sloan Digital Sky Survey não foi identificado dessa forma. Seria como tentar contar os grãos de areia de uma praia manualmente. Você pode escrever regras para distinguir galáxias de estrelas e objetos aleatórios (como pássaros e aviões), mas elas não serão muito precisas. Em vez disso, o projeto SKICAT (ferramenta de catalogação e análise de imagens celestes) usou um algoritmo de aprendizado. Partindo de chapas em que objetos eram rotulados com as categorias corretas, ele descobriu o que caracteriza cada objeto e aplicou o resultado a todas as chapas não rotuladas. Melhor ainda, conseguiu classificar objetos muito tênues para serem rotulados por humanos, que abrangiam grande parte da pesquisa.

Com o big data e o machine learning, podemos entender fenômenos muito mais complexos do que antes. Na maioria das áreas, tradicionalmente os cientistas usavam somente tipos de modelos muito limitados, como a progressão linear, em que a variação atribuída aos dados é sempre uma linha reta. Infelizmente, quase todos os fenômenos no mundo são não lineares. (Ou felizmente, já que caso contrário a vida seria muito tediosa – na verdade, não haveria vida.) O machine learning oferece um novo mundo de modelos não lineares. É como acender as luzes em uma sala em que antes só entrava um pouco do brilho da lua.

Em biologia, os algoritmos de aprendizado detectam onde os genes estão situados em uma molécula de DNA, onde os fragmentos supérfluos de RNA são unidos antes de proteínas serem sintetizadas, como as proteínas se misturam em suas formas características e como diferentes condições afetam a expressão de genes distintos. Em vez de testar milhares de drogas novas em laboratório, os

aprendizes preveem se elas funcionarão, e as mais promissoras são testadas. Eles também removem moléculas propensas a produzir efeitos colaterais indesejados, como o câncer. Isso evita a ocorrência de falhas caras, como drogas candidatas serem rejeitadas somente após os testes em humanos começarem.

O maior desafio, no entanto, é reunir essas informações em um todo coerente. Quais fatores podem levar a uma doença cardíaca e como eles interagem? Newton só precisou de três leis de movimento e uma de gravidade, mas o modelo completo de uma célula, de um organismo ou de uma sociedade é demais para um único ser humano montar. À medida que o conhecimento aumenta, os cientistas se especializam cada vez mais, porém ninguém consegue juntar as peças porque há muitas delas. Os cientistas entram em colaboração, mas a linguagem é um meio de comunicação muito lento. Eles tentam acompanhar as pesquisas uns dos outros, mas o volume de publicações é muito alto para conseguirem. Com frequência, é mais fácil refazer um experimento que encontrar a publicação que o relatou. O machine learning pode ajudar, procurando informações relevantes em material já publicado, traduzindo o jargão de uma área para o de outra e até mesmo estabelecendo conexões às quais os cientistas não tinham atentado. Cada vez mais o machine learning age como um concentrador gigante a partir do qual técnicas de modelagem inventadas em uma área se incorporam a outras.

Se os computadores não tivessem sido inventados, a ciência teria chegado a um impasse na segunda metade do século 20. Talvez isso não fosse notado tão imediatamente pelos cientistas porque eles se aferrariam a qualquer progresso limitado que ainda conseguissem obter, mas o teto desse progresso seria muito mais baixo. Da mesma forma, sem o machine learning, muitos cientistas teriam retornos menores nas décadas futuras.

Para conhecer o futuro da ciência, faremos um passeio em um laboratório do Manchester Institute of Biotechnology, local em que um robô chamado Adam trabalha com afinco para descobrir quais genes codificam quais enzimas da levedura. Adam tem um modelo do metabolismo da levedura e conhecimento geral de genes e proteínas. Ele cria hipóteses, projeta experimentos para testá-las, executa-os fisicamente, analisa os resultados e cria novas hipóteses até ficar satisfeito. Atualmente, os cientistas humanos ainda verificam independentemente as conclusões de Adam antes de aceitá-las, mas no futuro eles deixarão que cientistas robotizados verifiquem as hipóteses uns dos outros.

Um bilhão de Bill Clintons

Foi o machine learning que elegeu o presidente dos Estados Unidos em 2012. Os fatores que geralmente decidem eleições presidenciais – economia, qualidades apreciadas nos candidatos, e assim por diante – tiveram muita influência, e o resultado ficou restrito a alguns estados importantes. A campanha de Mitt Romney seguiu uma abordagem convencional de consulta, agrupando eleitores em amplas categorias e atendo-se ou não a cada uma delas. Neil Newhouse, perito de Romney em sondagem da opinião pública, disse: “Se conseguirmos ganhar dos políticos independentes em Ohio, vencemos a disputa”. Romney ultrapassou-os em 7%, mas mesmo assim não ganhou no estado nem a eleição.

Por outro lado, o presidente Obama contratou Rayid Ghani, um especialista em machine learning, como cientista-chefe de sua campanha, e Ghani deu início à maior operação de análise da história da política. Eles consolidaram todas as informações sobre eleitores em um único banco de dados; combinaram o que conseguiram obter em redes sociais, marketing e outras fontes; e previram quatro coisas para cada eleitor: qual a probabilidade de ele apoiar Obama, de comparecer às pesquisas, de reagir aos lembretes da campanha para fazer isso e de mudar sua opinião a partir de uma troca de ideias sobre um assunto específico. Com base nesses modelos de eleitores, toda noite a campanha executava 66.000 simulações da eleição e usava os resultados para direcionar seu pelotão de voluntários munidos com as informações de quem deveriam chamar, em quais portas deveriam bater e o que dizer.

Na política, como nos negócios e na guerra, não há nada pior que ver seu oponente realizar movimentos que você não entende e sobre os quais não sabe o que fazer até que seja tarde demais. Foi isso que aconteceu na campanha de Romney. Eles podiam ver o adversário comprando anúncios em determinadas emissoras a cabo de cidades específicas, mas não sabiam o porquê; sua bola de cristal estava muito embaçada. No fim das contas, Obama ganhou a preferência de todos os estados decisivos, exceto da Carolina do Norte, com margens maiores que o previsto até pelos mais confiáveis peritos em opinião pública. Estes, por sua vez, foram os que usaram (como Nate Silver) as técnicas de previsão mais sofisticadas; eles foram menos precisos que a campanha de Obama porque tinham menos recursos. Porém, foram muito mais precisos que as autoridades tradicionais, cujas previsões se basearam em sua experiência.

Você deve estar achando que a eleição de 2012 foi um acaso feliz: a maioria

das eleições não tem resultados tão próximos para o machine learning ser o fator decisivo. Porém, ele *fará* com que as eleições tenham resultados mais próximos no futuro. Na política, como em tudo, aprender faz parte da corrida armamentista. Na época de Karl Rove, ex-profissional de marketing direto e minerador de dados, os republicanos estavam na dianteira. Em 2012, eles recuaram e agora estão se aproximando novamente. Não sabemos quem estará na frente no próximo ciclo eleitoral, mas os dois partidos trabalharão duro para vencer. Isso significa entender melhor os eleitores e personalizar os discursos – e até mesmo escolher os candidatos – de acordo. O mesmo se aplica às plataformas dos partidos, durante e entre os ciclos eleitorais: se modelos de eleitores detalhados, baseados em dados estatísticos, mostrarem que uma plataforma atual não é vencedora, o partido a mudará. Como resultado, exceto nos eventos principais, diferenças entre os candidatos nas pesquisas serão menores e durarão menos. Se o resto continuar igual, os candidatos com os melhores modelos de eleitores vencerão, e os eleitores terão mais benefícios.

Um dos maiores talentos que um político pode ter é a habilidade de entender os eleitores, individualmente ou em pequenos grupos, e se dirigir (ou parecer estar se dirigindo) diretamente a eles. Bill Clinton é o exemplo paradigmático desse talento na história recente. O efeito do machine learning é como ter um dedicado Bill Clinton para cada eleitor. Todos esses mini-Clintons são uma sombra do Clinton real, mas têm a vantagem dos números; até mesmo Bill Clinton não tem como saber (mas gostaria) o que cada eleitor americano está pensando. Os algoritmos de aprendizado são definitivamente os políticos de varejo.

É claro que, como nas empresas, os políticos podem usar o conhecimento obtido com o machine learning tanto para o bem quanto para o mal. Por exemplo, eles poderiam fazer promessas inconsistentes para diferentes eleitores. Porém, os eleitores, a mídia e as organizações de defesa podem fazer sua própria mineração de dados e expor políticos que trapacearem. A corrida armamentista não se dá apenas entre candidatos, mas entre todos os participantes do processo democrático.

O resultado mais importante é que a democracia funciona melhor porque a largura de banda da comunicação entre eleitores e políticos é muito maior. Nesta época da internet de alta velocidade, a quantidade de informações que os representantes eleitos obtêm de nós com certeza ainda é do século 19: cerca de uma centena de bits a cada dois anos, o mesmo que caberia em uma cédula de

votação. Isso é complementado por consultas e talvez pela ocasional reunião por email ou na prefeitura, mas ainda é muito pouco. O big data e o machine learning mudam a equação. No futuro, contanto que os modelos de eleitores sejam precisos, os candidatos eleitos poderão perguntar aos eleitores milhares de vezes o que eles desejam e agir de acordo – sem precisar importunar os cidadãos reais de carne e osso.

Um por terra, dois pela internet

No ciberespaço, os algoritmos de aprendizado guarnecem as defesas da nação. Todos os dias, agressores estrangeiros tentam invadir computadores do Pentágono, de empresas de segurança e de outras organizações e agências governamentais. Suas táticas mudam continuamente; o que funcionou contra os ataques de ontem não têm efeito sobre os de hoje. Escrever um código que detectasse e bloqueasse cada ataque seria tão eficaz quanto a Linha Maginot,[4] e o comando cibernético do Pentágono sabe disso. Porém, o machine learning terá problemas se um ataque for o primeiro de seu tipo e não houver um anterior com o qual aprender. Em vez disso, os aprendizes constroem modelos de comportamento normal, dos quais há muitos, e de anomalias em flags. Em seguida, chamam a cavalaria (ou seja, os administradores do sistema). Se um dia a guerra cibernética explodir, os generais serão humanos, mas os soldados serão algoritmos. Os humanos são muito lentos e em menor número e seriam rapidamente derrotados por uma armada de bots. Precisamos de nossa própria armada de bots, e o machine learning é como o West Point dos bots.

A guerra cibernética é um exemplo de combate assimétrico, em que um lado não tem poder militar convencional equiparável ao do outro, mas mesmo assim pode causar danos graves. Alguns terroristas armados com nada mais que estiletes poderiam derrubar as Torres Gêmeas e matar milhares de inocentes. Atualmente, as maiores ameaças à segurança dos Estados Unidos podem ser classificadas como de combate assimétrico, e há uma arma eficaz contra todas elas: a informação. Se o inimigo não puder se esconder, ele não sobreviverá. A boa notícia é que temos muita informação, o que também é uma má notícia.

A Agência de Segurança Nacional (NSA, National Security Agency) adquiriu má fama por seu apetite insaciável por dados: de acordo com uma estimativa, todo dia ela intercepta mais de um bilhão de chamadas telefônicas e outros tipos de comunicação ao redor do planeta. No entanto, além dos problemas de

privacidade, ela não tem milhões de funcionários para avaliar todas essas chamadas e emails ou mesmo registrar quem está falando com quem. A grande maioria das chamadas é totalmente inofensiva, e é muito difícil escrever um programa para selecionar as chamadas suspeitas. Antigamente, a NSA usava a busca de palavras-chave, mas esse esquema é fácil de burlar. (É só chamar o bombardeio de “casamento” e a bomba de “ bolo de casamento”.) No século 21, essa é uma tarefa para o machine learning. O sigilo é a marca registrada da NSA, mas seu diretor declarou ao Congresso que a mineração de logs telefônicos já conseguiu evitar muitas ameaças terroristas.

Os terroristas podem se ocultar entre a multidão em um jogo de futebol, mas os aprendizes identificarão seus rostos. Eles podem fabricar bombas exóticas, mas os aprendizes as farejarão. Os aprendizes também podem fazer algo mais sutil: conectar eventos individuais aparentemente inócuos, mas que em conjunto seguem um padrão suspeito. Essa abordagem poderia ter evitado o 11 de Setembro. Há mais um truque: quando um programa de aprendizado é implantado, os vilões mudam seu comportamento para enganá-lo. Isso é o oposto do que ocorre no mundo natural, que sempre funciona da mesma forma. A solução é associar o machine learning à teoria dos jogos, algo em que trabalhei no passado: não aprender apenas a vencer o que seu oponente está fazendo agora; aprender a evitar que ele faça algo ao aprendiz. Decompor os custos e benefícios de diferentes ações, como se faz na teoria dos jogos, também pode ajudar a atingir o equilíbrio certo entre privacidade e segurança.

Durante a Batalha da Inglaterra, a Força Aérea Real (RAF, Royal Air Force) conteve a Luftwaffe apesar de estar em grande desvantagem. Os pilotos alemães não conseguiam entender como, aonde quer que fossem sempre se deparavam com a RAF. A Inglaterra tinha uma arma secreta: o radar, que detectava os aviões alemães bem antes de eles cruzarem seu espaço aéreo. O machine learning é como um radar que prevê o futuro. Ele não só reage aos movimentos do adversário, mas os prevê e evita.

Um exemplo mais próximo da vida cotidiana é o conhecido policiamento preventivo. Prevendo tendências criminosas e posicionando patrulhas de maneira estratégica em locais em que elas possam ser necessárias, assim como adotando outras medidas preventivas, a força policial de uma cidade pode fazer eficazmente o trabalho de uma força policial muito maior. Em vários aspectos, o cumprimento da lei é semelhante ao combate assimétrico, e muitas das mesmas

técnicas de machine learning são aplicáveis, seja na detecção de fraudes, na descoberta de redes criminosas ou na simples ronda policial.

O machine learning também está desempenhando um papel cada vez maior no campo de batalha. Os aprendizes podem ajudar a dissipar a névoa da guerra, analisando o reconhecimento de imagens, processando relatórios pós-combate e montando um cenário da situação para o comandante. O aprendizado alimenta o cérebro de robôs militares, ajudando-os a se orientar, adaptar-se ao terreno, diferenciar veículos inimigos de civis e detectar seus alvos. O AlphaDog da Darpa carrega equipamentos para os soldados. Drones podem voar de maneira autônoma com a ajuda de algoritmos de aprendizado; embora ainda sejam parcialmente controlados por pilotos humanos, a tendência é que um único piloto supervisione esquadras cada vez maiores. Nas forças armadas do futuro, haverá muito mais aprendizes que soldados, salvando inúmeras vidas.

Para onde estamos indo?

Tendências tecnológicas surgem e desaparecem o tempo todo. O que diferencia o machine learning é que, em todas essas mudanças, do surgimento ao esquecimento, ele continua crescendo. Seu primeiro grande estouro foi nas finanças, prevendo a ascensão e queda de ações, a partir do fim dos anos 1980. A tendência seguinte foi na mineração de bancos de dados corporativos, que no meio da década de 1990 começou a crescer muito, e em áreas como marketing direto, gerenciamento de relações com o cliente, pontuação de crédito e detecção de fraudes. Em seguida, vieram a web e o e-commerce, em que a personalização automatizada se tornou rapidamente obrigatória. Quando a queda das empresas “ponto com” esfriou os ânimos temporariamente, o uso de aprendizado para a pesquisa na web e a divulgação de propaganda ganhou fôlego. Para o bem ou para o mal, os ataques do 11 de Setembro colocaram o machine learning na linha de frente da guerra contra o terror. A Web 2.0 fez surgir vários aplicativos novos, desde os que fazem mineração em redes sociais aos que descobrem o que os blogueiros estão dizendo sobre nossos produtos. Paralelamente, cientistas de todos os tipos migravam cada vez mais para a modelagem de larga escala, com as áreas de biologia molecular e astronomia na liderança. A bolha no mercado de imóveis passou rápido; seu principal efeito foi uma migração bem-vinda de talentos de Wall Street para o Vale do Silício. Em 2011, os “memes” com “big data” se propagaram rapidamente, colocando o machine learning no centro do

futuro da economia global. Hoje é difícil encontrar uma área de atuação humana intocada pelo machine learning, o que inclui candidatas aparentemente improváveis, como música, esportes e degustação de vinhos.

Esse crescimento pode parecer notável, mas é apenas uma amostra do que está por vir. Apesar de sua utilidade, a geração de algoritmos de aprendizado em andamento na indústria é, na verdade, bem limitada. Quando os algoritmos que agora estão no laboratório chegarem às linhas de frente, a observação de Bill Gates de que uma revolução causada pelo machine learning equivaleria a dez Microsofts parecerá conservadora. E se as ideias que *realmente* fazem os olhos dos pesquisadores brilharem derem frutos, o machine learning trará não só uma nova era para a civilização, mas um novo estágio da evolução da vida na Terra.

O que torna isso possível? Como os algoritmos de aprendizado funcionam? O que eles podem fazer atualmente e como será a próxima geração? Como ocorrerá a revolução do machine learning? E a que oportunidades e perigos você deve ficar atento? É disso que trata este livro – continue lendo!

---

1 N. T.: A aprendizagem automática ou aprendizado de máquina (em inglês: “machine learning”) é um subcampo da inteligência artificial dedicado ao desenvolvimento de algoritmos e técnicas que permitam ao computador aprender, isto é, aperfeiçoar seu desempenho em alguma tarefa.

2 N. T.: Long tail, ou cauda longa, é um termo introduzido por Chris Anderson para designar nichos de mercado. O nome remete ao formato do gráfico gerado: os produtos mais populares alcançam o topo, porém não ocupam um espaço amplo no mercado, enquanto os nichos são pequenos e infinitos.

3 N.T.: Em tecnologia da informação, o termo big data (“megadados” em português) refere-se a um grande conjunto de dados armazenados.

4 N.T.: A Linha Maginot (em francês, “ligne Maginot”) foi uma linha de fortificações e de defesa construída pela França ao longo de suas fronteiras com a Alemanha e a Itália, após a Primeira Guerra Mundial, mais precisamente entre 1930 e 1936.

capítulo 2

O Algoritmo Mestre

Ainda mais surpreendente que a diversidade de aplicativos de machine learning existentes é o fato de que são os *mesmos* algoritmos que executam todas essas tarefas. Sem o machine learning, quando temos dois problemas para resolver, é preciso escrever dois programas diferentes. Eles podem usar parte da mesma infraestrutura, como a mesma linguagem de programação ou o mesmo sistema de banco de dados, mas um programa para, digamos, jogar xadrez não terá utilidade se você quiser processar aplicativos de cartão de crédito. Com o machine learning, o mesmo algoritmo pode fazer as duas coisas, contanto que você forneça a ele os dados adequados como fonte do aprendizado. Na verdade, apenas alguns algoritmos atuam na grande maioria dos aplicativos de machine learning, e iremos examiná-los nos próximos capítulos.

Por exemplo, considere o Naïve Bayes, um algoritmo de aprendizado que pode ser expresso como uma única equação curta. Dado um banco de dados de registros de pacientes – seus sintomas, resultados de testes e se eles tiveram ou não algum problema específico –, o Naïve Bayes pode aprender a diagnosticar o caso em uma fração de segundo, geralmente melhor que médicos que passaram muitos anos na faculdade de medicina. Ele também se sai melhor que sistemas médicos especializados que consomem milhares de homens-hora para ser construídos. O mesmo algoritmo é amplamente usado para aprender o comportamento de filtros de spam, um problema que à primeira vista não tem nenhuma relação com diagnósticos médicos. Outro aprendiz simples, chamado de algoritmo do vizinho mais próximo, tem sido usado para tudo, desde o reconhecimento de escrita manual até o controle de mãos robóticas e a recomendação de livros e filmes dos quais possamos gostar. E aprendizes de árvore de decisão conseguem decidir se seu aplicativo de cartão de crédito deve

ser aceito, encontrar junções de emenda em DNA e selecionar a próxima jogada em um jogo de xadrez.

Além de os mesmos algoritmos de aprendizado executarem uma variedade de tarefas infinita, eles são surpreendentemente simples se comparados com os algoritmos que substituem. A maioria dos aprendizes pode ser codificada em algumas centenas de linhas ou, talvez, em alguns milhares, se você adicionar muitos recursos. Por outro lado, os programas que eles substituem podem se estender por centenas de milhares ou até mesmo por milhões de linhas, e um único aprendiz pode produzir um número ilimitado de programas diferentes.

Se um número tão pequeno de aprendizes pode fazer tantas coisas, a pergunta lógica é: um único aprendiz poderia fazer tudo? Em outras palavras, um único algoritmo poderia aprender tudo que pode ser aprendido a partir de dados? Essa é uma tarefa muito difícil, já que inclui tudo que existe no cérebro de um adulto, tudo que a evolução criou e a soma de todo o conhecimento científico. Porém, na verdade, todos os principais aprendizes – inclusive o do vizinho mais próximo, o de árvores de decisão e o de redes bayesianas, uma generalização do Naïve Bayes – são universais da seguinte forma: se você lhes der uma quantidade suficiente de dados adequados, eles poderão se aproximar de qualquer função arbitrariamente semelhante – o que é o jargão matemático para aprender qualquer coisa. O problema é que os “dados suficientes” podem ser infinitos. Aprender a partir de dados finitos requer fazer suposições, como veremos, e diferentes aprendizes fazem suposições distintas, o que os torna bons para algumas coisas, mas não para outras.

Mas, e se, em vez de embutir as suposições no algoritmo, fizéssemos delas uma entrada explícita, junto aos dados, e permitíssemos que o usuário selecionasse quais usar ou até mesmo que declarasse novas? Há um algoritmo que possa receber qualquer dado e suposição e fornecer o conhecimento implícito neles? Acredito que sim. É claro que temos de impor alguns limites sobre quais serão as suposições; caso contrário, poderíamos trapacear dando ao algoritmo todo o conhecimento-alvo, ou algo aproximado, na forma de suposições. Porém, há muitas maneiras de fazer isso, desde limitar o tamanho da entrada a requerer que as suposições não sejam mais fortes que as dos aprendizes atuais.

A pergunta então passa a ser: qual nível de fragilidade as suposições podem ter para, mesmo assim, permitir que todo o conhecimento relevante seja derivado de

dados finitos? Observe a palavra *relevante*: só estamos interessados em conhecimento relativo ao nosso mundo, e não a mundos que não existem. Logo, a invenção de um aprendiz universal se resume à descoberta das regularidades mais profundas de nosso universo, aquelas que todos os fenômenos compartilham, e à formulação de uma maneira computacionalmente eficiente de combiná-las a dados. Como veremos, o requisito da eficiência computacional nos impede de usar apenas as leis da física como regularidades. No entanto, não implica que o aprendiz universal terá de ser tão eficiente quanto os mais especializados. Como ocorre com frequência na ciência da computação, estamos sacrificando a eficiência em nome da generalização. Isso também se aplica à quantidade de dados necessária ao aprendizado de um conhecimento-alvo específico: um aprendiz universal precisará de mais dados que um especializado, mas isso não será problema se tivermos a quantidade necessária – e quanto maior o volume de dados, maior a probabilidade de que seja esse o caso.

Temos, então, a hipótese central do livro:

Um único algoritmo de aprendizado universal pode obter todo o conhecimento – passado, presente e futuro – a partir de dados.

Chamo esse aprendiz de Algoritmo Mestre. Se um algoritmo assim for viável, inventá-lo seria um dos maiores feitos científicos de todos os tempos. Na verdade, o Algoritmo Mestre será a última coisa que teremos de inventar porque, uma vez que o lançarmos, ele inventará tudo o mais que puder ser inventado. Só teremos de fornecer uma quantidade suficiente do tipo de dado certo e ele descobrirá o conhecimento correspondente. Forneça um streaming de vídeo e ele aprenderá a ver. Forneça uma biblioteca e ele aprenderá a ler. Disponibilize os resultados de experimentos de física e ele descobrirá as leis da física. Forneça dados de cristalografia de DNA e ele descobrirá a estrutura do DNA.

Isso pode soar improvável: como um único algoritmo poderia aprender tantas coisas e coisas tão difíceis? Porém, na verdade, muitas linhas de evidência apontam para a existência de um Algoritmo Mestre. Vejamos quais são elas.

O argumento da neurociência

Em abril de 2000, uma equipe de neurocientistas do MIT divulgou na *Nature* os resultados de um experimento extraordinário. Eles reconectaram o cérebro de um furão, redirecionando as conexões dos olhos para o córtex auditivo (parte do

cérebro responsável por processar sons) e as conexões dos ouvidos para o córtex visual. Você deve estar pensando que o resultado foi um furão gravemente incapacitado, mas não: o córtex auditivo aprendeu a ver, o córtex visual aprendeu a ouvir, e o furão ficou bem. Em mamíferos normais, o córtex visual contém um mapa da retina: neurônios conectados a regiões próximas da retina ficam perto uns dos outros no córtex. Em vez disso, os furões reconectados desenvolveram um mapa da retina no córtex auditivo. Se a entrada visual tivesse sido redirecionada para o córtex somatossensorial, responsável pela percepção de toque, ele aprenderia a ver. Outros mamíferos também têm essa habilidade.

Em pessoas com cegueira congênita, o córtex visual pode assumir o controle de outras funções do cérebro. Em pessoas surdas, o córtex auditivo faz o mesmo. As pessoas cegas podem aprender a "ver" com a língua enviando imagens de vídeo de uma câmera acoplada à cabeça para um conjunto de eletrodos colocados na língua, com as altas voltagens correspondendo a pixels brilhantes e as baixas a pixels escuros. Ben Underwood era uma criança cega que aprendeu por conta própria a usar a ecolocalização para se locomover, como fazem os morcegos. Estalando sua língua e escutando os ecos, ele podia caminhar sem tropeçar em obstáculos, andar de skate e até mesmo jogar basquete. Tudo isso serve como evidência de que o cérebro usa o mesmo algoritmo de aprendizado em todos os locais, com as áreas dedicadas aos diferentes sentidos distinguidas apenas pelas diferentes entradas a que estão conectadas (por exemplo, olhos, ouvidos, nariz). Por sua vez, as áreas associativas adquirem sua função por estarem conectadas a várias regiões sensoriais, e as áreas "executivas" adquirem as suas conectando-se às áreas associativas e à saída motora.

A verificação do córtex em um microscópio leva à mesma conclusão. O mesmo padrão conectivo se repete em todos os locais. O córtex está organizado em colunas com seis camadas distintas, loops de feedback direcionando-se a outra estrutura cerebral chamada tálamo e um padrão recorrente de conexões inibitórias de curto alcance e estimulantes de alcance mais longo. Certo nível de variação está presente, mas ele lembra mais diferentes parâmetros ou configurações do mesmo algoritmo que algoritmos diferentes. Áreas sensoriais de nível inferior têm diferenças mais perceptíveis, mas, como os experimentos de reconexão mostram, elas não são cruciais. O cerebelo, a parte evolutiva mais antiga do cérebro responsável pelo controle motor de nível inferior, tem uma arquitetura claramente diferente e muito regular, construída a partir de vários

neurônios menores, logo parece que pelo menos o aprendizado motor usa um algoritmo diferente. Contudo, se o cerebelo de alguém for danificado, o córtex assumirá sua função. Parece que a evolução manteve o cerebelo em funcionamento não porque ele faz algo que o córtex não pode, mas porque é mais eficiente.

Os processamentos que ocorrem dentro da arquitetura cerebral também são semelhantes em todos os locais. Todas as informações do cérebro são representadas da mesma maneira por meio de padrões de acionamento elétrico dos neurônios. O mecanismo de aprendizado também é assim: memórias são formadas pelo reforço das conexões entre neurônios que são acionados juntos, com o uso de um processo bioquímico conhecido como potenciação de longo prazo. Isso não ocorre apenas nos humanos: diferentes animais têm cérebros semelhantes. O nosso é excepcionalmente grande, mas parece se basear nos mesmos princípios do [cérebro] de outros animais.

Outra linha de raciocínio para a uniformidade do córtex vem do que podemos chamar de pobreza do genoma. O número de conexões em nosso cérebro é um milhão de vezes maior que a quantidade de letras do genoma, logo não é fisicamente possível para o genoma especificar em detalhes como o cérebro está conectado.

No entanto, o argumento mais importante para o cérebro ser o Algoritmo Mestre é que ele é responsável por tudo que percebemos e imaginamos. Se existir algo que o cérebro não possa aprender, não saberemos que existe. Talvez apenas não o vejamos ou pensemos que é aleatório. De qualquer forma, se implementarmos o cérebro em um computador, esse algoritmo poderá aprender tudo que aprenderíamos. Logo, um caminho – talvez o mais popular – para a invenção do Algoritmo Mestre seja aplicar engenharia reversa ao cérebro. Jeff Hawkins tentou explicar a assunto em seu livro *On Intelligence*. Ray Kurzweil acha que a Singularidade – o surgimento de inteligência artificial que exceda enormemente a variedade humana – fará exatamente isso e apresenta sua opinião em seu livro *Como criar uma mente*. Todavia, essa é apenas uma entre várias abordagens possíveis, como veremos. Não é nem mesmo necessariamente a mais promissora, porque o cérebro é muito complexo e ainda estamos começando a decifrá-lo. Por outro lado, se não conseguirmos descobrir o Algoritmo Mestre, a Singularidade não ocorrerá tão cedo.

Nem todos os neurocientistas acreditam na uniformidade do córtex; precisamos

aprender mais antes de ter certeza. A questão do que o cérebro pode ou não aprender é algo muito debatido. Porém, se há algo que sabemos e que o cérebro não pode aprender, deve ter sido aprendido por evolução.

O argumento da evolução

A variedade infinita de vida é resultado de um único mecanismo: a seleção natural. O mais notável é que esse mecanismo é de um tipo muito familiar para os cientistas da computação: a pesquisa iterativa, em que resolvemos um problema testando várias soluções candidatas, selecionando e modificando as melhores e repetindo essas etapas enquanto for necessário. A evolução *é* um algoritmo. Parafraseando Charles Babbage, pioneiro da computação na era vitoriana, Deus não criou espécies, mas sim os algoritmos para a criação das espécies. As "intermináveis e lindas formas" sobre as quais Darwin fala na conclusão de *A origem das espécies* desmentem a teoria de uma uniformidade mais bela: todas as formas estão codificadas em sequências de DNA e surgem pela modificação e combinação delas. Quem adivinharia, dada somente uma descrição desse algoritmo, que ele produziria você e eu? Se a evolução pode aprender a nos criar, possivelmente também pode aprender tudo que exista para ser aprendido, caso a implementemos em um computador suficientemente poderoso. Na verdade, programas que evoluem por simulação da seleção natural são algo popular na área de machine learning. A evolução, portanto, é outro caminho promissor para se chegar ao Algoritmo Mestre.

A evolução é o exemplo definitivo de onde um simples algoritmo de aprendizado pode chegar se receber dados suficientes. Sua entrada é a experiência e o destino de todas as criaturas vivas que já existiram (agora *chamados* de big data). Por outro lado, a evolução vem sendo executada há mais de três bilhões de anos no computador mais poderoso da Terra: a própria Terra. Uma versão desse esquema em computador teria de ser mais rápida e usar menos dados que a original. Qual é o melhor modelo para o Algoritmo Mestre: a evolução ou o cérebro? Essa é a variante da área de machine learning para o debate "evolucionismo *versus* criacionismo". E, assim como a evolução e a criação se uniram para nos gerar, talvez o Algoritmo Mestre contenha elementos de ambas.

O argumento da física

Em um famoso ensaio de 1959, o físico e ganhador do Nobel, Eugene Wigner, mostrou admiração com o que chamou de “a irracional eficácia da matemática nas ciências naturais”. Que milagre faria leis induzidas de uma observação insuficiente serem aplicadas para muito além delas? De que forma as leis conseguem ser muito mais precisas que os dados em que se baseiam? Acima de tudo, por que a linguagem simples e abstrata da matemática consegue descrever precisamente tantos fenômenos de nosso mundo infinitamente complexo? Wigner considerou esse fato um mistério auspicioso e, ao mesmo tempo, insondável. No entanto, é assim que as coisas são, e o Algoritmo Mestre é uma extensão lógica dele.

Se o mundo fosse apenas uma confusão exuberante e ruidosa, haveria razões para duvidarmos da existência de um aprendiz universal. Porém, se tudo que vivenciamos é produto de algumas leis simples, faz sentido que um único algoritmo possa produzir tudo que possa ser produzido. Tudo que o Algoritmo Mestre tem de fazer é fornecer um atalho para as consequências das leis, substituindo derivações matemáticas incrivelmente longas por outras mais curtas baseadas em observações.

Por exemplo, acreditamos que as leis da física deram início à evolução, mas não sabemos como. Em vez disso, podemos induzir a seleção natural diretamente de observações, como Darwin fez. Incontáveis inferências erradas poderiam ser produzidas a partir dessas observações, mas a maioria delas nunca nos ocorrerá, porque nossas inferências são influenciadas pelo amplo conhecimento que temos do mundo, e esse conhecimento é consistente com as leis da natureza.

Ainda tem de ser conhecido quanto do caráter da lei física permeia áreas superiores como a biologia e a sociologia, mas o estudo do caos fornece vários exemplos surpreendentes de sistemas muito diferentes com comportamento semelhante, e a teoria da universalidade os explica. O conjunto de Mandelbrot é um bonito exemplo de como um procedimento iterativo muito simples pode fazer surgir uma inesgotável variedade de formas. Se as montanhas, os rios, as nuvens e as árvores são resultado desses procedimentos – e a geometria fractal mostra que são – talvez tais procedimentos sejam apenas diferentes parametrizações de um único procedimento que possamos induzir a partir deles.

Na física, com frequência as mesmas equações aplicadas a diferentes quantidades descrevem fenômenos de áreas totalmente distintas, como a

mecânica quântica, o eletromagnetismo e a dinâmica dos fluidos. A equação da onda, a equação de difusão, a equação de Poisson: uma vez que as descobrimos em uma área, podemos aplicá-las mais facilmente em outras; e quando aprendemos como resolvê-las em uma área, podemos resolvê-las em todas. Além disso, essas equações são muito simples e envolvem os mesmos poucos derivados de quantidades relacionadas a espaço e tempo. Como era de se esperar, são todas instâncias de uma equação mestre, e tudo que o Algoritmo Mestre precisa fazer é descobrir como instanciá-las para diferentes conjuntos de dados.

Outra linha de evidência vem da otimização, o ramo da matemática que tenta descobrir a entrada de uma função que produza sua melhor saída. Por exemplo, descobrir a sequência de compras e vendas de ações que maximize o retorno total é um problema de otimização. Na otimização, normalmente funções simples fazem surgir soluções surpreendentemente complexas. Ela desempenha um papel de destaque em quase todas as áreas da ciência, da tecnologia e dos negócios, inclusive em machine learning. Cada área faz otimizações dentro dos limites definidos em outras áreas. Tentamos aumentar nossa felicidade obedecendo às restrições econômicas, descobrir quais são as melhores soluções para as empresas dependendo da tecnologia disponível – que, por sua vez, é composta pelas melhores soluções que pudermos encontrar dentro das restrições da biologia e da física. Já a biologia é resultado de uma otimização obtida por evolução dentro das restrições da física e da química, e as próprias leis da física são soluções para problemas de otimização. Talvez, então, tudo que existe seja a solução progressiva de um problema de otimização mais abrangente, e o Algoritmo Mestre surge da formulação desse problema.

Os físicos e matemáticos não são os únicos que encontram conexões inesperadas entre áreas distintas. Em seu livro *Consiliência*, o famoso biólogo E. O. Wilson defende com paixão o argumento de que existe uniformidade no conhecimento universal, das ciências naturais às humanas. O Algoritmo Mestre é a expressão definitiva dessa uniformidade: se todo o conhecimento compartilha um padrão comum, o Algoritmo Mestre existe, e vice-versa.

No entanto, a física é única em sua simplicidade. Fora da física e da engenharia, o registro obtido pela matemática é mais variado. Em alguns casos, ele só é razoavelmente efetivo, e em outros, seus modelos são simples demais para serem úteis. Porém, essa tendência à simplificação excessiva vem das

limitações da mente humana e não das limitações da matemática. Grande parte do “hardware” do cérebro (ou melhor, wetware) é dedicada às sensações e aos movimentos, e para fazer cálculos matemáticos temos de pegar emprestadas partes dele que evoluíram para a linguagem. Os computadores não têm essas limitações e podem transformar facilmente big data em modelos muito complexos. O machine learning é o que obtemos quando a irracional eficácia da matemática encontra a irracional eficácia dos dados. A biologia e a sociologia nunca serão tão simples como a física, mas o método pelo qual descobrimos suas verdades pode ser.

O argumento da estatística

De acordo com uma das escolas de estatística, uma única fórmula simples é a base de todo o conhecimento. O teorema de Bayes, como a fórmula é conhecida, instrui como atualizar nossas crenças sempre que houver novas evidências. Inicialmente, um aprendiz bayesiano tem um conjunto de hipóteses sobre o mundo. Quando encontra novos dados, as hipóteses compatíveis com eles se tornam mais prováveis, e as menos compatíveis, menos prováveis (ou mesmo impossíveis). Após o recebimento de dados suficientes, uma única hipótese vence, ou algumas. Por exemplo, se eu estivesse procurando um programa que previsse exatamente os movimentos das ações, e uma ação que um programa candidato previu que subiria descesse, esse candidato perderia credibilidade. Após eu ter examinado vários candidatos, só os confiáveis permaneceriam e encapsulariam meu novo conhecimento do mercado de ações.

O teorema de Bayes é uma máquina que transforma dados em conhecimento. De acordo com os estatísticos bayesianos, ele é a *única* maneira correta de transformar dados em conhecimento. Se eles estiverem certos, o teorema de Bayes é o Algoritmo Mestre ou o mecanismo que o controla. Outros estatísticos têm ressalvas sobre o modo como o teorema de Bayes é usado e preferem outras formas de aprender a partir de dados. Na época anterior à computação, o teorema de Bayes só podia ser aplicado a problemas muito simples, e a ideia de que seria um aprendiz universal parecia improvável. No entanto, junto ao big data e ao big computing, o teorema encontrou seu lugar no vasto espaço das hipóteses e se espalhou para todas as áreas de conhecimento concebíveis. Se há um limite para o que esse teorema pode aprender, ainda não o encontramos.

O argumento da ciência da computação

Quando era veterano na universidade, passei um verão inteiro jogando Tetris, um videogame altamente viciante em que peças de vários formatos caem e temos de tentar colocá-las o mais perto possível umas das outras; o jogo termina quando a pilha de peças alcança o topo da tela. Não fazia ideia de que essa seria minha introdução ao NP-completo, o problema mais importante da ciência da computação teórica. Na verdade, em vez de ser uma tarefa irrelevante, ser bom em Tetris – *realmente* dominar o jogo – é uma das coisas mais úteis que podemos fazer. Se você conseguir resolver o problema do Tetris, poderá resolver milhares de problemas mais difíceis e importantes da ciência, tecnologia e gerenciamento – tudo de uma só vez. Isso ocorre porque, na essência, eles são o *mesmo* problema. Esse é um dos fatos mais espantosos de toda a ciência.

Descobrir como as proteínas assumem suas formas características; reconstruir a história evolutiva de um conjunto de espécies a partir de seu DNA; comprovar teoremas em lógica proposicional; detectar oportunidades de arbitragem em mercados com os custos das transações; inferir uma forma tridimensional em cenários bidimensionais; compactar dados em um disco; formar uma coalizão política estável; modelar a turbulência em fluxos interrompidos; encontrar o portfólio de investimentos mais seguro que forneça um retorno específico, a rota mais curta para visitar um conjunto de cidades, o melhor layout de componentes em um microchip, a melhor posição para sensores em um ecossistema, ou o estado de energia mais baixo de um vidro de spin; programar voos, aulas e empregos na indústria; otimizar a alocação de recursos, o fluxo do tráfego urbano, o bem-estar social e (o mais importante) seu placar no Tetris: todos esses são problemas NP-completos, o que significa que, se você conseguir resolver um deles, poderá resolver todos os problemas da classe NP, inclusive uns aos outros. Quem poderia adivinhar que esses problemas, superficialmente tão diferentes, na verdade são um só? Porém, se o são, faz sentido que um único algoritmo consiga aprender a resolver todos (ou, mais precisamente, todas as instâncias eficientemente resolvíveis).

P e NP são as classes mais importantes de problemas da ciência da computação. (Infelizmente, os nomes não são muito mnemônicos.) Um problema é da classe P quando podemos resolvê-lo de modo eficiente e é da classe NP quando podemos verificar sua solução de modo eficiente. A famosa questão P = NP investiga se todos os problemas eficientemente verificáveis

também são eficientemente resolvíveis. Graças ao NP-completo, a única coisa necessária para a resolução dessa questão é provar que *um* problema NP-completo é (ou não) eficientemente resolvível. NP não é a classe de problemas mais difícil da ciência da computação, mas talvez seja a classe "realista" mais difícil: se você não conseguir nem mesmo verificar a solução de um problema antes do mundo acabar, de que adianta tentar resolvê-lo? Os humanos são bons em resolver problemas NP aproximadamente, e em reciprocidade, problemas que achamos interessantes (como o do Tetris) com frequência têm "algo de NP". Uma das definições dadas à inteligência artificial diz que sua função é encontrar soluções heurísticas para problemas NP-completos. Normalmente, fazemos isso reduzindo-os à satisfatibilidade, o problema NP-completo canônico: uma fórmula lógica específica pode ser sempre verdadeira ou isso é autocontraditório? Se inventarmos um aprendiz que possa aprender a resolver a satisfatibilidade, ele terá uma boa chance de ser o Algoritmo Mestre.

Deixando o NP-completo de lado, a abrupta existência de computadores é ela própria um poderoso sinal de que há um Algoritmo Mestre. Se você voltasse no tempo para o início do século 20 e dissesse às pessoas que uma máquina seria inventada em breve e resolveria problemas de todas as áreas do conhecimento humano – a *mesma* máquina para *todos* os problemas –, ninguém acreditaria. Elas diriam que cada máquina só pode fazer uma única coisa: máquinas de costura não datilografam, e máquinas de escrever não costuram. Então, em 1936, Alan Turing imaginou um dispositivo curioso com uma fita e um cabeçote que lia e gravava símbolos, agora conhecido como máquina de Turing. Todos os problemas imagináveis que podem ser resolvidos pela dedução lógica podem ser resolvidos por uma máquina de Turing. Além disso, uma assim chamada máquina de Turing universal pode simular qualquer problema lendo sua especificação em fita – em outras palavras, ela pode ser programada para fazer qualquer coisa.

O Algoritmo Mestre é para a indução, o processo de aprender, o que a máquina de Turing é para a dedução. Ele pode aprender a simular qualquer outro algoritmo lendo exemplos de seu comportamento de entrada-saída. Assim como há muitos modelos de computação equivalentes a uma máquina de Turing, devem existir muitas formulações diferentes equivalentes a um aprendiz universal. O importante, no entanto, é encontrar a primeira dessas formulações, como Turing encontrou a primeira formulação do computador de uso geral.

Machine learners versus engenheiros do conhecimento

É claro que o Algoritmo Mestre tem um número pelo menos igual de céticos e defensores. Surge uma dúvida quando algo parece a panaceia. A resistência mais acirrada vem da eterna inimiga do machine learning: a engenharia do conhecimento. De acordo com seus defensores, o conhecimento não pode ser aprendido automaticamente; deve ser programado no computador por especialistas humanos. É claro que os aprendizes podem extrair algo dos dados, mas nada que poderíamos confundir com conhecimento *real*. Para os engenheiros do conhecimento, big data não é o novo petróleo; é o novo charlatanismo.

Nos primórdios da IA, o machine learning parecia o caminho óbvio para a existência de computadores com inteligência semelhante à humana; Turing e outros achavam que esse fosse o *único* caminho plausível. Então, os engenheiros do conhecimento revidaram, e por volta de 1970 o machine learning foi firmemente deixado em suspensão. Por um momento na década de 80, parecia que a engenharia do conhecimento estava prestes a dominar o mundo, com empresas e países investindo massivamente. Porém, houve um rápido desapontamento, e o machine learning começou sua inexorável ascensão, primeiro silenciosamente e depois administrando uma imensa onda de dados.

Apesar dos sucessos do machine learning, os engenheiros do conhecimento não se convenceram. Eles acreditam que em breve suas limitações aparecerão e o pêndulo se inclinará para o outro lado. Marvin Minsky, professor do MIT e pioneiro na área de inteligência artificial, é um membro de destaque desse grupo. Além de Minsky ser cético em relação ao machine learning como alternativa à engenharia do conhecimento, ele não acredita em *nenhuma* ideia de uniformização em inteligência artificial. Sua teoria da inteligência, como expressa em seu livro *A sociedade da mente*, poderia ser cruelmente caracterizada como “a mente é apenas uma coisa acontecendo após a outra”. *A sociedade da mente* é uma lista de centenas de ideias separadas, cada uma com sua própria etiqueta. O problema dessa abordagem da inteligência artificial é que ela não funciona; seria como se o computador colecionasse selos. Sem o machine learning, o número de ideias necessárias para a construção de um agente inteligente é infinito. Se um robô tivesse as mesmas capacidades de um ser humano, exceto a aprendizagem, rapidamente o humano o deixaria de lado.

Minsky foi um assíduo defensor do projeto Cyc, o mais notório fracasso da

história da inteligência artificial. O objetivo do Cyc era resolver o problema da inteligência artificial inserindo em um computador todo o conhecimento necessário. Quando o projeto começou nos anos 1980, seu líder, Doug Lenat, previu com confiança que o sucesso viria dentro de uma década. Trinta anos depois, o Cyc continua a crescer sem parar e ainda não incorporou o bom senso. Ironicamente, Lenat resolveu, talvez tarde demais, preencher o Cyc minerando a web, não porque o Cyc possa ler, mas porque não há outra maneira.

Mesmo se por algum milagre conseguíssemos terminar de codificar todas as peças necessárias, nossos problemas só estariam começando. Com o passar dos anos, vários grupos de pesquisa tentaram construir agentes inteligentes completos juntando algoritmos de visão, reconhecimento de voz, entendimento da linguagem, raciocínio, planejamento, movimentação, manipulação, e assim por diante. Sem uma estrutura unificada, essas tentativas rapidamente se tornaram uma intransponível muralha de complexidade: partes móveis em excesso, muitas interações, bugs demais para serem resolvidos por simples engenheiros de software humanos. Os engenheiros do conhecimento acreditam que a inteligência artificial seja apenas um problema de engenharia, mas ainda não alcançamos o ponto em que a engenharia possa nos conduzir pelo resto do caminho. Em 1962, quando Kennedy fez seu famoso discurso de conquista da Lua, ir à Lua era um problema de engenharia. Em 1662, não era, e é mais ou menos nesse ponto que a inteligência artificial está hoje.

Na indústria, não há sinal de que a engenharia do conhecimento consiga um dia competir com o machine learning, exceto em alguns nichos. Por que pagar especialistas para codificar conhecimento de maneira lenta e difícil em uma forma que os computadores possam entender, quando podemos extraí-lo de dados por um custo bem menor? Sem mencionar todas as coisas que os especialistas não sabem, mas que podemos descobrir a partir de dados. E quando não há dados disponíveis, raramente o custo de usar a engenharia do conhecimento excede o benefício. Imagine se os fazendeiros tivessem de fabricar cada talo de trigo, em vez de plantar as sementes e deixá-las crescer: passaríamos fome.

Outro crítico proeminente do machine learning é o linguista Noam Chomsky. Chomsky acredita que a linguagem deve ser inata, porque os exemplos de sentenças gramaticais que as crianças ouvem não são suficientes para aprender gramática. No entanto, essa observação apenas passa a responsabilidade do

aprendizado da linguagem para a evolução; não é um argumento contra o Algoritmo Mestre, mas sim contra ele ser algo como o cérebro. Além disso, se existir uma gramática universal (como Chomsky acredita), sua elucidação seria um passo adiante na elucidação do Algoritmo Mestre. Só seria diferente se a linguagem não tivesse nada em comum com outras habilidades cognitivas, o que é implausível dado que é algo evolutivamente recente.

Seja como for, se testarmos o argumento da "escassez de estímulo" fornecido por Chomsky, veremos que ele é demonstrativamente falso. Em 1969, J. J. Horning provou que gramáticas probabilísticas livres de contexto só podem ser aprendidas a partir de exemplos positivos, e resultados mais consistentes vieram a seguir. (Gramáticas livres de contexto são o sustento do linguista, e os modelos de versão probabilística mostram com que probabilidade cada regra deve ser usada.) Além disso, o aprendizado da linguagem não acontece isoladamente; as crianças encontram todo tipo de pistas dadas por seus pais e pelo ambiente. Se conseguimos aprender a linguagem após alguns anos de exemplos, isso ocorre parcialmente devido à semelhança entre sua estrutura e a estrutura do mundo. É nessa estrutura comum que estamos interessados, e sabemos por Horning e outros que ela é suficiente.

Quase sempre Chomsky é um crítico de qualquer aprendizado estatístico. Ele tem uma lista de coisas que os aprendizes estatísticos não podem fazer, mas ela está desatualizada em cinquenta anos. Chomsky parece querer equiparar o machine learning ao behaviorismo, em que o comportamento animal é reduzido à associação entre resposta e recompensa. Porém, machine learning não é behaviorismo. Os algoritmos de aprendizado modernos podem aprender representações internas sofisticadas e não apenas associações par a par entre estímulos.

No fim das contas, a prova está na prática. Os aprendizes de linguagem estatísticos funcionam e os sistemas de linguagem fabricados à mão não. O alerta inicial veio na década de 70, quando a Darpa, setor de pesquisa do Pentágono, organizou o primeiro projeto de reconhecimento de voz de larga escala. Para a surpresa de todos, um aprendiz sequencial simples do tipo que Chomsky ridicularizava venceu habilmente um sofisticado sistema baseado em conhecimento. Atualmente, aprendizes como esse são usados em quase todos os recursos de reconhecimento de voz, inclusive no do Siri. Fred Jelinek, chefe do grupo de reconhecimento de voz da IBM, ficou famoso por dizer ironicamente

“sempre que irrito um linguista, o desempenho do reconhecimento de voz melhora”. Presos no atoleiro da engenharia do conhecimento, os linguistas computacionais tiveram uma experiência de quase morte no fim dos anos 80. Desde então, métodos baseados em aprendizado inundaram a área a ponto de ser difícil encontrar uma palestra em uma conferência de linguística computacional que não aborde o aprendizado. Analisadores estatísticos decompõem a linguagem com precisão próxima a dos humanos, situação em que os analisadores codificados manualmente ficaram muito atrás. A tradução feita por máquina, a correção ortográfica, a rotulação de categorias morfossintáticas, a desambiguação de sentido de palavras, o fornecimento de respostas a perguntas, o diálogo, a sumarização: os melhores sistemas dessas áreas usam o aprendizado. O Watson, campeão computadorizado do *Jeopardy!*, não teria sido possível sem ele.

A isso Chomsky responderia dizendo que os sucessos da engenharia não são prova de validade científica. Por outro lado, se nossos prédios ruírem e nossos motores não funcionarem, talvez haja algo errado com a teoria da física. Chomsky acha que os linguistas devem se dedicar a falantes-ouvintes “ideais”, como definido por ele, e isso lhe permite ignorar coisas como a necessidade de estatística no aprendizado da linguagem. Logo, não é de surpreender que poucos pesquisadores ainda levem suas teorias a sério.

Outra fonte potencial de objeções ao Algoritmo Mestre é a noção, popularizada pelo psicólogo Jerry Fodor, de que a mente é composta por um conjunto de módulos com comunicação limitada apenas entre eles. Por exemplo, quando você assiste à TV, seu “cérebro superior” sabe que ela é apenas uma luz que pisca sobre uma superfície plana, mas seu sistema visual ainda vê formas tridimensionais. No entanto, mesmo se acreditarmos na modularidade da mente, isso não significa que diferentes módulos usem algoritmos de aprendizado distintos. O mesmo algoritmo operando sobre, digamos, informações visuais e verbais pode ser suficiente.

Críticos como Minsky, Chomsky e Fodor já tiveram defensores, mas felizmente sua influência diminuiu. Mesmo assim, ao tentar encontrar o Algoritmo Mestre devemos nos lembrar de suas críticas por duas razões. A primeira é que os engenheiros do conhecimento se depararam com muitos dos mesmos problemas que os machine learners encontram, e, mesmo não tendo sucesso, eles aprenderam várias lições valiosas. A segunda é que o aprendizado e

o conhecimento estão interligados de maneiras muito sutis, como descobriremos em breve. Lamentavelmente, com frequência os dois grupos não dialogam. Eles falam linguagens diferentes: o machine learning fala de probabilidade, e a engenharia do conhecimento, de lógica. Veremos posteriormente neste livro o que fazer sobre isso.

O cisne bica o robô

"Não importa quanto seu algoritmo seja inteligente, há algumas coisas que ele nunca aprenderá". Fora da inteligência artificial e da ciência cognitiva, as objeções mais comuns ao machine learning são variantes dessa alegação. Nassim Taleb se baseou nela insistentemente em seu livro *A lógica do cisne negro*. Alguns eventos simplesmente não são previsíveis. Se até hoje você só viu cisnes brancos, deve achar que a probabilidade de ver um cisne negro é zero. O baque financeiro de 2008 foi um "cisne negro".

É verdade que algumas coisas são previsíveis e outras não, e a primeira missão do machine learner é saber diferenciá-las. Porém, o objetivo do Algoritmo Mestre é aprender tudo que *possa* ser aprendido, e esse é um domínio muito mais amplo que Taleb e outros imaginam. A bolha imobiliária estava longe de ser um cisne negro; pelo contrário, foi amplamente prevista. A maioria dos modelos bancários não conseguiu detectar sua chegada, mas isso ocorreu devido às conhecidas limitações desses modelos e não às limitações do machine learning em geral. Os algoritmos de aprendizado são capazes de prever com precisão eventos raros nunca vistos antes; poderíamos dizer até que essa é a função principal do machine learning. Qual é a probabilidade de você ver um cisne negro se nunca viu um? Não seria uma pequena fração de espécies conhecidas que tardiamente começaram a produzir espécimes negros? Esse é apenas um exemplo simples; veremos vários outros mais complexos neste livro.

Uma objeção relacionada ouvida com frequência é: "Dados não podem substituir a intuição humana". Na verdade, é o oposto: a intuição humana não pode substituir os dados. Intuição é o que usamos quando não conhecemos os fatos, e já que normalmente não os conhecemos, ela é preciosa. Porém, quando a evidência está na nossa frente, por que negá-la? A análise estatística vence batedores de beisebol talentosos (como Michael Lewis memoravelmente documentou em *Moneyball – O homem que mudou o jogo*), vence especialistas na degustação de vinhos, e todo dia vemos novos exemplos do que ela pode

fazer. Devido à afluência de dados, o limite entre evidência e intuição está mudando rapidamente, e como ocorre em qualquer revolução, barreiras defensivas devem ser superadas. Se eu fosse o especialista em X na empresa Y, não gostaria de ser superado por alguém com dados. Há um ditado na indústria: "Ouça seus clientes, não a Hippo", em que Hippo é a abreviatura de "highest paid person's opinion" (opinião da pessoa de salário mais alto). Se quiser ser a autoridade de amanhã, aceite os dados em vez de combatê-los.

Também há quem diga que o machine learning pode encontrar regularidades estatísticas em dados, porém não descobrirá nada em um nível mais profundo, como as leis de Newton. Talvez ele ainda não o tenha feito, mas aposto que fará. Deixando de lado histórias de maçãs que caem, verdades científicas profundas não são os frutos de mais fácil acesso. A ciência passa por três fases, que podemos chamar de fases de Brahe, Kepler e Newton. Na fase de Brahe, coletamos uma grande quantidade de dados, como Tycho Brahe ao registrar pacientemente as posições dos planetas noite após noite, ano após ano. Na fase de Kepler, atribuímos leis empíricas aos dados, como Kepler fez aos movimentos dos planetas. Na fase de Newton, descobrimos as verdades mais profundas. Grande parte da ciência é composta por trabalho como o de Brahe e Kepler; os momentos "Newton" são raros. Atualmente, o big data faz o trabalho de bilhões de Brahes, e o machine learning, o trabalho de milhões de Keplers. Se houver mais momentos "Newton" para ocorrer (esperamos que sim), provavelmente eles virão dos algoritmos de aprendizado e dos surpresos cientistas de amanhã, ou pelo menos de uma combinação dos dois. (É claro que os prêmios Nobel irão para os cientistas, não importando se tiveram os principais insights ou apenas pressionaram um botão. Os algoritmos de aprendizado não têm ambições.) Veremos neste livro como seriam esses algoritmos e especularemos sobre o que eles poderiam descobrir – por exemplo, a cura do câncer.

O Algoritmo Mestre é uma raposa ou um porco-espinho?

Precisamos considerar mais uma possível objeção ao Algoritmo Mestre, talvez a mais séria de todas. Ela vem não de engenheiros do conhecimento ou especialistas descontentes, mas dos próprios especialistas em machine learning. Personificando um deles por um momento, eu diria: "Mas o Algoritmo Mestre não parece com a minha vida diária. Testo centenas de variações de muitos

algoritmos de aprendizado diferentes em algum problema específico e algoritmos distintos se saem melhor em determinados problemas. Como um único algoritmo poderia substituir todos?”.

Para o que a resposta é: realmente. Não seria bom se, em vez de testar centenas de variações de muitos algoritmos, só tivéssemos de testar centenas de variações de um único algoritmo? Se conseguirmos descobrir o que é importante e o que não é em cada variação, o que as partes importantes têm em comum e como se complementam, poderemos sintetizar um Algoritmo Mestre a partir delas. É o que faremos neste livro, ou tentaremos chegar o mais próximo possível. Talvez você, caro leitor, tenha suas próprias ideias ao avançar na leitura.

Que complexidade terá o Algoritmo Mestre? Milhares de linhas de código? Milhões? Não sabemos ainda, mas o machine learning tem um histórico muito interessante de algoritmos simples inesperadamente serem melhores que algoritmos sofisticados. Em uma famosa passagem de seu livro *As ciências do artificial*, o pioneiro da inteligência artificial e ganhador do prêmio Nobel, Herbert Simon, nos pede para pensar em uma formiga tentando dificilmente chegar em casa atravessando uma praia. O caminho da formiga é complexo, não porque ela própria seja complexa, mas porque o ambiente tem muitas dunas a serem superadas e pedras a serem contornadas. Se tentássemos modelar a trajetória da formiga programando cada caminho possível, estaríamos perdidos. Da mesma forma, no machine learning a complexidade está nos dados; tudo que o Algoritmo Mestre tem de fazer é assimilá-los, logo não devemos ficar surpresos se no fim das contas ele se mostrar simples. A mão humana é simples – quatro dedos e um polegar opositor – e mesmo assim ela pode fabricar e usar uma variedade infinita de ferramentas. O Algoritmo Mestre é para os algoritmos o que a mão é para as canetas, espadas, chaves de fenda e garfos.

Como Isaiah Berlin memoravelmente observou, alguns pensadores são como raposas – conhecem várias coisas menores – e outros são porcos-espinhos – são melhores em uma única coisa maior. O mesmo ocorre com os algoritmos de aprendizado. Espero que o Algoritmo Mestre seja um porco-espinho, mas mesmo se for uma raposa não chegaremos a ele com rapidez suficiente. O maior problema dos algoritmos de aprendizado atuais não é o fato de serem plurais, mas sim que, mesmo sendo úteis, eles não fazem tudo que gostaríamos que fizessem. Antes de conseguirmos descobrir verdades profundas com o machine learning, temos de descobrir verdades profundas sobre a própria técnica.

O que está em jogo?

Suponhamos que você tivesse sido diagnosticado com câncer e os tratamentos tradicionais – cirurgia, quimioterapia e radioterapia – falhassem. O que ocorrerá a seguir determinará se você vive ou morre. A primeira etapa é sequenciar o genoma do tumor. Empresas como a Foundation Medicine em Cambridge, Massachusetts, fazem isso: envie uma amostra do tumor e elas retornarão uma lista das mutações conhecidas relacionadas ao câncer encontradas em seu genoma. Isso é necessário porque cada caso de câncer é diferente e não há uma droga única que funcione para todos os casos. Os cânceres sofrem mutações ao se espalhar pelo corpo, e pela seleção natural as mutações mais resistentes às drogas tomadas têm mais probabilidade de crescer. A droga certa para você pode ser que só funcione para 5% dos pacientes, ou talvez seja preciso uma combinação de drogas que nunca foram testadas. Pode ser necessária uma nova droga projetada especificamente para o seu caso ou uma sequência de drogas para evitar as adaptações do câncer. Essas drogas também podem ter efeitos colaterais mortais para você, mas não para a maioria das pessoas. Nenhum médico terá como obter todas as informações necessárias para prever qual é o melhor tratamento para o seu caso, dado seu histórico clínico e o genoma de seu câncer. É uma tarefa ideal para o machine learning, porém os aprendizes de hoje não estão à altura. Cada um deles tem algumas das habilidades necessárias, mas faltam outras. O Algoritmo Mestre é o pacote completo. Será por intermédio de sua aplicação a vastas quantidades de dados de pacientes e drogas, combinada ao conhecimento extraído da literatura médica, que curaremos o câncer.

Um aprendiz universal é extremamente necessário em muitas outras áreas, de situações de vida e morte a casos menos graves. Imagine o sistema de recomendação ideal, um que recomendasse os livros, filmes e dispositivos que você escolheria se tivesse tempo para examinar todos. O algoritmo da Amazon não chega nem perto. Isso ocorre parcialmente porque ele não tem dados suficientes – principalmente, sabe apenas os itens que já compramos na Amazon –, mas mesmo se você lhe desse acesso ao seu fluxo de consciência completo desde o nascimento, ele não saberia o que fazer com as informações. Como transmutar o caleidoscópio de sua vida, a miríade de diferentes escolhas que fez, em um quadro coerente de quem você é e do que deseja? Essa tarefa está bem além do alcance dos aprendizes atuais, mas recebendo dados suficientes, o Algoritmo Mestre conseguirá conhecê-lo quase como o seu melhor amigo.

Algum dia haverá um robô em todos os lares, lavando os pratos, arrumando as camas e até mesmo tomando conta das crianças enquanto os pais trabalham. Se isso vai ocorrer rápido vai depender da dificuldade que tivermos para encontrar o Algoritmo Mestre. Se o melhor que pudermos fazer for combinar muitos aprendizes diferentes, cada um resolvendo apenas uma pequena parte do problema da inteligência artificial, não demoraremos a ter um muro de complexidade. Essa abordagem por partes funcionou para o *Jeopardy!*, mas poucas pessoas acreditam que os robôs domésticos de amanhã serão netos do Watson. Não quero dizer que o Algoritmo Mestre resolverá o problema da inteligência artificial sozinho; haverá grandes feitos de engenharia a serem realizados, e o Watson é uma boa amostra deles. Porém, a regra 80/20 é aplicável: o Algoritmo Mestre será 80% da solução e 20% do trabalho, logo certamente é o melhor ponto de partida.

O impacto do Algoritmo Mestre sobre a tecnologia não está restrito à inteligência artificial. Um aprendiz universal é uma arma fenomenal contra o monstro da complexidade. Sistemas que hoje são complexos demais para serem construídos deixarão de o ser. Os computadores farão mais coisas com menos ajuda nossa. Eles não repetirão sempre os mesmos erros e aprenderão com a prática, como as pessoas fazem. Como os mordomos nas histórias, em algumas situações até adivinharão o que desejamos antes de o expressarmos. Se atualmente eles nos tornam mais inteligentes, os computadores que executarem o Algoritmo Mestre farão com que nos achemos gênios.

O progresso tecnológico irá se acelerar de maneira notável, não apenas na ciência da computação, mas em muitas áreas diferentes. Isso, por sua vez, aumentará o crescimento econômico e ajudará a diminuir a pobreza. Com o Algoritmo Mestre ajudando a sintetizar e distribuir conhecimento, a inteligência de uma empresa será maior que a da soma de suas partes, não menor. Tarefas rotineiras serão automatizadas e substituídas por outras mais interessantes. Todas as tarefas serão executadas melhor que hoje, seja por uma pessoa mais treinada, um computador ou uma combinação dos dois. Ocorrerão menos crashes no mercado de ações e eles terão menor gravidade. Com uma refinada grade de sensores cobrindo todo o planeta e modelos aprendidos avaliando suas saídas a cada momento, não estaremos mais caminhando às cegas; a saúde de nosso planeta melhorará. Um modelo seu negociará tudo em seu nome, participando de jogos elaborados com os modelos de outras pessoas e entidades. Como resultado

de todas essas mudanças, nossa vida será mais longa, feliz e produtiva.

Já que o impacto potencial é tão grande, temos de tentar inventar o Algoritmo Mestre mesmo se as chances de sucesso forem baixas. E mesmo que demore muito, a procura de um aprendiz universal tem vários benefícios imediatos. Um deles é a melhor compreensão do machine learning ocasionada por uma visão unificada. Muitas decisões empresariais são tomadas com pouca compreensão das bases analíticas que as justificam, mas não precisa ser assim. Para usar uma tecnologia não precisamos dominar seu funcionamento interno, mas sim ter um bom modelo conceitual dela. Temos de saber como encontrar uma estação no rádio ou alterar o volume. Atualmente, os que não são especialistas em machine learning não têm um modelo conceitual do que um aprendiz faz. Os algoritmos que acionamos quando usamos o Google, o Facebook ou o conjunto mais recente de análise são um pouco como se uma limusine negra com as janelas tingidas aparecesse misteriosamente em nossa porta uma noite: devemos entrar? Aonde ela nos levará? É hora de ocupar o assento do motorista. Conhecer as suposições que diferentes aprendizes fazem nos ajudará a selecionar qual é o correto para a tarefa, em vez de adotar um aleatório surgido do nada – e, então, aturá-lo durante anos, redescobrindo dolorosamente o que devíamos saber desde o início. Sabendo o que os aprendizes otimizam, podemos garantir que eles otimizem aquilo em que estamos interessados, em vez de algo sobre o qual não temos conhecimento. Talvez o mais importante seja que, quando soubermos como um aprendiz específico chega às suas conclusões, saberemos o que fazer com essas informações – o que esperar, o que retornar para o fabricante e como obter um resultado melhor da próxima vez. E com o aprendiz universal que desenvolveremos neste livro como modelo conceitual, podemos fazer tudo isso sem sobrecarga cognitiva. Em sua essência, o machine learning é simples; só temos de remover as camadas de matemática e jargão para revelar a boneca russa interna.

Esses benefícios são aplicáveis à nossa vida tanto pessoal quanto profissional. Como tirar o melhor da trilha de dados que cada passo meu deixa no mundo moderno? Toda transação funciona em dois níveis: o que ela traz para nós e o que ensina ao sistema com o qual interagimos. Estar ciente disso é a primeira etapa para uma vida feliz no século 21. Ensine os aprendizes e eles o servirão; mas primeiro você precisa entendê-los. O que em minha função pode ser feito por um algoritmo de aprendizado, o que não pode e – o mais importante – como

posso me beneficiar do machine learning para fazê-lo melhor? O computador é sua ferramenta e não seu adversário. Armado com o machine learning, o gerente se torna um supergerente, o cientista um supercientista e o engenheiro um superengenheiro. O futuro pertence àqueles que souberem em um nível muito profundo como combinar sua área de especialização com o que o algoritmo faz melhor.

Porém, talvez o Algoritmo Mestre seja uma caixa de Pandora que seria melhor deixar fechada. Os computadores nos escravizarão ou mesmo nos exterminarão? O machine learning será instrumento de ditadores ou corporações maléficas? Conhecer o caminho que o machine learning está tomando nos permitirá saber com o que devemos ou não nos preocupar e o que fazer sobre isso. O cenário do *Exterminador do futuro*, em que uma inteligência artificial superior se torna consciente e subjuga a humanidade com um exército de robôs, não tem como ocorrer com os tipos de algoritmos de aprendizado que veremos neste livro. Só porque os computadores podem aprender não significa que adquirirão magicamente vontade própria. Os aprendizes são instruídos a atingir os objetivos para os quais os programamos; eles não conseguem mudá-los. Em vez disso, é preciso evitar que nos sirvam de maneira que prejudiquem mais do que ajudem, porque eles não sabem a diferença, e a solução é ensiná-los melhor.

Acima de tudo, temos de nos preocupar com o que o Algoritmo Mestre poderia fazer em mãos erradas. A primeira linha de defesa é assegurar que pessoas bem-intencionadas o alcancem antes – ou, se não estiver claro quem é bem-intencionado, assegurar que ele seja open source. A segunda é saber que, independentemente do nível de excelência do algoritmo de aprendizado, ele será tão bom quanto os dados que receber. Quem controlar os dados controlará o aprendiz. Nossa reação à dataficação (datafication) da vida não deve ser a de nos isolarmos em uma choupana na floresta – as florestas também estão cheias de sensores –, mas tentar controlar os dados que nos interessam. É bom que existam agentes de recomendação que encontrem e tragam o que queremos; ficaríamos perdidos sem isso. Porém, eles devem trazer o que queremos e não o que terceiros tentem nos empurrar. O controle dos dados e a posse dos modelos aprendidos com eles são as metas ao redor das quais muitas das batalhas do século 21 ocorrerão – entre governos, empresas, sindicatos e pessoas. Mas há o dever ético de compartilhar dados para o bem comum. O machine learning sozinho não curará o câncer; quem fará isso são os pacientes, compartilhando

seus dados para o bem de futuros doentes.

Uma teoria de tudo diferente

Hoje em dia a ciência está totalmente compartimentada, uma Torre de Babel em que cada subcomunidade fala seu próprio jargão e só enxerga o conteúdo de algumas subcomunidades adjacentes. O Algoritmo Mestre forneceria uma visão unificada de toda a ciência e possivelmente levaria a uma nova teoria de tudo. À primeira vista, pode parecer uma pretensão bizarra. O que o machine learning faz é induzir teorias a partir de dados. Como o próprio Algoritmo Mestre poderia se transformar em uma teoria? A teoria das cordas não é a teoria de tudo, e o Algoritmo Mestre também não é exatamente isso?

Para responder a essas perguntas, primeiro temos de entender o que é e o que não é uma teoria científica. Uma teoria é um conjunto de restrições para o que o mundo poderia ser e não uma descrição completa dele. Para obter esta última, você tem de combinar a teoria a dados. Por exemplo, considere a segunda lei de Newton. Ela diz que a força é igual à massa vezes a aceleração, ou $F = ma$. Não nos diz qual é a massa ou a aceleração de um objeto, ou as forças que atuam sobre ele. Ela só preconiza que, se a massa de um objeto for m e sua aceleração for a, então a força total deve ser ma. Isso remove alguns níveis de liberdade do universo, mas não todos. O mesmo ocorre com todas as outras teorias da física, inclusive da relatividade, mecânica quântica e a teoria das cordas, que são, na verdade, refinamentos da lei de Newton.

O poder de uma teoria está em quanto ela simplifica nossa descrição do mundo. Usando as leis de Newton, só precisamos saber a massa, a posição e a velocidade de todos os objetos em um determinado momento no tempo; sua posição e velocidade em todos os outros momentos advirão daí. Logo, as leis de Newton reduzem nossa descrição do mundo conforme o número de instantes distinguíveis na história do universo, no passado e no futuro. Impressionante! É claro que elas são só uma aproximação das leis reais da física, portanto as substituiremos pela teoria das cordas, ignorando todos os seus problemas e a questão de poder ou não ser validada empiricamente. Podemos fazer melhor? Sim, por duas razões.

A primeira é que, na realidade, nunca teremos dados suficientes para determinar o mundo completamente. Mesmo ignorando o princípio da incerteza, saber precisamente a posição e velocidade de todas as partículas existentes em

algum momento no tempo não é nem remotamente viável. E, já que as leis da física são caóticas, a incerteza aumenta com o tempo e rapidamente elas determinam muito pouco. Para descrever o mundo de maneira exata, precisamos de um lote atualizado de dados em intervalos regulares. Na verdade, as leis da física só nos dizem o que ocorre localmente. Isso reduz muito o seu poder.

O segundo problema é que, mesmo se tivéssemos um conhecimento total do mundo em algum momento no tempo, as leis da física ainda não nos permitiriam determinar seu passado e seu futuro. Isso ocorre porque o aumento do nível de computação necessário para obter essas previsões estaria além da capacidade de qualquer computador concebível. De fato, para simular perfeitamente o universo precisaríamos de outro universo idêntico. Esse é o motivo pelo qual a teoria das cordas é em grande parte irrelevante fora da física. As teorias que temos em biologia, psicologia, sociologia ou economia não são corolários das leis da física; elas tiveram de ser criadas a partir do zero. Presumimos que sejam aproximações do que as leis da física preveriam se aplicadas na escala de células, cérebros e sociedades, mas não há como saber.

Ao contrário das teorias de uma determinada área, que só têm poder dentro dessa área, o Algoritmo Mestre terá poder em todas as áreas. Dentro da área X, ele terá menos poder que a teoria prevalente na área, mas em todas as áreas – se considerarmos o mundo inteiro – terá amplamente mais poder que qualquer outra teoria. O Algoritmo Mestre é o embrião de todas as teorias; para obter a teoria X só teremos de adicionar a ele a quantidade mínima de dados necessária à sua indução. (No caso da física, seriam apenas os resultados de algumas centenas de experimentos-chave.) A consequência é que, incontestavelmente, o Algoritmo Mestre pode ser o melhor ponto de partida já visto para uma teoria de tudo. Com todo o respeito a Stephen Hawking, ele nos dirá mais sobre a mente de Deus que a teoria das cordas.

Alguém poderia dizer que procurar um aprendiz universal seria o epítome da húbris tecnológica. Mas sonhar não é húbris. Talvez o Algoritmo Mestre venha a assumir seu lugar entre as grandes quimeras, junto à pedra filosofal e à máquina de movimento perpétuo. Ou talvez seja mais como calcular a longitude no mar, um problema considerado muito difícil até um gênio solitário resolvê-lo. O mais provável é que seja um trabalho que atravesse gerações, erguido pedra a pedra como uma catedral. A única maneira de saber é um dia acordarmos cedo e começar a jornada.

Candidatos que não têm chance

Mas, se o Algoritmo Mestre existe, o que seria? Um candidato aparentemente óbvio é a memorização: apenas lembre-se de tudo que já viu; após algum tempo você terá visto tudo que há para ver e, portanto, saberá tudo que há para saber. O problema desse candidato é que, como disse Heráclito, nunca pisamos no mesmo rio duas vezes. Há muito mais para se ver do que jamais conseguiríamos. Não importa quantos flocos de neve tivermos visto, o próximo será diferente. Mesmo se você tivesse estado presente no Big Bang e em todos os eventos ocorridos desde então, teria visto somente uma pequena fração do que verá no futuro. Se tivesse sido testemunha da vida na Terra no intervalo de até dez mil anos atrás, isso não o teria preparado para o que está por vir. Alguém que cresceu em uma cidade não fica paralisado quando muda para outra, mas um robô que só tivesse a habilidade de memorização ficaria. O conhecimento também não é apenas uma longa lista de fatos. Ele é geral e tem estrutura. “Todos os humanos são mortais” é muito mais sucinto que sete bilhões de declarações de mortalidade, uma para cada ser humano. A memorização não nos dá nada disso.

Outro candidato a Algoritmo Mestre seria o microprocessador. Afinal, este que usamos em nosso computador pode ser visto como um algoritmo cuja tarefa é executar outros algoritmos, como uma máquina de Turing universal; e ele pode executar qualquer algoritmo imaginável, respeitando-se seus limites de memória e velocidade. Na verdade, para um microprocessador um algoritmo é apenas outro tipo de dado. O problema é que sozinho o microprocessador não sabe fazer nada; ele ficaria apenas ocioso o dia inteiro. De onde vêm os algoritmos que ele executa? Se forem codificados por um programador humano, não haverá aprendizado. No entanto, há uma sensação de que o microprocessador seja uma boa analogia para o Algoritmo Mestre. Ele não é o melhor hardware para a execução de um algoritmo específico. O ideal seria um circuito integrado específico para aplicativos (ASIC, application-specific integrated circuit) projetado de maneira muito precisa para esse algoritmo. Mesmo assim são os microprocessadores que usamos para quase todos os aplicativos, porque sua flexibilidade compensa sua relativa ineficiência. Se tivéssemos de construir um ASIC para cada aplicativo novo, a Revolução da Informação nunca teria ocorrido. Da mesma forma, o Algoritmo Mestre não é o melhor algoritmo para o aprendizado de algum conhecimento específico; o ideal seria um algoritmo que já codificasse grande parte do conhecimento (ou todo ele, tornando os dados

supérfluos). Contudo, o que queremos é induzir o conhecimento a partir de dados, porque é mais fácil e custa menos; logo, quanto mais genérico for o algoritmo de aprendizado, melhor ele será.

Uma candidata ainda mais extrema é a simples porta NOR: um switch lógico cuja saída é 1 somente se suas duas entradas forem 0. Lembre-se de que todos os computadores são compostos por portas lógicas construídas com base em transistores, e os processamentos podem ser reduzidos a combinações das portas AND, OR e NOT. Uma porta NOR é apenas uma porta OR seguida de uma porta NOT: a negação de uma disjunção, como em "Ficarei feliz contanto que não passe fome ou fique doente". AND, OR e NOT podem ser implementadas com o uso de portas NOR, logo NOR pode fazer tudo e, na verdade, é só o que alguns microprocessadores usam. Então, por que ela não pode ser o Algoritmo Mestre? Sua simplicidade é certamente imbatível. Infelizmente, uma porta NOR é o Algoritmo Mestre tanto quanto uma peça de Lego é o brinquedo universal. A peça de Lego pode ser um bloco de construção universal para brinquedos, mas uma pilha de Lego não se constitui sozinha espontaneamente em um brinquedo. O mesmo se aplica a outros esquemas simples de computação, como as redes de Petri ou os autômatos celulares.

Passando para alternativas mais sofisticadas, e quanto às consultas que qualquer mecanismo de banco de dados pode responder, ou os algoritmos simples de um pacote estatístico? Eles não são suficientes? Essas alternativas são tijolos de Lego maiores, mas continuam sendo apenas tijolos. Um mecanismo de banco de dados nunca descobre algo novo; ele informa apenas o que já sabe. Mesmo com todos os humanos que existem em um banco de dados sendo mortais, não ocorrerá a ele generalizar a mortalidade para outros humanos. (Os engenheiros de banco de dados empalidecem só de pensar nisso.) Grande parte do que se faz em estatística gira ao redor de testar hipóteses, mas antes alguém tem de formulá-las. Os pacotes estatísticos podem executar regressão linear e outros procedimentos simples, mas têm um limite muito baixo para o que podem aprender, não importando a quantidade de dados que recebam. Os melhores pacotes ficam na zona intermediária entre a estatística e o machine learning, mas há muitos outros tipos de conhecimento que eles não conseguem descobrir.

Certo, é hora de ser mais direto: o Algoritmo Mestre é a equação $U(X) = 0$. Ela não só cabe em uma camiseta como também em um selo de postagem. O quê? $U(X) = 0$ diz apenas que alguma função U (possivelmente muito complexa) de

alguma variável *X* (possivelmente também muito complexa) é igual a 0. Todas as equações podem ser reduzidas a essa fórmula; por exemplo, $F = ma$ é equivalente a $F - ma = 0$, logo, se você considerar $F - ma$ como uma função U de F, voilà: $U(F) = 0$. Normalmente, X pode ser qualquer entrada e U qualquer algoritmo, e com certeza o Algoritmo Mestre não pode ser mais geral que isso; se estamos procurando o algoritmo mais geral que pode ser encontrado, deve ser esse. É claro que estou apenas brincando, mas esse candidato inaceitável aponta para um perigo real do machine learning: a invenção de um aprendiz que seja tão geral que não tenha conteúdo suficiente para ser útil.

Mas qual é o menor conteúdo que um aprendiz deve ter para ser útil? Que tal as leis da física? Afinal, tudo que existe no mundo as segue (achamos que sim) e elas deram origem à evolução e (por intermédio desta) ao cérebro. Bem, talvez o Algoritmo Mestre esteja implícito nas leis da física, e, se estiver, temos de torná-lo explícito. Simplesmente fornecer dados para as leis da física não resultará em uma nova lei. Vejamos uma maneira de considerar o assunto: a teoria mestra de uma determinada área seria composta pelas leis da física compiladas em uma forma mais conveniente para essa área, mas, se assim fosse, precisaríamos de um algoritmo que encontrasse um atalho para levar dos dados da área à sua teoria, e não está claro se as leis da física podem ajudar nisso. Outro problema é que, se as leis da física fossem diferentes, presumivelmente o Algoritmo Mestre também conseguiria descobri-las em muitos casos. Os matemáticos gostam de dizer que Deus pode desobedecer às leis da física, mas mesmo ele não pode desafiar as leis da lógica. Pode até ser dessa forma, mas as leis da lógica são para dedução; precisamos de algo equivalente, mas para indução.

As cinco tribos do machine learning

É claro que não precisamos começar nossa busca do Algoritmo Mestre do zero. Temos algumas décadas de pesquisa em machine learning nas quais nos basear. Algumas das pessoas mais inteligentes do planeta dedicaram suas vidas a inventar algoritmos de aprendizado, e uma parte delas chegou inclusive a alegar que tinha um aprendiz universal em mãos. Vamos continuar o trabalho desses pioneiros, mas consideraremos essas reivindicações com certo cuidado, o que traz a pergunta: como saberemos se encontramos o Algoritmo Mestre? Quando o mesmo aprendiz, somente com mudanças de parâmetros e entradas mínimas além dos dados, conseguir entender vídeo e texto assim como os humanos e fizer

novas descobertas significativas em biologia, sociologia e outras ciências. Obviamente, de acordo com esse padrão, nenhum aprendiz comprovou ser o Algoritmo Mestre, mesmo no caso improvável de já existir um.

O Algoritmo Mestre não precisa começar do zero em cada novo problema. Talvez essa barreira seja muito alta para *qualquer* aprendiz superar e, com certeza, não é o que as pessoas fazem. Por exemplo, a linguagem não existe isoladamente; não poderíamos entender uma frase se não conhecêssemos o assunto ao qual ela se refere. Logo, ao aprender a ler, o Algoritmo Mestre precisa já ter aprendido a ver, ouvir e controlar um robô. Da mesma forma, um cientista não ajusta modelos cegamente aos dados; ele pode usar todo o seu conhecimento na área para atuar sobre o problema. Portanto, antes de fazer descobertas em biologia, o Algoritmo Mestre precisa ler todo o conhecimento necessário dessa área, contando com o fato de já ter aprendido a ler. O Algoritmo Mestre não é apenas um consumidor passivo de dados; ele pode interagir com seu ambiente e procurar os dados necessários ativamente, como Adam, o cientista robô, ou como qualquer criança explorando seu mundo.

A busca do Algoritmo Mestre é dificultada, mas também estimulada, pelas escolas de pensamento rivais que existem na área do machine learning. As principais são os simbolistas, os conexionistas, os evolucionários, os bayesianos e os analogistas. Cada tribo tem um conjunto de crenças básicas e um problema específico com o qual se preocupa mais. Ela encontrou a solução para esse problema, contando com as ideias das áreas da ciência que são suas aliadas, e tem um algoritmo mestre que a incorpora.

Para os simbolistas, toda a inteligência pode ser reduzida à manipulação de símbolos, da mesma forma que um matemático resolve equações substituindo uma expressão por outra. Os simbolistas acham que não podemos aprender do zero: é preciso que algum conhecimento inicial acompanhe os dados. Eles descobriram como incorporar conhecimento preexistente ao aprendizado e como combinar diferentes áreas de conhecimento dinamicamente para resolver novos problemas. Seu algoritmo mestre é a dedução inversa, que detecta qual conhecimento está faltando para que uma dedução seja aceita e a torna o mais geral possível.

Para os conexionistas, aprendizado é a função do cérebro e, portanto, o que precisamos fazer é torná-lo alvo de engenharia reversa. O cérebro aprende ajustando as forças das conexões entre neurônios, e o problema crucial é

descobrir quais conexões são responsáveis por quais erros e alterá-las de acordo. O algoritmo mestre dos conexionistas é a backpropagation, que compara a saída de um sistema com a saída desejada e altera sucessivamente as conexões em cada camada de neurônios para aproximar mais a saída do que ela deveria ser.

Os evolucionários acreditam que a mãe de todo o aprendizado é a seleção natural. Se ela nos criou, pode fazer qualquer coisa, e só precisamos simulá-la no computador. O problema-chave que os evolucionários resolvem é a estrutura do aprendizado: não só ajustando parâmetros, como faz a backpropagation, mas criando o cérebro que essas adaptações possam então ajustar. O algoritmo mestre dos evolucionários é a programação genética, que une e desenvolve programas de computador da mesma forma que a natureza une e desenvolve organismos.

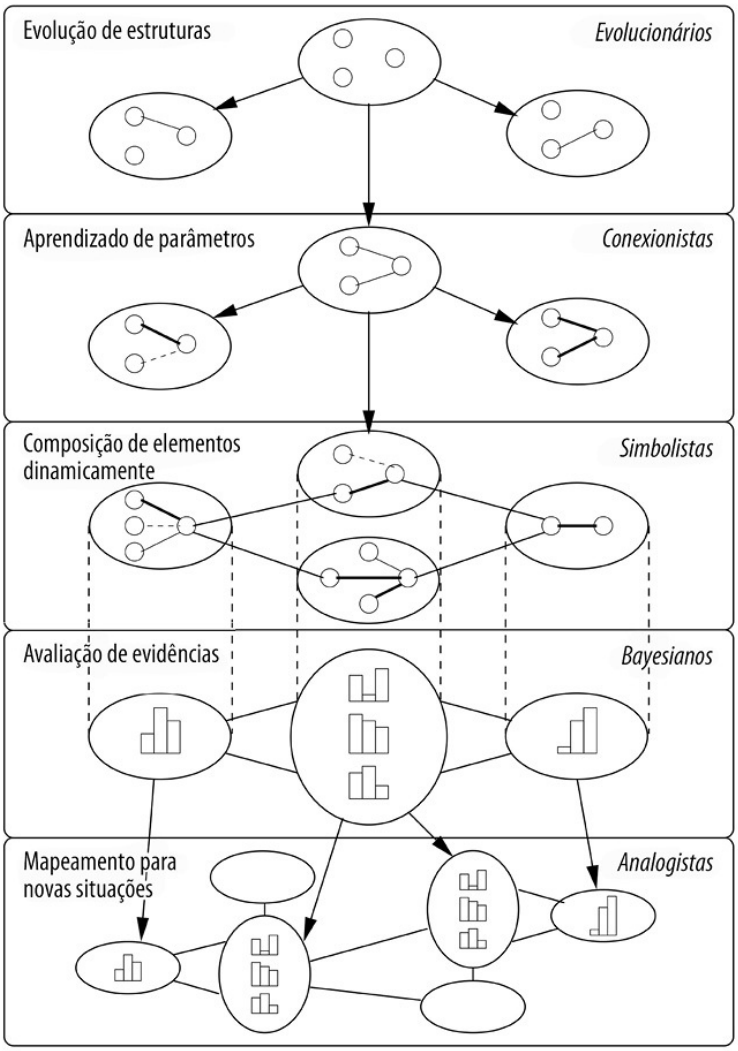
Os bayesianos se preocupam acima de tudo com a incerteza. Todo o conhecimento aprendido é incerto e o próprio aprendizado é um tipo de inferência incerta. O problema, então, passa a ser como lidar com informações com interferências, imperfeitas e até mesmo contraditórias sem se enganar. A solução é a inferência probabilística, e o algoritmo é o teorema de Bayes e seus derivados. O teorema de Bayes nos diz como incorporar novas evidências às nossas crenças, e os algoritmos de inferência probabilística fazem isso da maneira mais eficiente possível.

Para os analogistas, a chave do aprendizado é o reconhecimento de semelhanças entre situações e, a partir daí, a inferência de outras semelhanças. Se dois pacientes têm sintomas semelhantes, talvez eles tenham a mesma doença. O principal problema é julgar quanto duas coisas são semelhantes. O algoritmo mestre dos analogistas é a máquina de vetores de suporte, que descobre quais experiências devem ser lembradas e como combiná-las para fazer novas previsões.

A solução de cada tribo para seu problema central é um avanço brilhante conquistado com dificuldade. Porém, o Algoritmo Mestre verdadeiro deve resolver todos os cinco problemas e não apenas um. Por exemplo, para curar o câncer precisamos entender as redes metabólicas da célula: quais genes regulam quais outros genes, quais reações químicas as proteínas resultantes controlam e como a inclusão de uma nova molécula ao mix afetaria a rede. Seria tolo tentar aprender tudo isso do zero, ignorando o conhecimento que os biólogos acumularam esmeradamente em décadas. Os simbolistas sabem como combinar esse conhecimento a dados a partir de sequenciadores de DNA, microarranjos de

expressão gênica, e assim por diante, para produzir resultados aos quais não poderíamos chegar sozinhos. Contudo, o conhecimento obtido pela dedução inversa é puramente qualitativo; precisamos aprender não só quem interage com quem, mas também quanto, e a backpropagation pode fazê-lo. Porém, tanto a dedução inversa quanto a backpropagation ficariam suspensas no espaço sem alguma estrutura básica na qual pudéssemos fixar as interações e os parâmetros encontrados por elas, e a programação genética pode descobri-la. Nesse ponto, se tivéssemos conhecimento completo do metabolismo e todos os dados relevantes de um paciente específico, poderíamos descobrir um tratamento para ele. Todavia, na realidade as informações que temos são sempre muito incompletas, e até mesmo incorretas em alguns locais; mesmo assim precisamos avançar, e é para isso que existe a inferência probabilística. Nos casos mais difíceis, o câncer do paciente pode parecer muito diferente dos já vistos e todo o conhecimento aprendido falha. Algoritmos baseados em semelhanças podem ajudar encontrando analogias entre situações superficialmente muito diferentes, concentrando-se em suas semelhanças básicas e ignorando o resto.

Neste livro, sintetizaremos um único algoritmo com todas essas capacidades:



Nossa busca nos conduzirá pelo território de cada uma dessas cinco tribos. O cruzamento de fronteiras, em que elas se encontram, negociam e se enfrentam, será a parte mais difícil da jornada. Cada tribo tem uma peça do quebra-cabeça e devemos juntá-las. Os machine learners, como todos os cientistas, lembram os cegos e o elefante: um sente a tromba e acha que é uma cobra, o outro toca na perna e acha que é uma árvore, e o último encosta na presa e acha que é um touro. Nosso objetivo é tocar cada parte sem tirar conclusões, e quando tivermos tocado em todas elas tentaremos imaginar o elefante inteiro. É muito difícil combinar todas as peças em uma única solução – impossível, segundo alguns –, mas é o que faremos.

O algoritmo a que chegaremos ainda não será o Algoritmo Mestre, por razões que veremos, mas será o mais próximo a que alguém já chegou. E durante o percurso adquiriremos conhecimento suficiente para causar inveja a Creso. No

entanto, este livro é apenas a parte um da busca pelo Algoritmo Mestre. O protagonista da parte dois é você, caro leitor. Sua missão, caso a aceite, é fazer o resto do percurso e trazer o prêmio. Serei seu humilde guia na parte um, daqui até a fronteira do mundo conhecido. Por acaso o ouvi protestar que não sabe o suficiente ou que algoritmos não são o seu forte? Não tema. A ciência da computação ainda é jovem, e ao contrário do que ocorre em física ou biologia, você não precisa de um PhD para começar uma revolução. (É só perguntar para Bill Gates, aos senhores Sergey Brin e Larry Page ou a Mark Zuckerberg.) Insight e persistência são o que conta.

Está pronto? Nossa jornada começa com uma visita aos simbolistas, a tribo de raízes mais antigas.

capítulo 3

O problema de indução de Hume

Você é racionalista ou empírico?

Os racionalistas acreditam que os sentidos enganam e que o raciocínio lógico é o único caminho para o conhecimento. Os empíricos acreditam que o raciocínio é falível e que o conhecimento deve vir da observação e da experimentação. Os franceses são racionalistas; os anglo-saxões (como os franceses os chamam) são empíricos. Pessoas eruditas, advogados e matemáticos são racionalistas; jornalistas, médicos e cientistas são empíricos. *Assassinato por escrito* é uma série policial racionalista da TV; *CSI: Investigação criminal* é uma série empírica. Em ciência da computação, os teoristas e engenheiros do conhecimento são racionalistas; os hackers e machine learners são empíricos.

O racionalista gosta de planejar tudo antes de dar o primeiro passo. O empírico prefere fazer testes e ver como as coisas são. Não sei se há um gene para o racionalismo ou um para o empirismo, mas olhando para meus colegas cientistas da computação por vezes observei que essas tendências são quase como características da personalidade: algumas pessoas são totalmente racionalistas e não poderiam ser de outro modo; e outras são completamente empíricas e é isso que sempre serão. Os dois lados podem conversar e às vezes usam os resultados uns dos outros, mas só conseguem se entender até esse ponto. No fundo, cada um acredita que o que o outro faz é secundário e não muito interessante.

É provável que os racionalistas e os empíricos existam desde o surgimento do *Homo sapiens*. Antes de caçar, o homem primitivo Bob passava um longo tempo sentado em sua caverna pensando onde iria atacar. Enquanto isso, a mulher primitiva Alice estava fora pesquisando sistematicamente o território. Já que os dois tipos ainda nos rodeiam, é seguro dizer que nenhuma abordagem era melhor que a outra. Você pode pensar que o machine learning é o triunfo definitivo dos

empíricos, mas a verdade é mais sutil, como veremos em breve.

Racionalismo *versus* empirismo é uma das questões favoritas dos filósofos. Platão foi um dos primeiros racionalistas, e Aristóteles um dos primeiros empíricos. Porém, o debate se acalorou durante o Iluminismo, com um trio de grandes pensadores de cada lado: Descartes, Spinoza e Leibniz foram os principais racionalistas; Locke, Berkeley e Hume eram suas contrapartidas empíricas. Confiando no poder do raciocínio, os racionalistas inventaram teorias do universo que – para colocar de uma maneira sutil – não sobreviveram ao passar do tempo, mas também inventaram técnicas matemáticas básicas, como a álgebra e a geometria analítica. De modo geral, os empíricos eram mais práticos e sua influência pode ser sentida em todas as áreas, estendendo-se do método científico à Constituição dos Estados Unidos.

David Hume foi o mais importante entre os empíricos e o maior filósofo de língua inglesa de todos os tempos. Pensadores como Adam Smith e Charles Darwin o citam como uma de suas maiores influências. Também poderíamos dizer que ele é o padroeiro dos simbolistas. Hume nasceu na Escócia em 1711 e passou grande parte de sua vida na Edimburgo do século 18, uma cidade próspera com muita agitação intelectual. Um homem com inclinações à genialidade, ele foi, no entanto, um cético severo que dedicou quase todo o seu tempo a derrubar os mitos de sua época. Também levou a onda de pensamento empírico que Locke começara à sua conclusão lógica e fez uma pergunta que, desde então, pende como uma espada de Dâmocles sobre todo o conhecimento, do mais trivial ao mais avançado: como ter razão ao fazer generalizações a partir do que já vimos para chegar ao que não temos? Todo algoritmo de aprendizado é, de certa forma, uma tentativa de responder a essa pergunta.

A pergunta de Hume também é o ponto de partida de nossa jornada. Começaremos ilustrando-a com um exemplo da vida diária e encontrando sua encarnação moderna no famoso teorema “não existe almoço grátis”. Em seguida, veremos a resposta dos simbolistas a Hume. Isso nos levará ao problema mais importante do machine learning: o sobreajuste, ou padrões desvairados que na verdade não existem. Veremos como os simbolistas o resolvem e como o machine learning é basicamente um tipo de alquimia, a transmutação de dados em conhecimento com a ajuda de uma pedra filosofal. Para os simbolistas, a pedra filosofal é o próprio conhecimento. Nos próximos quatro capítulos estudaremos as soluções dos alquimistas das outras tribos.

Sair ou não sair?

Você tem uma amiga de quem gosta muito e deseja chamá-la para sair. No entanto, não sabe lidar com rejeições e só vai convidá-la se tiver certeza de que ela aceitará. É sexta à noite e lá está você com o celular na mão, tentando decidir se deve ou não chamá-la. Você se lembra de que na vez anterior ela disse não. Mas por quê? Duas vezes antes ela disse sim e na anterior a essas duas ela disse não. Talvez haja dias em que ela não goste de sair. Ou talvez ela goste de encontros dançantes e não de jantares? Tendo uma natureza muito sistemática, você larga o telefone e toma nota do que consegue se lembrar sobre as ocasiões anteriores:

Ocasião	Dia da semana	Tipo de encontro	Clima	TV à noite	Saímos?
1	Durante a semana	Jantar	Quente	Programação ruim	Não
2	Fim de semana	Dançar	Quente	Programação ruim	Sim
3	Fim de semana	Dançar	Quente	Programação ruim	Sim
4	Fim de semana	Dançar	Frio	Programação boa	Não
Hoje	Fim de semana	Dançar	Frio	Programação ruim	?

Então... o que vai ser? Sair ou não sair? Há um padrão que distinga o sim do não? E, o mais importante, o que o padrão diz sobre hoje?

É claro que não há um fator isolado que preveja corretamente a resposta: em alguns fins de semana ela gosta de sair e em outros não; às vezes ela gosta de dançar e às vezes não, e assim por diante. E quanto a uma combinação de fatores? Talvez ela goste de dançar nos fins de semana. Não, a ocasião número 4 elimina essa alternativa. Ou talvez ela só goste de sair em noites de fins de semana quentes. Bingo! Essa alternativa funciona! Caso em que, examinando o clima gelado lá fora, essa noite não parece promissora. Mas espere! E se ela gosta de dançar quando não há nada de bom na TV? Essa alternativa também funciona, e isso significa que hoje ela diria sim! Rápido, chame-a antes que fique muito tarde. Espere um pouco. Como você sabe que esse é o padrão certo? Você encontrou dois que batem com sua experiência anterior, mas fazem previsões opostas. Pensando bem, e se ela só gosta de dançar quando o tempo está bom? Ou ela sai nos fins de semana em que não há nada para ver na TV? Ou…

Nesse momento, você amassa frustrado suas notas e as joga no lixo. Não há como saber! O que você pode fazer? O fantasma de Hume concorda tristemente sobre seu ombro. *Você não tem uma base para escolher uma generalização e não outra*. Sim e não são respostas igualmente legítimas para a pergunta "O que ela dirá?". E o tempo está passando. Amarguradamente, você pega uma moeda em seu bolso e se prepara para lançá-la.

Você não é o único em apuros – nós também estamos. Acabamos de partir em nossa jornada para encontrar o Algoritmo Mestre e parece que já nos deparamos com um obstáculo insuperável. Há *alguma* maneira de aprender algo com o passado que possamos ter certeza de que é aplicável ao futuro? E se não houver, o machine learning não é uma empreitada sem esperança? A esse respeito, não está toda a ciência, até mesmo todo o conhecimento humano, fincada em terreno incerto?

Nem o big data resolveria o problema. Você poderia ser um supercasanova e ter saído com milhões de mulheres milhares de vezes cada, mas seu banco de dados mestre ainda não responderia o que *esta* mulher vai responder *desta* vez. Mesmo se hoje for exatamente como alguma ocasião anterior em que ela disse sim – mesmo dia da semana, mesmo tipo de encontro, mesma previsão meteorológica e mesmos programas na TV –, isso não significa que desta vez ela dirá sim. Pelo que parece, sua resposta é determinada por algum fator no qual você não pensou ou ao qual não tem acesso. Ou talvez não haja um equilíbrio ou uma razão para suas respostas: elas são aleatórias e você está apenas perdendo tempo tentando encontrar um padrão.

Os filósofos vêm debatendo o problema de indução de Hume desde que ele o formulou, mas ninguém deu uma resposta satisfatória. Bertrand Russell gostava de ilustrar o problema com a história do peru indutivista. Em sua primeira manhã na fazenda, o peru recebeu comida às 9:00, mas por ser bom em indução ele não tirou conclusões. Primeiro, coletou várias observações em muitos dias diferentes sob circunstâncias variadas. Tendo sido alimentado sistematicamente às 9:00 durante vários dias consecutivos, ele finalmente concluiu que sim, seria sempre alimentado às 9:00. Então, chegou a manhã da véspera de Natal e sua garganta foi cortada.

Seria bom se o problema de Hume fosse apenas uma charada filosófica interessante que pudéssemos ignorar, mas não podemos. Por exemplo, a atividade do Google se baseia em adivinhar quais páginas web estamos

procurando ao digitar algumas palavras-chave na caixa de pesquisa. Seu bem principal são logs massivos de consultas que pessoas inseriram no passado e os links em que elas clicaram nas páginas de resultados correspondentes. Mas o que fazer se alguém digitar uma combinação de palavras-chave que não exista no log? E, mesmo se existir, como ter certeza de que o usuário atual deseja as mesmas páginas dos anteriores?

Poderíamos simplesmente *presumir* que o futuro será como o passado? Essa é, com certeza, uma suposição arriscada. (Não funcionou para o peru indutivista.) Por outro lado, sem ela seria impossível obter todo o conhecimento, e assim é a vida. É melhor permanecer vivo, mesmo se for de maneira precária. Infelizmente, mesmo nos valendo dessa suposição, não estamos fora de perigo. Ela resolve os casos mais "triviais": se sou médico e o paciente B tem exatamente os mesmos sintomas do paciente A, presumo que o diagnóstico seja igual. Porém, se os sintomas do paciente B não coincidem exatamente com os de outras pessoas, fico sem pistas. Esse é o problema do machine learning: generalizar em casos que *nunca vimos*.

No entanto, talvez esse não seja um problema tão grande. Com dados suficientes, a maioria dos casos não se enquadraria na categoria "trivial"? Não. Vimos no capítulo anterior por que a memorização não funciona como aprendiz universal, contudo agora podemos torná-la mais quantitativa. Suponhamos que você tivesse um banco de dados com um trilhão de registros, cada um com mil campos booleanos (isto é, cada campo é a resposta para uma pergunta sim/não). É muita coisa. Que fração dos casos possíveis você viu? (Tente responder antes de prosseguir.) Bem, o número de respostas possíveis é duas para cada pergunta, logo para duas perguntas seria dois vezes dois (sim-sim, sim-não, não-sim e não-não), para três perguntas seria dois ao cubo ($2 \times 2 \times 2 = 2^3$) e para mil perguntas seria dois elevado a mil (2^{1000}). O trilhão de registros de seu banco de dados são um zilionésimo de 1% de 2^{1000}, em que "zilionésimo" significa "zero ponto 286 zeros seguidos por 1". Conclusão: não importa quantos dados haja – tera, peta, exa, zeta ou yotabyes –, basicamente você não viu *nada*. As chances de que o novo caso para o qual uma decisão precise ser tomada já esteja no banco de dados são tão pequenas que, sem generalização, você não tem nem como começar.

Se tudo isso parece um pouco abstrato, suponhamos que você fosse um grande provedor de email e precisasse rotular cada email recebido como spam ou não

spam. Poderia ter um banco de dados de um trilhão de emails recebidos, todos rotulados como spam ou não spam, mas isso não o ajudaria, já que as chances de cada novo email ser uma cópia exata de um anterior seriam abaixo de zero. Você não tem escolha, a não ser descobrir em um nível mais geral o que distingue o spam do não spam. E, de acordo com Hume, não há como fazê-lo.

Teorema "não existe almoço grátis"

Duzentos e cinquenta anos após Hume lançar sua bomba, David Wolpert, um físico que se tornou machine learner, deu a ela uma forma matemática elegante. Seu resultado, conhecido como teorema "não existe almoço grátis", define um limite para o nível de excelência de um aprendiz. O limite é muito baixo: nenhum aprendiz pode ser melhor do que a adivinhação aleatória! Podemos ir para casa: o Algoritmo Mestre se resume a cara ou coroa. Mas por que nenhum aprendiz pode ser melhor do que cara ou coroa? E, se for assim, por que o mundo está cheio de aprendizes altamente bem-sucedidos, que vão de filtros de spam (a surgir a qualquer momento) a carros autodirigíveis?

O teorema "não existe almoço grátis" é análogo à razão pela qual a aposta de Pascal falha. Em sua obra *Pensamentos* (*Pensées*), publicada em 1669, Pascal disse que deveríamos acreditar no Deus Cristão porque, se ele existir, isso nos dará a vida eterna, e se não existir, perdemos muito pouco. Esse foi um argumento incrivelmente sofisticado para a época, mas como Diderot salientou, um líder islâmico poderia proferir o mesmo argumento para a crença em Alá. E, se você escolher o deus errado, o preço a pagar é a danação eterna. Se colocarmos na balança, considerando a grande variedade de deuses, não estaríamos melhor acreditando em um deus do que em outro. Para cada deus que diz "faça isso", há outro que diz "faça aquilo". Também podemos simplesmente esquecer deus e aproveitar a vida sem restrições religiosas.

Substitua "deus" por "algoritmo de aprendizado" e "vida eterna" por "previsão precisa" e terá o teorema "não existe almoço grátis". Escolha seu aprendiz favorito. (Veremos muitos neste livro.) Para cada mundo em que ele se sair melhor do que a adivinhação aleatória, eu, o advogado do diabo, construirei outro no qual ele seja igualmente pior. Tudo que tenho de fazer é virar os rótulos de todas as ocorrências *não vistas*. Já que os rótulos das ocorrências observadas são equivalentes, não há como seu aprendiz distinguir o mundo do antimundo. Na média, ele será tão bom nos dois mundos quanto a adivinhação aleatória.

Portanto, na média, em todos os mundos possíveis, se emparelharmos cada mundo com seu antimundo, seu aprendiz será equivalente a jogar cara ou coroa.

No entanto, não desista ainda do machine learning ou do Algoritmo Mestre. Não estamos preocupados com todos os mundos possíveis, só com este no qual vivemos. Se soubermos algo sobre o mundo e incorporarmos isso ao nosso aprendiz, ele terá uma vantagem sobre a adivinhação aleatória. A essa questão Hume responderia que esse conhecimento deve ter vindo da indução, o que o torna falível. É verdade, mesmo se o conhecimento tiver sido impresso em nosso cérebro pela evolução, mas é um risco que temos de correr. Também poderíamos perguntar se há uma parcela de conhecimento tão incontestável, tão fundamental, que nos tornasse capazes de basear toda a indução nela. (Algo como a frase "Penso, logo existo" de Descartes, embora seja difícil descobrir como transformá-la em um algoritmo de aprendizado.) Acho que a resposta é sim, e veremos o que é essa parcela de conhecimento no Capítulo 9.

Por enquanto, a consequência prática do teorema "não existe almoço grátis" é que não há algo como aprendizado sem conhecimento. Os dados sozinhos não são suficientes. Começar do zero somente o levará ao zero. O machine learning é um tipo de bomba de conhecimento: podemos usá-lo para extrair conhecimento a partir de dados, mas primeiro temos de encher a bomba.

O machine learning é o que os matemáticos chamam de problema mal formulado: ele não tem apenas uma solução. Vejamos um pequeno problema mal formulado: quais são os dois números que, somados, resultam em 1.000? Supondo que os números sejam positivos, há quinhentas respostas possíveis: 1 e 999, 2 e 998, e assim por diante. A única maneira de resolver um problema mal formulado é introduzindo suposições adicionais. Se eu lhe disser que o segundo número é o triplo do primeiro, bingo: a resposta é 250 e 750.

Tom Mitchell, um grande simbolista, chama isso de "a futilidade do aprendizado não tendencioso". Na vida cotidiana, *tendencioso* é uma palavra pejorativa: noções preconcebidas são ruins. Porém, no machine learning, noções preconcebidas são indispensáveis; não é possível aprender sem elas. Na verdade, elas também são indispensáveis para a cognição humana, mas são programadas no cérebro e as assumimos como verdadeiras. Tendências acima e além dessas é que são questionáveis.

Aristóteles disse que nada vai para o intelecto sem antes ter passado pelos sentidos. Leibniz acrescentou: "Exceto o próprio intelecto". O cérebro humano

não é uma lousa vazia, até porque não é uma lousa. Uma lousa é passiva, algo em que escrevemos, mas o cérebro processa ativamente as informações que recebe. A memória é a lousa em que ele escreve e, inicialmente, está de fato vazia. Por outro lado, um computador é uma lousa vazia até o programarmos; o próprio processo ativo tem de ser gravado na memória antes que algo possa ocorrer. Nosso objetivo é descobrir o programa mais simples que podemos escrever, de modo que ele continue a escrever a si próprio lendo dados, sem limite, até saber tudo que há para saber.

O machine learning tem um inevitável elemento de jogo de azar. No primeiro filme da série *Dirty Harry*, Clint Eastwood persegue um ladrão de banco, atirando repetidamente nele. No fim, o ladrão está caído perto de uma arma, sem saber se deve pegá-la. Foram dados seis ou apenas cinco tiros? Harry é solidário (por assim dizer): "Você tem de saber a resposta para a pergunta 'Pareço com sorte?'. E quanto a você?". Essa é a pergunta que os machine learners tem de fazer a si mesmos todo dia ao ir para o trabalho. Sinto-me com sorte hoje? Como a evolução, o machine learning não acerta o tempo todo; na verdade, erros são a regra e não a exceção. Porém, isso não é problema porque descartamos os erros e ficamos com os acertos, e o resultado final é o que importa. Quando adquirimos algum conhecimento novo, ele se torna a base para a indução de ainda mais conhecimento. A única pergunta é por onde começar.

Enchendo a bomba do conhecimento

Na obra *Principia*, junto com suas três leis do movimento, Newton enuncia quatro regras da indução. Embora sejam bem menos conhecidas do que as leis da física, elas são igualmente importantes. A regra-chave é a terceira, que poderíamos parafrasear assim:

> *Princípio de Newton: O que quer que seja verdadeiro em tudo que já vimos é verdadeiro em tudo no universo.*

Não é exagero dizer que essa declaração aparentemente inócua é a essência da revolução newtoniana e da ciência moderna. As leis de Kepler são aplicáveis a exatamente seis entidades: os planetas do sistema solar conhecidos em sua época. As leis de Newton são aplicáveis a cada partícula de matéria do universo. O salto em generalidade entre as duas é surpreendente e é consequência direta do princípio de Newton. Somente esse princípio já é uma bomba de conhecimento

de poder fenomenal. Sem ele não haveria leis da natureza, apenas uma colcha de retalhos incompleta composta por pequenas regularidades.

O princípio de Newton é a primeira regra não escrita do machine learning. Induzimos as regras mais amplamente aplicáveis e só reduzimos seu escopo quando os dados nos forçam. À primeira vista pode parecer ridiculamente otimista, mas vem funcionando para a ciência há mais de trezentos anos. É claro que poderíamos imaginar um universo tão variado e caprichoso a ponto de o princípio de Newton falhar sistematicamente, mas esse não é nosso universo.

No entanto, o princípio de Newton é apenas a primeira etapa. Ainda precisamos descobrir o que é verdadeiro em tudo que já vimos – extrair as regularidades a partir dos dados brutos. A solução-padrão é presumir que conhecemos a *forma* da verdade, e a tarefa do aprendiz é revelá-la. Por exemplo, na questão do encontro você poderia presumir que a resposta de sua amiga seria determinada por um único fator, caso em que o aprendizado consistiria apenas em verificar cada fator conhecido (dia da semana, tipo de encontro, clima e programação na TV) para saber se ele prevê corretamente a resposta em cada situação. É claro que o problema é que nenhum dos fatores resolve! Você jogou e falhou. Então, flexibilize um pouco as suposições. E se a resposta de sua amiga for determinada por uma conjunção de dois fatores? Com quatro fatores, cada um com dois valores possíveis, há vinte e quatro possibilidades a serem verificadas (seis pares de fatores a serem selecionados vezes duas opções para o valor de cada fator). Agora temos um número exagerado de opções: quatro conjunções de dois fatores preveem corretamente o resultado! O que fazer? Se estiver se sentindo com sorte, você pode simplesmente selecionar uma delas e esperar que aconteça o melhor. Porém, a alternativa mais sensata é a democracia: faça uma votação e escolha a previsão vencedora.

Se todas as conjunções de dois fatores falharem, você pode testar todas as conjunções de qualquer número de fatores. Os machine learners e os psicólogos chamam essas conjunções de “conceitos conjuntivos”. As definições encontradas em dicionários são conceitos conjuntivos: uma cadeira tem um assento *e* um espaldar *e* algum número de pernas. Remova qualquer um desses elementos e ela não será mais uma cadeira. Um conceito conjuntivo é o que Tolstoi tinha em mente quando escreveu a frase de abertura de *Anna Kariênina*: “Todas as famílias felizes se parecem, cada família infeliz é infeliz à sua maneira”. O mesmo ocorre com as pessoas. Para ser feliz, você precisa de saúde, amor,

amigos, dinheiro, um trabalho do qual goste, e assim por diante. Remova algum desses elementos e será infeliz.

No machine learning, os exemplos de um conceito são chamados de exemplos positivos e os contraexemplos são exemplos negativos. Se você estiver tentando aprender a reconhecer gatos em imagens, imagens de gatos serão exemplos positivos e imagens de cães serão exemplos negativos. Se criasse um banco de dados de famílias tiradas da literatura mundial, os Kariênin seriam um exemplo negativo de família feliz e haveria poucos exemplos positivos.

É típico do machine learning começar com suposições restritivas e abrandá-las de maneira gradual se não explicarem os dados, e o processo costuma ser executado automaticamente pelo aprendiz, sem qualquer ajuda nossa. Primeiro, ele testa todos os fatores isoladamente, depois todas as conjunções de dois fatores, de três fatores, e assim por diante. Porém, temos um problema: há *muitos* conceitos conjuntivos e tempo insuficiente para testar todos.

O caso do encontro é ilusório porque é muito pequeno (quatro variáveis e quatro exemplos). Mas suponhamos que agora você gerencie um serviço de encontros online e precise descobrir quais casais combinam. Se cada usuário de seu sistema preencher um questionário com respostas a cinquenta perguntas sim/não, cada correspondência possível seria caracterizada por uma centena de atributos, cinquenta de cada membro do futuro casal. Com base nos casais que marcaram um encontro e relataram o resultado, você pode encontrar uma definição conjuntiva para o conceito de uma "boa união"? Há 3^{100} definições possíveis para serem testadas. (As três opções de cada atributo são sim, não e não faz parte do conceito.) Mesmo com o computador mais rápido do mundo, os casais não existirão mais – e sua empresa terá falido – quando você terminar, a menos que tenha sorte e uma definição muito curta atenda. Tantas regras e tão pouco tempo. Precisamos fazer algo mais inteligente.

Aqui está uma maneira. Não seja pessimista e comece a supor que todas as uniões são boas. Em seguida, tente excluir as uniões que não tenham algum atributo. Repita isso para cada atributo e selecione um que exclua as piores uniões e o menor número de boas uniões. Agora sua definição será semelhante a algo como, digamos, "É um bom casal somente quando ele cede". Tente adicionar cada um dos outros atributos e selecione um que exclua as uniões ruins que duraram mais e as boas que duraram menos. Talvez a definição passe a ser "É um bom casal somente quando ele cede e ela também". Tente adicionar um

terceiro atributo a esses dois, e assim por diante. Quando tiver excluído todas as uniões ruins, terá terminado: você tem uma definição do conceito que inclui todos os exemplos positivos e exclui todos os negativos. Vejamos: “Um casal é uma boa combinação somente quando os dois cedem, ele gosta de cães e ela não gosta de gatos”. Jogue os dados fora e mantenha somente essa definição, já que ela encapsula tudo que é relevante para você. Com certeza esse algoritmo terminará seu processamento em um período de tempo aceitável, e também é o primeiro aprendiz real que encontramos neste livro!

Como conquistar o mundo

No entanto, os conceitos conjuntivos não nos levam muito longe. O problema é que, como disse Rudyard Kipling, “Há várias maneiras de construir canções tribais, e todas elas estão corretas”. Os conceitos reais são disjuntivos. Cadeiras podem ter quatro pernas ou uma, e às vezes nenhuma. Você pode ganhar no xadrez de inúmeras maneiras diferentes. Emails contendo a palavra *Viagra* são provavelmente spam, mas emails contendo “grátis!!!” também são. Além disso, todas as regras têm exceções. Algumas famílias são disfuncionais e, mesmo assim, felizes. Os pássaros voam, a menos que sejam pinguins, avestruzes, casuares ou quiuís (ou tenham quebrado uma asa, estejam em gaiolas etc.).

O que precisamos é aprender conceitos que sejam definidos por um conjunto de regras, não por uma única regra, como:

Se você gostou de Guerra nas estrelas, episódios IV-VI, gostará de Avatar.
Se você gostou de Jornada nas estrelas: a nova geração e Titanic, gostará de Avatar.
Se você é membro do Sierra Club e lê livros de ficção científica, gostará de Avatar.
Ou:
Se seu cartão de crédito foi usado na China, Canadá e Nigéria ontem, ele foi roubado.
Se seu cartão de crédito foi usado duas vezes após as 23:00 em um dia útil, ele foi roubado.
Se seu cartão de crédito foi usado para comprar um dólar de gasolina, ele foi roubado.

(Para que você entenda a última regra, ladrões de cartão crédito costumavam comprar rotineiramente um dólar de gasolina para verificar se um cartão roubado era válido antes de mineradores de dados descobrirem a tática.)[1]

Podemos aprender conjuntos de regras como esse em intervalos de uma regra de cada vez usando o algoritmo que vimos para o aprendizado de conceitos conjuntivos. Após aprender cada regra, descartamos os exemplos positivos que ela considera, para que a próxima regra leve em consideração o máximo possível

de exemplos positivos restantes, e continuamos fazendo isso até que todos tenham sido considerados. É um exemplo de “dividir para conquistar”, a mais antiga estratégia do manual da conquista dos cientistas. Também podemos melhorar o algoritmo que encontra uma única regra mantendo um número *n* de hipóteses, não apenas uma, e estendendo a cada etapa todas elas de todas as maneiras possíveis e mantendo os *n* melhores resultados.

Descobrir regras dessa forma foi invenção de Ryszard Michalski, cientista da computação polonês. Kalusz, a cidade natal de Michalski, foi sucessivamente parte da Polônia, Rússia, Alemanha e Ucrânia, o que pode tê-lo deixado mais atento aos conceitos disjuntivos do que a maioria das pessoas. Após imigrar para os Estados Unidos em 1970, ele acabou encontrando a escola simbolista de machine learning, junto com Tom Mitchell e Jaime Carbonell. Michalski tinha personalidade arrogante. Se você desse uma palestra em uma conferência de machine learning, haveria grandes chances de no fim ele levantar a mão para dizer que essa era apenas uma releitura de uma de suas antigas ideias.

Conjuntos de regras são populares para varejistas quando decidem quais mercadorias armazenarão. Normalmente, eles usam uma abordagem mais exaustiva do que “dividir e conquistar”, procurando todas as regras que prevejam consistentemente a compra de cada item. O Walmart foi pioneiro nessa área. Uma de suas primeiras descobertas foi que quando alguém compra fraldas também costuma comprar cervejas. Como assim? Uma interpretação seria que a mãe manda o pai ao supermercado para comprar fraldas e, como compensação emocional, ele compra uma caixa de cervejas. Sabendo disso, agora o supermercado consegue vender mais cervejas colocando-as perto das fraldas, o que nunca teria ocorrido a eles sem a mineração de regras. A regra “cervejas e fraldas” ganhou status de lenda entre os mineradores de dados (embora alguns aleguem que seja apenas uma lenda urbana). De qualquer forma, muito tempo se passou desde os problemas de design de circuitos digitais que Michalski tinha em mente quando começou a pensar na indução de regras na década de 60. Quando inventamos um novo algoritmo de aprendizado, não temos noção das coisas para as quais ele será usado.

Minha primeira experiência direta com o aprendizado de regras foi quando, tendo acabado de mudar para os Estados Unidos para começar a pós-graduação, fiz um pedido de cartão de crédito. O banco me enviou uma carta dizendo “Sentimos muito, mas seu pedido foi rejeitado devido a TEMPO

INSUFICIENTE NO ENDEREÇO ATUAL e AUSÊNCIA DE HISTÓRICO DE COMPRAS A CRÉDITO” (ou outras palavras semelhantes em maiúsculas). Soube imediatamente que ainda havia muito pesquisa a ser feita em machine learning.

Entre a cegueira e o delírio

Conjuntos de regras são muito mais poderosos do que conceitos conjuntivos. Na verdade, são tão poderosos que podemos representar *qualquer* conceito usando-os. Não é difícil entender o porquê. Se você me der uma lista completa de todas as instâncias de um conceito, posso transformar cada instância em uma regra que especifique todos os atributos dessa instância, e o conjunto de todas as regras será a definição do conceito. Voltando ao exemplo do encontro, uma regra seria: *Se for uma noite quente de fim de semana, não houver nada de bom na TV e você propuser uma ida à boate, ela dirá sim*. A tabela só contém alguns exemplos, mas se contivesse todos os $2 \times 2 \times 2 \times 2 = 16$ exemplos possíveis, com cada um sendo rotulado como “Convite aceito” ou Convite não aceito”, seria complicado transformar cada exemplo positivo em uma regra.

O poder dos conjuntos de regras é uma faca de dois gumes. Por um lado, você sabe que sempre é possível encontrar um conjunto de regras que case perfeitamente os dados. Porém, antes de começar a achar que tirou a sorte grande, lembre-se de que são grandes as chances de encontrar um conjunto de regras totalmente irrelevante. Não se esqueça do teorema “não existe almoço grátis”: não é possível aprender sem conhecimento. E presumir que o conceito pode ser definido por um conjunto de regras é o mesmo que não presumir nada.

Um conjunto de regras inútil seria aquele que só abordasse os exemplos positivos que você viu e nada mais. Esse conjunto de regras vai parecer 100% exato, mas é uma ilusão: ele preverá que qualquer exemplo novo é negativo e, portanto, considerará erroneamente os positivos. Se no geral houver mais exemplos positivos do que negativos, isso será ainda pior do que cara ou coroa. Imagine um filtro de spam que só decidisse que um email é spam se for uma cópia exata de uma mensagem anterior rotulada como tal. Ele aprenderá facilmente e parecerá funcionar bem para os dados rotulados, mas você pode acabar não tendo um filtro de spam. Infelizmente, nosso algoritmo “dividir para conquistar” poderia aprender um conjunto de regras como esse.

Em sua história “Funes, o Memorioso”, Jorge Luis Borges fala sobre um

jovem com uma memória perfeita. À primeira vista isso pode parecer um dom, mas na verdade é uma terrível maldição. Funes pode lembrar a forma exata das nuvens no céu em um momento arbitrário no passado, porém tem dificuldades para entender que um cão visto de perfil às 15:14 é o mesmo visto de frente às 15:15. Sua própria face no espelho o surpreende sempre que ele a vê. Funes não pode generalizar; para ele, duas coisas só são iguais se se parecem até o último detalhe. Um aprendiz de regras ilimitado seria como Funes e também não funcionaria. Aprender é esquecer os detalhes e, ao mesmo tempo, lembrar as partes importantes. Os computadores são esse tipo de sábio idiota: podem se lembrar de tudo, mas não é isso que queremos que façam.

O problema não está restrito à memorização indiscriminada de instâncias. Sempre que um aprendiz encontra um padrão nos dados que não é verdadeiro no mundo real, dizemos que ele sobreajustou os dados. O sobreajuste é o problema central do machine learning. Foram escritos mais artigos sobre ele do que qualquer outro tópico. Um aprendiz poderoso, seja simbolista, conexionista ou de qualquer outro tipo, tem de se preocupar com padrões delirantes. A única maneira segura de evitá-los é restringir severamente o que o aprendiz pode guardar, por exemplo, exigindo que o padrão seja um conceito conjuntivo curto. Infelizmente, isso é um exagero e não permite que o aprendiz veja a maioria dos padrões verdadeiros detectáveis nos dados. Logo, um bom aprendiz está sempre na linha tênue entre a cegueira e o delírio.

Os humanos não são imunes ao sobreajuste. Poderíamos até dizer que ele é a causa-raiz de muitos de nossos males. Considere uma menininha branca que, ao ver um bebê latino no shopping, deixasse escapar “Olhe, mãe, uma empregada bebê!” (evento real). Isso não significa que ela seja uma intolerante inata. Em vez disso, exagerou na generalização a partir das poucas empregadas latinas que viu em sua curta vida. O mundo está cheio de latinas que têm outras ocupações, mas ela ainda não as viu. Nossas crenças são baseadas na experiência, o que proporciona um retrato muito incompleto do mundo, e é fácil tirar conclusões falsas. Ser inteligente e informado também não o imuniza contra o sobreajuste. Aristóteles cometeu um sobreajuste quando disse que é preciso força para manter um objeto em movimento. Galileu mostrou sua genialidade ao intuir que objetos intocados se mantêm em movimento sem precisar visitar o espaço sideral para testemunhar o fato.

No entanto, os algoritmos de aprendizado são particularmente propensos ao

sobreajuste, porque têm capacidade quase ilimitada para encontrar padrões em dados. No tempo necessário para um humano encontrar um único padrão, um computador pode encontrar milhões. No machine learning, a maior vantagem do computador – a habilidade de processar grandes quantidades de dados e repetir interminavelmente as mesmas etapas sem cansar – também é seu calcanhar de Aquiles. E é surpreendente o que podemos encontrar se buscarmos o suficiente. *O código da Bíblia*, um best-seller de 1998, alega que a Bíblia contém previsões de eventos futuros que podemos descobrir pulando letras em intervalos regulares e montando palavras a partir das letras em que cairmos. Infelizmente, há tantas maneiras de fazer isso que podemos ter certeza de encontrar "previsões" em qualquer texto suficientemente longo. Os céticos responderam encontrando-as em *Moby Dick* e nas decisões judiciais da Suprema Corte, além das menções a Roswell e óvnis no Gênesis. John von Neumann, um dos pioneiros da ciência da computação, disse em uma frase famosa que "com quatro parâmetros posso gerar um elefante e com cinco posso fazê-lo balançar a tromba". Hoje aprendemos rotineiramente modelos com milhões de parâmetros, o bastante para dar a cada elefante no mundo seu próprio balanço de tromba. Alguém disse até que *mineração de dados* significa "torturar os dados até que confessem".

Algo que exacerba seriamente o sobreajuste é o ruído. Ruído em machine learning significa simplesmente erros nos dados ou eventos aleatórios que não podem ser previstos. Suponhamos que sua amiga goste de ir dançar quando não há nada interessante na TV, mas você se confundiu na ocasião número 3 e anotou que *havia* algo de bom na TV naquela noite. Se tentar criar um conjunto de regras que faça uma exceção para essa noite, provavelmente acabará obtendo uma resposta pior do que se a ignorasse. Ou suponhamos que ela estivesse de ressaca por ter saído na noite anterior e disse não quando normalmente teria dito sim. A menos que você saiba sobre a ressaca, aprender um conjunto de regras que considere esse exemplo será contraproducente: seria melhor não classificá-lo como um não. E fica pior: o ruído pode tornar impossível criar um conjunto de regras consistente. Observe que as ocasiões 2 e 3 são, na verdade, indistinguíveis: elas têm os mesmos atributos. Se sua amiga dissesse sim na ocasião 2 e não na ocasião 3, nenhuma regra conseguiria defini-las corretamente.

O sobreajuste ocorre quando temos hipóteses demais sem dados suficientes para diferenciá-las. A má notícia é que, até mesmo para o simples aprendiz conjuntivo, o número de hipóteses aumenta exponencialmente com o número de

atributos. O crescimento exponencial é algo assustador. Uma bactéria *E. coli* pode se dividir em duas aproximadamente a cada quinze minutos; com nutrientes suficientes ela pode se transformar em uma massa de bactérias do tamanho da Terra em cerca de um dia. Quando o número de coisas que um algoritmo precisa fazer cresce exponencialmente com o tamanho de sua entrada, os cientistas da computação chamam isso de explosão combinatória. No machine learning, o número de instâncias possíveis de um conceito é uma função exponencial do número de atributos: quando os atributos são booleanos, cada novo atributo dobra o número de instâncias possíveis pegando cada instância anterior e estendendo-a com um sim ou não para esse atributo. Por sua vez, o número de conceitos possíveis é uma função exponencial do número de instâncias possíveis: já que um conceito rotula cada instância como positiva ou negativa, o acréscimo de uma instância dobra o número de conceitos possíveis. Como resultado, o número de conceitos é uma função exponencial de uma função exponencial do número de atributos! Em outras palavras, o machine learning é uma explosão combinatória de explosões combinatórias. Talvez devêssemos simplesmente desistir e não perder nosso tempo com um problema tão impossível.

Felizmente, algo ocorre no aprendizado que elimina um dos exponenciais, deixando um único problema exponencial intratável e "comum". Suponhamos que você tivesse um saquinho cheio de definições de conceitos, cada uma escrita em um pedaço de papel, retirasse um aleatoriamente e visse com que exatidão ele corresponde aos dados. Uma definição ruim conseguirá corresponder a, digamos, todos os mil exemplos de seus dados tanto quanto uma moeda consegue mostrar cara mil vezes seguidas. "Uma cadeira tem quatro pernas e é vermelha ou tem um assento, mas não tem pernas" provavelmente definirá algumas cadeiras, mas não todas as que você viu, e também definirá algumas outras coisas, mas não todas. Logo, se uma definição aleatória corresponder corretamente a mil exemplos, é muito improvável que seja a definição errada, ou pelo menos estará bem próxima da real. E se a definição corresponder a um milhão de exemplos, poderemos praticamente ter certeza de que ela está certa. De que outra forma ela definiria todos esses exemplos corretamente?

É claro que um algoritmo de aprendizado real não tentaria tirar apenas uma definição aleatoriamente do saquinho; tiraria várias, e elas não seriam escolhidas de modo aleatório. Quanto mais definições ele testar, maior a probabilidade de

que uma acidentalmente corresponda a todos os exemplos. Se você executar um milhão de vezes mil jogadas de cara ou coroa, é praticamente garantido que pelo menos uma execução só mostre cara, e um milhão é um número razoavelmente baixo de hipóteses a serem consideradas. Por exemplo, esse é quase o número de conceitos conjuntivos possíveis se os exemplos tiverem só treze atributos. (É bom ressaltar que você não precisa testar explicitamente os conceitos um a um; se o melhor conceito que encontrar usando o aprendiz conjuntivo apresentar correspondência com todos os exemplos, o efeito será o mesmo.)

Conclusão: o aprendizado é uma corrida entre a quantidade de dados que você tem e o número de hipóteses considerado. Mais dados reduzem exponencialmente o número de hipóteses que sobrevivem, mas, se no início você tiver muitas delas, talvez algumas hipóteses inúteis permaneçam no final. Como regra prática, se o aprendiz só considerar um número exponencial de hipóteses (por exemplo, todos os conceitos conjuntivos possíveis), o equivalente exponencial dos dados compensará esse número e tudo ficará bem, contanto que você tenha muitos exemplos e não tantos atributos. Por outro lado, se ele considerar um número duplamente exponencial (por exemplo, todos os conjuntos de regras possíveis), os dados compensarão somente um dos exponenciais e haverá problemas. É possível até mesmo descobrir antecipadamente de quantos exemplos você precisará para ter certeza absoluta de que a hipótese selecionada pelo aprendiz está próxima da verdadeira, contanto que ela corresponda a todos os dados; em outras palavras, para que a hipótese esteja provavelmente quase correta. O cientista Leslie Valiant da Universidade Harvard recebeu o Turing Award, o Nobel da ciência da computação, por inventar esse tipo de análise, que ele descreve em seu livro adequadamente intitulado *Probably Approximately Correct*.

A precisão em que podemos confiar

Na prática, uma análise no estilo de Valiant tende a ser muito pessimista e demanda mais dados do que costumamos ter. Então, como saber se podemos acreditar no que um aprendiz nos informa? Simples: não acredite em nada até ter verificado em dados *que o aprendiz não viu*. Se os padrões que o aprendiz usou como hipótese também funcionarem em novos dados, você poderá ficar bastante seguro de que eles são reais. Caso contrário, saberá que houve sobreajuste. Esse é apenas o método científico aplicado ao machine learning: não é suficiente uma

nova teoria explicar evidências passadas porque é fácil inventar uma teoria que faça isso; a teoria também deve fazer novas previsões, e você só deve aceitá-la depois que as previsões forem verificadas experimentalmente. (E mesmo assim, apenas de maneira provisória, porque evidências futuras ainda podem demonstrar que ela é falsa.)

A relatividade geral de Einstein só foi amplamente aceita quando Arthur Eddington confirmou empiricamente sua previsão de que o Sol curva a luz de estrelas distantes. Porém, você não precisa esperar que novos dados cheguem se confiar em seu aprendiz. Em vez disso, pegue os dados que possui e divida-os aleatoriamente em um conjunto de treinamento, que você passará para o aprendiz, e um conjunto de teste, que ocultará dele e usará para verificar sua precisão. A precisão em dados retidos para avaliação é o padrão-ouro do machine learning. Você pode escrever um artigo sobre um excelente algoritmo de aprendizado novo que inventou, mas, se este não for significativamente mais preciso do que os anteriores em dados retidos, o artigo não será publicável.

A precisão em dados ainda não vistos é um teste rigoroso; tanto é que, na verdade, muitos experimentos científicos não passam nele. Isso não o torna inútil, já que a ciência não se baseia apenas em previsões; ela também se baseia em explicação e compreensão. Porém, se no fim das contas seus modelos não fizerem previsões precisas com dados novos, você não terá certeza se entendeu ou conseguiu explicar o fenômeno subjacente. E para o machine learning, testar com dados não vistos é indispensável porque é a única maneira de saber se o aprendiz cometeu ou não um sobreajuste.

A precisão do conjunto de teste também não é à prova de erros. Conforme a lenda, em um aplicativo militar antigo um aprendiz simples detectou tanques com 100% de precisão tanto no conjunto de treinamento quanto no conjunto de teste, cada um contendo cem imagens. Surpreendente – ou suspeito? Na verdade, as imagens de tanques eram mais claras do que as demais e eram a única coisa que o aprendiz estava detectando. Atualmente, temos conjuntos de teste maiores, mas a qualidade do conjunto de dados não é necessariamente melhor, logo é preciso ter cuidado. A avaliação empírica minuciosa desempenhou um papel importante no crescimento do machine learning, transformando-o de uma área nova em uma madura. Até o fim da década de 80, os pesquisadores de cada tribo acreditavam principalmente em sua própria retórica, presumiam que seu paradigma era basicamente melhor e se comunicavam pouco com outros grupos.

Então, simbolistas como Ray Mooney e Jude Shavlik começaram a comparar sistematicamente os diferentes algoritmos usando os mesmos conjuntos de dados, e – adivinhem – nenhum era claramente melhor do que o outro. Hoje a rivalidade continua, porém há muito mais intercâmbio. A existência de uma estrutura experimental comum e de um grande repositório de conjuntos de dados mantido pelo grupo de machine learning da Universidade da Califórnia, em Irvine, fez maravilhas para impulsionar o avanço. E como veremos, nossa maior esperança de criar um aprendiz universal está na síntese de ideias de diferentes paradigmas.

É claro que não é suficiente saber que está havendo sobreajuste; temos de poder evitá-lo. Isso significa parar de enquadrar os dados de maneira perfeita mesmo se pudermos. Um método é usar testes de significância estatística para verificar se os padrões que estamos vendo realmente existem. Por exemplo, uma regra que abordasse trezentos exemplos positivos *versus* cem negativos e uma que abordasse três positivos *versus* um negativo seriam 75% exatas nos dados de treinamento, mas a primeira regra é quase certamente melhor do que cara ou coroa, enquanto a segunda não é, já que quatro lançamentos de uma moeda não viciada poderiam facilmente resultar três vezes em cara. Ao construir uma regra, se em algum momento não conseguirmos encontrar condições que melhorem de maneira significativa sua precisão, então devemos parar, mesmo se ela ainda abordar alguns exemplos negativos. Isso reduzirá a precisão do conjunto de treinamento da regra, mas provavelmente o tornará uma generalização mais exata, que é o que queremos.

No entanto, ainda não estamos a salvo. Se eu testar uma única regra e ela for 75% precisa em quatrocentos exemplos, é provável que seja confiável. Porém, se eu testar um milhão de regras e a melhor tiver 75% de precisão em quatrocentos exemplos, ela não será confiável, porque o resultado pode ter ocorrido acidentalmente. Esse é o problema que encontramos ao escolher um fundo mútuo. Os fundos da Clairvoyant Services foram os melhores no mercado durante dez anos seguidos. Bem, o gerente deve ser um gênio. Ou não? Se tivermos mil fundos para escolher, haverá grandes chances de que um seja líder no mercado por dez anos seguidos, mesmo se estiverem sendo gerenciados por macacos jogadores de dardos. A literatura científica também é afetada por esse problema. Os testes de significância são o padrão-ouro que decide se um resultado de pesquisa é publicável, mas se várias equipes estiverem procurando

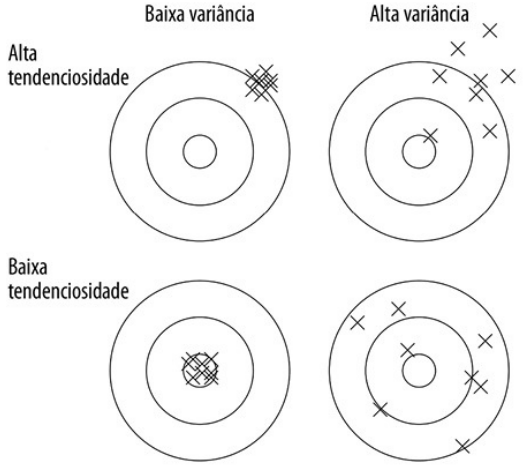
um resultado e só uma o encontrar, há chances de que ela não o tenha encontrado, ainda que não seja possível saber disso lendo seu artigo aparentemente consistente. Uma solução seria publicar também resultados negativos, assim saberíamos sobre as tentativas que falharam, mas que não foram usadas. No machine learning, podemos rastrear quantas regras testamos e ajustar nossos testes de significância de acordo, mas os testes tenderiam a descartar regras boas junto com as ineficientes. Um método melhor é aceitar que algumas hipóteses falsas inevitavelmente passarão, mantendo sua quantidade sob controle ao rejeitar as de significância suficientemente baixa e testando as hipóteses restantes com dados adicionais.

Outro método popular é preferir hipóteses simples. O algoritmo “dividir para conquistar” opta implicitamente por usar regras mais simples porque para de adicionar condições a uma regra assim que aborda apenas exemplos positivos e para de adicionar regras assim que todos os exemplos positivos são considerados. Porém, para combater o sobreajuste, precisamos adotar uma preferência mais forte por regras simples, que nos faça parar de adicionar condições antes mesmo de todos os exemplos negativos serem considerados. Por exemplo, podemos extrair uma penalidade proporcional ao tamanho da regra a partir de sua precisão e usá-la como medida de avaliação.

A preferência por hipóteses mais simples é popularmente conhecida como navalha de Occam, mas no contexto do machine learning isso leva a uma interpretação errada. “As entidades não devem ser aumentadas além do necessário”, como a navalha é frequentemente parafraseada, significa apenas escolher a teoria mais simples que englobe os dados. Talvez Occam ficasse perplexo com a noção de que devemos preferir uma teoria que *não* considere a evidência de maneira exata, já que isso resultará em uma generalização melhor. Teorias simples são preferíveis porque incorrem em um custo cognitivo menor (para nós) e em um custo computacional também menor (para nossos algoritmos) e não porque necessariamente esperemos que sejam mais precisas. Pelo contrário, até mesmo nossos modelos mais elaborados em geral são simplificações exageradas da realidade. Mesmo entre teorias que enquadram perfeitamente os dados, sabemos pelo teorema “não existe almoço grátis” que não há garantia de que a mais simples resultará em generalizações melhores; na verdade, alguns dos melhores algoritmos – como os de boosting e os de máquinas de vetores de suporte – aprendem o que parecem ser modelos

desnecessariamente complexos. (Veremos por que eles funcionam nos Capítulos 7 e 9.)

Se a precisão do conjunto de teste de seu aprendiz o desapontar, você precisa diagnosticar o problema. Foi cegueira ou delírio? No machine learning, os termos técnicos usados para esses extremos são *tendenciosidade* e *variância*. Um relógio que está sempre uma hora atrasado tem tendenciosidade alta, mas pouca variância. Se em vez disso ele se alternar de maneira errática entre rápido e lento, mas em média informar a hora certa, terá alta variância e baixa tendenciosidade. Suponhamos que você estivesse em um bar com alguns amigos, bebendo e jogando dardos. Eles não sabem, mas você vem praticando há anos e domina o jogo. Todos os seus dardos acertam na mosca. Você tem baixa tendenciosidade e baixa variância, o que é mostrado no canto inferior esquerdo deste diagrama:



Seu amigo Ben também é muito bom, mas exagerou um pouco na bebida. Seus dardos estão espalhados, mas ele continua dizendo em voz alta que em média tem acertado na mosca. (Talvez devesse ter sido um estatístico.) Esse é o caso de baixa tendenciosidade e alta variância, mostrado no canto inferior direito. A namorada de Ben, Ashley, é bastante regular, mas tem tendência a mirar mais alto e à direita. Ela tem baixa variância e alta tendenciosidade (canto superior esquerdo). Os dardos de Cody, que é de fora da cidade e nunca tinha jogado, estão espalhados e longe do centro. Ele tem tanto alta tendenciosidade quanto alta variância (canto superior direito).

Você pode estimar a tendenciosidade e a variância de um aprendiz comparando suas previsões após aprender variações aleatórias do conjunto de treinamento. Se ele estiver repetindo os mesmos erros, o problema é a tendenciosidade, e você

precisa de um aprendiz mais flexível (ou apenas diferente). Se não houver padrão nos erros, o problema é a variância, e você deve testar um aprendiz menos flexível ou obter mais dados. A maioria dos aprendizes tem um parâmetro que podemos alterar para torná-los mais ou menos flexíveis, como o limite dos testes de significância ou a penalidade referente ao tamanho do modelo. Ajustar esse parâmetro é seu primeiro recurso.

A indução é o inverso da dedução

O problema maior, no entanto, é que no início quase todos os aprendizes sabem muito pouco, e não há ajuste de parâmetros que os levem à linha de chegada. Sem a orientação do conhecimento fornecido pelo cérebro de um adulto, eles podem facilmente se desviar do objetivo. Ainda que seja o que a maioria dos aprendizes faz, simplesmente presumir que conhecemos a forma da verdade (por exemplo, que ela é um pequeno conjunto de regras) não é algo com que possamos contar. Um empírico rigoroso diria que isso é tudo que um recém-nascido tem, codificado na arquitetura de seu cérebro, e realmente as crianças cometem mais sobreajustes do que os adultos, mas gostaríamos de aprender mais rápido do que uma criança. (Dezoito anos é muito tempo, sem contar a faculdade.) O Algoritmo Mestre precisa ter desde o início um grande corpo de conhecimento, seja ele fornecido por humanos ou aprendido em execuções anteriores, e usá-lo para orientar novas generalizações a partir de dados. É isso que os cientistas fazem, e é o oposto de uma lousa vazia. O algoritmo de indução da regra “dividir para conquistar” não pode fazê-lo, mas há outra maneira de aprender regras que o façam.

A chave é saber que a indução é apenas o inverso da dedução, da mesma forma que a subtração é o inverso da adição ou a integração é o inverso da diferenciação. Essa ideia foi proposta pela primeira vez por William Stanley Jevons no fim dos anos 1800. Steve Muggleton e Wray Buntine, uma equipe anglo-australiana, projetou o primeiro algoritmo prático baseado nela em 1988. A estratégia de pegar uma operação conhecida e obter seu oposto tem um histórico famoso na matemática. Sua aplicação à adição levou à invenção dos inteiros, porque sem números negativos nem sempre a adição tem um elemento inverso ($3 – 4 = –1$). Da mesma forma, sua aplicação à multiplicação levou aos números racionais. Quando aplicada à elevação ao quadrado, temos os números complexos. Podemos aplicá-la à dedução? Um exemplo clássico de raciocínio

dedutivo é:

Sócrates é humano.
Todos os humanos são mortais.
Logo...?

A primeira declaração é um fato sobre Sócrates e a segunda é uma regra geral sobre os humanos. O que vem a seguir? Que Sócrates é mortal, claro, pela aplicação da regra a Sócrates. No raciocínio indutivo temos o fato inicial e o derivado, e procuramos uma regra que nos permita inferir o último a partir do primeiro:

Sócrates é humano.
...?
Logo, Sócrates é mortal.

Uma regra para esse raciocínio seria: *Se Sócrates é humano, então ele é mortal*. Essa resposta serve, mas não é muito útil porque se refere especificamente a Sócrates. Mas podemos aplicar o princípio de Newton e generalizar a regra para todas as entidades: *Se uma entidade é humana, então ela é mortal*. Ou, mais sucintamente: *Todos os humanos são mortais*. Obviamente, seria precipitado induzir essa regra apenas a partir de Sócrates, mas sabemos fatos semelhantes sobre outros humanos:

Platão é humano. Platão é mortal.
Aristóteles é humano. Aristóteles é mortal.
E assim por diante.

Para cada par de fatos, construímos a regra que nos permite inferir o segundo fato a partir do primeiro e a generalizamos pelo princípio de Newton. Quando a mesma regra geral é induzida repetidamente, podemos ter alguma confiança de que ela é verdadeira.

Até agora não fizemos nada que o algoritmo “dividir para conquistar” não fizesse. No entanto, suponhamos que, em vez de saber que Sócrates, Platão e Aristóteles são humanos, soubéssemos apenas que eles são filósofos. Ainda queremos concluir se eles são mortais, e induzimos anteriormente ou alguém nos disse que todos os humanos são mortais. O que está faltando? Uma regra diferente: *Todos os filósofos são humanos*. Essa também é uma generalização válida (pelo menos até termos inteligência artificial e os robôs começarem a filosofar) e ela “preenche a lacuna” de nosso raciocínio:

Sócrates é filósofo.

Todos os filósofos são humanos.
Todos os humanos são mortais.
Logo, Sócrates é mortal.

Também podemos induzir regras puramente a partir de outras regras. Se sabemos que todos os filósofos são humanos e mortais, podemos induzir que todos os humanos são mortais. (Não podemos induzir que todos os mortais são humanos porque conhecemos outras criaturas mortais, como cães e gatos. Por outro lado, cientistas, artistas e outros profissionais são humanos e mortais, o que reforça a regra.) Em geral, quanto mais regras e fatos tivermos inicialmente, mais oportunidades teremos para induzir novas regras usando a "dedução inversa". E quanto maior for o número de regras que induzirmos, mais regras poderemos induzir. É um círculo virtuoso de criação de conhecimento, limitado somente pelo risco de sobreajuste e pelo custo computacional. Mas nesse caso, também, a existência de um conhecimento inicial ajuda: se em vez de uma única lacuna maior tivermos muitas lacunas menores para preencher, as etapas de nossas induções serão menos arriscadas e, portanto, menos propensas ao sobreajuste. (Por exemplo, dado o mesmo número de situações, induzir que todos os filósofos são humanos é menos arriscado que induzir que todos os humanos são mortais.)

É difícil inverter uma operação porque o inverso não é exclusivo. Por exemplo, um número positivo tem duas raízes quadradas, uma positiva e uma negativa ($2^2 = (–2)^2 = 4$). Integrar a derivada de uma função, que é a abordagem mais usada, só recupera a função até uma constante. A derivada de uma função nos diz quanto essa função sobe ou desce em cada ponto. A soma de todas as alterações nos retorna a função, exceto por não sabermos onde ela começou; podemos "deslizar" a função integrada para cima ou para baixo sem alterar a derivada. Para facilitar, é possível ser rigoroso com a função assumindo que a constante aditiva é zero. A dedução inversa tem um problema semelhante, e o princípio de Newton é uma solução. Por exemplo, de *Todos os filósofos gregos são humanos* e *Todos os filósofos gregos são mortais* podemos induzir que *Todos os humanos são mortais*, ou apenas que *Todos os gregos são mortais*. Porém, por que ficar com a generalização mais modesta? Em vez disso, podemos presumir que todos os humanos são mortais até encontrarmos uma exceção. (O que, de acordo com Ray Kurzweil, ocorrerá em breve.)

Nesse meio tempo, uma aplicação importante da dedução inversa é prever se novas drogas terão efeitos colaterais danosos. Falhas durante testes com animais

e ensaios clínicos são a principal razão para drogas novas demandarem muitos anos e bilhões de dólares para serem desenvolvidas. Fazendo generalizações a partir de estruturas moleculares tóxicas conhecidas, podemos formar regras que eliminem compostos aparentemente promissores, aumentando muito as chances de ocorrerem testes bem-sucedidos com os remanescentes.

Aprendendo a curar o câncer

Quase sempre a dedução inversa é uma ótima maneira de fazermos novas descobertas em biologia, e essa é a primeira etapa da cura do câncer. De acordo com o Dogma Central, tudo que acontece em uma célula viva é, em última instância, controlado por seus genes, via proteínas cuja síntese eles iniciam. Na verdade, uma célula é como um computador minúsculo e o DNA é o programa executado nele: altere o DNA e uma célula da pele pode se transformar em um neurônio, ou a célula de um rato pode se transformar em uma célula humana. Em um programa de computador, todos os bugs são falha do programador. Porém, em uma célula, bugs podem surgir espontaneamente quando a exposição à radiação ou um erro de cópia transforma um gene em outro, quando um gene é copiado acidentalmente duas vezes, e assim por diante. Na maioria das situações essas mutações fazem a célula morrer de maneira silenciosa, mas em outras a célula começa a crescer e se dividir incontrolavelmente e surge um câncer.

Curar o câncer significa impedir que as células danificadas se reproduzam sem prejudicar as células saudáveis. Para isso temos de saber em que elas diferem e, em particular, como seus genomas diferem, já que todo o resto vem daí. Felizmente, o sequenciamento genético está se tornando rotineiro e acessível. Com isso podemos aprender a prever quais drogas funcionarão contra os genes cancerígenos. Essa abordagem é oposta à quimioterapia tradicional, que afeta todas as células indiscriminadamente. Aprender quais drogas funcionam contra quais mutações requer um banco de dados de pacientes, dos genomas de seus cânceres, das drogas testadas e dos resultados. As regras mais simples codificam correspondências do tipo uma-para-uma entre genes e drogas, como em: *Se o gene BCR-ABL estiver presente*, *use Gleevec*. (O gene BCR-ABL causa um tipo de leucemia e o medicamento Gleevec consegue curá-lo em nove entre dez pacientes.) Quando o sequenciamento de genomas de câncer e a confrontação de resultados dos tratamentos se tornarem prática-padrão, muitas outras regras como essa serão descobertas.

No entanto, isso é só o começo. A maioria dos cânceres envolve uma combinação de mutações ou só pode ser curada por drogas que ainda não foram descobertas. A próxima etapa é aprender regras com condições mais complexas, envolvendo o genoma do câncer, o genoma e o histórico médico do paciente, efeitos colaterais conhecidos causados pelas drogas, e assim por diante. Porém, no fim das contas precisamos é de um modelo de como a célula inteira funciona, que nos permita simular no computador o efeito tanto das mutações de um paciente específico quanto de diferentes combinações de drogas, existentes ou especulativas. Nossas principais fontes de informação para a construção desses modelos são sequenciadores de DNA, microarranjos de expressão gênica e a literatura biológica. É na combinação dessas fontes que a dedução inversa pode ajudar.

Adam, o cientista robô que vimos no Capítulo 1, fornece um exemplo. O objetivo de Adam é descobrir como as células de levedura funcionam. Inicialmente, ele possui um conhecimento básico da genética e do metabolismo da levedura e muitos dados de expressão gênica obtidos das células. Então, ele usa a dedução inversa para criar hipóteses sobre quais genes são expressos com quais proteínas, projeta experimentos com microarranjos para testá-las, revisa suas hipóteses e repete o processo. Se um gene será expresso é algo que vai depender de outros genes e das condições do ambiente, e a teia de interações resultante pode ser representada na forma de um conjunto de regras, como:

Se a temperatura está alta, o gene A é expresso.
Se o gene A é expresso e o gene B não, o gene C é expresso.
Se o gene C é expresso, o gene D não é expresso.

Se soubéssemos a primeira e a terceira regras e tivéssemos dados de microarranjo em que a uma temperatura alta B e D não fossem expressos, poderíamos induzir a segunda regra por dedução inversa. Quando tivermos essa regra, talvez até verificando-a com um experimento de microarranjo, poderemos usá-la como base de inferências indutivas adicionais. Da mesma forma, podemos agrupar as sequências de reações químicas de acordo com as proteínas que executam sua tarefa.

Contudo, não é suficiente saber quais genes regulam quais outros genes e como as proteínas organizam a teia de reações químicas da célula. Também precisamos saber quanto de cada espécie molecular é produzido. Os microarranjos de DNA e outros experimentos podem fornecer esse tipo de

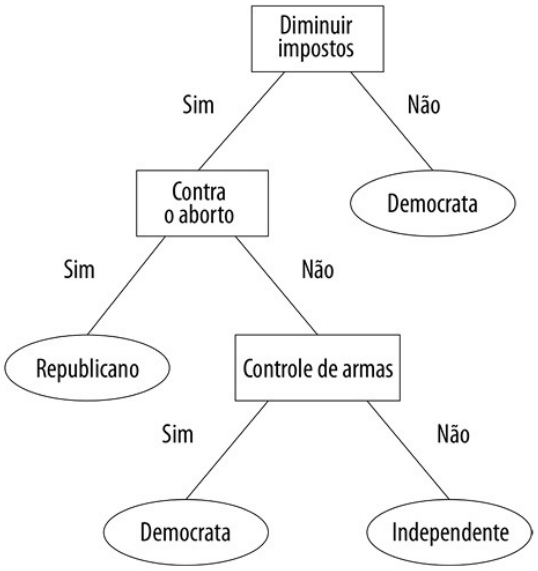
informação quantitativa, mas a dedução inversa, com seu caráter lógico “tudo ou nada”, não é muito boa para lidar com essa tarefa. Para tal precisamos dos métodos conexionistas que veremos no próximo capítulo.

Um jogo de vinte perguntas

Outra limitação da dedução inversa é que ela é computacionalmente muito intensa, o que dificulta o escalonamento para conjuntos de dados massivos. Para eles, o melhor algoritmo simbolista é a indução de árvores de decisão. As árvores de decisão podem ser consideradas uma resposta ao que fazer se regras de mais de um conceito servirem à mesma instância. Como decidir, nessa situação, a que conceito a instância pertence? Se nos depararmos com um objeto parcialmente oculto com superfície plana e quatro pernas, como saber se é uma mesa ou uma cadeira? Uma opção é ordenar as regras, por exemplo, pela diminuição da precisão, e escolher a primeira que atender. Outra é deixar que as regras decidam. Alternativamente, as árvores de decisão asseguram *a priori* que cada instância só apresente correspondência com uma única regra. É isso que ocorre quando cada par de regras difere em pelo menos um atributo de teste, o que permite organizar esse conjunto de regras em uma árvore de decisão. Por exemplo, considere estas regras:

Se você é a favor da diminuição de impostos e contra o aborto, é republicano.
Se é contra a diminuição de impostos, é democrata.
Se é a favor da diminuição de impostos, pró-escolha e contra o controle de armas, é independente.
Se é a favor da diminuição de impostos, pró-escolha e a favor do controle de armas, é democrata.

Essas regras podem ser organizadas na seguinte árvore de decisão:



Uma árvore de decisão é como participar de um jogo de vinte perguntas com um exemplo. A partir da raiz, cada nó pergunta sobre o valor de um único atributo, e, dependendo da reposta, seguimos uma ou outra ramificação. Quando chegamos a uma folha, lemos o conceito previsto. Cada caminho da raiz a uma folha corresponde a uma regra. Se isso lembra os incômodos menus telefônicos que temos de percorrer ao ligar para um serviço de atendimento ao cliente, não é por acaso: um menu telefônico é uma árvore de decisão. O computador do outro lado da linha está fazendo um jogo de vinte perguntas para descobrir o que você deseja, e cada menu é uma pergunta.

De acordo com a árvore de decisão anterior, você é republicano, democrata ou independente; não é possível se encaixar em mais de uma opção ou não se encaixar em nenhuma. Conjuntos de conceitos com essa propriedade são chamados de conjuntos de classes, e o algoritmo que os prevê é um classificador. Um único conceito define implicitamente duas classes: o próprio conceito e sua negação. (Por exemplo, spam e não spam.) Os classificadores são o tipo de machine learning mais disseminado.

Podemos aprender com árvores de decisão usando uma variante do algoritmo “dividir para conquistar”. Primeiro, pegamos um atributo na raiz para testar. Em seguida, nos concentramos nos exemplos que descem em cada ramificação e aplicamos a eles o próximo teste. (Por exemplo, verificamos se quem é a favor da diminuição de impostos é contra o aborto ou pró-escolha.) Repetimos isso para cada novo nó induzido até que todos os exemplos de uma ramificação tenham a mesma classe, ponto em que rotulamos essa ramificação com a classe.

Uma dúvida inevitável é como selecionar o melhor atributo para ser testado em um nó. A precisão – o número de exemplos previstos corretamente – não funciona muito bem, porque não estamos tentando prever uma classe específica, mas sim separar gradualmente as classes até que cada ramificação seja "pura". Isso nos traz à mente o conceito de entropia da teoria da informação. A entropia de um conjunto de objetos pode ser medida pelo nível de desordem existente nele. Se um grupo de 150 pessoas tiver 50 republicanos, 50 democratas e 50 independentes, sua entropia política será máxima. Por outro lado, se todos forem republicanos, a entropia será zero (considerando-se a afiliação partidária). Logo, para aprender usando uma boa árvore de decisão, devemos selecionar em cada nó o atributo que, em média, gere a entropia de classe mais baixa em todas as suas ramificações, o que é avaliado de acordo com quantos exemplos entrarem em cada ramificação.

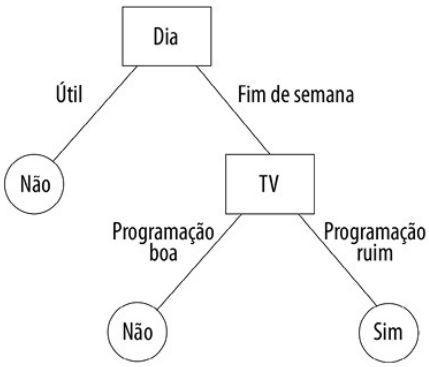
Assim como no aprendizado de regras, não queremos induzir uma árvore que preveja perfeitamente as classes de todos os exemplos de treinamento, porque poderíamos cometer sobreajuste. Como antes, podemos usar testes de significância ou uma penalidade para o tamanho da árvore e evitar isso.

A existência de uma ramificação para cada valor de um atributo funciona se o atributo for discreto, mas e quanto a atributos numéricos? Se tivéssemos uma ramificação para cada valor de uma variável contínua, a árvore seria infinitamente ampla. Uma solução simples é selecionar alguns limites-chave por entropia e usá-los. Por exemplo, a temperatura do paciente está acima ou abaixo de 37 graus Celsius? Isso, combinado com outros sintomas, pode ser tudo que o médico precisa saber sobre a temperatura do paciente para decidir se ele tem uma infecção.

As árvores de decisão são usadas em muitas áreas. Elas foram introduzidas no machine learning por intermédio de um trabalho em psicologia. Earl Hunt e alguns colegas as usaram nos anos 1960 para modelar como os humanos adquirem novos conceitos, e um de seus alunos de pós-graduação, J. Ross Quinlan, tentou usá-las posteriormente no jogo de xadrez. Seu objetivo original era prever o resultado final de partidas com rei e torre *versus* rei e cavalo de acordo com as posições no tabuleiro. A partir dessas origens modestas, as árvores de decisão têm se desenvolvido para ser, de acordo com os pesquisadores, o algoritmo de machine learning mais amplamente usado. Não é difícil entender o porquê: elas são fáceis de compreender, seu aprendizado é

rápido e geralmente são muito precisas sem grandes ajustes. Quinlan é o pesquisador mais proeminente da escola simbolista. Um australiano calmo e realista, ele tornou as árvores de decisão o padrão-ouro da classificação, melhorando-as implacavelmente ano após ano e redigindo artigos muito claros sobre elas.

Independentemente do que você quiser prever, há uma boa chance de que alguém tenha usado uma árvore de decisão para isso. O Kinect da Microsoft usa árvores de decisão para descobrir onde várias partes de nosso corpo estão a partir da saída de sua câmera com sensor de profundidade; ele então usa seus movimentos para controlar o console de jogos Xbox. Em uma disputa apertada em 2002, árvores de decisão previram três de cada quatro decisões judiciais da Suprema Corte, enquanto um grupo de especialistas acertou menos de 60%. Milhares de usuários de árvores de decisão não podem estar errados, você pensa, e projeta uma para prever a resposta de sua amiga a um convite para sair:



De acordo com essa árvore, hoje à noite ela dirá sim. Você respira fundo, pega o telefone e disca seu número.

Os simbolistas

A principal crença dos simbolistas é que a inteligência pode ser reduzida em sua totalidade à manipulação de símbolos. Um matemático resolve equações trocando símbolos de lugar e substituindo-os por outros de acordo com regras predefinidas. O mesmo ocorre com um especialista em lógica fazendo deduções. Conforme essa hipótese, a inteligência não depende do substrato; não importa se as manipulações de símbolos são executadas em um quadro-negro, pelo ligamento e desligamento de transistores, pela ativação de neurônios ou em uma brincadeira com jogos de montar. Se você tiver uma configuração com o poder

de uma máquina de Turing universal, poderá fazer qualquer coisa. Um software pode ficar totalmente separado do hardware, e se seu problema é descobrir como as máquinas conseguem aprender, (felizmente) não precisa se preocupar com o último além de comprar um PC ou ciclos na nuvem da Amazon.

Os machine learners simbolistas compartilham essa crença no poder da manipulação de símbolos com outros cientistas da computação, psicólogos e filósofos. O psicólogo David Marr defende que qualquer sistema de processamento de informações deve ser estudado em três níveis: o das propriedades básicas do problema que ele está resolvendo, o dos algoritmos e representações usados para resolver o problema e o de como esses algoritmos são fisicamente implementados. Por exemplo, a adição pode ser definida por um conjunto de axiomas não importando como ela é executada; números podem ser expressos de diferentes maneiras (romanos e arábicos, por exemplo) e somados com o uso de diferentes algoritmos, e estes podem ser implementados com um ábaco, uma calculadora portátil ou até mesmo, de maneira muito ineficiente, em nossa cabeça. O aprendizado é um caso básico de faculdade cognitiva que podemos estudar proveitosamente de acordo com os níveis de Marr.

O machine learning simbolista é uma ramificação da escola de engenharia do conhecimento da inteligência artificial. Nos anos 1970, os chamados sistemas baseados em conhecimento obtiveram alguns sucessos impressionantes, na década de 80 se espalharam rapidamente, mas depois desapareceram. A principal razão para isso foi o mal-afamado gargalo do conhecimento: extrair conhecimento de especialistas e codificá-lo como regras é muito difícil, trabalhoso e propenso a erros para ser viável para a maioria dos problemas. Deixar que o computador aprenda automaticamente a, digamos, diagnosticar doenças examinando bancos de dados de sintomas de pacientes anteriores e dos resultados correspondentes mostrou-se muito mais fácil do que entrevistar médicos indefinidamente. De repente, o trabalho de pioneiros como Ryszard Michalski, Tom Mitchell e Ross Quinlan tinham uma nova relevância e, desde então, a área não parou de crescer. (Outro problema importante foi que os sistemas baseados em conhecimento tinham dificuldades para lidar com a incerteza, algo sobre o qual veremos mais detalhes no Capítulo 6.)

Devido a suas origens e princípios orientadores, o machine learning simbolista se encontra mais próximo das outras áreas da inteligência artificial do que as demais escolas. Se a ciência da computação fosse um continente, o aprendizado

simbolista compartilharia uma longa fronteira com a engenharia do conhecimento. O conhecimento é trocado nas duas direções – conhecimento inserido manualmente para uso em aprendizes e conhecimento induzido para inclusão em bases de conhecimento –, mas o abismo racionalista-empírico divide essa fronteira e não é fácil cruzá-lo.

O simbolismo é o caminho mais curto para o Algoritmo Mestre. Ele não exige saber como a evolução ou o cérebro funciona e evita as complexidades matemáticas do bayesianismo. Conjuntos de regras e árvores de decisão são fáceis de entender, logo sabemos do que o aprendiz é capaz. Isso facilita descobrir o que ele está fazendo de certo e errado, corrigir os erros e ter confiança nos resultados.

Apesar da popularidade das árvores de decisão, a dedução inversa é o melhor ponto de partida para o Algoritmo Mestre. Ela tem a propriedade crucial de incorporar conhecimento facilmente – e sabemos que o problema de Hume torna essa propriedade essencial. Além disso, os conjuntos de regras são uma maneira exponencialmente mais compacta de representar a maioria dos conceitos do que as árvores de decisão. É fácil converter uma árvore de decisão em um conjunto de regras: cada caminho da raiz até uma folha se torna uma regra e não há ampliação. Por outro lado, no pior caso, converter um conjunto de regras em uma árvore de decisão requer a conversão de cada regra em uma miniárvore de decisão e, então, a substituição de cada folha da árvore da regra 1 por uma cópia da árvore da regra 2, de cada folha de cada cópia da regra 2 por uma cópia da regra 3, e assim por diante, causando uma ampliação massiva.

A dedução inversa é como se tivéssemos um supercientista examinando sistematicamente as evidências, considerando possíveis induções, confrontando as mais fortes e usando-as junto com outras evidências para construir hipóteses adicionais – tudo isso na velocidade dos computadores. É claro e belo, pelo menos para o gosto simbolista. Por outro lado, apresenta sérias desvantagens. O número de induções possíveis é vasto e, a menos que fiquemos próximos do conhecimento inicial, é fácil se perder. A dedução inversa é facilmente perturbada pelo ruído: como descobrir quais etapas dedutivas estão faltando se as próprias premissas ou conclusões estiverem erradas? E o mais grave é que raramente conceitos reais podem ser definidos de maneira concisa por um conjunto de regras. Eles não são tão claros: há uma grande área cinzenta entre, digamos, spam e não spam. Os conceitos reais requerem a avaliação e o acúmulo

de evidências frágeis até que surja um retrato claro. Diagnosticar uma doença envolve dar mais peso a alguns sintomas do que a outros e não se importar com evidências incompletas. Ninguém jamais conseguiu aprender um conjunto de regras que reconheçam um gato ao examinar os pixels de uma imagem e provavelmente ninguém jamais conseguirá.

Os conexionistas, em particular, são grandes críticos do aprendizado simbolista. Segundo eles, conceitos que não podemos definir com regras lógicas são apenas a ponta do iceberg; há muita coisa ocorrendo abaixo da superfície que o raciocínio formal não consegue ver, da mesma forma que grande parte do que ocorre em nossas mentes é subconsciente. Você não pode simplesmente construir um cientista automatizado sem corpo e esperar que ele produza alguma coisa com significado – primeiro é preciso dar-lhe algo como um cérebro real, conectado a sentidos reais, e fazê-lo crescer no mundo real, talvez tropeçando aqui e ali. Mas como construir esse cérebro? Aplicando engenharia reversa à concorrência. Se quisermos aplicar engenharia reversa a um carro, temos de olhar embaixo do capô. Para aplicar engenharia reversa ao cérebro, é preciso olhar dentro do crânio.

---

1 N.T.: Essa prática ocorria nos Estados Unidos.

capítulo 4

Como seu cérebro aprende?

A regra de Hebb, como ficou conhecida, é o princípio básico do conexionismo. Na verdade, a área deriva seu nome da crença de que o conhecimento é armazenado nas conexões entre os neurônios. Donald Hebb, psicólogo canadense, a concebeu dessa forma em seu livro de 1949, *The Organization of Behavior A organização do comportamento*: "Quando um axônio da célula *A* está suficientemente perto da célula *B* e participa de maneira repetida ou persistente de sua ativação, ocorre algum processo de crescimento ou alteração metabólica em uma das células ou em ambas, de modo que a eficiência da célula *A*, como uma das células que ativa a célula *B*, é aumentada". Com frequência, essa regra é parafraseada como "Neurônios que se ativam juntos tendem a permanecer juntos".

A regra de Hebb foi uma confluência de ideias da psicologia e da neurociência, com a incorporação de uma saudável dose de especulação. Aprender por associação era um dos temas favoritos dos empíricos britânicos, de Locke e Hume a John Stuart Mill. Em *Principles of Psychology*, William James enuncia um princípio geral da associação que é muito semelhante à regra de Hebb, com os neurônios sendo substituídos por processos cerebrais e o aumento da eficiência pela propagação da excitação. Por volta da mesma época, o grande neurocientista espanhol Santiago Ramón y Cajal estava fazendo as primeiras observações detalhadas do cérebro, corando neurônios individuais com o método de Golgi, recentemente inventado, e catalogando o que via como um botânico classifica novas espécies de árvores. Na época de Hebb, os neurocientistas não conheciam muito bem o funcionamento dos neurônios, mas ele foi o primeiro a propor um mecanismo pelo qual foi possível codificar as associações.

No aprendizado simbolista, há uma correspondência um-para-um entre os

símbolos e os conceitos que eles representam. Por outro lado, as representações conexionistas são distribuídas: cada conceito é representado por muitos neurônios e cada neurônio participa da representação de muitos conceitos. Os neurônios que ativam um ao outro formam o que Hebb chamou de montagem de células. Os conceitos e as memórias são representados no cérebro por montagens de células. Cada montagem pode incluir neurônios de diferentes regiões do cérebro e sobrepor outras montagens. A montagem de células para "perna" inclui a montagem de células para "pé", que inclui montagens para a imagem de um pé e o som da palavra *pé*. Se você perguntar a um sistema simbolista onde o conceito "Nova York" está representado, ele apontará para o local preciso na memória em que o conceito está armazenado. Em um sistema conexionista, a resposta é "cada pedaço dele está armazenado em vários locais".

Outra diferença entre o aprendizado simbolista e o conexionista é que o primeiro é sequencial e o outro é paralelo. Na dedução inversa, descobrimos em etapas as novas regras necessárias para chegarmos à conclusão desejada a partir das premissas. Nos modelos conexionistas, todos os neurônios aprendem simultaneamente de acordo com a regra de Hebb. Isso reflete as diferentes propriedades dos computadores e do cérebro. Os computadores fazem tudo executando uma pequena etapa de cada vez, como quando somam dois números ou desligam um comutador, e o resultado é que precisam de muitas etapas para fazer algo útil; porém, essas etapas são muito rápidas, porque os transistores podem ser ativados e desativados bilhões de vezes por segundo. Já o cérebro executa um grande número de processamentos em paralelo, com bilhões de neurônios operando ao mesmo tempo, mas cada processamento ocorre de forma lenta, porque os neurônios são ativados no máximo mil vezes por segundo.

O número de transistores de um computador está chegando perto do número de neurônios de um cérebro humano, mas o cérebro tem um número maior de conexões. Em um microprocessador, um transistor típico fica conectado diretamente a uma quantidade pequena de outros transistores, e a tecnologia de semicondutor planar limita severamente a melhoria do desempenho do computador. Por outro lado, um neurônio tem milhares de sinapses. Se você estivesse descendo a rua e encontrasse um conhecido, só demoraria cerca de um décimo de segundo para reconhecê-lo. Na velocidade de ativação dos neurônios, isso é tempo suficiente apenas para cem etapas de processamento, mas nessas cem etapas seu cérebro consegue vasculhar toda a memória, encontrar a melhor

correspondência e adaptá-la para o novo contexto (roupas diferentes, outra iluminação, e assim por diante). Em um cérebro, cada etapa de processamento pode ser muito complexa e envolver várias informações, como ocorre na representação distribuída.

Isso não significa que não podemos simular um cérebro com um computador; afinal, é isso que os algoritmos conexionistas fazem. Já que um computador é uma máquina de Turing universal, ele pode implementar não só os processamentos do cérebro como qualquer outro, se dermos tempo e memória suficientes. Particularmente, o computador pode usar a velocidade para compensar a falta de conectividade, empregando a mesma conexão mil vezes para simular mil conexões. Na verdade, atualmente a principal limitação dos computadores em comparação com o cérebro é o consumo de energia: o cérebro usa a mesma quantidade de energia que uma lâmpada pequena, enquanto o consumo do Watson poderia iluminar um prédio comercial inteiro.

No entanto, para simular um cérebro precisamos de mais do que a regra de Hebb; temos de entender como o cérebro é construído. Cada neurônio é como uma pequena árvore, com um grande número de raízes – os dendritos – e um tronco fino e sinuoso – o axônio. O cérebro é uma floresta com bilhões dessas árvores, mas há algo de incomum nelas. Os galhos de cada árvore estabelecem conexões – sinapses – com as raízes de milhares de outras árvores, formando um emaranhado massivo diferente de qualquer coisa que já vimos. Alguns neurônios têm axônios curtos e outros têm axônios excessivamente longos, estendendo-se facilmente de um lado a outro do cérebro. Se fossem esticados, os axônios de seu cérebro se estenderiam da Terra à Lua.

E essa selva crepita com eletricidade. Faíscas correm ao longo dos troncos das árvores e geram mais faíscas em árvores vizinhas. De vez em quando, uma área inteira da selva se agita freneticamente antes de se acalmar de novo. Quando você balança o pé, uma série de descargas elétricas, chamadas potenciais de ação, percorre a medula espinhal e a perna até alcançar os músculos do pé e fazer com que se movam. O cérebro em ação é uma sinfonia de faíscas elétricas. Se você pudesse se sentar dentro dele e observar o que acontece enquanto lê esta página, a cena que veria faria até mesmo a mais ocupada metrópole de ficção científica parecer relaxada. O resultado desse padrão tão complexo de ativação de neurônios é a sua consciência.

Na época de Hebb, não havia uma maneira de medir a força sináptica ou

alterações ocorridas nela, imagine descobrir a biologia molecular da alteração sináptica. Atualmente, sabemos que as sinapses crescem (ou se formam de novo) quando o neurônio pós-sináptico é ativado logo após o pré-sináptico. Como todas as células, os neurônios têm diferentes concentrações de íons dentro e fora deles, criando uma voltagem ao longo de sua membrana. Quando o neurônio pré-sináptico é ativado, pequenas bolsas liberam moléculas neurotransmissoras na lacuna sináptica. Elas fazem com que canais sejam abertos na membrana do neurônio pós-sináptico, deixando entrar íons de potássio e sódio e alterando a voltagem ao longo da membrana como resultado. Se neurônios pré-sinápticos suficientes forem ativados próximos uns dos outros, a voltagem aumentará repentinamente e um potencial de ação descerá pelo axônio do neurônio pós-sináptico. Isso também fará com que os canais de íons se tornem mais responsivos e novos canais apareçam, reforçando a sinapse. Pelo que sabemos, é assim que os neurônios aprendem.

A próxima etapa é transformar esse conhecimento em um algoritmo.

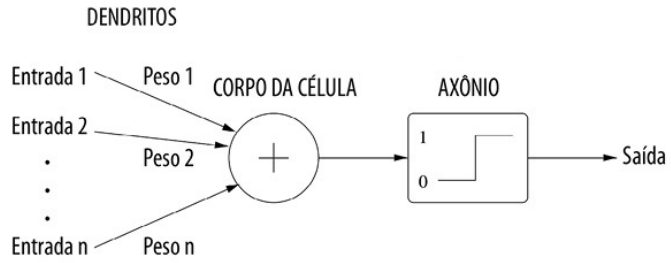
Ascensão e queda do perceptron

O primeiro modelo formal de um neurônio foi proposto por Warren McCulloch e Walter Pitts em 1943. Ele se parecia um pouco com as portas lógicas das quais os computadores são compostos. Uma porta OR é ativada quando pelo menos uma de suas entradas está ativada, e uma porta AND é ativada quando todas elas estão ativadas. Um neurônio de McCulloch-Pitts é ativado quando o número de suas entradas ativas ultrapassa um limite. Se o limite for um, o neurônio agirá como uma porta OR; se o limite for igual ao número de entradas, ele agirá como uma porta AND. Além disso, um neurônio de McCulloch-Pitts pode impedir que outro neurônio seja ativado, o que modela tanto as sinapses inibitórias quanto as portas NOT. Logo, uma rede de neurônios pode executar todas as operações de um computador. No passado, os computadores costumavam ser chamados de cérebros eletrônicos, o que não era uma simples analogia.

O que o neurônio de McCulloch-Pitts não faz é aprender. Para que o fizesse, teríamos de dar pesos variáveis às conexões entre os neurônios, resultando no que é chamado de perceptron. Os perceptrons foram inventados no fim dos anos 1950 por Frank Rosenblatt, um psicólogo da Universidade Cornell. Palestrante carismático e personagem jovial, Rosenblatt fez mais do que qualquer outra pessoa para dar forma aos primeiros dias do machine learning. O nome

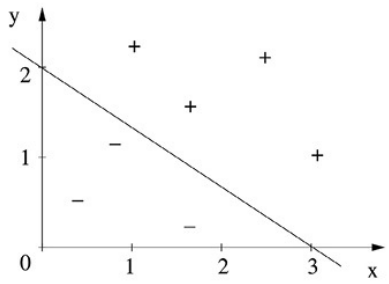
perceptron deriva de seu interesse pela aplicação de seus modelos a tarefas perceptivas, como o reconhecimento de voz e de caracteres. Em vez de implementar perceptrons em softwares, que naquela época eram muito lentos, Rosenblatt construiu seus próprios dispositivos. Os pesos foram implementados em resistores variáveis, como os encontrados em interruptores de diminuição de luz, e o aprendizado dos pesos era executado por motores elétricos que giravam os botões dos resistores. (Sem dúvida, alta tecnologia!)

Em um perceptron, um peso positivo representa uma conexão excitatória e um peso negativo uma conexão inibitória. O perceptron retorna 1 se a soma ponderada de suas entradas estiver acima do limite e 0 quando ela está abaixo. Variando os pesos e o limite, podemos alterar a função que o perceptron calcula. É claro que essa explicação ignora muitos dos detalhes de como os neurônios funcionam, mas queremos manter o assunto o mais simples possível; nosso objetivo é desenvolver um algoritmo de aprendizado de uso geral, não construir um modelo realista do cérebro. Se alguns dos detalhes que ignoramos se mostrarem importantes, os adicionaremos posteriormente. No entanto, apesar de nossas abstrações simplificadoras, ainda podemos ver como cada parte desse modelo corresponde a uma parte do neurônio:



Quanto maior for o peso de uma entrada, mais forte será a sinapse correspondente. O corpo da célula soma todas as entradas ponderadas e o axônio aplica uma função degrau ao resultado. A caixa do axônio no diagrama mostra a representação gráfica de uma função degrau: 0 para valores de entrada baixos, mudando abruptamente para 1 quando a entrada alcança o limite.

Suponhamos que um perceptron tivesse duas entradas contínuas x e y. (Em outras palavras, x e y podem assumir qualquer valor numérico, não apenas 0 e 1.) Então, cada exemplo poderia ser representado por um ponto no plano, e a fronteira entre exemplos positivos (para os quais o perceptron retorna 1) e negativos (quando o valor de retorno é 0) seria uma linha reta:



Isso ocorre porque a fronteira é o conjunto de pontos em que a soma ponderada é exatamente igual ao limite, e uma soma ponderada é uma função linear. Por exemplo, se os pesos forem 2 para x e 3 para y e o limite for 6, a fronteira será definida pela equação $2x + 3y = 6$. O ponto $x = 0$, $y = 2$ fica na fronteira, e para permanecer nela temos de cruzá-la três passos a cada dois passos descidos, para que o ganho em x compense a perda em y. Os pontos resultantes formam uma linha reta.

Aprender os pesos de um perceptron significa variar a direção da linha reta até todos os exemplos positivos estarem em um lado e todos os negativos estarem no outro. Em uma única dimensão, a fronteira é um ponto; em duas, é uma linha reta; em três, é um plano; e em mais de três, é um hiperplano. É difícil visualizar coisas no hiperespaço, mas a matemática funciona da mesma forma. Em n dimensões, temos n entradas e o perceptron tem n pesos. Para decidir se o perceptron será ou não ativado, multiplicamos cada peso pela entrada correspondente e comparamos a soma de todos eles com o limite.

Se todas as entradas tiverem peso um e o limite for a metade do número de entradas, o perceptron será ativado se mais da metade de suas entradas forem ativadas. Em outras palavras, o perceptron é como um pequeno parlamento em que a maioria vence. (Ou talvez não tão pequeno, se considerarmos que pode ter milhares de membros.) No entanto, de modo geral não é democrático, porque nem todos têm direito a voto. Uma rede neural é mais como uma rede social, em que alguns amigos íntimos valem mais do que os milhares de amigos do Facebook. E são os amigos que inspiram mais confiança que o influenciam mais. Se um amigo recomendar um filme e você o assistir e gostar dele, provavelmente da próxima vez seguirá seu conselho novamente. Por outro lado, se ele elogiar filmes de que você não gostou, começará a ignorar suas opiniões (e talvez a amizade até diminua).

É assim que o algoritmo do perceptron de Rosenblatt aprende os pesos.

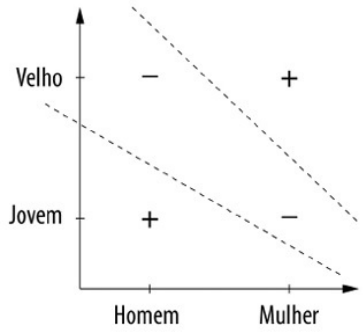
Considere a célula-avó, um dos experimentos de raciocínio favoritos dos

neurocientistas cognitivos. A célula-avó é um neurônio existente no cérebro que é ativado sempre que vemos nossa avó, e só nessa situação. Se as células-avós existem ou não é uma questão em aberto, mas projetaremos uma para usar no machine learning. Um perceptron aprende a reconhecer uma avó da seguinte forma. As entradas enviadas para a célula são os pixels brutos da imagem ou características associadas a ela, como *olhos castanhos*, que recebem o valor 1 se a imagem tiver um par de olhos castanhos; caso contrário recebem 0. Inicialmente, todas as conexões que vão das características até o neurônio têm pesos aleatórios baixos, como as sinapses de nosso cérebro no nascimento. Então, mostramos ao perceptron uma série de imagens, algumas de nossa avó e outras não. Se ele for ativado ao ver uma imagem da avó, ou não for ativado ao ver alguma outra coisa, então não é preciso ocorrer aprendizado. (Se não está quebrado, não conserte.) Porém, se o perceptron não for ativado quando estiver olhando para a avó, isso significa que a soma ponderada de suas entradas deve ter sido maior, logo devemos aumentar os pesos das entradas que foram ativadas. (Por exemplo, se sua avó tiver olhos castanhos, o peso dessa característica aumentará.) Inversamente, se o perceptron for ativado quando não deveria ser, devemos diminuir os pesos das entradas ativas. São os erros que conduzem o aprendizado. Com o tempo, as características que representarem nossa avó adquirirão pesos altos e as que não representarem receberão pesos baixos. Quando o perceptron for ativado sempre que ver a avó, e somente nessa situação, o aprendizado terá terminado.

O perceptron provocou muita empolgação. Era simples e mesmo assim conseguia reconhecer letras impressas e sons de voz apenas sendo treinado com exemplos. Um colega de Rosenblatt na Universidade Cornell provou que, se os exemplos positivos e negativos pudessem ser separados por um hiperplano, o perceptron o encontraria. Para Rosenblatt e outros, uma compreensão genuína de como o cérebro aprende parecia possível, e com ela obteríamos um poderoso algoritmo de aprendizado de uso geral.

Então, o perceptron encontrou um obstáculo. Os engenheiros do conhecimento ficaram irritados com a pretensão de Rosenblatt e com ciúmes de toda a atenção e patrocínio que as redes neurais, especificamente os perceptrons, receberam. Um deles foi Marvin Minsky, antigo colega de classe de Rosenblatt na Bronx High School of Science e, na época, líder do grupo de inteligência artificial do MIT. (Ironicamente, ele obteve o grau de PhD em redes neurais, mas se

desiludiu.) Em 1969, Minsky e seu colega Seymour Papert publicaram *Perceptrons*, um livro detalhando as deficiências do algoritmo epônimo, com vários exemplos de coisas simples que eles não conseguiam aprender. A mais simples – e, portanto, a mais condenatória – era a função exclusive-OR, ou XOR na abreviação, que é verdadeira quando uma de suas entradas é verdadeira, mas não ambas. Por exemplo, o público-alvo mais leal da Nike é supostamente os adolescentes do sexo masculino e as mulheres de meia-idade. Em outras palavras, você comprará tênis da Nike se for "jovem XOR mulher". Jovem é correto, mulher é correto, mas ambos não. Você também não será um alvo promissor das propagandas da Nike se não for nem jovem nem mulher. O problema da função XOR é que não há uma linha reta capaz de separar os exemplos positivos dos negativos. Esta figura mostra duas candidatas que falharam:



Já que os perceptrons só podem aprender fronteiras lineares, não conseguem aprender a função XOR. E se não podem fazer nem isso, não são um modelo muito bom de como os cérebros aprendem, ou candidatos viáveis para o Algoritmo Mestre.

No entanto, um perceptron só modela o aprendizado de um único neurônio, e, embora Minsky e Papert tenham admitido que camadas de neurônios interconectados poderiam aprender mais, eles não enxergavam uma maneira de simulá-las. Ninguém enxergava. O problema é que não há uma maneira clara de alterar os pesos dos neurônios das camadas "ocultas" para reduzir os erros cometidos pelos da camada externa. Cada neurônio oculto influencia a saída por meio de vários caminhos, e cada erro tem mil causadores. Quem devemos culpar? Ou, inversamente, quem deve receber os créditos pelas saídas corretas? Esse problema de atribuição de crédito surge sempre que tentamos aprender um modelo complexo e é um dos problemas centrais do machine learning.

Os *perceptrons* eram matematicamente irrepreensíveis, intensos em sua clareza

e desastrosos em seus efeitos. Na época, o machine learning estava associado principalmente a redes neurais, e a maioria dos pesquisadores (para não mencionar os patrocinadores) concluiu que a única maneira de construir um sistema inteligente era programá-lo explicitamente. Pelos quinze anos seguintes, a engenharia do conhecimento assumiria o papel central, e o machine learning parecia destinado ao esquecimento.

Físico cria cérebro de vidro

Se a história do machine learning fosse um filme de Hollywood, o vilão seria Marvin Minsky. Ele seria a rainha má que deu à Branca de Neve uma maçã envenenada, fazendo-a adormecer. (Em um ensaio de 1988, Seymour Papert chegou até a se comparar, ironicamente, ao caçador que a rainha enviou para matar Branca de Neve na floresta.) E o príncipe encantado seria um físico do Caltech chamado John Hopfield. Em 1982, Hopfield notou uma surpreendente analogia entre o cérebro e os vidros de spin, um material exótico muito apreciado pelos físicos estatísticos. Isso desencadeou um renascimento conexionista que culminou, alguns anos depois, na invenção dos primeiros algoritmos capazes de resolver o problema da atribuição de crédito, prenunciando uma nova era em que o machine learning substituiu a engenharia do conhecimento como paradigma dominante da inteligência artificial.

Os vidros de spin não são realmente vidros, embora tenham algumas propriedades semelhantes. Em vez disso, eles são materiais magnéticos. Cada elétron é um pequeno ímã determinado por seu spin, que pode apontar “para cima” ou “para baixo”. Em materiais como o ferro, os spins dos elétrons tendem a se alinhar: quando um elétron com o spin para baixo é rodeado por elétrons com os spins para cima, ele costuma subir de nível. Quando a maioria dos spins de um pedaço de ferro se alinha, ele se transforma em um ímã. Em ímãs comuns, a força da interação entre spins adjacentes é a mesma para todos os pares, mas em um vidro de spin ela pode variar; pode até mesmo ser negativa, fazendo com que spins próximos apontem para direções opostas. A energia de um ímã comum é mais baixa quando todos os seus spins se alinham, mas em um vidro de spin as coisas não são tão simples. Na verdade, encontrar o estado de energia mais baixa de um vidro de spin é um problema NP-completo, o que significa que quase todos os outros problemas de otimização difíceis de resolver podem ser reduzidos a ele. Logo, um vidro de spin não permanece necessariamente em seu

estado geral de energia mais baixa; assim como a água da chuva pode escorrer morro abaixo até chegar a um lago em vez de alcançar o oceano, em vez de evoluir para o nível global, um vidro de spin pode ficar preso em um nível mínimo local, um estado com energia menor do que todos os estados que podem ser alcançados a partir dele com a ascensão de nível.

Hopfield notou uma semelhança interessante entre vidros de spin e redes neurais: os spins de um elétron respondem ao comportamento de seus vizinhos como ocorre com um neurônio. No caso do elétron, ele sobe de nível quando a soma ponderada dos vizinhos excede um limite, caso contrário desce de nível (ou permanece no mesmo). Inspirado por essa descoberta, ele definiu um tipo de rede neural que evolui com o tempo da mesma forma que um vidro de spin e postulou que os estágios de energia mínimos da rede são suas memórias. Cada um desses estados tem uma "bacia de atração" de estados iniciais e converge para ela, e é assim que a rede pode executar o reconhecimento de padrões: por exemplo, se uma das memórias for o padrão de pixels preto e branco formado pelo dígito nove e a rede enxergar um nove distorcido, ela convergirá para o "ideal" e o reconhecerá. Abruptamente, um vasto corpo de teorias da física podia ser aplicado ao machine learning e um grande número de físicos estatísticos foi para a área, ajudando-a a sair do nível mínimo local em que estava presa.

No entanto, um vidro de spin ainda é um modelo muito pouco realista do cérebro. Em primeiro lugar, as interações entre spins são simétricas e as conexões entre neurônios não o são. Outro grande problema que o modelo de Hopfield ignorou é que neurônios reais são estatísticos: eles não se ativam ou desativam deterministicamente de acordo com suas entradas; em vez disso, à medida que a soma ponderada das entradas aumenta, os neurônios se tornam mais propensos a ser ativados, mas não há garantia de que serão. Em 1985, David Ackley, Geoff Hinton e Terry Sejnowski substituíram os neurônios determinísticos das redes de Hopfield por neurônios probabilísticos. Agora, uma rede neural tinha uma distribuição probabilística ao longo de seus estados, com a ocorrência dos estados de energia mais alta sendo exponencialmente menos provável do que a dos de energia mais baixa. Na verdade, a probabilidade de a rede estar em um estado específico era proporcionada pela conhecida distribuição de Boltzmann da termodinâmica, logo eles chamaram sua rede de máquina de Boltzmann.

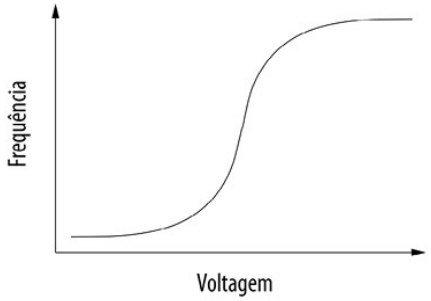
A máquina de Boltzmann tem uma combinação de neurônios sensoriais e

ocultos (semelhantes, por exemplo, à retina e ao cérebro, respectivamente). Ela aprende se alternando entre os estados de vigília e suspensão, como acontece com os humanos. Quando em vigília, os neurônios sensoriais são ativados conforme imposto pelos dados, e os ocultos evoluem de acordo com a dinâmica da rede e a entrada sensorial. Por exemplo, se a rede receber uma imagem do número nove, os neurônios correspondentes aos pixels pretos da imagem permanecerão ativados, os outros ficarão desativados e os ocultos serão ativados aleatoriamente de acordo com a distribuição de Boltzmann, dados os valores desses pixels. Durante a suspensão, a máquina descansa, deixando tanto os neurônios sensoriais quanto os ocultos livres para relaxar. Um pouco antes do raiar de um novo dia, ela compara as estatísticas de seus estados durante a suspensão e durante as atividades do dia anterior e altera os pesos das conexões para que fiquem de acordo. Se dois neurônios tendem a ser acionados juntos durante o dia, mas não tanto durante a suspensão, o peso de sua conexão sobe; se ocorrer o oposto, ele desce. Por intermédio dessa atividade dia após dia, as correlações previstas entre os neurônios sensoriais evoluem até chegarem às reais. Nesse ponto, a máquina de Boltzmann terá aprendido um bom modelo dos dados e resolvido eficazmente o problema da atribuição de crédito.

Geoff Hinton continuou o experimento e testou muitas variações da máquina de Boltzmann nas décadas seguintes. Hinton, um psicólogo transformado em cientista da computação e trineto de George Boole, inventor do cálculo lógico usado em todos os computadores digitais, é o líder mundial dos conexionistas. Tentou por mais tempo e com mais afinco entender como o cérebro funciona do que qualquer outra pessoa. Ele conta que chegou em casa um dia em um estado de grande empolgação, dizendo “Consegui! Descobri como o cérebro funciona!”. Sua filha respondeu: “De novo não, pai!”. A última paixão de Hinton é o aprendizado profundo, que veremos posteriormente neste capítulo. Ele também estava envolvido no desenvolvimento da backpropagation, um algoritmo ainda melhor do que as máquinas de Boltzmann para a resolução do problema de atribuição de crédito que examinaremos a seguir. Em princípio, as máquinas de Boltzmann poderiam resolver o problema da atribuição de crédito, mas na prática o aprendizado era muito lento e difícil, tornando essa abordagem inviável para a maioria das aplicações. O divisor de águas seguinte envolveria se livrar de outra simplificação excessiva que provinha da época de McCulloch e Pitts.

A curva mais importante do mundo

No que diz respeito a seus vizinhos, um neurônio só pode estar em um entre dois estados: ativo ou inativo. No entanto, esse esquema não considera uma sutileza importante. Os potenciais de ação têm vida curta; a voltagem chega ao pico por uma pequena fração de segundo e retorna imediatamente ao seu estado de descanso. E raramente um único pico é registrado no neurônio de destino; são necessários vários picos nos dois neurônios para que ele seja ativado. Um neurônio típico tem picos ocasionalmente na ausência de estímulo, gera cada vez mais picos à medida que o estímulo cresce e chega a um nível de saturação com a taxa de picos mais veloz que ele pode reunir, além da qual o aumento dos estímulos não tem efeito. Em vez de uma porta lógica, um neurônio é mais como um conversor de voltagem para frequência. A curva da frequência de acordo com a voltagem tem esta aparência:



Essa curva, que parece um S alongado, é conhecida sob diversas denominações, como curva logística, sigmoide ou S. Examine-a atentamente, porque ela é a curva mais importante do mundo. No início, a saída aumenta lentamente acompanhando a entrada, tão lentamente que parece constante. Em seguida, ela começa a mudar mais rapidamente, depois muito rapidamente, e então de maneira cada vez mais lenta, até se tornar constante novamente. A curva de transferência de um transistor, em que as voltagens de entrada e saída estão relacionadas, também é uma curva S. Logo, tanto os computadores quanto o cérebro estão cheios de curvas S. Porém, não é só isso. A curva S representa a forma de transições de fase de todos os tipos: a probabilidade de um elétron mudar seu spin de acordo com o campo aplicado, a magnetização do ferro, a gravação de um bit de memória em um disco rígido, a abertura de um canal iônico em uma célula, o derretimento do gelo, a evaporação da água, a expansão inflacionária do universo primordial, o equilíbrio pontuado da evolução, mudanças de paradigma da ciência, a disseminação de novas tecnologias, o

white flight1 proveniente de vizinhanças multiétnicas, rumores, epidemias, revoluções, a queda de impérios e muito mais. O livro *O ponto da virada* poderia igualmente (a não ser pelo viés comercial) ter sido chamado de *A curva S*. Um terremoto é uma transição de fase na posição relativa de duas placas tectônicas adjacentes. Um barulho à noite é apenas o som das microscópicas "placas tectônicas" das paredes de sua casa mudando de lugar, logo não se assuste. Joseph Schumpeter disse que a economia evolui por quedas e saltos: as curvas S são a forma da destruição criativa. O efeito de ganhos e perdas financeiros sobre nossa felicidade segue uma curva S, portanto não se preocupe demais. A probabilidade de que uma fórmula lógica aleatória seja satisfatória – um exemplo típico de problema NP-completo – passa por uma fase de transição de quase 1 para quase 0 à medida que o tamanho da fórmula aumenta. Os físicos estatísticos dedicam suas vidas ao estudo das transições de fase.

No livro *O Sol também se levanta* de Hemingway, quando perguntam a Mike Campbell como ele foi à falência, sua resposta é: "De duas maneiras. Gradualmente e então repentinamente". O mesmo poderia ser dito do Lehman Brothers. Essa é a essência de uma curva S. Uma das regras de previsão do futurista Paul Saffo é: procure curvas S. Quando você não conseguir acertar a temperatura do chuveiro – primeiro está fria demais e depois fica rapidamente muito quente –, a culpa é da curva S. Quando estiver fazendo pipoca, observe o progresso da curva S: inicialmente nada acontece, então alguns grãos de milho estouram, depois muitos, todos então estouram em uma repentina explosão de fogos de artifício, depois mais alguns, e a pipoca está pronta. Cada movimento de nossos músculos segue uma curva S: lento, depois rápido e então lento novamente. Os personagens de desenhos animados ganharam mais naturalidade quando os animadores da Disney descobriram esse esquema e começaram a copiá-lo. Seus olhos se movem em curvas S, fixando-se em uma coisa e depois em outra, acompanhando sua consciência. As mudanças de humor são transições de fase. Assim como o são o nascimento, a adolescência, apaixonar-se, casar-se, engravidar, começar a trabalhar, perder o emprego, mudar-se para uma nova cidade, ser promovido, aposentar-se e morrer. O universo é uma vasta sinfonia de transições de fase, do cósmico ao microscópico, das coisas mundanas às que transformam nossas vidas.

A curva S não é importante apenas como modelo; ela também é o pau para toda obra da matemática. Se você prestar atenção em sua seção intermediária,

verá que ela se aproxima de uma linha reta. Muitos fenômenos que consideramos lineares são na verdade curvas S, porque nada pode crescer sem limite. A favor da relatividade, e contra Newton, a aceleração não aumenta linearmente com a força, mas segue uma curva S centralizada em zero. O mesmo ocorre com a corrente elétrica como função da voltagem dos resistores encontrados nos circuitos eletrônicos, ou em uma lâmpada (até o filamento derreter, o que é outra transição de fase). Se você olhar a curva S à distância, ela parecerá uma função degrau, com a saída repentinamente mudando de 0 para 1 ao alcançar o limite. Portanto, dependendo das voltagens de entrada, a mesma curva representa o funcionamento de um transistor tanto em computadores digitais quanto em dispositivos analógicos, como os amplificadores e sintonizadores de rádio. A primeira parte de uma curva S é efetivamente um exponencial, e perto do ponto de saturação ela se aproxima da queda exponencial. Quando alguém falar em crescimento exponencial, pergunte a si mesmo: com que rapidez ele se transformará em uma curva S? Quando a explosão demográfica diminuirá, a lei de Moore perderá força ou a singularidade deixará de ocorrer? Calcule a diferencial de uma curva S e obterá uma curva de sino: lenta, rápida, de lenta se tornando baixa, alta, baixa. Adicione uma sucessão de curvas S escalonadas para cima e para baixo e obterá algo próximo a uma onda senoidal. Na verdade, todas as funções podem ser representadas com alguma fidedignidade por uma soma de curvas S: quando a função sobe, adicionamos uma curva S; quando ela desce, subtraímos a curva. O aprendizado das crianças não é um aperfeiçoamento contínuo, mas um acúmulo de curvas S. A mudança tecnológica também. Cerre os olhos ao fixá-los no horizonte da cidade de Nova York e verá uma sucessão de curvas S, tão nítidas quanto a extremidade de um arranha-céu.

O mais importante para nós é que as curvas S levam a uma nova solução para o problema da atribuição de crédito. Se o universo é uma sinfonia de transições de fase, iremos modelá-lo com uma. É isso que o cérebro faz: ele ajusta o sistema interno de transições de fase ao que existe externamente. Portanto, substituiremos a função degrau do perceptron por uma curva S e veremos o que acontece.

Subindo montanhas no hiperespaço

No algoritmo do perceptron, o sinal de erro é um esquema “tudo ou nada”: ou

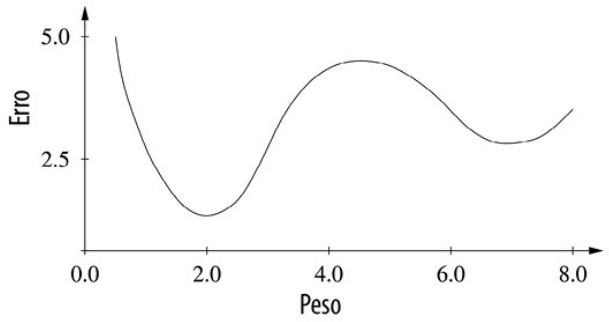
ele funciona corretamente ou não funciona. Isso não ajuda muito, principalmente se tivermos uma rede de vários neurônios. Podemos saber que o neurônio de saída está errado (“Essa não é minha avó!”), mas e quanto a algum neurônio oculto dentro do cérebro? E o que significaria esse neurônio estar certo ou errado? Se a saída dos neurônios for contínua em vez de binária, o quadro muda. Para começar, agora sabemos *quanto* o neurônio de saída está errado: é uma função da diferença entre ele e a saída desejada. Se precisarmos que o neurônio esteja bem ativo (“Oi, vovó!”) e ele estiver só um pouco, isso é melhor do que se não estivesse nada ativo. O mais importante é que agora podemos propagar o erro para os neurônios ocultos: se precisarmos que o neurônio de saída esteja mais ativo e o neurônio A se conectar a ele, quanto mais A estiver ativo, mais devemos reforçar sua conexão; mas se A for inibido por um neurônio B, então B deve ficar menos ativo, e assim por diante. De acordo com o feedback proveniente de todos os neurônios aos quais estiver conectado, cada neurônio decidirá quanto deve ficar mais ou menos ativo. Com base nisso e na atividade de *seus* neurônios de entrada, ele deve reforçar ou enfraquecer suas conexões. Preciso ficar mais ativo e o neurônio B está me inibindo? Diminua seu peso. O neurônio C está ativo, mas sua conexão comigo é fraca? Reforce-a. No próximo ciclo, os neurônios que forem meus “clientes” ao longo da rede me dirão se estou me saindo bem.

Sempre que a “retina” do aprendiz vê uma nova imagem, esse sinal se propaga para frente pela rede até produzir uma saída. A comparação desta com a saída desejada gera um sinal de erro, que então se propaga de volta pelas camadas até alcançar a retina. De acordo com esse sinal de retorno e com as entradas que recebeu durante a passagem para frente, cada neurônio ajusta seus pesos. À medida que a rede vê cada vez mais imagens da avó e de outras pessoas, os pesos convergem gradualmente para valores que permitam que ela diferencie os dois tipos de imagens. A backpropagation, como esse algoritmo é chamado, é incrivelmente mais poderosa do que o algoritmo do perceptron. Um único neurônio só pode aprender linhas retas. Com neurônios ocultos suficientes, um perceptron multicamadas, como é chamado, pode representar fronteiras arbitrariamente intrincadas. Isso torna a backpropagation – ou simplesmente a backprop – o algoritmo mestre dos conexionistas.

A backprop é o exemplo de uma estratégia muito comum tanto na natureza quanto na tecnologia: se estiver com pressa para chegar ao topo da montanha,

suba pelo lado mais íngreme possível. O termo técnico para isso é subida de gradiente (se quiser chegar ao topo) ou descida de gradiente (se quiser ir ao fundo do vale). As bactérias conseguem encontrar comida nadando para cima até o gradiente de concentração de, digamos, moléculas de glicose, e evitam elementos tóxicos nadando para baixo de seu gradiente. Todos os tipos de coisas, de asas de aviões a sistemas de antenas, podem ser otimizados pela descida de gradiente. A backprop é uma maneira eficiente desse esquema para um perceptron multicamadas: mantenha-se ajustando os pesos para diminuir o erro e pare quando todos os ajustes falharem. Com a backprop, não precisamos descobrir como ajustar os pesos de cada neurônio a partir do zero, o que seria muito lento; podemos fazer isso camada a camada, ajustando cada neurônio com base no modo como ajustamos os neurônios aos quais ele se conecta. Se você tivesse de descartar seu kit de ferramentas de machine learning inteiro em uma emergência exceto por uma ferramenta, recomendaria ficar com a descida de gradiente.

Mas a backprop resolve o problema do machine learning? Podemos simplesmente agrupar uma grande pilha de neurônios, esperar que ela faça seu truque e ganhar um prêmio Nobel por descobrir como o cérebro funciona? Infelizmente, a vida não é tão fácil. Suponhamos que sua rede tivesse um único peso e este fosse o gráfico do erro gerado por ele:



O peso ótimo, em que o erro é mais baixo, é 2,0. Se a rede começar com um peso de 0,75, por exemplo, a backprop chegará ao nível ótimo em algumas etapas, como uma bola rolando montanha abaixo. Porém, se a rede começar com 5,5, a backprop descerá para 7,0 e ficará presa aí. A backprop, com suas mudanças de peso incrementais, não sabe como encontrar o erro mínimo global, e os erros locais podem ser arbitrariamente ruins, como confundir a avó com um chapéu. Com um único peso, você poderia testar todos os valores possíveis em incrementos de 0,01 e encontrar o peso ótimo dessa forma. Todavia, com

milhares de pesos, imagine com milhões ou bilhões, essa opção não é viável porque o número de pontos na grade sobe exponencialmente com o número de pesos. O mínimo global está oculto em algum local na insondável vastidão do hiperespaço – boa sorte em sua busca.

Suponhamos que você tivesse sido raptado e o abandonassem vendado em algum lugar dos Himalaias. Sua cabeça está doendo e sua memória também não está boa. Tudo que você sabe é que precisa chegar ao topo do monte Everest. O que você faz? Dá um passo para frente e quase cai em uma ravina. Após tomar fôlego decide ser um pouco mais sistemático. Você sente cuidadosamente o solo ao redor com seu pé até encontrar o ponto mais alto e anda energicamente para esse ponto. Em seguida, repete a operação. Pouco a pouco, você sobe cada vez mais. Após algum tempo, os passos dados conduzem para baixo e você para. Essa é a subida de gradiente. Se os Himalaias fossem apenas o monte Everest e este fosse um cone perfeito, funcionaria muito bem. Porém, é mais provável que, quando você chegar a um local em que comece a ser conduzido para baixo, ainda esteja muito longe do topo. Você está apenas em um sopé em algum lugar e está preso. É isso que ocorre com a backprop, exceto por ela subir montanhas no hiperespaço em vez de em 3D. Se sua rede tiver um único neurônio, simplesmente aumentar os pesos para valores melhores um passo de cada vez o levará ao topo. Porém, com um perceptron multicamadas, a paisagem é muito escarpada; boa sorte na busca do pico mais alto.

Foi parcialmente por isso que Minsky, Papert e outros não conseguiram descobrir como realizar o aprendizado com perceptrons multicamadas. Eles podiam imaginar a substituição de funções degrau por curvas S e a execução da descida de gradiente, mas se depararam com o problema dos mínimos locais do erro. Naquela época os pesquisadores não confiavam em simulações; eles demandavam prova matemática de que um algoritmo podia funcionar, e não há uma prova assim para a backprop. Porém, acabou ficando claro que quase sempre um mínimo local é suficiente. Com frequência, a superfície do erro se parece com os espinhos de um porco-espinho, com muitos picos e vales íngremes, mas não importa se o vale mais baixo foi encontrado; podemos usar qualquer vale. Melhor ainda, um mínimo local pode ser preferível porque ele tem menos probabilidade de sobreajustar os dados do que o mínimo global.

O hiperespaço é uma faca de dois gumes. Por um lado, quanto maior é o número de dimensões, mais espaço há para superfícies altamente intrincadas e

soluções ótimas. Por outro, para ficar preso a um "local optimum"[2] você precisa estar confinado a *cada* dimensão, logo é mais difícil ficar preso em muitas dimensões do que em três. No hiperespaço há passagens montanhosas em todos os (hiper) locais. Portanto, com a ajuda de um xerpa[3], com frequência a backprop encontra um caminho que leva a um conjunto de pesos adequado. Talvez o caminho leve apenas ao vale místico de Shangri-La, não ao mar, mas por que reclamar se no hiperespaço há milhões de Shangri-Las, cada um com bilhões de passagens montanhosas que levam ao mar?

No entanto, cuidado ao dar muita relevância aos pesos encontrados pela backprop. Lembre-se de que podem existir outros igualmente bons. O aprendizado em perceptrons multicamadas é um processo caótico no sentido de que quando começamos em locais um pouco diferentes acabamos encontrando soluções muito diversas. O fenômeno é o mesmo, esteja a pequena diferença nos pesos iniciais ou nos dados de treinamento, e se manifesta em todos os aprendizes poderosos, não apenas na backprop.

Poderíamos eliminar o problema das soluções ótimas removendo as curvas S e deixando cada neurônio retornar a soma ponderada de suas entradas. Isso tornaria a superfície de erro mais plana, permanecendo um único mínimo – o global. Entretanto, o problema é que uma função linear composta por funções lineares continua sendo apenas uma função linear, logo uma rede de neurônios lineares não é melhor do que um único neurônio. Um cérebro linear, não importa o tamanho, é mais estúpido do que um verme. As curvas S são um ponto de equilíbrio adequado entre a estupidez das funções lineares e a severidade das funções degrau.

A vingança do perceptron

A backprop foi inventada em 1986 por David Rumelhart, psicólogo da Universidade da Califórnia, em San Diego, com a ajuda de Geoff Hinton e Ronald Williams. Entre outras coisas, eles mostraram que a backprop pode aprender funções XOR, o que permitiu que os conexionistas se vingassem de Minsky e Papert. Como vimos no exemplo da Nike, homens jovens e mulheres de meia-idade são os mais prováveis compradores de tênis da marca. Podemos representar isso com uma rede de três neurônios: um neurônio é ativado quando vê um homem jovem, o outro quando vê uma mulher de meia-idade e o terceiro quando um dos outros neurônios é ativado. E com a backprop podemos aprender

os pesos apropriados, resultando em um detector bem-sucedido de clientes da Nike. (Estamos quase lá, Marvin[4].)

Em uma demonstração inicial do poder da backprop, Terry Sejnowski e Charles Rosenberg treinaram um perceptron multicamadas para ler em voz alta. Seu sistema NETtalk examinou o texto, selecionou os fonemas corretos de acordo com o contexto e os passou para um sintetizador de voz. O NETtalk não só criou generalizações precisas para novas palavras, algo que os sistemas baseados em conhecimento não podiam fazer, como aprendeu a falar de maneira incrivelmente semelhante à humana. Sejnowski costumava fascinar audiências em reuniões de pesquisa reproduzindo uma fita com o progresso do NETtalk: no início balbuciando, depois começando a fazer sentido e, então, falando suavemente com erros apenas ocasionais. (Você pode encontrar amostras no YouTube ao digitar “sejnowski nettalk”.)

O primeiro grande sucesso das redes neurais foi em previsões no mercado de ações. Já que conseguiam detectar pequenas não linearidades em dados com muito ruído, elas superaram os modelos lineares que então eram prevalentes em finanças e usados em larga escala. Um fundo de investimentos típico treinaria uma rede separada para cada grupo de um grande número de ações, deixaria as redes selecionarem as mais promissoras e, então, analistas humanos decidiriam em quais ações investir. No entanto, alguns fundos foram além e deixaram os próprios aprendizes comprar e vender. Exatamente como tudo isso se deu é um segredo muito bem guardado, mas não deve ser acidente o fato de os machine learners largarem os hedge funds a uma taxa alarmante.

Os modelos não lineares são importantes não só no mercado de ações. Cientistas de todos os lugares usam a regressão linear porque é isso que conhecem, porém os fenômenos que eles estudam costumam ser não lineares, e um perceptron multicamadas pode modelá-los. Os modelos lineares não enxergam transições de fase; as redes neurais as absorvem como uma esponja.

Outro sucesso inicial notável das redes neurais foi aprender a dirigir um carro. Os carros autônomos surgiram para o público em geral com as Darpa Grand Challenges[5] em 2004 e 2005, mas uma década antes, pesquisadores da Universidade Carnegie Mellon já tinham treinado com sucesso um perceptron multicamadas para dirigir um carro detectando a pista em imagens de vídeo e virando o volante apropriadamente. O carro da Carnegie Mellon conseguiu cruzar a América de costa a costa com uma visão muito borrada (30 × 32 pixels),

um cérebro menor do que o de um verme e pouca ajuda do copiloto humano. (O projeto foi chamado de “No Hands Across America”.) Talvez esse não tenha sido o primeiro carro realmente autônomo, mas em comparação não deixou nada a dever à maioria dos motoristas adolescentes.

Atualmente temos tantas aplicações para a backprop que é impossível contar. À medida que sua fama foi crescendo, passamos a conhecer melhor sua história. Descobriu-se que, como ocorre com frequência na ciência, a backprop foi inventada mais de uma vez. Yann LeCun e outros pesquisadores chegaram a ela na França quase ao mesmo tempo que Rumelhart. Um estudo sobre backprop foi rejeitado pela principal conferência de inteligência artificial no início dos anos 1980 porque, de acordo com os examinadores, Minsky e Papert já tinham provado quer os perceptrons não funcionavam. Na verdade, Rumelhart é creditado como inventor da backprop pelo teste de Colombo: Colombo não foi a primeira pessoa a descobrir a América, mas sim a última. Ocorre que Paul Werbos, um aluno de pós-graduação de Harvard, propôs um algoritmo semelhante em sua tese de PhD em 1974. E, por uma grande ironia, Arthur Bryson e Yu-Chi Ho, dois estudiosos da teoria do controle, fizeram o mesmo ainda antes: em 1969, no mesmo ano em que Minsky e Papert publicaram *Perceptrons*! Na verdade, a própria história do machine learning mostra por que precisamos de algoritmos de aprendizado. Se em 1969 existissem algoritmos que encontrassem automaticamente estudos relacionados na literatura científica, eles poderiam ter evitado décadas de tempo perdido e antecipado quem tinha conhecimento de quais descobertas.

Entre as muitas ironias da história do perceptron, talvez a mais triste seja a de que Frank Rosenblatt morreu em um acidente de barco na baía de Chesapeake no ano de 1969 e não viveu para ver o segundo ato de sua criação.

Modelo completo de uma célula

Uma célula viva é um exemplo básico de um sistema não linear. A célula executa todas as suas funções transformando matérias-primas em produtos finais por intermédio de uma complexa teia de reações químicas. Podemos descobrir a estrutura dessa rede usando métodos simbolistas como a dedução inversa, que vimos no último capítulo, mas para construir um modelo completo de uma célula é preciso adotar uma abordagem quantitativa, aprendendo os parâmetros que unem os níveis de expressão de diferentes genes, associando as variáveis

ambientais às variáveis internas, e assim por diante. É difícil porque não há um relacionamento linear simples entre essas quantidades. Em vez disso, a célula mantém sua estabilidade conectando loops de feedback, o que leva a um comportamento muito complexo. A backpropagation é adequada para esse problema graças à sua habilidade de aprender eficientemente funções não lineares. Se tivéssemos um mapa completo dos caminhos metabólicos da célula e observações suficientes de todas as variáveis relevantes, em princípio a backprop poderia aprender um modelo detalhado, com um perceptron multicamadas para prever cada variável de acordo com suas causas imediatas.

No entanto, para o futuro próximo só teremos conhecimento parcial das redes metabólicas das células e poderemos observar apenas uma fração das variáveis que gostaríamos. Aprender modelos úteis apesar das informações ausentes e das inconsistências inevitáveis nas informações disponíveis é uma tarefa para métodos bayesianos, que veremos no Capítulo 6. O mesmo ocorre para a definição de previsões para um paciente específico, com o modelo em mãos: a evidência disponível é necessariamente ruidosa e incompleta e a inferência bayesiana a aproveita o melhor que pode. Se o objetivo for curar o câncer, ajuda o fato de não precisarmos necessariamente entender todos os detalhes de como as células do tumor funcionam, mas saber apenas o suficiente para desativá-las sem danificar as células normais. No Capítulo 6, também veremos como orientar o aprendizado em direção ao objetivo evitando ao mesmo tempo o que não sabemos e não precisamos saber.

Para uso mais imediato, sabemos que poderíamos utilizar a dedução inversa para inferir a estrutura das redes da célula a partir de dados e de conhecimento prévio, mas há uma explosão combinatória de maneiras de aplicá-la e precisamos de uma estratégia. Já que as redes metabólicas foram projetadas pela evolução, talvez simulá-la em nossos algoritmos de aprendizado seja um caminho a seguir. No próximo capítulo, veremos como fazê-lo.

Aprofundando-se no cérebro

Quando a backprop começou a ser usada, os conexionistas tinham a esperança de aprender rapidamente redes cada vez maiores até que elas se aproximassem de cérebros artificiais, respeitando-se os limites do hardware. Não foi isso que ocorreu. Aprender redes com uma única camada oculta era fácil, mas além desse limite as coisas ficaram rapidamente muito difíceis. Redes com algumas

camadas só funcionavam se projetadas cuidadosamente para a aplicação (por exemplo, o reconhecimento de caracteres). Além desse limite a backprop falhava. À medida que adicionamos camadas, o sinal de erro se torna mais difuso, como um rio ramificando-se em afluentes cada vez menores, até chegarmos ao nível de gotas de chuva que não são registradas. Aprender com dúzias ou centenas de camadas ocultas, como no cérebro, permanecia um sonho distante, e em meados dos anos 1990 a empolgação com os perceptrons multicamadas diminuiu. Um pequeno núcleo de conexionistas aguentou firme, mas em geral a atenção dada à área de machine learning mudou de foco. (Examinaremos esses territórios nos Capítulos 6 e 7.)

Hoje, no entanto, o conexionismo ressurgiu. Estamos aprendendo redes mais profundas do que em qualquer outra época e elas estão definindo novos padrões para a visão, o reconhecimento de voz, a descoberta de medicamentos e outras áreas. O novo campo do aprendizado profundo está na primeira página do *New York Times*. Se olharmos com mais cuidado... surpresa: é o velho e confiável mecanismo da backprop ainda de pé. O que mudou? Pouca coisa, dizem os críticos: apenas temos computadores mais velozes e maior volume de dados. Ao que Hinton e outros respondem: exatamente, estávamos certos o tempo todo!

Na verdade, os conexionistas fizeram progressos reais. Um dos protagonistas dessa última mudança na montanha-russa conexionista é um dispositivo modestamente pequeno chamado autoencoder. Um autoencoder é um perceptron multicamadas cuja saída é igual à sua entrada. Entra uma foto de nossa avó e é ela própria que sai – a mesma foto de nossa avó. À primeira vista parece uma ideia tola: que utilidade poderia ter tal dispositivo? O segredo é tornar a camada oculta muito menor do que as camadas de entrada e saída, assim a rede não pode aprender simplesmente a copiar a entrada na camada oculta e esta na saída, caso em que também poderíamos jogar tudo fora. Porém, se a camada oculta for pequena, ocorre algo interessante: a rede é forçada a codificar a entrada em menos bits, para poder ser representada na camada oculta, e depois decodificar esses bits novamente para o tamanho total. Poderia, por exemplo, aprender a codificar uma imagem de um milhão de pixels de nossa avó apenas como a palavra de três caracteres *avó*, ou algum código curto inventado por ela, e aprender simultaneamente a decodificar “avó” para uma imagem de nossa querida vovozinha. Logo, um autoencoder não é diferente de uma ferramenta de compactação de arquivos, com duas vantagens importantes: ele descobre como

compactar coisas por conta própria e, como as redes de Hopfield, pode transformar uma imagem com interferência e distorções em uma clara e limpa.

Os autoencoders foram descobertos nos anos 1980, mas era muito difícil aprender com eles, ainda que tivessem uma única camada oculta. Descobrir como empacotar muitas informações na mesma pequena quantidade de bits é um problema dificílimo (um código para a avó, outro um pouco diferente para o avô, outro para Jennifer Aniston etc.). A paisagem no hiperespaço é irregular demais para fornecer um bom pico; as unidades ocultas precisam aprender o que poderíamos considerar um excesso de funções XOR das entradas. Logo, os autoencoders acabaram não sendo adotados. O truque que levou uma década para ser descoberto é tornar a camada oculta maior do que as camadas de entrada e saída. Como assim? Na verdade, essa é apenas metade do truque: a outra metade é forçar a grande maioria das unidades ocultas a se desativarem em um momento específico. Esse esquema impede que a camada oculta copie a entrada e – o mais importante – torna o aprendizado muito mais fácil. Se permitirmos que diferentes bits representem entradas distintas, estas não terão mais que competir para ativar os mesmos bits. Além disso, agora a rede tem muito mais parâmetros, portanto o hiperespaço em que você está tem muito mais dimensões e há mais maneiras de evitar o que de outra forma seriam máximos locais. Esse é o chamado autoencoder esparso, e ele é um truque limpo.

No entanto, ainda não vimos o aprendizado profundo. A próxima boa ideia é empilhar autoencoders esparsos uns sobre os outros como um club sandwich. A camada oculta do primeiro autoencoder passa a ser a camada de entrada/saída do segundo, e assim por diante. Já que os neurônios são não lineares, cada camada oculta aprende uma representação mais sofisticada da entrada, baseada na anterior. Dado um extenso conjunto de imagens de rostos, o primeiro autoencoder aprende a codificar características locais como ângulos e manchas, o segundo usa essas informações para codificar características faciais como a ponta do nariz ou a íris do olho, o terceiro memoriza narizes e olhos inteiros, e assim por diante. Para concluir, a camada do topo poderia ser um perceptron convencional que aprendesse a reconhecer nossa avó a partir das características gerais fornecidas pela camada abaixo dela – muito mais fácil do que usar informações brutas fornecidas por uma única camada oculta ou tentar executar a backprop em todas as camadas de uma só vez. A rede Google Brain celebrizada pelo *New York Times* é um sanduíche de nove camadas de autoencoders e outros

ingredientes que aprende a reconhecer gatos em vídeos do YouTube. Com um bilhão de conexões, na época ela era a maior rede a ter executado um processo de aprendizado. Não é de surpreender que Andrew Ng, um dos líderes do projeto, também seja um dos principais proponentes da ideia de que a inteligência humana se resume a um único algoritmo e tudo que precisamos fazer é descobri-lo. Ng, cuja afabilidade esconde uma forte ambição, acredita que autoencoders esparsos empilhados podem nos levar mais perto de resolver o problema da inteligência artificial do que qualquer coisa que já vimos.

Os autoencoders empilhados não são o único tipo de aprendiz profundo. Há outro baseado nas máquinas de Boltzmann, e ainda outro – redes neurais convolucionais – baseado em um modelo do córtex visual. No entanto, apesar de seus sucessos notáveis, ainda estão longe de ser como o cérebro. A rede do Google pode reconhecer faces de gatos vistas de frente; os humanos podem reconhecer gatos em qualquer pose e até mesmo quando é difícil distinguir a face. Trata-se de uma rede ainda muito superficial; só três de suas nove camadas são autoencoders. Um perceptron multicamadas é um modelo admissível do cerebelo, a parte do cérebro responsável pelo controle motor de nível inferior, mas o córtex é outra história. Em primeiro lugar, estão faltando as conexões retrógradas necessárias à propagação de erros, além de também ser o local onde a mágica do aprendizado reside. Em seu livro *On Intelligence*, Jeff Hawkins defende o uso de algoritmos de aprendizado baseados na organização do córtex, mas até agora nenhum desses algoritmos conseguiu competir com as redes profundas atuais.

Isso pode mudar à medida que conhecermos melhor o cérebro. Inspirada pelo projeto genoma humano, a nova área da conectômica tenta mapear cada sinapse do cérebro. A União Europeia está investindo um bilhão de euros para construir um modelo completo dele. A iniciativa americana Brain, com 100 milhões de dólares de financiamento somente em 2014, tem metas semelhantes. No entanto, os simbolistas não acreditam que esse caminho leve ao Algoritmo Mestre. Mesmo se pudermos imaginar o cérebro inteiro no nível de sinapses individuais, precisaremos (ironicamente) de algoritmos de machine learning melhores para transformar essas imagens em diagramas de conexões; fazê-lo de forma manual está fora de questão. E o pior é que, ainda que tivéssemos um mapa completo do cérebro, ficaríamos perdidos para descobrir o que ele faz. O sistema nervoso do verme *C. elegans* é composto somente por 302 neurônios e foi totalmente

mapeado em 1986, mas temos apenas um conhecimento fragmentário do que ele faz. Precisamos de conceitos de nível mais alto para entender a confusão dos detalhes de nível inferior, eliminando os que forem específicos do wetware6 ou apenas peculiaridades da evolução. Não construímos aviões aplicando engenharia reversa às penas, e eles não batem asas. Em vez disso, os designs dos aviões são baseados nos princípios da aerodinâmica, aos quais todos os objetos voadores devem obedecer. Ainda não entendemos os princípios de raciocínio análogos.

Talvez a conectômica seja um exagero. Já foi ouvido de conexionistas que a backprop é o Algoritmo Mestre e só precisamos aumentar sua escala. Porém, os simbolistas tratam a ideia com desdém. Eles citam uma longa lista de coisas que os humanos podem fazer, mas as redes neurais não. Veja, por exemplo, o raciocínio lógico. Ele envolve combinar informações que podem nunca ter sido vistas juntas antes. Mary comeu um sapato no almoço? Não, porque Mary é uma pessoa, pessoas só comem o que é comestível e sapatos não são comestíveis. Os sistemas simbólicos não têm dificuldades para fazer isso – eles simplesmente encadeiam as regras relevantes –, mas os perceptrons multicamadas não conseguem fazê-lo; quando terminam o aprendizado, apenas processam a mesma função repetidamente. As redes neurais não são composicionais, e a composicionalidade desempenha um papel importante na cognição humana. Outro grande problema é que os humanos – e modelos simbólicos, como os conjuntos de regras e as árvores de decisão – podem explicar seu raciocínio, enquanto as redes neurais são grandes pilhas de números que ninguém consegue entender.

Porém, se os humanos têm tantas habilidades que seu cérebro não aprendeu ajustando sinapses, de onde elas vêm? A menos que você acredite em mágica, a resposta deve ser a evolução. Quem não acredita no conexionismo e tem coragem para defender suas convicções tem a incumbência de descobrir como a evolução aprendeu tudo que um bebê sabe ao nascer – e quanto mais coisas achar que são inatas, maior será a tarefa. Contudo, se conseguir descobrir como se dá o processo e programar um computador para executá-lo, seria indelicado negar que inventou pelo menos uma das versões do Algoritmo Mestre.

---

1 N.T.: Migração dos setores mais ricos das cidades grandes para cidades menores ou bairros mais afastados que começou nos EUA nos anos 60 e 70 por causa do aumento da violência, do crescimento dos guetos, do aumento de carros e da diversidade racial nos espaços anteriormente frequentados apenas

pela elite majoritariamente WASP (White, Anglo-Saxon and Protestant, ou branco, anglo-saxão e protestante).

2 N.T.: Local optimum é um termo da matemática aplicada e da ciência da computação. O local optimum de um problema de otimização combinatória é uma solução que é ótima dentre um conjunto de soluções vizinhas. O plural é local optima, que traduzi como soluções ótimas.

3 N.T.: Nativo do Nepal que presta serviços de carregador em expedições ao Himalaia. É uma brincadeira do autor.

4 N.T.: Alusão ao personagem de desenho animado Marvin, o Marciano, que estampa camisetas da Nike.

5 N.T.: A Darpa Grand Challenge é uma competição de automóveis com direção automática.

6 N.T.: O termo wetware é usado para descrever a integração dos conceitos da construção física conhecida como “sistema nervoso central” e a construção mental conhecida como “a mente humana”.

capítulo 5

Evolução: o algoritmo de aprendizado da natureza

Robotic Park é uma enorme fábrica de robôs rodeada por vinte e seis mil quilômetros quadrados de selva, espaço urbano e outros ambientes. Ao redor da selva está o mais alto e espesso muro já construído, com postos de sentinelas, holofotes e torres de atiradores. O muro tem duas finalidades: manter os intrusos do lado de fora e os habitantes do parque – milhões de robôs lutando pela sobrevivência e pelo controle da fábrica – do lado de dentro. Os robôs vencedores podem gerar descendentes, e sua reprodução é realizada pela programação dos bancos internos de impressoras 3D. Gradualmente, eles se tornam mais inteligentes, rápidos – e mortais. Robotic Park é controlada pelo exército dos Estados Unidos e sua meta é desenvolver o soldado definitivo.

Essa fábrica ainda não existe, mas algum dia pode existir. Eu a propus como um experimento de raciocínio em um workshop da Darpa alguns anos atrás e um dos oficiais militares presentes disse prosaicamente: “É viável”. Sua voluntariedade pode parecer menos surpreendente se você considerar que o exército já controla um modelo completo de vila afegã no deserto da Califórnia, com a presença de aldeões, para o treinamento de suas tropas, e alguns bilhões de dólares seria um preço pequeno a pagar pelo soldado definitivo.

Os primeiros passos para a construção do Robotic Park já foram dados. No laboratório Creative Machines Lab de Hod Lipson na Universidade Cornell, robôs inacreditavelmente bem modelados estão aprendendo a rastejar e voar, talvez até mesmo enquanto você lê este texto. Um deles se parece com uma torre sinuosa de tijolos de borracha, outro com um helicóptero com asas de libélula, e ainda há outro parecido com um Lego “metamorfo”. Esses robôs não foram

projetados por engenheiros humanos, mas sim criados pela evolução, o mesmo processo que fez surgir a diversidade de vida na Terra. Embora no início eles evoluam dentro de uma simulação computadorizada, uma vez que pareçam suficientemente aptos para agir no mundo real, versões sólidas são fabricadas de modo automático pela impressão 3D. Os robôs ainda não estão prontos para ocupar seu lugar no mundo, mas percorreram um longo caminho desde a sopa primordial de peças simuladas com a qual começaram.

O algoritmo que evoluiu esses robôs foi inventado por Charles Darwin no século 19. Na época, ele não o considerava um algoritmo, em parte porque ainda faltava uma sub-rotina essencial. Quando James Watson e Francis Crick a forneceram em 1953, o palco estava montado para o segundo estágio da evolução: *in silico* em vez de *in vivo* e um bilhão de vezes mais rápido. Seu profeta foi um indivíduo do meio-oeste dos Estados Unidos, de face rosada e sempre sorridente, chamado John Holland.

Algoritmo de Darwin

Como muitos dos primeiros pesquisadores do machine learning, Holland começou trabalhando em redes neurais, mas seus interesses tomaram um rumo diferente quando, enquanto aluno de pós-graduação na Universidade de Michigan, leu a obra clássica de Ronald Fisher *The Genetical Theory of Natural Selection*. Nela, Fisher, que também foi o fundador da estatística moderna, formulou a primeira teoria matemática da evolução. Brilhante como era, Holland percebeu que a teoria de Fisher deixou de fora a essência da evolução. Fisher considerou cada gene isoladamente, mas a adaptabilidade de um organismo é uma função complexa de todos os seus genes. Se estes são independentes, as frequências relativas de suas variantes convergem rapidamente para o ponto de adaptabilidade máxima e permanecem em equilíbrio daí em diante. Porém, se os genes interagem, a evolução – a busca pela adaptabilidade máxima – é muito mais complexa. Com mil genes, cada um com duas variantes, o genoma tem 2^{1000} estados possíveis, e nenhum planeta no universo é remotamente grande ou antigo o suficiente para ter testado todos. Mesmo assim a evolução na Terra conseguiu gerar alguns organismos notavelmente adaptados, e a teoria da seleção natural de Darwin explica como, pelo menos qualitativamente. Holland decidiu transformá-la em um algoritmo.

Mas primeiro era preciso se formar. Cautelosamente, ele escolheu um tópico

mais conservador para sua dissertação – circuitos booleanos com ciclos – e em 1959 obteve o primeiro PhD mundial em ciência da computação. No entanto, seu orientador de tese de doutoramento, Arthur Burks, o encorajou a se interessar por computação evolucionária e o ajudou a conseguir um emprego na universidade em Michigan e a protegê-lo de colegas mais experientes que não achavam que aquilo fosse ciência da computação. O próprio Burks foi mais receptivo porque tinha sido colaborador próximo de John von Neumann, que provou que era factível criar máquinas autorreprodutivas. Na verdade, acabou ficando para ele a tarefa de terminar o trabalho quando von Neumann morreu de câncer em 1957. Foi notável von Neumann conseguir provar que essas máquinas podiam existir, dado o estado primitivo da genética e da ciência da computação na época. Contudo, seu autômato fazia apenas cópias exatas de si mesmo; a evolução de autômatos teve de esperar por Holland.

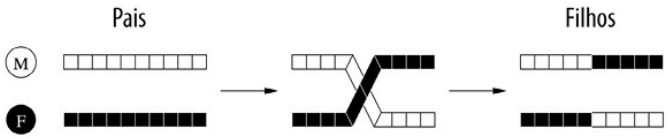
A entrada-chave de um algoritmo genético, como ficou conhecida a criação de Holland, é uma função de adaptabilidade. Dados um programa candidato e algum fim que ele deva atingir, a função de adaptabilidade atribui ao programa uma pontuação numérica que reflete com que excelência ele atende à finalidade. Na seleção natural, é questionável se a adaptabilidade pode ser interpretada dessa forma: embora a adaptabilidade de uma asa para o voo faça sentido intuitivamente, a evolução como um todo não tem um fim conhecido. No entanto, no machine learning é fácil encontrarmos algo como uma função de adaptabilidade. Se precisássemos de um programa que pudesse diagnosticar um paciente, seria melhor que ele diagnosticasse corretamente 60% dos pacientes de nosso banco de dados do que se estivesse correto apenas 55% das vezes e, portanto, uma possível função de adaptabilidade seria a fração de casos diagnosticados corretamente.

Nesse aspecto, os algoritmos genéticos são muito semelhantes à reprodução seletiva. Darwin começa *A origem das espécies* com uma discussão sobre o assunto, como um trampolim para o conceito mais difícil, que era a seleção natural. Todas as plantas e animais domésticos que conhecemos hoje são resultado, geração após geração, da seleção e fecundação dos organismos que melhor serviam aos nossos propósitos: o milho com as maiores espigas, as árvores com os frutos mais doces, as ovelhas com mais lã, os cavalos mais resistentes. Os algoritmos genéticos fazem o mesmo, exceto por reproduzirem programas e não criaturas vivas, e o surgimento de uma geração só demanda

alguns segundos de tempo do computador em vez do tempo de existência de um ser vivo.

A função de adaptabilidade representa o papel que o ser humano desempenha no processo. Porém, a parte mais sutil é a da natureza. Começando com uma população de indivíduos não muito adaptados – possivelmente aleatórios –, o algoritmo genético tem de fornecer variações que possam então ser selecionadas de acordo com a adaptabilidade. Como a natureza faz isso? Darwin não sabia. É aí que entra a parte genética do algoritmo. Da mesma forma que o DNA codifica um organismo como uma sequência de pares de bases, podemos codificar um programa como uma sequência de bits. Em vez de 0 e 1, o alfabeto do DNA tem quatro caracteres – as quatro bases adenina, timina, citosina e guanina –, mas essa é uma diferença superficial. Variações, sejam nas sequências de DNA ou nas sequências de bits, podem ser geradas de muitas maneiras. A abordagem mais simples é a mutação pontual: a mudança de um bit aleatório na sequência de bits ou a alteração de uma única base em uma sequência de DNA. Porém, para Holland, o poder real dos algoritmos genéticos estava em algo mais complicado: no sexo.

Resumida às suas noções básicas (sem risinhos, por favor), a reprodução sexual consiste na troca de material entre cromossomos da mãe e do pai, um processo chamado crossing-over (sobrecruzamento). Ele produz dois novos cromossomos, um composto pelo cromossomo da mãe até o ponto de crossing-over e depois composto pelo do pai, e o outro é o oposto:



Um algoritmo genético funciona imitando esse processo. Em cada geração, ele promove o encontro dos indivíduos mais adaptados, produzindo dois filhos de cada par de pais pelo crossing-over de suas sequências de bits em um ponto aleatório. Após a aplicação de mutações pontuais às novas sequências, é permitido que elas vaguem à vontade em seu mundo virtual. Elas então retornam com uma pontuação de adaptabilidade individual e o processo se repete. Cada geração é mais apta que a anterior, e o processo termina quando a adaptabilidade desejada é alcançada ou o tempo acaba.

Por exemplo, suponhamos que quiséssemos evoluir uma regra para a filtragem

de spam. Se dez mil palavras diferentes aparecessem nos dados de treinamento, cada regra candidata poderia ser representada por uma sequência de vinte mil bits, dois para cada palavra. O primeiro bit correspondente à palavra *free* será igual a 1 se emails contendo *free* forem detectados pela regra, e 0 se não forem. O segundo bit é o oposto: 1 se emails que *não* tiverem *free* forem detectados, e 0 se não forem. Logo, se os dois bits forem iguais a 1, os emails serão detectados pela regra não importando se contêm *free*, e a regra não considerará essa palavra. Por outro lado, se os dois bits forem 0, nenhum email será detectado pela regra, já que um dos bits sempre falhará, e os emails passarão pelo filtro (caramba!). Um email só corresponderá totalmente a uma regra se seu padrão inteiro de palavras presentes e ausentes for permitido por ela. A adaptabilidade da regra seria, digamos, a porcentagem de emails que ela classificar corretamente. Começando com uma população de sequências aleatórias, cada uma representando uma regra com condições aleatórias, agora o algoritmo genético pode evoluir regras cada vez melhores, sobrecruzando e modificando repetidamente as sequências mais aptas em cada geração. Por exemplo, se a população atual incluir as regras *Se o email tiver a palavra* free (grátis)*, então é spam* e *Se o email tiver a palavra* easy (fácil)*, então é spam*, seu sobrecruzamento gerará a regra provavelmente mais adequada *Se o email tiver* free *e* easy*, então é spam*, contanto que o ponto de sobrecruzamento não caia entre os dois bits correspondentes a uma dessas palavras. Ele também gerará a regra *Todos os emails são spam*, que resulta da remoção das duas condições, mas não é provável que essa regra gere muitos descendentes na próxima geração.

Já que nosso objetivo é produzir o melhor filtro de spam que pudermos, em vez de simular fielmente a seleção natural real, podemos trapacear modificando o algoritmo para que ele atenda às nossas necessidades. Uma maneira como os algoritmos genéticos costumam trapacear é permitindo a imortalidade. (Pena não ser possível fazer isso na vida real.) Dessa forma, um indivíduo altamente apto não compete apenas com sua própria geração para se reproduzir, mas também com seus filhos, e então com seus netos, bisnetos, e assim por diante, enquanto continuar sendo um dos indivíduos mais aptos da população. Por outro lado, no mundo real, o melhor que um indivíduo altamente apto pode fazer é passar adiante metade de seus genes para muitos filhos, e provavelmente cada um deles será menos apto devido aos genes herdados de seu outro pai. A imortalidade

evita esse retrocesso e com alguma sorte permite que o algoritmo alcance mais cedo a adaptabilidade desejada. É claro que, já que os humanos mais aptos da história, conforme medido pelo número de descendentes, são gente como Genghis Khan – ancestral de um entre duzentos homens vivos atualmente –, talvez não seja tão ruim que na vida real a imortalidade seja *vetada*.

Se quisermos evoluir um conjunto inteiro de regras de filtragem de spam, não apenas uma, podemos representar um conjunto candidato de *n* regras com uma sequência de *n* × 20.000 bits (20.000 para cada regra, supondo dez mil palavras diferentes nos dados, como antes). Regras contendo 00 para alguma palavra desaparecerão do conjunto de resultados, já que não correspondem a nenhum email, como já vimos. Se um email apresentar correspondência com alguma regra, ele será classificado como spam; caso contrário será válido. Podemos continuar deixando que a adaptabilidade seja a porcentagem de emails classificados corretamente, mas para combater o sobreajuste é provável que tenhamos de subtrair dela uma penalidade proporcional ao número total de condições ativas no conjunto de regras.

Podemos ser ainda mais extravagantes ao permitir que regras de conceitos intermediários evoluam e alterar essas regras na hora da execução. Por exemplo, poderíamos evoluir as regras *Se o email tiver a palavra loan (empréstimo), então é um scam* e *Se o email for um scam (golpe), então é spam*. Já que o resultado de uma regra não é mais sempre um spam, precisamos introduzir bits adicionais nas sequências das regras para representar seus resultados. É claro que o computador não usaria literalmente a palavra *scam (golpe)*; ela surgiu de alguma sequência de bits arbitrária para representar o conceito, mas isso basta para o que queremos. Conjuntos de regras como esse, que Holland chamou de sistemas classificadores, são um dos cavalos de batalha da tribo de machine learning que ele fundou: os evolucionários. Como os perceptrons multicamadas, os sistemas classificadores enfrentam o problema da atribuição de crédito – qual é a adaptabilidade de regras de conceitos intermediários? –, e Holland projetou o chamado algoritmo da brigada do balde (bucket brigade algorithm) para resolvê-lo. No entanto, os sistemas classificadores são muito menos usados do que os perceptrons multicamadas.

Comparados ao modelo simples do livro de Fisher, os algoritmos genéticos são um passo à frente. Darwin se desculpou por sua falta de habilidade matemática, mas se ele tivesse vivido um século depois provavelmente teria, em vez disso,

sentido falta da destreza em programação. Na verdade, é extremamente difícil capturar a seleção natural com um conjunto de equações, mas expressá-la como um algoritmo é outra coisa, e pode responder o que de outra forma seriam perguntas incômodas. Por que as espécies aparecem repentinamente no registro fóssil? Onde está a evidência de que elas evoluíram gradualmente de espécies anteriores? Em 1972, Niles Eldredge e Stephen Jay Gould propuseram que a evolução é composta por uma série de "equilíbrios pontuados", alternando períodos longos de estase com curtas explosões de rápida mudança, como a explosão Cambriana. Isso gerou um debate acalorado, com os críticos da teoria apelidando-a de "evolução por solavancos" e Eldredge e Gould retrucando que o gradualismo é a "evolução arrastada". A experiência obtida com os algoritmos genéticos dá suporte à abordagem dos solavancos. Se você executar um algoritmo genético para cem mil gerações e observar a população a cada mil, provavelmente o gráfico da adaptabilidade contra o tempo parecerá uma escada irregular, com melhorias repentinas seguidas por períodos nivelados que tendem a se tornar mais longos com o tempo. Também não é difícil entender o porquê. Quando o algoritmo alcançar um máximo local de adaptabilidade – um pico na paisagem da adaptabilidade –, ele permanecerá aí por um longo tempo até uma mutação ou um sobrecruzamento com mais sorte levar um indivíduo da inclinação para um pico mais alto, ponto em que esse indivíduo se multiplicará e subirá a ladeira a cada nova geração. E quanto mais alto for o pico atual, mais tempo demorará para que isso ocorra. É claro que a evolução natural é mais complicada do que isso: em primeiro lugar, o ambiente pode mudar, fisicamente ou porque outros organismos evoluíram, e um organismo que estava em um pico de adaptabilidade pode repentinamente estar sob pressão para evoluir mais uma vez. Logo, embora úteis, os algoritmos genéticos atuais estão longe de ter chegado ao fim da história.

O dilema entre explorar e usufruir

Observe quanto os algoritmos genéticos diferem dos perceptrons multicamadas. A backprop acolhe uma única hipótese em determinado momento e a hipótese muda gradualmente até se estabelecer em um local optimum. Os algoritmos genéticos consideram toda uma população de hipóteses a cada etapa e essas hipóteses podem dar grandes saltos de uma geração à outra, graças ao sobrecruzamento. A backprop age deterministicamente após definir os pesos

iniciais com valores baixos aleatórios. Os algoritmos genéticos, por sua vez, são cheios de escolhas aleatórias: qual hipótese permanecerá ativa e será submetida ao sobrecruzamento (com as hipóteses mais adequadas sendo as candidatas mais prováveis), onde devemos cruzar duas sequências, quais bits modificar. A backprop aprende os pesos de uma arquitetura de rede predefinida; redes mais densas são mais flexíveis, mas também mais difíceis de aprender. Os algoritmos genéticos não fazem suposições *a priori* sobre as estruturas que aprenderão, a não ser sobre sua forma geral.

Devido a tudo isso, os algoritmos genéticos têm menos probabilidades do que a backprop de ficarem presos em um local optimum e, em princípio, estão mais propensos a oferecer algo realmente novo. Porém, também são muito mais difíceis de analisar. Como saber se um algoritmo genético levará a algo útil em vez de ficar andando em círculos? A chave é pensar em termos de blocos de construção. Cada subconjunto de bits de uma sequência codifica um bloco de construção útil, e quando sobrecruzamos duas sequências esses blocos se unem em um maior, que por sua vez se torna matéria-prima de uma nova construção. Holland gosta de usar retratos falados para ilustrar o poder dos blocos de construção. Na época anterior aos computadores, um desenhista de retratos falados desenhava rapidamente o retrato de um suspeito a partir de entrevistas com testemunhas oculares, que selecionavam uma boca em um conjunto de tiras de papel demonstrando formatos de boca típicos e faziam o mesmo para os olhos, o nariz, o queixo, e assim por diante. Com apenas dez partes básicas e dez opções para cada parte, esse sistema fornecia dez bilhões de faces diferentes, mais do que o número de pessoas na Terra.

No machine learning, como nas demais áreas da ciência da computação, não há nada melhor do que fazermos esse tipo de explosão combinatória operar a nosso favor em vez de contra nós. O interessante nos algoritmos genéticos é que cada sequência contém implicitamente um número exponencial de blocos de construção, conhecidos como esquemas, e assim a busca é muito mais eficiente do que parece. Isso ocorre porque cada subconjunto de bits da sequência é um esquema, representando alguma combinação possivelmente adequada de propriedades, e uma sequência tem um número exponencial de subconjuntos. Podemos representar um esquema substituindo os bits da sequência que não fazem parte dele por . *Por exemplo, a sequência 110 contém os esquemas* **, **0, *1*, 1**, *10*, *11*, 1*0 e 110. Obteremos um esquema diferente para cada

escolha de bits a serem incluídos; já que temos duas escolhas para cada bit (incluir/não incluir), temos 2^n esquemas. Inversamente, um esquema específico pode ser representado por muitas sequências diferentes em uma população e será avaliado implicitamente sempre que as sequências o forem. Suponhamos que a probabilidade de uma hipótese sobreviver na próxima geração fosse proporcional à sua adaptabilidade. Holland mostrou que, nesse caso, quanto mais próximos os representantes de um esquema de uma geração estiverem da média, mais podemos esperar que eles apareçam na próxima geração. Logo, embora o algoritmo genético manipule sequências explicitamente, ele pesquisa implicitamente o espaço muito maior dos esquemas. Com o tempo, os esquemas mais adaptados passam a dominar a população, portanto, ao contrário de alguém andando em círculos, os algoritmos genéticos encontram o caminho de casa.

Um dos problemas mais importantes do machine learning – e da vida – é o dilema entre explorar e usufruir. Se você encontrou algo que funciona, deve continuar usando-o? Ou seria melhor testar novas alternativas, sabendo que pode ser perda de tempo, mas também levar a uma solução melhor? Você preferiria ser um boiadeiro ou um fazendeiro? Criar uma empresa ou gerenciar uma já existente? Namorar ou não ter compromisso? Uma crise da meia-idade representa o anseio por explorar após muitos anos usufruindo da mesma situação. Em um impulso, você voa até Las Vegas, determinado a apostar as economias de toda uma vida na tentativa de se tornar um milionário. Entra no primeiro cassino e se depara com uma fila de caça-níqueis. O melhor seria o que, em média, desse-lhe o melhor retorno, mas você não sabe qual. É preciso testar cada máquina um número de vezes suficiente para descobrir. Porém, se passar muito tempo fazendo isso, perderá dinheiro em máquinas que não compensam. Inversamente, se você se arriscar e escolher ao acaso uma máquina que pareceu boa nas primeiras jogadas, mas na verdade não é a melhor, perderá seu dinheiro jogando nela pelo resto da noite. Esse é o dilema entre explorar e usufruir. A cada vez que você joga, é preciso escolher entre repetir a melhor jogada que fez até agora, que lhe deu o melhor retorno, ou testar outras jogadas e coletar informações que possam levar a retornos ainda melhores. Com duas máquinas caça-níqueis, Holland mostrou que a estratégia ótima é lançar uma moeda viciada a cada vez, de forma que a moeda se torne exponencialmente mais viciada com o tempo. (No entanto, não me processe se não funcionar para você. Lembre-se de que no final a casa sempre vence.) Quanto melhor parecer uma

máquina caça-níqueis, mais você deve jogar nela, mas nunca desista totalmente da outra, para o caso de ela acabar se mostrando melhor.

Um algoritmo genético seria como o chefe de uma gangue de jogadores, jogando em máquinas caça-níqueis de todos os cassinos da cidade ao mesmo tempo. Dois esquemas competirão entre si se incluírem os mesmos bits e diferirem em pelo menos um, como *10 e* 11, e *n* esquemas em competição são como *n* máquinas caça-níqueis. Cada conjunto de esquemas em competição é um cassino, e o algoritmo genético descobre simultaneamente a máquina vencedora de cada cassino, seguindo a estratégia ótima de jogar nas máquinas que pareçam melhores com uma frequência exponencialmente maior. Muito inteligente.

Em *O guia do mochileiro das galáxias*, uma raça alienígena constrói um enorme supercomputador para responder à pergunta definitiva, e após um longo tempo ele emite “42”. Porém, o computador também destaca que os alienígenas não sabem qual é a pergunta, logo eles constroem um computador ainda maior para encontrá-la. Infelizmente, esse computador – também conhecido como planeta Terra – é destruído para dar caminho a uma autoestrada espacial minutos antes de terminar seu processamento de vários milhões de anos. Não temos como saber qual é a pergunta, mas talvez ela fosse: em qual máquina caça-níqueis você deve jogar?

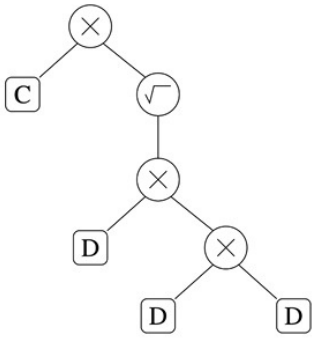
Sobrevivência dos programas mais aptos

Durante as primeiras décadas, a comunidade dos algoritmos genéticos era composta principalmente por John Holland, seus alunos e os alunos desses alunos. Por volta de 1983, o maior problema que os algoritmos genéticos tinham conseguido resolver foi aprender a controlar sistemas de tubulação de gás. Entretanto, mais ou menos na mesma época em que as redes neurais faziam seu retorno, o interesse pela computação evolucionária tomou fôlego. A primeira conferência internacional de algoritmos genéticos aconteceu em Pittsburgh em 1985, e uma explosão cambriana de variantes de algoritmos genéticos estava a caminho. Algumas delas tentaram modelar a evolução com maior proximidade – afinal, o algoritmo genético básico era apenas uma aproximação muito imperfeita – e outras seguiram caminhos bem diferentes, combinando ideias evolucionárias com conceitos da ciência da computação que teriam intrigado Darwin.

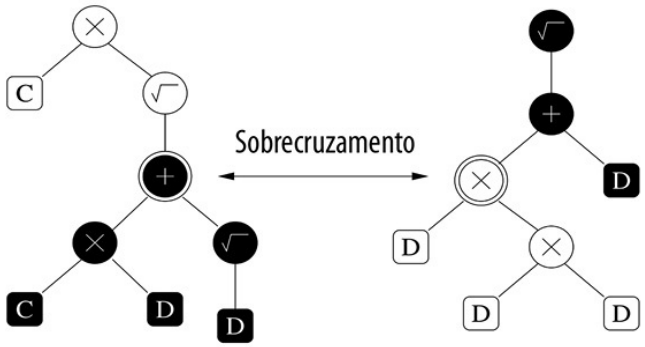
Um dos alunos mais aplicados de Holland foi John Koza. Em 1987, ao voltar

para a Califórnia de uma conferência na Itália, ocorreu-lhe uma ideia luminosa. Em vez de evoluir coisas comparativamente simples como regras *Se... então...* e controladores de tubulação de gás, por que não evoluir programas de computador totalmente desenvolvidos? E se for esse o objetivo, por que usar sequências de bits como representação? Um programa é na verdade uma árvore de chamadas de sub-rotinas, logo é melhor sobrecruzar essas subárvores do que inseri-las em sequências de bits e correr o risco de destruir sub-rotinas perfeitamente boas ao sobrecruzá-las em um ponto aleatório.

Por exemplo, suponhamos que você quisesse evoluir um programa para calcular a duração do ano em um planeta, representada por *T*, a partir de sua distância média do Sol, representada por *D*. De acordo com a terceira lei de Kepler, *T* é a raiz quadrada de *D* ao cubo, vezes uma constante *C* que depende das unidades usadas para o tempo e a distância. Um algoritmo genético deve poder descobrir isso examinando dados de Tycho Brahe sobre movimentos planetários, como fez Kepler. Na abordagem de Koza, *D* e *C* são as folhas da árvore de um programa, e as operações que as combinam, como multiplicar e tirar a raiz quadrada, são os nós internos. A árvore de programa a seguir calcula *T* corretamente:



Em programação genética, como Koza chamou seu método, sobrecruzamos duas árvores de programa trocando aleatoriamente duas de suas subárvores. Por exemplo, o sobrecruzamento dessas duas árvores nos nós realçados gera o programa correto para o cálculo de *T* como um dos filhos:



Podemos medir a adaptabilidade (ou a falta dela) de um programa pela discrepância entre sua saída e a saída correta nos dados de treinamento. Por exemplo, se o programa dissesse que um ano na Terra tem trezentos dias, isso subtrairia sessenta e cinco pontos de sua adequação. Começando com uma população de árvores de programa aleatórias, a programação genética usa o sobrecruzamento, a mutação e a sobrevivência para gerar gradualmente programas melhores até um nível satisfatório.

É claro que o cálculo da duração do ano em um planeta é um problema muito simples, envolvendo apenas multiplicação e raízes quadradas. Em geral, as árvores de programa podem incluir todo o conjunto de estruturas de programação, como instruções *Se... então...*, loops e recursão. Um exemplo mais ilustrativo do que a programação genética é capaz de fazer seria descobrir a sequência de ações que um robô precisa executar para atingir um objetivo. Suponhamos que eu pedisse ao meu "officebot" para me trazer um grampeador do armário que fica no corredor. O robô tem um extenso conjunto de comportamentos disponíveis, como percorrer o corredor, abrir uma porta, pegar um objeto, e assim por diante. Por sua vez, cada um deles pode ser composto por vários subcomportamentos: mover a mão em direção ao objeto ou pegá-lo em vários pontos possíveis, por exemplo. Cada comportamento pode ser ou não executado dependendo dos resultados de comportamentos anteriores, pode ter de ser repetido algumas vezes *etc.* O desafio é montar a estrutura de comportamentos e subcomportamentos certa, junto com os parâmetros de cada comportamento, como, por exemplo, a distância que a mão deve percorrer. A partir dos comportamentos "atômicos" do robô e de suas combinações possíveis, a programação genética pode montar um comportamento complexo que atinja o objetivo desejado. Vários pesquisadores evoluíram estratégias para robôs jogadores de futebol dessa forma.

Uma consequência do sobrecruzamento de árvores de programa e não de

sequências de bits é que os programas resultantes podem ter qualquer tamanho, tornando o aprendizado mais flexível. No entanto, a tendência geral é o aumento com as árvores crescendo para tamanhos cada vez maiores à medida que a evolução se prolonga (também conhecido como “sobrevivência do mais gordo”). Os evolucionários podem tirar algum consolo do fato de que programas escritos por humanos não são diferentes (Microsoft Windows: quarenta e cinco milhões de linhas de código, e só aumenta) e de que os códigos que construímos não permitem uma solução tão simples como a inclusão de penalidade por complexidade na função de adaptabilidade.

O primeiro sucesso da programação genética, em 1995, foi no design de circuitos eletrônicos. A partir de uma pilha de componentes eletrônicos como transistores, resistores e capacitores, o sistema de Koza reinventou um design já patenteado de um filtro passa-baixa, um circuito que pode ser usado para atividades como reforçar os tons graves de uma trilha de dance music. Desde então ele adotou como esporte a reinvenção de dispositivos patenteados, produzindo-os às dúzias. O próximo grande passo veio em 2005, quando o US Patent and Trademark Office forneceu uma patente para um sistema de otimização de fábrica projetado geneticamente. Se o teste de Turing envolvesse enganar um examinador de patentes em vez de uma pessoa em uma conversa, 25 de janeiro de 2005 seria uma data a entrar para os livros de história.

A confiança de Koza se sobressai mesmo em uma área que não é conhecida por gerar profissionais acanhados. Koza vê a programação genética como uma máquina de invenções, um Thomas Edison de silício para o século 21. Ele e outros evolucionários acreditam que a programação genética pode aprender qualquer programa, o que a torna sua aposta na competição pelo Algoritmo Mestre. Em 2004, eles instituíram a premiação anual Humie Awards para reconhecer criações genéticas “que rivalizem com os humanos”; trinta e nove foram premiadas até o momento.

Para que serve o sexo?

Apesar de seus sucessos e dos insights que forneceram sobre questões como o gradualismo *versus* o equilíbrio pontuado, os algoritmos genéticos deixaram um grande mistério sem solução: o papel do sexo na evolução. Os evolucionários dão muita importância ao sobrecruzamento, mas os membros das outras tribos acham que o trabalho não compensa. Nenhum dos resultados teóricos de Holland

mostra que o sobrecruzamento ajuda de verdade; a mutação é suficiente para aumentar exponencialmente a frequência dos esquemas mais aptos na população com o tempo. E intuir por "blocos de construção" é interessante, mas traz rapidamente problemas, mesmo quando a programação genética é usada. À medida que blocos maiores evoluem, o sobrecruzamento também se torna cada vez mais propenso a quebrá-los. Além disso, quando aparece um indivíduo altamente adaptado, seus descendentes tendem a ocupar rapidamente a população, deixando de fora esquemas possivelmente melhores que ficaram presos em indivíduos que no geral eram menos aptos. Isso reduz muito a busca por variações do campeão de adaptabilidade. Os pesquisadores inventaram vários esquemas para preservar a diversidade na população, mas até agora os resultados são inconclusivos. Certamente, os engenheiros usam muito os blocos de construção, porém combiná-los envolve, bem, muita engenharia; não é só uma questão de uni-los de alguma forma conhecida, e não está claro se o sobrecruzamento pode ajudar.

Sem o sexo, os evolucionários ficariam apenas com a mutação para dar poder ao seu mecanismo. Se o tamanho da população for substancialmente maior do que o número de genes, há chances de que cada mutação pontual esteja representada nele, e a pesquisa passa a ser um tipo de escalada de montanha: teste todas as variações de etapa única possíveis, selecione a melhor e repita. (Ou selecione várias entre as melhores, caso em que a abordagem se chama beam search.) Os simbolistas, em particular, usam isso o tempo todo para aprender conjuntos de regras, embora não achem que seja um tipo de evolução. Para não ficar presa no máximo local, a escalada de montanha pode ser aperfeiçoada com a aleatoriedade (mover-se para baixo na montanha com alguma probabilidade) e recomeços aleatórios (após algum tempo, pular para um estado aleatório e continuar a partir daí). Isso é suficiente para a descoberta de boas soluções para problemas; se o benefício de adicionar o sobrecruzamento justifica o custo computacional extra é uma questão que continua em aberto.

Ninguém sabe por que o sexo é difuso na natureza. Várias teorias foram propostas, mas nenhuma foi amplamente aceita. A líder é a hipótese da Rainha Vermelha, popularizada por Matt Ridley no livro de mesmo nome. Como a Rainha Vermelha disse para Alice em *Através do espelho*, "É preciso correr o máximo que puder para continuar no mesmo lugar". Por essa perspectiva, os organismos estão em uma perpétua corrida armamentista com os parasitas, e o

sexo ajuda a manter a população variada, para que nenhum germe possa infectá-la em sua totalidade. Se for essa a resposta, então o sexo é irrelevante para o machine learning, pelo menos até programas aprendidos terem de competir com vírus de computador por tempo e memória do processador. (Danny Hillis alega que introduzir deliberadamente parasitas em coevolução em um algoritmo genético pode ajudá-lo a escapar do máximo local aumentando gradualmente a dificuldade, mas isso é algo que ainda não foi testado.) Christos Papadimitriou e colegas mostraram que o sexo otimiza não a adaptabilidade, mas o que eles chamam de miscibilidade: a habilidade de um gene gerar em média um bom resultado quando combinado com outros genes. Isso pode ser útil quando a função de adaptabilidade não for conhecida ou constante, como na seleção natural, mas no machine learning e na otimização a escalada de montanha tende a se sair melhor.

Os problemas da programação genética não param aí. Na verdade, mesmo seus sucessos podem não ser tão genéticos como os evolucionários gostariam. Veja o design de circuitos, que foi um sucesso emblemático da programação genética. Em geral, até designs relativamente simples requerem muito trabalho de pesquisa, e não está claro quanto os resultados devem à força física em vez de à esperteza genética. Para amenizar o aumento da crítica, Koza incluiu em seu livro de 1992, *Genetic Programming*, experimentos mostrando que a programação genética se saía melhor que candidatos da geração aleatória em problemas de síntese de circuitos booleanos, mas a margem de vitória era pequena. Em seguida, na ICML (International Conference on Machine Learning – Conferência Internacional de Machine Learning) de 1995 em Lake Tahoe, Califórnia, Kevin Lang publicou um artigo que mostrava que a escalada de montanha vencia a programação genética nos mesmos problemas, com frequência por uma grande diferença. Koza e outros evolucionários tinham repetidamente tentado publicar artigos na ICML, uma das principais vitrines da área, mas para sua crescente frustração continuaram sendo recusados devido à validação empírica insuficiente. Decepcionado por seus artigos serem rejeitados, ver o artigo de Lang fez Koza perder a paciência. Ele produziu rapidamente um artigo de vinte e três páginas, no formato de duas colunas da ICML, refutando as conclusões de Lang e acusando os examinadores da ICML de conduta científica imprópria. Então, colocou uma cópia em cada assento do auditório da conferência. Dependendo de nosso ponto de vista, podemos considerar que a

gota d’água foi o artigo de Lang ou a resposta de Koza; seja como for, o incidente em Lake Tahoe marcou a separação final entre os evolucionários e o resto da comunidade de machine learning, com os evolucionários se retirando. Os programadores genéticos começaram a fazer sua própria conferência, que se fundiu à de algoritmos genéticos para formar a Gecco, Genetic and Evolutionary Computing Conference. Quanto à corrente de machine learning em voga na época, eles praticamente a esqueceram. Um triste desenlace, porém não foi a primeira vez na história que o sexo era o culpado de uma separação.

O sexo pode não ter sido bem-sucedido no machine learning, mas, como consolo, desempenhou de outras formas um papel predominante na evolução tecnológica. A pornografia foi a não reconhecida “aplicação matadora” da World Wide Web, sem mencionar da imprensa, da fotografia e do mercado de vídeo antes dela. O vibrador foi o primeiro dispositivo elétrico portátil, antecedendo o telefone celular em um século. As lambretas ganharam as ruas na Europa pós-guerra, particularmente na Itália, porque permitiam que casais jovens se distanciassem de suas famílias. Facilitar encontros amorosos certamente foi uma das “aplicações matadoras” do fogo quando o *Homo erectus* o descobriu um milhão de anos atrás; da mesma forma, uma das principais condutoras do aumento do realismo em robôs humanoides será a indústria de robôs sexuais. O sexo parece ser o fim, e não o meio, que justificará a evolução tecnológica.

Natureza evolucionista

Evolucionários e conexionistas têm algo importante em comum: ambos projetam algoritmos de aprendizado inspirados na natureza. Porém, a partir daí seguem caminhos diferentes. Os evolucionários se concentram na estrutura do aprendizado; para eles, ajustar uma estrutura evoluída otimizando parâmetros é secundário. Por outro lado, os conexionistas preferem pegar uma estrutura simples e codificada à mão com muitas conexões e deixar o aprendizado de pesos fazer o trabalho. Essa é a versão do machine learning para a controvérsia do criacionismo *versus* evolucionismo, e há bons argumentos dos dois lados.

Por um lado, a evolução produziu muitas coisas surpreendentes, mas nenhuma tanto como você. Com ou sem o sobrecruzamento, a estrutura em evolução é parte essencial do Algoritmo Mestre. O cérebro pode aprender qualquer coisa, mas não pode evoluir um cérebro. Se conhecêssemos toda a sua arquitetura, poderíamos implementá-la em hardware, porém estamos longe disso; obter ajuda

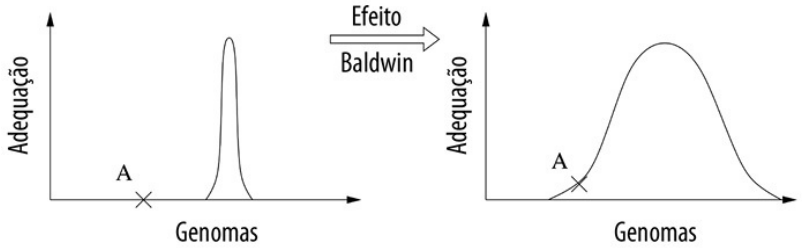
da evolução simulada em computador é fácil. Além disso, também queremos evoluir os cérebros dos robôs, sistemas com sensores arbitrários e a superinteligência artificial. Não há razão para pararmos no design do cérebro humano se houver designs melhores para essas tarefas. Por outra perspectiva, a evolução é muito lenta. A vida inteira de um organismo gera uma única informação sobre seu genoma: sua adaptabilidade, refletida no número de filhos. Esse é um desperdício colossal de informações, que o aprendizado neural evita adquirindo-as quando são usadas (por assim dizer). Como conexionistas do porte de Geoff Hinton gostam de ressaltar, não há vantagem em carregar no genoma informações que não possamos obter prontamente a partir dos sentidos. Quando um recém-nascido abre os olhos, o mundo das imagens inunda sua mente; o cérebro só tem de organizá-lo. No entanto, o que precisa ser especificado no genoma é a arquitetura da máquina que faz a organização.

Como no debate criacionismo *versus* evolucionismo, nenhum dos lados tem a resposta completa; a chave é descobrir como combiná-los. O Algoritmo Mestre não é a programação genética nem a backprop, mas tem de incluir os elementos básicos de ambas: aprendizado da estrutura e aprendizado dos pesos. Pelo ponto de vista convencional, a natureza faz sua parte primeiro – evolui um cérebro – e então a evolução assume, preenchendo o cérebro com informações. Podemos reproduzir isso facilmente em algoritmos de aprendizado. Primeiro, aprenda a estrutura da rede, usando (por exemplo) a escalada de montanha para decidir quais neurônios se conectam com quais outros neurônios: tente adicionar cada nova conexão possível à rede, mantenha a que traga mais melhorias ao desempenho e repita o processo. Em seguida, aprenda os pesos das conexões empregando a backprop, e seu cérebro novinho em folha estará pronto para ser usado.

Porém, há uma sutileza importante tanto na evolução natural quanto na artificial. Precisamos aprender os pesos de cada estrutura candidata ao longo do percurso, não apenas da final, para ver como ela se sai na luta pela vida (no caso natural) ou nos dados de treinamento (no caso artificial). A estrutura que queremos selecionar a cada etapa é a que se sair melhor após o aprendizado dos pesos, não antes. Logo, na verdade, a natureza não vem antes da evolução; em vez disso, elas se alternam, com cada rodada de aprendizado “evolutivo” preparando o terreno para a próxima rodada de aprendizado “criacionista” e vice-versa. A natureza se desenvolve de acordo com as etapas evolutivas pelas

quais passa. O crescimento evolutivo das áreas associativas do córtex se baseia no aprendizado neural das áreas sensoriais, sem o qual ele seria inútil. Os gansinhos seguem sua mãe por toda parte (comportamento evoluído), mas para isso eles precisam reconhecê-la (habilidade aprendida). Se você foi o primeiro ser que os gansinhos viram ao nascer, eles o seguirão, como Konrad Lorenz mostrou de maneira memorável. O cérebro do recém-nascido já codifica características do ambiente, mas não explicitamente; a evolução o otimizou para extrair essas características da entrada esperada. Da mesma forma, em um algoritmo que aprenda iterativamente tanto a estrutura quanto os pesos, cada nova estrutura será implicitamente uma função dos pesos aprendidos em etapas anteriores.

De todos os genomas possíveis, poucos correspondem a organismos viáveis. Logo, o cenário típico da adaptabilidade é composto por vastas planícies com picos altos ocasionais, o que dificulta muito a evolução. Se você andar pelo Kansas vendado, não saberá qual caminho tomar para chegar às Montanhas Rochosas e perambulará por muito tempo antes de chegar ao sopé e começar a escalada. Porém, se combinar a evolução com o aprendizado neural, algo interessante ocorrerá. Se estiver em terreno plano, mas não muito longe do sopé, o aprendizado neural o levará lá, e quanto mais perto você estiver do sopé, maior a probabilidade de ele o ajudar. É como ser capaz de vasculhar o horizonte: isso não o ajudará em Wichita, mas em Denver você verá as Montanhas Rochosas à distância e tomará o caminho certo. Agora Denver parece mais conveniente do que quando você estava vendado. O efeito final é a ampliação dos picos de adaptabilidade, o que torna possível encontrar o caminho até eles a partir de locais antes muito difíceis, como o ponto A neste gráfico:



Em biologia, isso se chama efeito Baldwin, em homenagem a J. M. Baldwin, que o propôs em 1896. Na evolução baldwiniana, os comportamentos aprendidos antes se tornam geneticamente programados depois. Se mamíferos da família dos cães aprenderem a nadar, eles terão mais chances de evoluir para focas – como o fizeram – do que se se afogarem. Logo, o aprendizado individual pode

influenciar a evolução sem recorrer ao lamarquismo. Geoff Hinton e Steven Nowlan demonstraram o efeito Baldwin no machine learning usando algoritmos genéticos para evoluir a estrutura da rede neural e observando que a adaptabilidade só aumentou com o tempo quando o aprendizado individual foi permitido.

Quem aprender mais rápido vence

A evolução procura boas estruturas, e o aprendizado neural as preenche: essa combinação é a mais fácil das etapas que executaremos para chegar ao Algoritmo Mestre. Isso pode surpreender quem está familiarizado com as intermináveis reviravoltas da controvérsia entre natureza e evolução, que ocorre há 2.500 anos e continua forte. No entanto, ver a vida pela perspectiva de um computador explica muita coisa. A "natureza" para um computador é o programa que ele executa e a "evolução" é composta pelos dados recebidos. É claramente um absurdo perguntar qual é mais importante; não há saída sem o programa e os dados, e a saída não é, por exemplo, 60% proveniente do programa e 40% dos dados. Esse é o tipo de pensamento linear contra o qual a familiaridade com o machine learning o imunizará.

Por outro lado, você deve estar se perguntando por que não terminamos aqui. É certo que se tivéssemos combinado os dois algoritmos mestres da natureza, a evolução e o cérebro, isso seria tudo que poderíamos alcançar. Infelizmente, o que temos até agora é apenas uma imagem rudimentar de como a natureza aprende, boa o suficiente para muitas aplicações, mas ainda uma sombra pálida do que seria o ideal. Por exemplo, o desenvolvimento do embrião é uma parte crucial da vida, porém não há nada semelhante no machine learning: o "organismo" depende muito do genoma e podemos estar perdendo algo importante aí. Outra razão é que não ficaríamos satisfeitos mesmo se tivéssemos descoberto todas as nuances de como a natureza funciona. Em primeiro lugar, ela é muito lenta. A evolução leva bilhões de anos para aprender e o cérebro leva toda uma existência. A cultura é melhor: posso incluir o aprendizado de uma existência em um livro e você o lerá em algumas horas. Contudo, os algoritmos de aprendizado têm de poder aprender em minutos ou segundos. Quem aprender mais rápido vence, seja o efeito Baldwin acelerando a evolução, a comunicação verbal acelerando o aprendizado humano, ou computadores descobrindo padrões à velocidade da luz. O machine learning é o último capítulo da corrida

armamentista da vida na Terra, e um hardware mais veloz é apenas metade da equação. A outra metade é um software mais inteligente.

Acima de tudo, o objetivo do machine learning é encontrar o melhor algoritmo de aprendizado possível, por qualquer meio disponível, e a evolução e o cérebro não podem fornecê-lo. Os produtos da evolução têm muitas deficiências óbvias. Por exemplo, o nervo óptico dos mamíferos está conectado à parte frontal da retina, não à parte de trás, criando um ponto cego desnecessário – e notório – próximo à fóvea, a área de visão mais nítida. A biologia molecular das células vivas é tão confusa que com frequência os biólogos moleculares brincam que só pessoas que não sabem nada sobre o assunto acreditam em design inteligente. A arquitetura do cérebro pode ter falhas semelhantes – o cérebro tem várias restrições que os computadores não têm, como memória de curto prazo muito limitada –, e não há razão para ficarmos presos a elas. Além disso, sabemos de muitas situações em que os humanos parecem fazer continuamente o que é errado, como Daniel Kahneman ilustra com detalhes em seu livro *Rápido e devagar: duas formas de pensar*.

Ao contrário dos conexionistas e dos evolucionários, os simbolistas e os bayesianos não acreditam em emulação da natureza. Eles querem descobrir a partir de princípios básicos o que os aprendizes devem fazer – e isso inclui a nós, humanos. Se quisermos aprender a diagnosticar o câncer, por exemplo, não é suficiente dizer "é assim que a natureza aprende; façamos o mesmo". Há muita coisa em jogo. Erros custam vidas. Os médicos devem fazer diagnósticos da maneira mais precisa que puderem, com métodos semelhantes aos que os matemáticos usam para provar teoremas, ou se aproximar disso o máximo que puderem, dado que raramente é possível ser tão rigoroso. Eles precisam avaliar as evidências para reduzir as chances de darem um diagnóstico errado; ou serem mais exatos, para que quanto mais danoso for um erro, menos probabilidades tenham de cometê-lo. (Por exemplo, não conseguir encontrar um tumor que existe é potencialmente muito pior do que inferir um que não exista.) É preciso tomar decisões *ótimas* e não apenas decisões que pareçam boas.

Esse é o exemplo de uma tensão que percorre quase todas as áreas da ciência e da filosofia: a divisão entre teorias descritivas e normativas, entre "é assim que acontece" e "é assim que deve ser". No entanto, os simbolistas e os bayesianos gostam de ressaltar que descobrir como devemos aprender também pode nos ajudar a entender como na verdade aprendemos porque, presumivelmente, as

duas atividades não estão totalmente relacionadas – longe disso. Em particular, comportamentos que são importantes para a sobrevivência e tiveram um longo tempo para evoluir não devem estar longe do nível ótimo. Não somos muito bons em responder perguntas escritas sobre probabilidades, mas somos muito bons em selecionar instantaneamente movimentos de braços e mãos para alcançar um alvo. Muitos psicólogos têm usado modelos simbolistas e bayesianos para explicar aspectos do comportamento humano. Os simbolistas dominaram as primeiras décadas da psicologia cognitiva. Nos anos 1980 e 1990, os conexionistas imperaram, mas agora os bayesianos estão em ascensão.

Para os problemas mais difíceis – aqueles que gostaríamos de resolver, mas não conseguimos, como a cura do câncer – é provável que abordagens puramente inspiradas pela natureza apresentem poucas informações para serem bem-sucedidas, mesmo com o uso de quantidades massivas de dados. Podemos em princípio aprender um modelo completo das redes metabólicas de uma célula pela combinação de pesquisa estrutural, com ou sem sobrecruzamento, e aprendizado de parâmetros com a backpropagation, mas há um número excessivo de soluções ótimas insatisfatórias às quais não devemos nos prender. Precisamos raciocinar com amostras maiores, montando-as e desmontando-as conforme necessário e usando a dedução inversa para preencher as lacunas. E precisamos que nosso aprendizado seja guiado pelo objetivo de encontrar um diagnóstico ótimo para o câncer e descobrir os melhores medicamentos para curá-lo.

O aprendizado ótimo é o objetivo central dos bayesianos e eles não têm dúvidas de que descobriram como alcançá-lo. Por favor, me acompanhem...

capítulo 6

Na igreja do reverendo Bayes

A silhueta escura da catedral se ergue na noite. Luz emana de seus vitrais, projetando complexas equações nas ruas e prédios externos. Ao se aproximar, você ouve cantos sendo entoados do lado de dentro. Parece ser latim, ou talvez matemática, mas o peixe Babel1 em seu ouvido os traduz para o português: “Gire a manivela! Gire a manivela!”. Assim que você entra, o canto se dissolve em um “Aaaah!” de satisfação e em um murmúrio dizendo “*A posteriori*! *A posteriori*!”. Você examina a multidão. Uma enorme tábua de pedra se estende sobre o altar com uma fórmula gravada em letras de trezentos centímetros: $P(A|B) = P(A)\ P(B|A) / P(B)$ Enquanto você fixa os olhos na inscrição sem compreendê-la, seu Google Glass pisca para ajudá-lo: “Teorema de Bayes”. Agora a multidão começa a cantar “Mais dados! Mais dados!”. Vítimas de sacrifício enfileiradas estão sendo implacavelmente empurradas em direção ao altar. Subitamente, você percebe que está no meio da fila – tarde demais. À medida que a manivela vai surgindo acima, você grita: “Não! Não quero ser um ponto de dados! Deixem-me iiiir!”.

Você acorda suando frio. Em seu colo está um livro chamado *O Algoritmo Mestre*. Tendo se livrado do pesadelo, você retoma a leitura de onde a havia interrompido.

O teorema que controla o mundo

O caminho para o aprendizado ótimo começa com uma fórmula da qual muitas pessoas já ouviram falar: o teorema de Bayes. Porém, aqui o veremos sob uma perspectiva totalmente nova e constataremos que ele é bem mais poderoso do que é possível notar em seu uso diário. Em sua essência, o teorema de Bayes é apenas uma regra simples para a atualização de nosso nível de crença em uma

hipótese quando obtemos novas evidências: se as evidências confirmarem a hipótese, a probabilidade de ela estar correta será maior; caso contrário, as chances serão menores. Por exemplo, se você fizer um teste de AIDS e o resultado for positivo, a probabilidade de que tenha a doença aumentará. As coisas ficam mais interessantes quando temos muitas evidências, como os resultados de vários testes. Para combinar todos os resultados sem causar uma explosão combinatória, precisamos fazer suposições simplificadoras. E tudo pode ficar ainda mais interessante se considerarmos muitas hipóteses ao mesmo tempo, como todos os diferentes diagnósticos possíveis de um paciente. Pode ser complicado calcular a probabilidade de existência de cada doença a partir dos sintomas do paciente em um período de tempo razoável. Quando soubermos como fazer tudo isso, estaremos prontos para aprender pelo método bayesiano. Para os bayesianos, aprender é "apenas" outra aplicação do teorema de Bayes, com modelos completos como hipóteses e dados como evidências: à medida que temos mais dados, alguns modelos se tornam mais prováveis e outros menos prováveis, até idealmente um se destacar como o claro vencedor. Os bayesianos inventaram tipos de modelos muito inteligentes. Comecemos então.

Thomas Bayes foi um padre inglês do século 18 que, sem perceber, se tornou o ponto central de uma nova religião. Você deve estar se perguntando como algo assim acontece, mas lembre-se de que também ocorreu com Jesus: o cristianismo, como o conhecemos, foi inventado por São Paulo, enquanto Jesus se tornou o ponto central da fé judaica. Da mesma forma, o bayesianismo, como o conhecemos, foi inventado por Pierre-Simon de Laplace, um francês que nasceu cinco décadas após Bayes. Bayes foi o pregador que descreveu pela primeira vez uma nova maneira de pensar sobre a probabilidade, mas foi Laplace que converteu esses insights no teorema que ostenta o nome de Bayes.

Um dos maiores matemáticos de todos os tempos, talvez Laplace seja mais conhecido por sua visão do determinismo newtoniano: Dada por um instante uma inteligência que pudesse compreender todas as forças pelas quais a natureza é animada e a respectiva situação dos seres que a compõem – uma inteligência suficientemente vasta para submeter estes dados à análise –, ela abrangeria na mesma fórmula os movimentos dos maiores corpos do universo e os do mais leve átomo. Para ela, nada seria incerto, e o futuro, tal como o passado, estaria presente a seus olhos.

Isso é irônico, já que Laplace também foi o pai da teoria das probabilidades,

que ele acreditava ser apenas o bom senso reduzido a cálculos. No âmago de seus experimentos em probabilidade estava a preocupação com a pergunta de Hume. Por exemplo, como sabemos se o Sol nascerá amanhã? Ele tem agido assim todos os dias até hoje, mas isso não garante que continuará a fazê-lo. A resposta de Laplace tinha duas partes. A primeira é o que agora chamamos de princípio da indiferença, ou princípio de razão insuficiente. Acordamos um dia – digamos, no começo de tudo, que para Laplace foi cerca de cinco mil anos atrás – e, após uma bela tarde, vemos o Sol se pôr. Ele voltará? Nunca tínhamos visto o Sol nascer e não há uma razão em particular para acreditarmos que isso ocorrerá ou que não ocorrerá novamente. Logo, devemos considerar que os dois cenários têm probabilidades iguais de ocorrer e a probabilidade de o Sol nascer de novo é de 50%. Porém, Laplace não parou aí; se o passado pode servir de guia para o futuro, a cada dia que o Sol nascer deve aumentar nossa certeza de que ele continuará a fazê-lo. Após cinco mil anos, a probabilidade de que o Sol nasça mais uma vez amanhã deve estar muito próxima de um, mas não exatamente nesse ponto, já que não podemos ter certeza absoluta. A partir desse teste de raciocínio, Laplace derivou a chamada regra de sucessão, que estima a probabilidade de o Sol nascer de novo após ter nascido n vezes, como $(n + 1) / (n + 2)$. Quando $n = 0$, esse valor representa apenas 1/2; e à medida que n aumenta, o mesmo ocorre com a probabilidade, se aproximando de 1 quando n chega perto do infinito.

Essa regra provém de um princípio mais geral. Suponhamos que você acordasse no meio da noite em um planeta estranho. Ainda que tudo que consiga ver seja o céu estrelado, tem razões para acreditar que o Sol nascerá em algum momento, porque a maioria dos planetas gira do redor de si próprios e de seu Sol. Logo, sua estimativa quanto à probabilidade correspondente deve ser maior que 1/2 (digamos, 2/3). Chamamos isso de *probabilidade a priori* de que o Sol nascerá, já que ela antecede a descoberta de qualquer evidência. Ela não é baseada na contagem do número de vezes que o Sol nasceu neste planeta no passado, porque você não estava lá para ver; em vez disso, reflete suas crenças *a priori* sobre o que ocorrerá, de acordo com seu conhecimento geral do universo. Porém, agora as estrelas começaram a desaparecer, portanto sua certeza de que o Sol realmente nascerá neste planeta aumentou, com base em sua experiência na Terra. Ela passou a ser uma *probabilidade a posteriori*, porque surge após a ocorrência de alguma evidência. O céu começa a clarear e a probabilidade *a*

posteriori passa para outro patamar. Por fim, uma parte do disco luminoso do Sol aparece acima do horizonte e talvez atinja “a torre do Sultão com um dardo de luz”, como no verso de abertura do *Rubaiyat*. A menos que você esteja tendo visões, agora é certo que o Sol nascerá.

A pergunta crucial é como a probabilidade *a posteriori* deve evoluir com o surgimento de mais evidências. A resposta é dada pelo teorema de Bayes. Podemos considerá-lo em termos de causa e efeito. O nascer do Sol faz as estrelas sumirem e o céu clarear, mas esta última ocorrência é uma evidência mais forte do nascer do dia, já que as estrelas podem sumir no meio da noite, por exemplo, devido à chegada de um nevoeiro. Portanto, a probabilidade de ocorrência do nascer do Sol deve aumentar mais após vermos o céu clarear do que se virmos as estrelas sumirem. Em notação matemática, dizemos que *P*(*nascer-do-sol* | *céu-claro*), a probabilidade de ocorrer o nascer do Sol dado que o céu está clareando, é maior que *P*(*nascer-do-sol* | *desaparecimento-das-estrelas*), sua probabilidade condicional se as estrelas estiverem sumindo. De acordo com o teorema de Bayes, quanto maior for a probabilidade de o efeito ocorrer dada a causa, maior a probabilidade da causa ter ocorrido dado o efeito: se *P*(*céu-claro* | *nascer-do-sol*) for maior que *P*(*desaparecimento-das-estrelas* | *nascer-do-sol*), talvez porque alguns planetas estejam tão distantes de seu sol que as estrelas continuem a brilhar após o amanhecer, então *P*(*nascer-do-sol* | *céu-claro*) também será maior que *P*(*nascer-do-sol* | *desaparecimento-das-estrelas*).

No entanto, não acaba aí. Se observarmos um efeito que ocorreria mesmo sem a causa, certamente essa não será uma evidência muito forte de a causa estar presente. O teorema de Bayes incorpora essa hipótese dizendo que *P*(*causa* | *efeito*) diminui com *P*(*efeito*), a probabilidade *a priori* do efeito (isto é, sua probabilidade na ausência de qualquer conhecimento das causas). Para concluir, se outros fatores forem iguais, quanto maior for a probabilidade *a priori* de uma causa, maior deve ser sua probabilidade *a posteriori*. Se juntarmos tudo isso, o teorema de Bayes diz que *P*(*causa* | *efeito*) = *P*(*causa*) × *P*(*efeito* | *causa*) / *P*(*efeito*).

Substitua *causa* por *A* e *efeito* por *B*, omita o sinal de multiplicação para simplificar e terá a fórmula de trezentos centímetros da catedral.

É claro que essa é apenas uma declaração do teorema e não uma prova. Porém, a prova é surpreendentemente simples. Podemos ilustrá-la com um exemplo do

diagnóstico médico, uma das “aplicações matadoras” da inferência bayesiana. Suponhamos que você fosse médico e tivesse feito o diagnóstico de cem pacientes no mês passado. Quatorze tinham gripe, vinte tinham febre e onze tinham ambas. A probabilidade condicional de ocorrer febre causada pela gripe é, portanto, de onze em quatorze, ou 11/14. O condicionamento reduz o tamanho do universo considerado, nesse caso, de todos os pacientes a apenas pacientes gripados. No universo de todos os pacientes, a probabilidade de ocorrência de febre é de 20/100; no universo de pacientes gripados é de 11/14. A probabilidade de um paciente ter gripe *e* febre é igual à fração de pacientes que têm gripe vezes a fração *dos* que têm febre: $P(gripe, febre) = P(gripe) \times P(febre \mid gripe) = 14/100 \times 11/14 = 11/100$. No entanto, também poderíamos ter feito isso de maneira inversa: $P(gripe, febre) = P(febre) \times P(gripe \mid febre)$. Logo, já que as duas equações são iguais a $P(gripe,febre)$, $P(febre) \times P(gripe \mid febre) = P(gripe) \times P(febre \mid gripe)$. Divida os dois lados por $P(febre)$ e obterá $P(gripe \mid febre) = P(gripe) \times P(febre \mid gripe) / P(febre)$. É isso! Esse é o teorema de Bayes, com a gripe como causa e a febre como efeito.

Acontece que os humanos não são muito bons em inferência bayesiana, pelo menos quando o raciocínio verbal está envolvido. O problema é que tendemos a negligenciar a probabilidade *a priori* da causa. Se após fazer o teste de HIV o resultado fosse positivo e o teste só fornecesse 1% de falsos positivos, você entraria em pânico? À primeira vista, parece que agora suas chances de estar com AIDS são de 99%. Porém, ficaremos calmos e aplicaremos o teorema de Bayes passo a passo: $P(HIV \mid positivo) = P(HIV) \times P(positivo \mid HIV) / P(positivo)$. $P(HIV)$ é a prevalência do HIV na população geral, que é de cerca de 0,3% nos Estados Unidos. $P(positivo)$ é a probabilidade de que o teste dê positivo não importando se você tem ou não AIDS; digamos que seja de 1%. Logo, $P(HIV \mid positivo) = 0{,}003 \times 0{,}99 / 0{,}01 = 0{,}297$. Muito diferente de 0,99! A razão é que o HIV é raro na população geral. O teste ter dado positivo aumenta suas chances de ter AIDS em duas ordens de magnitude, mas isso não chega nem à metade das chances. Se seu teste de HIV der positivo, o melhor a fazer é ficar calmo e fazer um teste definitivo. É provável que você esteja bem.

O teorema de Bayes é útil porque geralmente o que conhecemos é a probabilidade dos efeitos dadas as causas, mas o que queremos saber é a probabilidade das causas dados os efeitos. Por exemplo, sabemos que uma porcentagem de pacientes gripados tem febre, todavia o que queremos saber

realmente é a probabilidade de um paciente com febre estar gripado. O teorema de Bayes nos permite passar de um ao outro. No entanto, sua relevância vai muito além disso. Para os bayesianos, essa fórmula de aparência inocente é o $F = ma$ do machine learning, a base a partir da qual um vasto número de resultados e aplicações flui. E qualquer que seja o Algoritmo Mestre, ele deve ser "apenas" uma implementação computacional do teorema de Bayes. Coloquei *apenas* entre aspas porque implementar o teorema de Bayes em um computador pode ser terrivelmente difícil para todos os problemas, exceto os mais simples, por razões que veremos em breve.

O uso do teorema de Bayes como base em estatística e no machine learning é frustrante não só pela dificuldade computacional, mas também por uma extrema controvérsia. Não é vergonhoso querer saber por que, afinal ele não é consequência direta da noção de probabilidade condicional, como vimos no exemplo da gripe? Na verdade, o problema não é com a fórmula propriamente dita. A controvérsia está em como os bayesianos obtêm as probabilidades que são usadas nela e o que essas probabilidades significam. Para a maioria dos estatísticos, a única maneira legítima de estimar probabilidades é contando a frequência com que os eventos correspondentes ocorrem. Por exemplo, a probabilidade de ocorrer febre é de 0,2 porque vinte entre cem pacientes observados a tiveram. Essa é a interpretação "frequentista" da probabilidade, e a escola de pensamento dominante da estatística derivou seu nome dela. Porém, observe que no exemplo do nascer do Sol, e no princípio da indiferença de Laplace, fizemos algo diferente: extraímos uma probabilidade do nada. O que justifica assumir *a priori* que a probabilidade de o Sol nascer é de 1/2, 2/3 ou qualquer que seja ela? A resposta dos bayesianos é que uma probabilidade não é uma frequência, mas sim um grau subjetivo de crença. Logo, nós é que resolvemos o que fazer dela, e tudo que a inferência bayesiana nos permite fazer é atualizar nossas crenças *a priori* com novas evidências para obter nossas crenças *a posteriori* (o que também é conhecido como "girar a manivela bayesiana"). A devoção dos bayesianos a essa ideia é quase religiosa, o bastante para aguentar duzentos anos de ataques, que ainda não terminaram. E com o surgimento de computadores com poder suficiente para executar a inferência bayesiana, e os conjuntos de dados massivos que a acompanham, eles estão começando a se sobressair.

Todos os modelos estão errados, mas alguns são úteis

Um médico não faz um diagnóstico de gripe baseado apenas em se alguém tem febre; ele considera todo um conjunto de sintomas, inclusive se a pessoa está com tosse, garganta inflamada, nariz escorrendo, dor de cabeça, calafrios, e assim por diante. Logo, o que precisamos calcular é $P(gripe \mid febre, tosse, garganta\ inflamada, nariz\ escorrendo, dor\ de\ cabeça, calafrios, ...)$. Pelo teorema de Bayes, sabemos que isso é proporcional a $P(febre, tosse, garganta\ inflamada, nariz\ escorrendo, dor\ de\ cabeça, calafrios, ... \mid gripe)$. Porém, agora nos deparamos com um problema. Como estimar essa probabilidade? Se cada sintoma é uma variável booleana (você o tem ou não) e o médico considera n sintomas, um paciente poderia ter 2^n combinações possíveis de sintomas. Se tivermos, digamos, vinte sintomas e um banco de dados de dez mil pacientes, só vimos uma pequena fração de cerca de um milhão de combinações possíveis. Pior, para estimar de maneira exata a probabilidade de uma combinação específica, precisamos pelo menos de dezenas de observações dela, o que significa que o banco de dados teria de incluir dezenas de milhões de pacientes. Se adicionarmos mais dez sintomas, precisaremos de mais pacientes do que o número de pessoas no planeta Terra. Com cem sintomas, mesmo se pudéssemos obter como que por mágica os dados, não haveria espaço suficiente em todos os discos rígidos do mundo para o armazenamento de todas as probabilidades. E se chegar um paciente com uma combinação de pacientes não vistos antes, não saberemos como diagnosticá-lo. Estaremos diante de nossa velha inimiga: a explosão combinatória.

Portanto, faremos o que a vida já costuma exigir de nós: tentar uma solução conciliatória. Faremos suposições simplificadoras para reduzir a algo gerenciável o número de probabilidades que temos de estimar. Uma suposição muito simples e popular é a de que todos os efeitos são independentes, dada a causa. Isso significa que, por exemplo, ter febre não alterará a probabilidade de também termos tosse, se soubermos que estamos com gripe. Matematicamente, estamos dizendo que $P(febre, tosse \mid gripe)$ é apenas $P(febre \mid gripe) \times P(tosse \mid gripe)$. Veja bem: cada um desses sintomas é fácil de estimar a partir de um pequeno número de observações. Foi o que fizemos para a febre na seção anterior, e não seria diferente para a tosse ou qualquer outro sintoma. O número de observações necessárias deixou de aumentar exponencialmente com o número de sintomas; na verdade, ele não está aumentando de forma alguma.

Observe que o que estamos dizendo é apenas que febre e tosse ocorrem independentemente quando estamos gripados, não obrigatoriamente. É claro que se não soubermos se temos gripe, a febre e a tosse estarão altamente correlacionadas, já que há maior probabilidade de termos tosse se já tivermos febre. *P*(*febre, tosse*) não é igual a *P*(*febre*) × *P*(*tosse*). Tudo que estamos dizendo é que, se soubermos que temos gripe, saber se temos febre não nos dará informações *adicionais* sobre se temos tosse. Da mesma forma, se você não souber que o Sol está para nascer e as estrelas começarem a desaparecer, sua expectativa de que o céu clareie aumentará; mas se souber que o nascer do Sol é iminente, ver as estrelas desaparecerem não fará diferença.

Repare também que foi graças ao teorema de Bayes que pudemos usar esse truque. Se quiséssemos estimar diretamente *P*(*gripe | febre, tosse etc.*), sem antes transformar em *P*(*febre, tosse etc. | gripe*) usando o teorema, ainda precisaríamos de um número exponencial de probabilidades, uma para cada combinação de sintomas e de gripado/não gripado.

Um aprendiz que usa o teorema de Bayes e supõe que os efeitos são independentes dada a causa se chama classificador Naïve Bayes. Ele tem esse nome porque, bem, essa é uma suposição muito ingênua.[2] Na verdade, ter febre torna mais provável que haja tosse, mesmo se soubermos que estamos gripados, já que (por exemplo) é mais provável que estejamos com uma gripe forte. Porém, o machine learning é a arte de fazer falsas suposições e mesmo assim ter um bom resultado. Como o estatístico George Box disse em uma frase famosa: “Todos os modelos estão errados, mas alguns são úteis”. É melhor fazer a estimativa de um modelo mais simples para a qual haja dados suficientes do que a de um modelo perfeito para a qual não haja dados. O economista Milton Friedman chegou a argumentar em um ensaio altamente influente que as melhores teorias são as mais simples, contanto que suas previsões sejam precisas, porque elas explicam mais com menos. Para mim isso parece um exagero, mas demonstra que, contrariando a frase de Einstein, a ciência costuma progredir fazendo as coisas da maneira mais simples possível.

Ninguém sabe ao certo quem inventou o Naïve Bayes. Esse algoritmo é mencionado sem atribuição em um livro de reconhecimento de padrões de 1973, mas só foi aceito nos anos 1990, quando os pesquisadores notaram que, surpreendentemente, com frequência ele era mais exato do que aprendizes muito mais sofisticados. Eu era aluno de pós-graduação na época, e, quando decidi

tardiamente incluir o Naïve Bayes em meus experimentos, fiquei admirado ao ver que ele se saía melhor do que todos os outros algoritmos que estava comparando, exceto um – felizmente, o algoritmo que desenvolvia para minha tese, ou talvez não estivesse aqui agora.

Agora o algoritmo Naïve Bayes é amplamente usado. Por exemplo, ele forma a base de muitos filtros de spam. Tudo começou quando David Heckerman, um proeminente pesquisador bayesiano que também é médico, teve a ideia de tratar o spam como uma doença cujos sintomas fossem as palavras do email: *Viagra* é um sintoma, assim como *free* (grátis), mas o nome de nosso melhor amigo provavelmente sinaliza um email legítimo. Podemos então usar o Naïve Bayes para classificar emails como spam e não spam, contanto que os remetentes de spam gerem emails selecionando palavras aleatoriamente. É claro que essa é uma suposição ridícula: ela só seria verdadeira se as frases não tivessem sintaxe e conteúdo. Porém, naquele verão, Mehran Sahami, então aluno de pós-graduação em Stanford, testou a suposição durante um estágio na Microsoft Research, e ela forneceu ótimos resultados. Quando Bill Gates perguntou a Heckerman como isso aconteceu, ele disse que para identificar spam não precisamos entender os detalhes da mensagem; é suficiente obter o sentido geral vendo as palavras que foram usadas.

Um mecanismo de busca básico também usa um algoritmo muito semelhante ao Naïve Bayes para decidir quais páginas web serão retornadas em resposta a uma consulta. A principal diferença é que, em vez de spam/não spam, ele tenta prever o que é relevante/irrelevante. A lista de problemas de previsão aos quais o Naïve Bayes tem sido aplicado é praticamente interminável. Peter Norvig, diretor de pesquisa do Google, disseme uma vez que ele era o aprendiz mais amplamente usado na empresa, e o Google usa machine learning em tudo que faz. Não é difícil entender por que o Naïve Bayes seria popular entre os funcionários do Google. Fora a incrível precisão, ele permite escalonamento; aprender com um classificador Naïve Bayes é apenas uma questão de contar quantas vezes cada atributo ocorre junto a cada classe e demora pouco mais do que ler os dados no disco.

Ironicamente, você poderia usar o Naïve Bayes em uma escala muito maior do que a do Google: para modelar todo o universo. Na verdade, se você acredita em um Deus onipotente, pode modelar o universo como uma vasta distribuição Naïve Bayes em que tudo o que ocorre é independente dada a vontade de Deus.

É claro que o problema é que não podemos ler a mente de Deus, mas no Capítulo 8 investigaremos como aprender modelos Naïve Bayes mesmo quando não conhecemos as classes dos exemplos.

Pode não parecer à primeira vista, mas o Naïve Bayes está intimamente relacionado com o algoritmo do perceptron. O perceptron soma pesos e o Naïve Bayes multiplica probabilidades, porém, se você usar um logaritmo, o último será reduzido ao primeiro. Ambos podem ser considerados generalizações de regras simples *Se... então...*, em que cada antecedente pode pesar mais ou menos em direção à conclusão em vez de ser um caso "todos ou nenhum". Esse é apenas um dos exemplos das conexões mais profundas existentes entre os aprendizes que tentam ser um Algoritmo Mestre. Talvez conscientemente você não conheça o teorema de Bayes (bem, agora conhece), mas de certa forma cada um dos dez bilhões de neurônios de seu cérebro é uma pequena instância dele.

O Naïve Bayes é um bom modelo conceitual de aprendiz para ser usado na leitura de notícias: ele captura a correlação em pares entre cada entrada e a saída, o que com frequência é tudo que necessitamos para entender referências a algoritmos de aprendizado em notícias. Porém, é claro que o machine learning não consiste apenas de correlações em pares, da mesma forma que o cérebro não é composto de um único neurônio. A ação começa quando procuramos padrões mais complexos.

De Eugene Onegin ao Siri

Em 1913, na véspera da Primeira Guerra Mundial, o matemático russo Andrei Markov publicou um artigo aplicando probabilidade à, entre tantas coisas possíveis, poesia. Nesse artigo, ele modelou um clássico da literatura russa, *Eugene Onegin* de Pushkin, usando o que agora chamamos de cadeia de Markov. Em vez de presumir que cada letra era gerada de maneira aleatória independentemente do resto, ele introduziu um mínimo básico de estrutura sequencial: permitiu que a probabilidade de ocorrência de cada letra dependesse da letra imediatamente anterior a ela. Ele mostrou que, por exemplo, vogais e consoantes tendem a se alternar, logo, se você encontrar uma consoante, haverá uma probabilidade maior de que a próxima letra (se ignorarmos a pontuação e espaços em branco) seja uma vogal do que haveria se as letras fossem independentes. Isso pode não parecer muito, mas na época anterior à dos computadores era preciso passar horas contando caracteres manualmente, e a

ideia de Markov era uma grande novidade. Se $Vogal_i$ for uma variável booleana considerada verdadeira se a *i*-ésima letra de *Eugene Onegin* for uma vogal e falsa se for uma consoante, podemos representar o modelo de Markov com um gráfico como este na forma de cadeia, com uma seta entre dois nós indicando dependência direta entre as variáveis correspondentes:

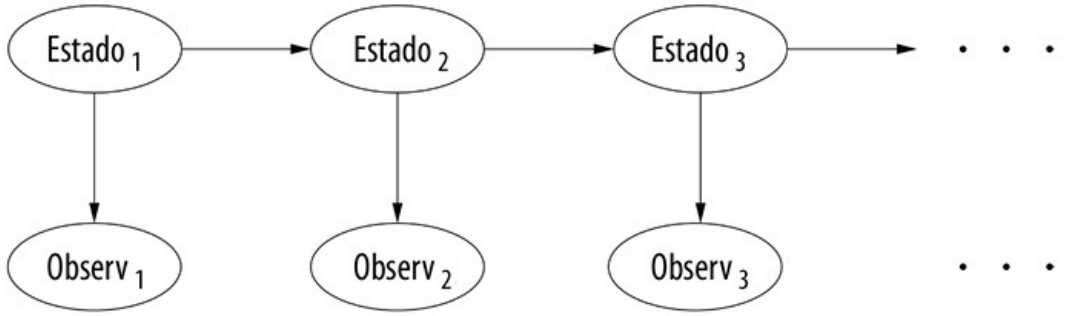
Markov presumiu (de maneira errada, mas útil) que as probabilidades são as mesmas em todos os locais do texto. Portanto, precisamos estimar só três probabilidades: $P(Vogal_1 = Verdadeiro)$, $P(Vogal_{i+1} = Verdadeiro \mid Vogal_i = Verdadeiro)$ e $P(Vogal_{i+1} = Verdadeiro \mid Vogal_i = Falso)$. (Já que a soma das probabilidades resulta em um, a partir delas podemos obter imediatamente $P(Vogal_1 = Falso)$ etc.) Assim como no algoritmo Naïve Bayes, podemos ter o número de variáveis que quisermos sem que o número de probabilidades aumente muito, mas agora as variáveis dependem realmente umas das outras.

Se medirmos não só a probabilidade da ocorrência de vogais *versus* consoantes, mas também a probabilidade de ocorrência de cada letra do alfabeto uma após a outra, podemos nos divertir gerando novos textos com as mesmas estatísticas de *Onegin*: selecione a primeira letra, em seguida selecione a segunda com base na primeira, e assim por diante. É claro que o resultado será totalmente inarticulado, porém se cada letra se basear em várias letras anteriores em vez de em apenas uma, começará a soar mais como o falar desconexo de alguém que exagerou na bebida, localmente coerente mesmo sendo globalmente sem significado. Ainda não é suficiente para passar no teste de Turing, mas modelos como esse são um componente-chave de sistemas de tradução por máquina, como o Google Tradutor, que nos permite ver a web inteira (ou quase) em português, não importando o idioma em que as páginas foram originalmente escritas.

O PageRank, algoritmo que originou o Google, é ele próprio uma cadeia de Markov. A ideia de Larry Page era a de que, quando muitos links levam a uma página web, provavelmente ela é mais importante do que páginas com poucos links, e os links provenientes de páginas relevantes também devem ser considerados mais importantes. Isso produz um regresso infinito, mas podemos manipulá-lo com uma cadeia de Markov. Imagine um usuário da web passando

de uma página à outra seguindo links aleatoriamente: os estados dessa cadeia de Markov são páginas web em vez de caracteres, o que a torna um problema amplamente maior, porém a matemática é a mesma. A pontuação de uma página é então a fração do tempo que o usuário da web passa nela ou, de modo equivalente, sua probabilidade de acabar acessando a página após navegar sem destino por muito tempo.

Cadeias de Markov podem ser vistas em todos os lugares e elas constituem um dos tópicos mais estudados na matemática, mas mesmo assim são um tipo muito limitado de modelo probabilístico. Podemos ir além com um modelo como este:



Os estados formam uma cadeia de Markov, como antes, mas não os vemos; temos de inferi-los a partir das observações. Isso se chama modelo oculto de Markov, ou HMM (hidden Markov model). (O nome é enganoso, já que são os estados que estão ocultos e não o modelo.) Os HMMs são a base de sistemas de reconhecimento de voz como o Siri. No reconhecimento de voz, os estados ocultos são palavras escritas, as observações são os sons falados para o Siri e o objetivo é inferir as palavras a partir dos sons. O modelo tem dois componentes: a probabilidade de ocorrência da próxima palavra dada a atual, como em uma cadeia de Markov, e a probabilidade de vários sons serem ouvidos dada a palavra que está sendo pronunciada. (Como a inferência é feita é um problema fascinante ao qual voltaremos após a próxima seção.) Além do seu uso no Siri, usamos um HMM sempre que falamos no celular. Isso ocorre porque nossas palavras são enviadas pelo ar como um fluxo de bits e estes são adulterados quando em trânsito. O HMM descobre então os bits que foram enviados (estado oculto) a partir dos bits recebidos (observações), o que ele só tem condições de fazer quando o número de bits adulterados não é excessivo.

Os HMMs também são a ferramenta favorita dos biólogos computacionais. Uma proteína é uma sequência de bases. Se quisermos prever, por exemplo, como a proteína se dobrará em uma forma 3D, podemos tratar os aminoácidos

como observações e o tipo de dobradura em cada ponto como o estado oculto. Da mesma forma, podemos usar um HMM para identificar os locais em que a transcrição de genes é iniciada no DNA e muitas outras propriedades.

Se os estados e as observações forem variáveis contínuas em vez de discretas, o HMM passará a ser o que é conhecido como filtro de Kalman. Os economistas usam filtros de Kalman para remover o ruído de séries de quantidades temporais como as do PIB, inflação e desemprego. Os valores "reais" do PIB são os estados ocultos; a cada etapa temporal, o valor real deve ser semelhante não só ao observado, mas também ao valor real anterior, já que a economia raramente dá saltos abruptos. O filtro de Kalman encontra um equilíbrio entre esses dois valores, gerando uma curva mais suave que continue em concordância com as observações. Quando um míssil vai em direção ao alvo, é um filtro de Kalman que o mantém no caminho certo. Sem ele, o homem não teria chegado à Lua.

Tudo está conectado, mas não diretamente

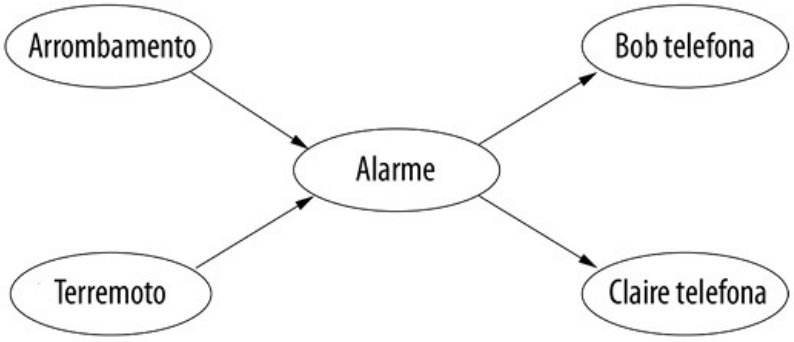
Os HMMs são bons na modelagem de sequências de todos os tipos, mas não chegam perto da flexibilidade das regras *Se... então...* dos simbolistas, em que qualquer elemento pode aparecer como antecedente, e o elemento consequente de uma regra pode, por sua vez, ser o antecedente de alguma regra posterior do fluxo. No entanto, se na prática permitirmos o uso de uma estrutura tão arbitrária, o número de probabilidades a serem aprendidas explodirá. Por muito tempo ninguém soube como resolver esse difícil problema e os pesquisadores recorreram a esquemas *ad hoc*, como anexar estimativas de certeza às regras e combiná-las de alguma forma. Se A implica B com certeza de 0,8 e B implica C com certeza de 0,7, talvez A implique C com certeza de 0,8 × 0,7.

O problema desses esquemas é que eles podem cometer erros graves. A partir das duas regras perfeitamente razoáveis *Se o irrigador está ligado, então a grama ficará molhada* e *Se a grama está molhada, então choveu*, posso inferir a regra absurda *Se o irrigador está ligado, então choveu*. Um problema mais traiçoeiro é que se usarmos regras com níveis de confiança estaremos propensos à dupla contagem de evidências. Suponhamos que você lesse no *New York Times* que alienígenas pousaram na Terra. Pode ser uma brincadeira, ainda que não seja primeiro de abril. Porém, você acaba de ver a mesma notícia no *Wall Street Journal*, no *USA Today* e no *Washington Post*. Começa a entrar em pânico, da mesma forma que os ouvintes da infame transmissão de rádio de *A guerra dos*

mundos, feita por Orson Welles, não perceberam que se tratava de uma dramatização. No entanto, se você verificar as letras miúdas e notar que os quatro jornais tiraram a história da Associated Press, voltará a suspeitar que é uma brincadeira, desta vez de um repórter da AP. Os sistemas de regras não têm como lidar com isso, nem o Naïve Bayes. Se ele usar características como *Noticiado no* New York Times como indicadoras de que uma notícia é verdadeira, tudo o que poderá fazer é adicionar *Noticiado na AP*, o que só piorará as coisas.

A virada ocorreu no início da década de 1980, quando Judea Pearl, professor de ciência da computação da Universidade da Califórnia, Los Angeles, inventou uma nova representação: as redes bayesianas. Pearl é um dos cientistas da computação mais famosos do mundo e seus métodos são usados em machine learning, inteligência artificial e muitas outras áreas. Ele ganhou o Turing Award, o Prêmio Nobel da ciência da computação, em 2012.

Pearl percebeu que não seria problema haver uma complexa rede de dependências entre variáveis aleatórias, contanto que cada variável só dependesse diretamente de outras poucas variáveis. Podemos representar essas dependências na forma de um diagrama, como os que vimos para as cadeias de Markov e os HMMs, exceto pelo fato de que agora o diagrama pode ter qualquer estrutura (mas as setas não podem formar loops fechados). Um dos exemplos favoritos de Pearl é o do alarme contra arrombamento. O alarme que temos em casa deve disparar se um arrombador tentar invadir, mas ele também poderia ser acionado por um terremoto. (Em Los Angeles, onde Pearl vive, os terremotos são quase tão frequentes quanto os arrombamentos.) Se uma noite você ficasse trabalhando até tarde e seu vizinho Bob telefonasse para dizer que acabou de ouvir seu alarme disparar, mas sua vizinha Claire não ouvisse, seria o caso de chamar a polícia? Aqui está o diagrama de dependências:



Quando há uma seta de um nó para outro no diagrama, dizemos que o primeiro

nó é pai do segundo. Logo, os pais de *Alarme* são *Arrombamento* e *Terremoto*, e ele é o único pai de *Bob telefona* e *Claire telefona*. Uma rede bayesiana é um diagrama de dependências como esse, junto com uma tabela para cada variável, fornecendo a probabilidade de ocorrência de cada combinação de valores de seus pais. No caso de *Arrombamento* e *Terremoto* só precisamos de uma probabilidade de cada um, já que eles não têm pais. No caso de *Alarme* precisamos de quatro: a probabilidade de ele ser acionado mesmo se não houver arrombamento ou terremoto, a probabilidade de ser acionado se houver um arrombamento, mas não um terremoto, e assim por diante. Para *Bob telefona* precisamos de duas probabilidades (de acionamento ou não do alarme) e o mesmo para Claire.

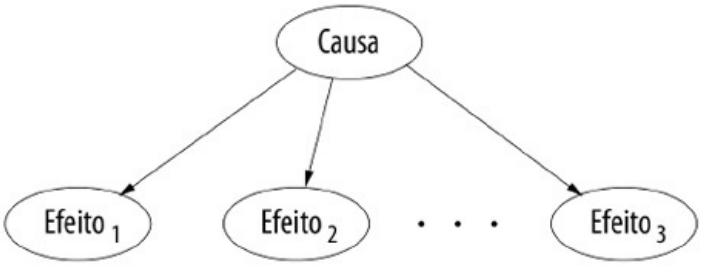
O ponto crucial é: o telefonema de Bob depende de *Arrombamento* e *Terremoto*, mas só por intermédio de *Alarme*. Seu telefonema é *condicionalmente independente* de *Arrombamento* e *Terremoto* dado *Alarme*, e o mesmo ocorre para o de Claire. Se o alarme não disparar, os vizinhos continuarão dormindo e o arrombador não será perturbado. Além disso, dado o *Alarme*, Bob e Claire são independentes. Sem essa estrutura de independência, você precisaria aprender $2^5 = 32$ probabilidades, uma para cada estado possível das cinco variáveis. (Ou 31, se formos perfeccionistas, já que a última pode ser considerada implícita.) Com as independências condicionais, você só precisa de $1 + 1 + 4 + 2 + 2 = 10$, uma economia de 68%. E isso para esse pequeno exemplo; com centenas ou milhares de variáveis, a economia chega perto de 100%.

A primeira lei da ecologia, de acordo com o biólogo Barry Commoner, diz que tudo está conectado a todo o resto. Isso pode ser verdade, mas também tornaria impossível entender o mundo, se não fosse pela independência condicional: tudo está conectado, mas apenas indiretamente. Para me afetar, algo que ocorra a um quilômetro e meio de distância deve antes afetar alguma coisa em minha vizinhança, mesmo se for apenas pela propagação de luz. Como alguém brincou certa vez, o espaço é a razão pela qual nada nos acontece. Colocado de outra forma, a estrutura do espaço é um exemplo de independência condicional.

No exemplo do arrombamento, a tabela completa de trinta e duas possibilidades nunca é representada explicitamente, mas ela está implícita no conjunto de tabelas menores e na estrutura do diagrama. Para obter *P*(*Arrombamento*, *Terremoto*, *Alarme*, *Bob telefona*, *Claire telefona*), tudo que

tenho de fazer é multiplicar *P*(*Arrombamento*), *P*(*Terremoto*), *P*(*Alarme* | *Arrombamento*, *Terremoto*), *P*(*Bob telefona* | *Alarme*) e *P*(*Claire telefona* | *Alarme*). O mesmo ocorre em qualquer rede bayesiana: para obter a probabilidade de um estado completo, basta multiplicar as probabilidades contidas nas linhas correspondentes das tabelas individuais das variáveis. Logo, contanto que as independências condicionais se mantenham, não serão perdidas informações se passarmos para a representação mais compacta. E dessa forma poderemos calcular facilmente as probabilidades de estados extremamente incomuns, inclusive de estados que nunca foram observados. As redes bayesianas desmentem a concepção errônea de que o machine learning não consegue prever eventos muito raros, ou "cisnes negros", como Nassim Taleb os chama.

Em retrospecto, podemos ver que o algoritmo Naïve Bayes, as cadeias de Markov e os HMMs são casos especiais de redes bayesianas. A estrutura do Naïve Bayes é:



As cadeias de Markov representam a suposição de que o futuro é condicionalmente independente do passado dado o presente. Adicionalmente, os HMMs presumem que cada observação só depende do estado ao qual ela está associada. As redes bayesianas são para os bayesianos o que a lógica é para os simbolistas: uma língua franca que nos permite representar de maneira elegante uma grande variedade de situações e projetar algoritmos que funcionem uniformemente em todas elas.

Podemos considerar uma rede bayesiana como um "modelo generativo", uma receita para a geração probabilística de um estado do mundo: primeiro ela decide independentemente se houve um arrombamento e/ou um terremoto; com base nisso define se o alarme disparou; e, então, define se Bob e Claire telefonaram. A rede bayesiana conta uma história. A ocorreu e levou a B; ao mesmo tempo, C também ocorreu, e B e C juntos causaram D. Para calcular a probabilidade de uma história específica ocorrer, só temos de multiplicar as probabilidades de todas as suas diferentes ramificações.

Uma das aplicações mais interessantes das redes bayesianas é a modelagem de como os genes regulam uns aos outros em células vivas. Bilhões de dólares foram gastos na tentativa de descobrir correlações em pares entre genes individuais e doenças específicas, mas o resultado foi desapontadamente baixo. Se pensarmos bem, isso não é tão surpreendente: o comportamento de uma célula provém de interações complexas entre os genes e o ambiente, e um único gene tem poder preditivo limitado. Porém, com redes bayesianas, podemos revelar essas interações, contanto que tenhamos os dados necessários, e com a disseminação dos microarranjos de DNA sua disponibilidade é cada vez maior.

Após ser pioneiro na aplicação do machine learning à filtragem de spam, David Heckerman se voltou para o uso de redes bayesianas na luta contra a AIDS. O vírus da AIDS é um adversário forte porque sofre mutações rapidamente, o que dificulta que uma vacina ou droga específica o contenha por muito tempo. Heckerman notou que esse era o mesmo jogo de gato e rato que os filtros de spam jogam com mensagens indesejadas e decidiu aplicar uma lição que aprendeu aí: atacar o elo mais fraco. No caso do spam, os elos mais fracos são os URLs que temos de usar para receber o pagamento do cliente. Para o HIV, eles são pequenas regiões da proteína do vírus que não podem mudar sem feri-lo. Se ele pudesse treinar o sistema imunológico para reconhecer essas regiões e atacar as células que as exibem, talvez obtivesse uma vacina contra a AIDS. Heckerman e seus colaboradores usaram uma rede bayesiana para ajudar a identificar as regiões vulneráveis e desenvolveram um mecanismo de distribuição de vacina que poderia ensinar o sistema imunológico a atacar apenas essas regiões. O mecanismo de distribuição funcionou em ratos, e agora testes clínicos estão sendo preparados.

Com frequência ocorre que, mesmo após considerarmos todas as independências condicionais, alguns nós de uma rede bayesiana ainda têm muitos pais. Certas redes têm tantas setas que, quando as imprimimos, a página fica toda preta. (O físico Mark Newman as chama de “ridiculogramas”.) Um médico precisa diagnosticar simultaneamente todas as doenças que um paciente pode ter, não apenas uma, e cada doença é mãe de muitos sintomas diferentes. Uma febre pode ser causada por qualquer número de condições além da gripe, mas é inútil tentar prever sua probabilidade dada cada combinação de condições possível. Porém, nem tudo está perdido. Em vez de uma tabela especificando a probabilidade condicional do nó para cada estado de seus pais, podemos

aprender uma distribuição mais simples. A opção mais popular é uma versão probabilística da operação lógica OR: qualquer causa individual pode provocar febre, mas cada causa tem uma probabilidade específica de não provocá-la, mesmo quando é suficiente para fazê-lo. Heckerman e outros aprenderam redes bayesianas que diagnosticam centenas de doenças infecciosas dessa forma. O Google usa uma rede bayesiana gigante desse tipo em seu sistema AdSense para selecionar automaticamente anúncios a serem inseridos em páginas web. A rede associa um milhão de variáveis de conteúdo umas às outras e a doze milhões de palavras e frases por intermédio de mais de trezentos milhões de setas, tudo aprendido a partir de cem bilhões de fragmentos de texto e consultas.

Em um exemplo menos complexo, o Xbox Live da Microsoft usa uma rede bayesiana para classificar jogadores e unir os de aptidão semelhante. O resultado de um jogo é uma função probabilística dos níveis de aptidão dos oponentes, e usando o teorema de Bayes podemos inferir a aptidão de um jogador a partir dos resultados de seus jogos.

O problema da inferência

Há uma grande armadilha em tudo isso, infelizmente. O fato de a rede bayesiana nos permitir representar uma distribuição de probabilidade de maneira compacta não significa que também possamos raciocinar eficientemente com ela. Suponhamos que você quisesse calcular *P*(*Arrombamento* | *Bob telefonou*, *Claire não*). Pelo teorema de Bayes, você sabe que essa probabilidade é semelhante a *P*(*Arrombamento*) *P*(*Bob telefonou*, *Claire não* | *Arrombamento*) / (*Bob telefonou*, *Claire não*) ou, igualmente, *P*(*Arrombamento*, *Bob telefonou*, *Claire não*) / *P*(*Bob telefonou*, *Claire não*). Se tivesse a tabela completa com as probabilidades de todos os estados, você poderia obter essas duas probabilidades somando as linhas correspondentes da tabela. Por exemplo, *P*(*Bob telefonou*, *Claire não*) é a soma das probabilidades de todas as linhas em que Bob telefona e Claire não. Porém, a rede bayesiana não fornece a tabela completa. É possível construí-la a partir de tabelas individuais, mas essa abordagem demanda tempo e espaço exponenciais. O que queremos na verdade é calcular *P*(*Arrombamento* | *Bob telefonou*, *Claire não*) sem construir a tabela completa. Resumindo, esse é o problema da inferência nas redes bayesianas.

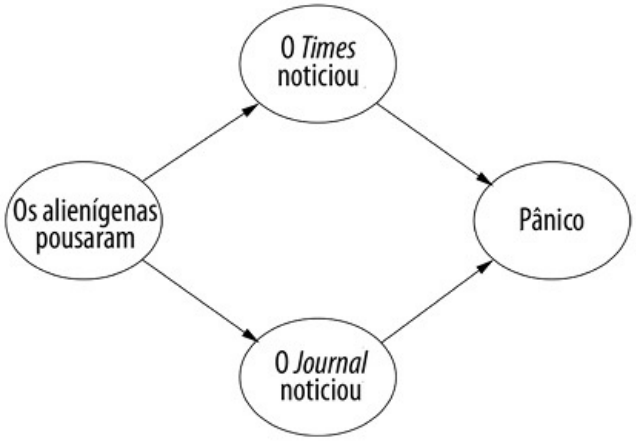
Em muitos casos podemos fazer isso e evitar o aumento exponencial. Suponhamos que você estivesse liderando um pelotão em fila indiana pelo

território inimigo tarde da noite e quisesse verificar se todos os seus soldados continuam o seguindo. Poderia parar e contá-los você mesmo, mas perderia muito tempo. Uma solução mais inteligente é simplesmente perguntar ao primeiro soldado atrás de você: “Quantos soldados estão o seguindo?”. Cada soldado faria a mesma pergunta ao soldado seguinte, até o último dizer “Nenhum”. Agora o penúltimo soldado pode dizer “Um”, e assim por diante, até o primeiro soldado, com cada soldado somando um ao número de soldados atrás dele. Você saberá então quantos soldados ainda o seguem e não precisou nem mesmo parar.

O aplicativo Siri usa a mesma ideia para calcular, a partir dos sons capturados no microfone, a probabilidade de alguém ter dito “Call the police” (Chame a polícia). Imagine a frase “Call the police” como um pelotão de palavras marchando pela página em fila indiana. *Police* quer saber sua probabilidade de ocorrer, mas para isso ele precisa saber a probabilidade de *the*; e *the* por sua vez precisa saber a probabilidade de *call*. Logo, *call* calcula sua probabilidade e a passa para *the*, que faz o mesmo e passa o resultado para *police*. Agora *police* sabe sua probabilidade, devidamente informado por cada palavra da frase, mas não tivemos de construir a tabela completa com oito possibilidades (a primeira palavra será ou não *call*, a segunda será ou não *the* e a terceira será ou não *police*). Na verdade, o Siri considera todas as palavras que poderiam aparecer em cada posição e não apenas se a primeira palavra é ou não *call*, e assim por diante, mas o algoritmo é o mesmo. Talvez ele considere, com base nos sons, que a primeira palavra foi *call* ou *tell*, a segunda foi *the* ou *her* e a terceira foi *police* ou *please*. Individualmente, as palavras mais prováveis poderiam ser *call*, *the* e *please*. Porém, isso formaria a frase absurda “Call the please”, portanto, considerando as outras palavras, o Siri conclui que a frase é “Call the police”. Ele faz a chamada, e com um pouco de sorte a polícia chegará a tempo de capturar o arrombador.

A mesma ideia também funciona quando o diagrama é uma árvore em vez de uma cadeia. Se em vez de um pelotão você estiver no comando de um exército inteiro, poderá perguntar a cada um dos comandantes de sua companhia quantos soldados estão atrás dele e somar suas respostas. Por sua vez, cada comandante da companhia perguntará a cada um dos comandantes de seu pelotão, e assim por diante. Porém, se o diagrama formar loops, isso será um problema. Se houver um oficial de ligação que seja membro de dois pelotões, ele será contado

duas vezes; na verdade, todos que estiverem atrás dele serão contados duas vezes. É isto que ocorrerá no cenário “os alienígenas pousaram”, se você quiser calcular, digamos, a probabilidade de haver pânico:



Uma solução é combinar *O* Times *noticiou* e *O* Journal *noticiou* em uma única megavariável com quatro valores: *SimSim* se os dois jornais tiverem noticiado, *SimNão* se o *Times* noticiar uma aterrissagem, mas o *Journal* não *etc.* Isso transformará o diagrama em uma cadeia de três variáveis e tudo ficará bem. No entanto, sempre que você adicionar uma fonte de notícias, o número de valores da megavariável dobrará. Se em vez de duas fontes de notícias você tiver cinquenta, a megavariável terá 2^{50} valores. Logo, esse método não poderá levá-lo mais longe, e nenhum outro método conhecido o fará.

O problema é pior do que parece porque as redes bayesianas têm setas “invisíveis” que acompanham as visíveis. *Arrombamento* e *Terremoto* são fatos independentes *a priori*, mas quando o alarme soa eles se entrelaçam: o alarme nos faz suspeitar de um arrombamento, mas se ouvirmos no rádio que houve um terremoto, presumiremos que foi ele que o disparou. O terremoto *explicou* o alarme, tornando o arrombamento menos provável, portanto os dois são dependentes. Em uma rede bayesiana, todos os pais da mesma variável são interdependentes dessa maneira e, por sua vez, isso introduz dependências adicionais, com frequência tornando o diagrama resultante muito mais denso do que o original.

A pergunta crucial da inferência é se conseguiremos fazer o diagrama preenchido “parecer uma árvore” sem o tronco ficar muito grosso. Se a megavariável do tronco tiver um número excessivo de valores possíveis, a árvore crescerá de maneira descontrolada até cobrir todo o planeta, como os baobás de *O pequeno príncipe*. Na árvore da vida, cada espécie é um galho, mas dentro de

cada galho há um diagrama, com cada criatura tendo dois pais, quatro avós, algum número de filhos, e assim por diante. A "grossura" de um galho é igual ao tamanho da população da espécie. Quando os galhos são muito grossos, nossa única opção é recorrer à inferência aproximada.

Uma solução, deixada como exercício por Pearl em seu livro sobre redes bayesianas, é imaginar que o diagrama não tem loops e apenas continuar propagando as probabilidades para trás e para frente até que elas convirjam. Isso é conhecido como propagação de crença em loops (loopy belief propagation), tanto porque funciona em diagramas com loops quanto porque é uma ideia maluca. Surpreendentemente, funciona bem em vários casos. Por exemplo, é um dos métodos mais avançados para a comunicação wireless, com as variáveis aleatórias sendo os bits da mensagem, codificados de maneira inteligente. Porém, a propagação de crença em loops também pode convergir para as respostas erradas ou oscilar indefinidamente. Outra solução, que se originou na física, mas foi importada para o machine learning e estendida por Michael Jordan e outros, é aproximar uma distribuição intratável de uma tratável e otimizar os parâmetros da última para torná-la a mais próxima possível da primeira.

No entanto, a opção mais popular é afogar as mágoas em álcool e andar tropegamente noite adentro. O termo técnico para isso é *métodos de Monte Carlo via cadeias de Markov*, ou MCMC (Markov chain Monte Carlo). A parte "Monte Carlo" vem do fato de o método envolver risco, como uma visita ao cassino de mesmo nome, e a parte "cadeias de Markov" é porque envolve usar uma sequência de etapas, cada uma dependendo apenas da anterior. A ideia do MCMC é fazer uma caminhada aleatória, como fazem os beberrões, pulando de um estado a outro da rede de forma que, no longo prazo, o número de vezes que cada estado é visitado seja proporcional à sua probabilidade. Podemos então estimar a probabilidade de ocorrer um arrombamento, digamos, como o número de vezes que visitamos um estado em que houve arrombamento. Uma cadeia de Markov "bem-comportada" converge para uma distribuição estável, logo, após algum tempo, ela sempre fornece aproximadamente as mesmas respostas. Por exemplo, quando embaralhamos cartas, depois de algum tempo todas as ordens das cartas têm probabilidades iguais de ocorrer, não importando a ordem inicial; portanto, sabemos que se houver n ordens possíveis, a probabilidade de cada uma será de $1/n$. O truque no MCMC é projetarmos uma cadeia de Markov que

convirja para a distribuição de nossa rede bayesiana. Uma opção fácil é percorrer repetidamente as variáveis, testando cada uma de acordo com sua probabilidade condicional dado o estado de seus vizinhos. As pessoas costumam se referir ao MCMC como um tipo de simulação, mas ele não é isso: a cadeia de Markov não simula nenhum processo real; nós a projetamos para gerar eficientemente amostras a partir de uma rede bayesiana, que também não é um modelo sequencial.

Se rastrearmos as origens do MCMC, voltaremos à época do Projeto Manhattan, em que os físicos precisaram estimar a probabilidade de nêutrons colidirem com átomos e dispararem uma reação em cadeia. Porém, em décadas mais recentes, ele deflagrou tamanha revolução que com frequência é considerado um dos algoritmos mais importantes de todos os tempos. O MCMC é bom não só para calcular probabilidades, mas também para integrar qualquer função. Sem ele, os cientistas estavam restritos a funções que podiam integrar analiticamente, ou a integrais bem-comportadas e de poucas dimensões que pudessem aproximar como uma série de trapezoides. Com o MCMC, eles têm liberdade para construir modelos complexos, sabendo que o computador fará o trabalho pesado. De sua parte, provavelmente os bayesianos devem o aumento da popularidade de seus métodos mais ao MCMC do que a qualquer outra coisa.

A desvantagem é que o MCMC costuma ser muito lento ao fazer a convergência, ou nos engana parecendo tê-la feito quando na verdade não a fez. Geralmente, as distribuições de probabilidades reais têm picos muito altos, com vastas áreas inúteis de probabilidades minúsculas pontuadas por repentinos montes Everest. A cadeia de Markov então converge para o pico mais próximo e permanece nele, gerando estimativas de probabilidade muito tendenciosas. É como se o ébrio seguisse o cheiro do álcool até o bar mais próximo e ficasse lá a noite toda, em vez de perambular pela cidade como queríamos que ele fizesse. Por outro lado, se em vez de usar uma cadeia de Markov apenas gerássemos amostras independentes, como fazem os métodos mais simples de Monte Carlo, não teríamos um cheiro para seguir e talvez nem mesmo encontrássemos aquele primeiro bar; seria como lançar dardos em um mapa da cidade, esperando que eles acertassem em cheio nos pubs.

A inferência em redes bayesianas não está restrita ao cálculo de probabilidades. Ela também inclui encontrar a explicação mais provável para a evidência, como a doença que melhor explica os sintomas ou as palavras que

melhor explicam os sons que o Siri ouviu. Isso não é o mesmo que selecionar a palavra mais provável a cada etapa, porque palavras que são individualmente prováveis dados os sons que produzem podem não ter probabilidade de ocorrer juntas, como no exemplo “Call the please”. No entanto, tipos de algoritmos semelhantes também funcionam para essa tarefa (e, na verdade, são o que a maioria dos reconhecedores de voz usa). O mais importante é que a inferência inclui tomar as melhores decisões, conduzidas não só pelas probabilidades de diferentes resultados, mas também pelos custos (ou utilidades, para usar o termo técnico) correspondentes. O custo de ignorar um email enviado pelo seu chefe pedindo que você faça algo até amanhã é muito maior do que o de encontrar um spam, logo com frequência é melhor permitir o recebimento de um email mesmo que ele pareça um spam.

Os carros autônomos e outros robôs são exemplos básicos da inferência probabilística em ação. À medida que o carro percorre a vizinhança, simultaneamente constrói um mapa do território e detecta sua localização com certeza cada vez maior. De acordo com um estudo recente, os taxistas de Londres têm a parte posterior de seu hipocampo, uma região do cérebro envolvida na construção de memória e mapas espaciais, aumentada conforme vão aprendendo o layout da cidade. Talvez eles usem algoritmos de inferência probabilística semelhantes, com a perceptível diferença de que no caso dos humanos beber não parece ajudar.

Aprendendo da maneira bayesiana

Agora que sabemos (mais ou menos) como resolver o problema da inferência, estamos prontos para aprender redes bayesianas a partir de dados, porque para os bayesianos o aprendizado é apenas outro tipo de inferência probabilística. Tudo que temos de fazer é aplicar o teorema de Bayes com as hipóteses como as possíveis causas e os dados como o efeito observado: $P(\textit{hipótese} \mid \textit{dados}) = P(\textit{hipótese}) \times P(\textit{dados} \mid \textit{hipótese}) / P(\textit{dados})$ A hipótese pode ser tão complexa quanto uma rede bayesiana inteira ou tão simples quanto a probabilidade de uma moeda exibir cara. No último caso, os dados são apenas o resultado de uma série de lançamentos da moeda. Se, digamos, obtivermos setenta caras em cem lançamentos, um frequentista estimaria a probabilidade de ocorrência de cara como de 0,7. Isso é explicado pelo chamado princípio da máxima verossimilhança: de todas as probabilidades possíveis de ocorrência de cara, 0,7

é a mais provável para a exibição de setenta caras em cem lançamentos. A verossimilhança de uma hipótese é *P*(*dados* | *hipótese*) e o princípio diz que devemos escolher a hipótese que a maximize. Contudo, os bayesianos fazem algo mais sutil. Eles apontam que nunca sabemos ao certo qual hipótese é a verdadeira e, portanto, não devemos selecionar apenas uma hipótese, como um valor de 0,7 para a probabilidade de ocorrência de cara; em vez disso, devemos calcular a probabilidade *a posteriori* de cada hipótese possível e acolher todas elas ao fazer previsões. A soma das probabilidades de todas as hipóteses deve ser um, logo, se uma se tonar mais provável, a probabilidade das outras diminuirá. Na verdade, para um bayesiano não há essa certeza que chamamos de verdade; temos uma distribuição *a priori* entre as hipóteses, após vermos os dados ela se torna a distribuição *a posteriori*, como dado pelo teorema de Bayes, e isso é tudo.

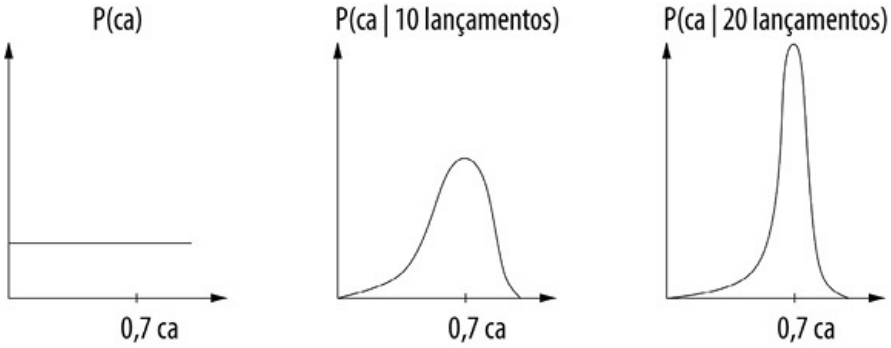
Essa abordagem diverge radicalmente da maneira como fazemos ciência. É como dizer: “Nem Copérnico nem Ptolomeu estavam certos; devemos prever as trajetórias futuras dos planetas presumindo que a Terra gira ao redor do Sol e vice-versa e calcular a média dos resultados”.

É claro que se trata de uma média ponderada, com o peso da hipótese sendo sua probabilidade *a posteriori*, logo uma hipótese que explique melhor os dados será a mais adequada. Mesmo assim, como algumas pessoas brincam, ser bayesiano significa nunca ter de dizer que você está certo.

É evidente que considerar várias hipóteses em vez de apenas uma é um grande incômodo. No caso de aprender uma rede bayesiana, devemos fazer previsões calculando a média de todas as redes bayesianas possíveis, incluindo todas as estruturas de diagramas possíveis e todos os valores de parâmetros possíveis para cada estrutura. Em alguns casos, podemos calcular a média dos parâmetros de forma fechada, mas com várias estruturas não temos essa alternativa. É preciso recorrer a, por exemplo, usar o MCMC no espaço das redes, pulando de uma rede possível à outra à medida que a cadeia de Markov progride. Se combinarmos toda essa complexidade e custo computacional com a noção controversa dos bayesianos de que não há algo como a realidade objetiva, não será difícil entender por que os frequentistas dominaram a ciência no último século.

No entanto, há uma vantagem e algumas razões importantes para preferirmos a maneira bayesiana. A vantagem é que, na maior parte do tempo, quase todas as

hipóteses acabam produzindo uma probabilidade *a posteriori* pequena, e podemos ignorá-la com segurança. Na verdade, considerar apenas a hipótese mais provável é geralmente uma aproximação muito boa. Suponhamos que nossa distribuição *a priori* para o problema do lançamento da moeda considerasse que todas as probabilidades de ocorrência de cara fossem igualmente possíveis. A observação dos resultados de sucessivos lançamentos produz uma concentração cada vez maior da distribuição nas hipóteses que melhor se adequam aos dados. Por exemplo, se *ca* representar as possíveis probabilidades de ocorrência de cara e uma moeda exibir cara 70% do tempo, veremos algo assim:



A probabilidade *a posteriori* após cada lançamento se torna a probabilidade *a priori* do próximo lançamento e, lançamento após lançamento, teremos cada vez mais certeza de que *ca* = 0,7. Se considerarmos apenas a hipótese mais provável (*ca* = 0,7 nesse caso), a abordagem bayesiana será muito semelhante à frequentista, mas com uma diferença crucial: os bayesianos consideram a probabilidade *a priori* *P*(*hipótese*) e não apenas a verossimilhança *P*(*dados* | *hipótese*). (A probabilidade *a priori* *P*(*dados*) pode ser ignorada porque é a mesma para todas as hipóteses e, portanto, não afeta a escolha da vencedora.) Se quisermos presumir que todas as hipóteses são igualmente prováveis *a priori*, a abordagem bayesiana se resumirá ao princípio da máxima verossimilhança. Logo, os bayesianos poderiam dizer para os frequentistas: “Veja, o que vocês fazem é um caso especial do que fazemos, mas pelo menos explicitamos nossas suposições”. E se as hipóteses não forem igualmente prováveis *a priori*, a suposição implícita pela máxima verossimilhança de que elas o sejam levará a respostas erradas.

Essa parece ser uma discussão teórica, mas tem enormes consequências práticas. Se observarmos apenas um lançamento da moeda, e ela exibir cara, a máxima verossimilhança dirá que a probabilidade de ocorrência de cara deve ser igual a um. Essa definição pode ser muito imprecisa e nos deixar

angustiosamente despreparados para o fato de a moeda exibir coroa. Depois que tivermos visto vários lançamentos, a estimativa será mais confiável, mas em muitos problemas, nunca são observados lançamentos suficientes, não importa o volume de dados. Suponhamos que a palavra *supercalifragilisticexpialidocious* nunca aparecesse em um spam em nossos dados de treinamento e aparecesse uma vez em um email falando sobre *Mary Poppins*. Um filtro de spam Naïve Bayes com estimativas de probabilidade de máxima verossimilhança decidirá então que um email que a contenha não pode ser spam, não importando se todas as outras palavras do email avisem "Spam! Spam!". Por outro lado, um bayesiano daria à palavra uma probabilidade baixa, mas não nula, de aparecer em um spam, permitindo que as outras palavras a sobreponham.

O problema ficará pior se tentarmos aprender não só a estrutura de uma rede bayesiana, mas também seus parâmetros. Podemos fazê-lo com a escalada de montanha, começando com uma rede vazia (sem setas), adicionando a seta que mais aumente a verossimilhança e continuando nesse ritmo até que nenhuma seta cause melhoria. Infelizmente, isso não demora a levar ao sobreajuste massivo, com uma rede que atribua probabilidade zero a todos os estados que não apareçam nos dados. Os bayesianos podem fazer algo muito mais interessante. Podem usar a distribuição *a priori* para representar o conhecimento de especialistas sobre o problema – sua resposta à pergunta de Hume. Por exemplo, podemos projetar uma rede bayesiana inicial para o diagnóstico clínico entrevistando médicos, perguntando a eles quais sintomas acham que dependem de quais doenças e adicionando as setas correspondentes. Essa é a "rede *a priori*", e a distribuição *a priori* pode penalizar redes alternativas pelo número de setas que adicionarem ou removerem dela. Porém, os médicos também falham, logo deixaremos que os dados sobreponham suas opiniões: se o aumento na verossimilhança proveniente da inclusão de uma seta compensar a penalidade, nós a adicionaremos.

É claro que os frequentistas conhecem esse problema, e sua resposta é, por exemplo, multiplicar a verossimilhança por um fator que penalize redes mais complexas. Porém, nesse ponto o frequentismo e o bayesianismo se tornaram indistinguíveis, e chamar a função de pontuação de "verossimilhança penalizada" ou "probabilidade *a posteriori*" é na verdade apenas uma questão de gosto.

Apesar da convergência dos pensamentos frequentista e bayesiano em alguns

aspectos, permanece a diferença filosófica sobre o significado de probabilidade. Considerá-la subjetiva provoca náuseas em muitos cientistas, mas também permite usos que de outra forma seriam proibidos. Se você for um frequentista, só estimará probabilidades de eventos que possam ocorrer mais de uma vez. Logo, uma pergunta como “Qual é a probabilidade de Hillary Clinton vencer Jeb Bush na próxima eleição presidencial?” não pode ser respondida, porque nunca houve uma eleição que os colocasse em disputa. Porém, para um bayesiano, uma probabilidade é um grau subjetivo de crença, portanto ele tem liberdade para fazer uma suposição embasada, e o cálculo da inferência manterá a consistência de todas as suas suposições.

O método bayesiano não é aplicável apenas ao aprendizado de redes bayesianas e seus casos especiais. (Apesar de seu nome, as redes bayesianas não são necessariamente bayesianas: os frequentistas também podem aprendê-las, como acabamos de ver.) Podemos inserir uma distribuição *a priori* em qualquer classe de hipóteses – conjuntos de regras, redes neurais, programas – e então atualizá-las com sua verossimilhança de acordo com os dados. O ponto de vista dos bayesianos é o de que cabe a nós escolher a representação, mas é preciso aprendê-la usando o teorema de Bayes. Nos anos 1990, eles planejaram uma espetacular tomada de poder na Conference on Neural Information Processing Systems (NIPS), a principal vitrine da pesquisa conexionista. Os chefes do bando (por assim dizer) eram David MacKay, Radford Neal e Michael Jordan. MacKay, um inglês que foi aluno de John Hopfield no Caltech e depois se tornou consultor-chefe de ciência no Departamento de Energia da Inglaterra, mostrou como aprender da maneira bayesiana com perceptrons multicamadas. Neal apresentou os conexionistas ao MCMC, e Jordan os introduziu na inferência variacional. Para concluir, eles mostraram que no limite é possível “integrar” os neurônios em um perceptron multicamadas, deixando um tipo de modelo bayesiano que não lhes fazia referência. Não demorou muito para a palavra *neural* do título de um artigo enviado para a NIPS se tornar previsor de rejeição. Alguns pesquisadores brincaram que a conferência deveria mudar seu nome para BIPS, abreviação de Bayesian Information Processing Systems.

Markov pesa a evidência

Porém, algo engraçado aconteceu na tentativa de conquistar o mundo. Pesquisadores usuários de modelos bayesianos começaram a notar que era

possível obter melhores resultados com o ajuste das probabilidades de maneiras ilegítimas. Por exemplo, elevar *P*(*palavras*) a alguma potência no reconhecimento de voz melhorava a precisão, mas isso não era mais o teorema de Bayes. O que estava ocorrendo? Parecia que a culpa era das falsas suposições de independência que os modelos generativos fazem. A estrutura de diagrama simplificada torna os modelos algo que pode ser aprendido e que vale a pena manter, mas uma abordagem melhor é aprender apenas os parâmetros mais adequados para a tarefa em questão, não importando se forem probabilidades. A importância real do Naïve Bayes é que ele fornece um conjunto de características pequeno e informativo a partir do qual é possível prever a classe e é uma maneira rápida e robusta de aprender os parâmetros correspondentes. Em um filtro de spam, cada característica é a ocorrência de uma palavra específica do spam, e o parâmetro correspondente é a frequência com que ela acontece; o mesmo é verdadeiro para o não spam. Visto dessa forma, o Naïve Bayes pode ser ótimo, no sentido de fazer as melhores previsões possíveis, mesmo em muitos casos em que suas suposições de independência são claramente violadas. Quando percebi isso e publiquei um artigo sobre o assunto em 1996, a desconfiança que as pessoas tinham em relação ao Naïve Bayes desapareceu, ajudando-o a decolar. Contudo, também foi um passo em direção a um tipo de modelo diferente, que nas últimas duas décadas vem cada vez mais substituindo as redes bayesianas no machine learning: as redes de Markov.

A rede de Markov é um conjunto de características e pesos correspondentes, que juntos definem uma distribuição de probabilidade. Uma característica pode ser tão simples quanto *Esta é uma balada* ou tão elaborada quanto *Esta é uma balada de um artista de hip hop, com riff de saxofone e progressão de acordes descendente*. O site Pandora usa um extenso conjunto de características, que ele chama de Music Genome Project, para selecionar as canções a serem reproduzidas para o usuário. Para quem gosta de baladas, o peso da característica correspondente aumentará, e a pessoa estará mais propensa a ouvi-las ao ativar o Pandora. Se ela também gostar de artistas de hip hop, o peso dessa característica também aumentará. As canções com mais probabilidades de serem ouvidas agora serão as que têm as duas características, ou seja, baladas de artistas de hip hop. Se a pessoa não gostar de baladas ou artistas de hip hop, mas gostar desses dois estilos apenas se combinados, ela precisará da característica mais elaborada *Balada de um artista de hip hop*. As características do Pandora são criadas

manualmente, mas nas redes de Markov também podemos aprender características usando a escalada de montanha, semelhante a como ocorre na indução de regras. Seja como for, a descida de gradiente é uma boa maneira de aprender os pesos.

Como as redes bayesianas, as redes de Markov podem ser representadas por diagramas, mas eles têm arcos não direcionados em vez de setas. Duas variáveis estarão conectadas, significando que dependem diretamente uma da outra, se aparecerem juntas em alguma característica, como ocorre com *Balada* e *De um artista de hip hop* em *Balada de um artista de hip hop*.

As redes de Markov são um componente básico em muitas áreas, como a visão computacional. Por exemplo, um carro autônomo precisa segmentar cada imagem que ele vê na estrada, no céu e no campo. Uma opção é rotular cada pixel como uma das três paisagens de acordo com sua cor, mas isso não é satisfatório. As imagens têm ruídos e variam muito, e o carro verá erroneamente pedras espalhadas por toda a pista e trechos da estrada no céu. No entanto, sabemos que pixels que são vizinhos em uma imagem geralmente fazem parte do mesmo objeto, e podemos introduzir um conjunto de características correspondente: para cada par de pixels vizinhos, a característica será verdadeira se eles pertencerem ao mesmo objeto, caso contrário será falsa. Agora imagens com blocos grandes e contíguos pertencentes à estrada e ao céu terão muito mais probabilidades de ocorrer do que imagens sem esses blocos, e o carro seguirá em linha reta em vez de virar continuamente para a esquerda e a direita a fim de evitar pedras imaginárias.

As redes de Markov podem ser treinadas para maximizar a verossimilhança dos dados como um todo ou a verossimilhança condicional do que se deseja prever dado o que se sabe. Para o Siri, a verossimilhança dos dados como um todo é $P(palavras, sons)$ e a verossimilhança condicional em que estamos interessados é $P(palavras \mid sons)$. Otimizando a última, podemos ignorar $P(sons)$, que apenas nos desvia de nosso objetivo. E já que a ignoramos, podemos ser arbitrariamente complexos. Isso é muito melhor do que a suposição irreal dos HMMs de que os sons dependem somente das palavras correspondentes, sem nenhuma influência dos arredores. Na verdade, se o Siri só quisesse descobrir quais palavras você acabou de falar, talvez não precisasse se preocupar com probabilidades; teria de garantir apenas que as palavras corretas tivessem pontuação maior do que as incorretas ao somar os pesos de suas

características – por segurança, pontuação muito maior.

Os analogistas levam essa linha de raciocínio à sua conclusão lógica, como veremos no próximo capítulo. Na primeira década do novo milênio, foram eles que monopolizaram a NIPS. Agora, mais uma vez os conexionistas estão no comando, sob a bandeira do aprendizado profundo. Algumas pessoas dizem que a pesquisa se dá em ciclos, todavia ela parece mais uma espiral, com loops se formando em direção ao progresso. No machine learning, a espiral converge para o Algoritmo Mestre.

Lógica e probabilidade: o par incompatível

Pode parecer que os bayesianos e os simbolistas se dão muito bem, dado que ambos acreditam em uma abordagem do aprendizado por primeiros princípios, em vez de inspirada pela natureza. Longe disso. Os simbolistas não gostam de probabilidades e contam piadas como "Quantos bayesianos são necessários para trocar uma lâmpada? Eles não têm certeza. Pensando bem, não sabem nem se a lâmpada está queimada". Agora, falando sério, os simbolistas apontam para o alto preço que pagamos pela probabilidade. Subitamente, a inferência se torna muito mais cara, todos esses números são difíceis de entender, temos de lidar com probabilidades *a priori*, e hordas de hipóteses zumbis nos perseguem indefinidamente. A possibilidade de agregar conhecimento dinamicamente, tão cara aos simbolistas, se foi. E o pior, não sabemos como aplicar distribuições de probabilidade a muitas coisas que precisamos aprender. A rede bayesiana é uma distribuição que se estende por um vetor de variáveis, mas e quanto às distribuições por redes, bancos de dados, bases de conhecimento, linguagens, planos e programas de computador, para citar algumas? Todas elas são facilmente manipuladas em lógica, e é claro que um algoritmo que não possa aprendê-las não será o Algoritmo Mestre.

Por sua vez, os bayesianos apontam para a fragilidade da lógica. Se tivermos uma regra como *Pássaros voam*, será impossível haver um mundo sem ao menos um pássaro que não voe. Se tentarmos corrigir adicionando exceções, como *Pássaros voam, a menos que sejam pinguins*, isso nunca terá fim. (E quanto aos avestruzes? Pássaros em gaiolas? Pássaros mortos? Pássaros com asas quebradas? Asas molhadas?) Um médico o diagnostica com câncer, e você decide obter uma segunda opinião. Se o segundo médico discordar, você não terá saída. Não poderá pesar as duas opiniões; terá de acreditar em ambas. E então

acontece uma catástrofe: porcos voam, o movimento perpétuo é possível e a Terra não existe – porque em lógica tudo pode ser inferido a partir de uma contradição. Além disso, se o conhecimento for aprendido a partir de dados, nunca teremos certeza de que ele é verdadeiro. Por que os simbolistas pensam o contrário? É claro que Hume desaprovaria tamanha desatenção.

Os bayesianos e os simbolistas concordam que suposições *a priori* são inevitáveis, mas diferem nos tipos de conhecimento *a priori* que permitem. Para os bayesianos, o conhecimento entra na distribuição *a priori* que se estende pela estrutura e pelos parâmetros do modelo. Em princípio, a probabilidade *a priori* do parâmetro pode ser a que quisermos, mas ironicamente os bayesianos tendem a selecionar probabilidades *a priori* não informativas (como atribuir a mesma probabilidade a todas as hipóteses) porque é mais fácil fazer cálculos com elas. Seja qual for o caso, os humanos não são muito bons em estimar probabilidades. Para a estrutura, as redes bayesianas fornecem uma maneira intuitiva de incorporar conhecimento: desenhe uma seta de A a B se acha que A causa diretamente B. Porém, os simbolistas são muito mais flexíveis: você pode fornecer como conhecimento *a priori* para o seu aprendiz qualquer coisa que possa codificar em lógica, e praticamente qualquer coisa pode ser codificada em lógica – contanto que seja como preto e branco.

Obviamente, precisamos tanto da lógica quanto da probabilidade. A cura do câncer é um bom exemplo. Uma rede bayesiana pode modelar um único aspecto de como as células funcionam, como a regulação dos genes ou a dobradura das proteínas, mas só a lógica pode juntar todas as peças em um quadro coerente. Por outro lado, a lógica não pode lidar com informações incompletas ou com ruído, que são difusas em biologia experimental, mas as redes bayesianas podem manipulá-las tranquilamente.

O aprendizado bayesiano opera com uma única tabela de dados, em que cada coluna representa uma variável (por exemplo, o nível de expressão de um gene) e cada linha representa uma instância (que poderia ser um experimento de microarranjo, com o nível de expressão observado para cada gene). Não há problema se a tabela tiver “lacunas” e erros de medição porque podemos usar a inferência probabilística para preencher as lacunas e incluir os erros em uma média. Porém, se tivermos mais de uma tabela, o aprendizado bayesiano chegará a um impasse. Ele não saberá como, por exemplo, combinar dados de expressão de genes com dados sobre quais segmentos de DNA são convertidos em

proteínas e, por sua vez, como as formas tridimensionais dessas proteínas as fazem se fixar em diferentes partes da molécula de DNA, afetando a expressão de outros genes. Em lógica, podemos facilmente escrever regras relacionando todos esses aspectos e aprendê-las a partir de combinações relevantes de tabelas – mas somente se estas não tiverem lacunas ou erros.

Foi muito fácil combinar conexionismo e evolucionismo: basta evoluir a estrutura da rede e aprender os parâmetros por backpropagation. Todavia, unificar lógica e probabilidade é um problema muito mais difícil. Tentativas começam por volta da época de Leibniz, que foi um pioneiro em ambas. Alguns dos melhores filósofos e matemáticos dos séculos 19 e 20, como George Boole e Rudolf Carnap, trabalharam muito nisso, mas não foram longe. Mais recentemente, cientistas da computação e pesquisadores de inteligência artificial se juntaram ao grupo. Contudo, com a virada do milênio, o melhor que tivemos foram sucessos parciais, como adicionar algumas estruturas lógicas a redes bayesianas. A maioria dos especialistas acreditava ser impossível unificar lógica e probabilidade. As expectativas de criação de um Algoritmo Mestre não pareciam boas, particularmente porque nem o algoritmo evolucionário nem o conexionista podiam lidar com informações incompletas ou com múltiplos conjuntos de dados.

Felizmente, o problema foi resolvido, e agora o Algoritmo Mestre parece estar mais próximo. No Capítulo 9 veremos como ocorreu essa proeza e a traremos de lá. Mas primeiro precisamos incluir no quebra-cabeça uma peça muito importante que ainda está faltando: como aprender a partir de um número mínimo de dados. Isso parece ser desnecessário em uma época de abundância de dados, mas a verdade é que nos deparamos com frequência com grandes volumes de dados sobre algumas partes do problema que queremos resolver e quase nada sobre outras. É aí que entra uma das ideias mais importantes do machine learning: a analogia. Todas as tribos que estudamos até agora têm uma coisa em comum: aprendem um modelo explícito do fenômeno que está sendo considerado, seja um conjunto de regras, um perceptron multicamadas, um programa genético ou uma rede bayesiana. Quando elas não têm dados suficientes para fazê-lo, ficam confusas. Mas os analogistas podem aprender a partir de tão pouco quanto um exemplo porque nunca formam um modelo. Vejamos o que fazem então.

---

1 O peixe Babel é um animal fictício da série *O guia do mochileiro das galáxias*, de Douglas Adams. Ele

aparece no primeiro livro da série como uma espécie de peixe que pode fazer a tradução instantânea entre dois idiomas.

2 N.T.: A frase parece sem sentido, mas isso muda se considerarmos que “ingênuo” em inglês é naive, e o autor cria então uma correlação com o nome do classificador.

capítulo 7

Você é o que parece

Frank Abagnale Jr. é um dos vigaristas mais notórios da história. Abagnale, representado por Leonardo DiCaprio no filme de Spielberg *Prenda-me se for capaz*, falsificou cheques no valor de milhões de dólares, fingiu ser um advogado e um professor universitário e viajou pelo mundo como um piloto falso da Pan Am, tudo antes de seu vigésimo primeiro aniversário. Porém, talvez sua façanha mais surpreendente tenha sido posar de médico por quase um ano na Atlanta do fim dos anos 1960. A prática de medicina supostamente requer muitos anos de faculdade, uma licença, residência etc., mas Abagnale conseguiu burlar todas essas sutilezas e nunca foi chamado a responder por isso.

Imagine por um momento tentar tamanha proeza. Você se esgueira para dentro do consultório de um médico ausente, e em pouco tempo entra um paciente e lhe conta todos os seus sintomas. Agora você tem de diagnosticá-lo, só que não sabe nada de medicina. Tudo o que tem é um armário cheio de arquivos de pacientes: seus sintomas, diagnósticos, tratamentos pelos quais passaram e assim por diante. O que fazer? A saída mais fácil é procurar nos arquivos o paciente cujos sintomas pareçam mais com os do atual e fazer um diagnóstico. Se sua postura médica for tão convincente quanto a de Abagnale, esse esquema pode servir. A mesma ideia se aplica a áreas que vão além da medicina. Se você fosse um jovem presidente e se deparasse com uma crise mundial, como Kennedy quando uma aeronave espiã norte-americana revelou mísseis nucleares soviéticos sendo instalados em Cuba, provavelmente não haveria um roteiro pronto para ser seguido. Em vez disso, você procuraria situações históricas análogas às atuais e tentaria aprender com elas. Os chefes do Estado-Maior solicitaram com urgência um ataque a Cuba, mas Kennedy, tendo acabado de ler *Canhões de agosto*, um relato campeão em vendas da deflagração da Primeira Guerra Mundial, sabia

muito bem como seria fácil aquilo descambar para uma guerra generalizada e talvez tenha poupado o mundo de uma guerra nuclear.

A analogia foi a fagulha que deu início a muitos dos maiores avanços científicos da história. A teoria da seleção natural nasceu quando Darwin, ao ler *Ensaio sobre a população* de Malthus, ficou perplexo com os paralelos entre a luta pela sobrevivência na economia e na natureza. O modelo de átomo criado por Bohr surgiu ao ver o átomo como um sistema solar em miniatura, com elétrons como os planetas e o núcleo como o sol. Kekulé descobriu a forma anelar da molécula de benzeno após imaginar uma cobra mordendo sua própria cauda.

O raciocínio analógico tem uma linhagem intelectual distinta. Aristóteles o expressou em sua lei da semelhança: quando duas coisas são semelhantes, pensar em uma tende a nos fazer pensar na outra. Empíricos como Locke e Hume seguiram seus passos. A verdade, disse Nietzsche, é um exército móvel de metáforas. Kant também era um entusiasta. William James acreditava que "esse senso de similaridade é o leme e a espinha dorsal de nosso raciocínio". Alguns psicólogos contemporâneos argumentam até que a cognição humana em sua totalidade é uma teia de analogias. Dependemos dela para saber andar em uma cidade desconhecida e para entender expressões como "ver a luz" e "de cabeça erguida". Adolescentes que usam "tipo" em cada frase que dizem provavelmente, tipo, concordariam que essa analogia é importante, cara.

Dado tudo isso, não é de surpreender que a analogia desempenhe um papel proeminente no machine learning. No entanto, ela começou a participar de maneira lenta e inicialmente foi ofuscada pelas redes neurais. Sua primeira versão algorítmica apareceu em um obscuro relatório técnico escrito em 1951 por dois estatísticos de Berkeley, Evelyn Fix e Joe Hodges, e só foi publicada em um periódico popular décadas depois. Porém, nesse meio tempo, outros artigos sobre o algoritmo de Fix e Hodges começaram a aparecer e a se multiplicar até ele se tornar o mais pesquisado em toda a ciência da computação. O algoritmo do vizinho mais próximo, como é chamado, é a primeira parada de nosso passeio pelo aprendizado baseado em analogia. A segunda serão as máquinas de vetores de suporte, uma ideia que tomou o machine learning de assalto na virada do milênio e só recentemente foi ofuscada pelo aprendizado profundo. A terceira e última é o raciocínio analógico totalmente desenvolvido, que há várias décadas é um componente básico da psicologia e da inteligência artificial e é objeto de

estudo quase pelo mesmo tempo no machine learning.

A tribo dos analogistas é a menos coesa entre as cinco tribos. Ao contrário das outras, que têm uma forte identidade e ideais em comum, eles são mais como um conjunto de pesquisadores independentes, unidos apenas pela crença em julgamentos de similaridade como base do aprendizado. Alguns, como os adeptos da máquina de vetores de suporte, podem até desaprovar serem considerados parte desse grupo. Porém, não param de surgir modelos profundos, e acho que eles se beneficiariam muito caso se unissem em torno dessa causa comum. A similaridade é uma das ideias centrais do machine learning, e analogistas de todos os tipos são seus guardiões. Talvez em uma década futura, o machine learning seja dominado pela analogia profunda, combinando no mesmo algoritmo a eficiência do vizinho mais próximo, a sofisticação matemática das máquinas de vetores de suporte e o poder e a flexibilidade do raciocínio analógico. (Acabei de revelar um de meus projetos de pesquisa secretos.)

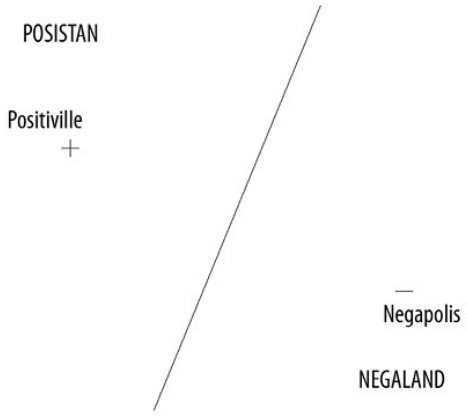
Iguale-se se for capaz

O vizinho mais próximo é o algoritmo de aprendizado mais simples e rápido já inventado. Na verdade, poderíamos dizer que é o algoritmo mais rápido de qualquer tipo que já foi inventado. Ele consiste em não fazer absolutamente nada e, portanto, leva zero hora para ser executado. Não é possível vencer isso. Se você quiser aprender a reconhecer faces e tiver um vasto banco de dados de imagens rotuladas como face/não face, apenas deixe-o parado lá. Não se preocupe, seja feliz. Sem sabê-lo, essas imagens já formarão implicitamente um modelo do que é uma face. Suponhamos que você fosse o Facebook e quisesse identificar automaticamente faces em fotos que as pessoas enviaram como um passo anterior a serem marcadas com os nomes de seus amigos. É bom não ser preciso fazer nada, dado que os usuários do Facebook fazem o upload de mais de trezentos milhões de fotos por dia. A aplicação de qualquer um dos aprendizes que vimos até agora a essas fotos, com a possível exceção do Naïve Bayes, demandaria vários computadores. E o Naïve Bayes não é suficientemente inteligente para reconhecer faces.

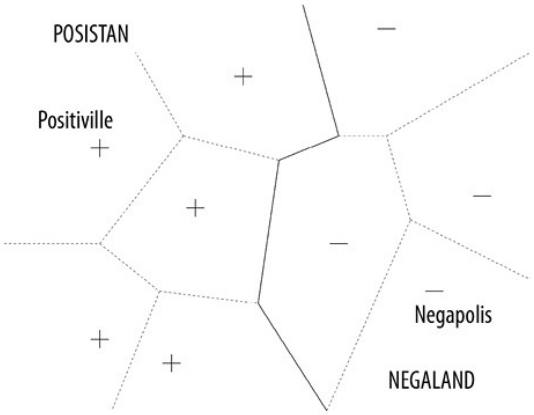
É claro que há um preço a pagar, e ele surge na hora da prova. Jane User acabou de fazer o upload de uma nova imagem. É uma face? A resposta do vizinho mais próximo é: encontre a imagem mais semelhante no banco de dados de fotos rotuladas do Facebook – seu “vizinho mais próximo” –, e se ela contiver

uma face, essa também terá. Muito simples, mas agora você tem de vasculhar potencialmente bilhões de fotos em (idealmente) uma fração de segundo. Como um aluno preguiçoso que não se preocupa em estudar para a prova, o vizinho mais próximo é pego de surpresa e tem de se virar. Mas ao contrário do que ocorre na vida real, em que sua mãe o ensinou a nunca deixar para amanhã o que pode fazer hoje, no machine learning a procrastinação pode compensar. Na verdade, todo o gênero de aprendizado do qual o vizinho mais próximo faz parte é às vezes chamado de "aprendizagem preguiçosa", e nesse contexto não há nada de pejorativo no termo.

A razão para os aprendizes preguiçosos serem um pouco mais inteligentes do que parecem é que seus modelos, embora implícitos, podem na verdade ser extremamente sofisticados. Considere o caso extremo em que tivéssemos apenas uma instância de cada classe. Por exemplo, gostaríamos de descobrir onde fica a fronteira entre dois países, mas só conhecemos a localização de suas capitais. A maioria dos aprendizes ficaria confusa, mas o vizinho mais próximo adivinha facilmente que a fronteira é uma linha reta disposta a meio caminho entre as duas cidades:



Os pontos da linha estão à mesma distância das duas capitais; os pontos à esquerda da linha estão mais próximos de Positiville, logo o algoritmo do vizinho mais próximo presume que eles façam parte de Posistan e vice-versa. É claro que o ideal seria que essa fosse uma fronteira exata, mas uma aproximação provavelmente é bem melhor do que nada. No entanto, se soubéssemos da existência de muitas cidades nos dois lados da fronteira é que as coisas ficariam interessantes:



O algoritmo do vizinho mais próximo pode formar implicitamente uma fronteira muito intrincada, ainda que tudo que faça seja lembrar onde ficam as cidades e atribuir os pontos de acordo! Podemos considerar a “região metropolitana” de uma cidade como todos os pontos que estejam mais próximos dela do que de qualquer outra cidade; os limites entre as regiões metropolitanas são mostrados como linhas tracejadas no diagrama. Agora Posistan é apenas a união das regiões metropolitanas de todas as suas cidades, como o é Negaland. Por outro lado, uma árvore de decisão (por exemplo) só poderia formar fronteiras que se estendessem alternadamente de norte a sul e de leste a oeste, o que provavelmente não ficaria muito próximo da fronteira real. Logo, mesmo que os aprendizes de árvore de decisão sejam mais “ávidos” e tentem arduamente no momento do aprendizado descobrir onde ficam as fronteiras, o algoritmo “preguiçoso” do vizinho mais próximo acaba sendo melhor.

A aprendizagem preguiçosa vence porque é muito mais difícil formar um modelo global, como uma árvore de decisão, do que apenas descobrir individualmente onde ficam os pontos específicos da consulta. Imagine tentar definir o que é uma face com uma árvore de decisão. Poderíamos dizer que ela tem dois olhos, um nariz e uma boca, mas o que é um olho e como o encontrar em uma imagem? E se os olhos da pessoa estiverem fechados? Definir com segurança uma face no nível de cada pixel é muito difícil, principalmente dadas as diferentes expressões, poses, contextos e condições de iluminação com as quais ela pode aparecer. Em vez disso, o vizinho mais próximo toma um atalho: se a imagem do banco de dados mais semelhante àquela que Jane acabou de inserir for de uma face, a de Jane também será. Para esse esquema funcionar, o banco de dados precisa conter uma imagem que seja suficientemente semelhante à nova – por exemplo, uma face com pose, iluminação *etc.* semelhante –,

portanto quanto maior o banco de dados, melhor. No caso de um problema bidimensional como descobrir a fronteira entre dois países, um banco de dados pequeno é o bastante. Para um problema muito difícil como identificar faces, em que a cor de cada pixel é uma dimensão de variação, é preciso um banco de dados enorme. Mas atualmente nós os temos. Aprender a partir deles pode ser muito custoso para um aprendiz ávido, que trace explicitamente o limite entre faces e não faces. Porém, para o vizinho mais próximo, o limite está implícito nas localizações dos pontos de dados e na medida das distâncias, e o único custo é o tempo da consulta.

A ideia de formar um modelo local em vez de global não se aplica apenas à classificação. Os cientistas costumam usar regressão linear para prever variáveis contínuas, mas a maioria dos fenômenos é não linear. Por sorte, no âmbito local eles são lineares porque é possível obter com uma linha reta algo que localmente se aproxime de uma curva suave. Logo, se em vez de tentar adequar uma linha reta a todos os dados a adequarmos somente aos pontos próximos do ponto da consulta, teremos um algoritmo muito poderoso de regressão não linear. A preguiça compensa. Se Kennedy precisasse de uma teoria de relações internacionais completa para decidir o que fazer com os mísseis soviéticos em Cuba, problemas ocorreriam. Em vez disso, ele viu uma analogia entre a crise e a deflagração da Primeira Guerra Mundial, e essa analogia o direcionou para as decisões corretas.

O vizinho mais próximo pode salvar vidas, como Steven Johnson relatou em *O mapa fantasma*. Em 1854, Londres enfrentou uma epidemia de cólera, que matou uma entre oito pessoas em partes da cidade. A teoria prevalente na época de que a cólera era causada pelo “ar ruim” não ajudou a impedir sua disseminação. Mas John Snow, um físico que não acreditava na teoria, teve uma ideia melhor. Ele marcou em um mapa de Londres os locais de todos os casos conhecidos da doença e o dividiu nas regiões mais próximas a cada fonte de água pública. Heureca: quase todas as mortes ocorreram na “região metropolitana” de uma fonte específica, localizada na Broad Street no bairro do Soho. Inferindo que a água contida nesse poço estava contaminada, Snow convenceu os habitantes locais a desativarem a fonte, e a epidemia cessou. Esse episódio originou a ciência da epidemiologia, mas também foi o primeiro sucesso do algoritmo do vizinho mais próximo – quase um século antes de sua invenção oficial.

Com o uso do vizinho mais próximo, cada ponto de dados é seu próprio classificador e prevê a classe para todos os exemplos de consulta em que é vencedor. O vizinho mais próximo é como um exército de formigas, em que cada soldado faz pouco individualmente, mas que quando se unem podem mover montanhas. Quando a carga de uma formiga é muito pesada, ela pode compartilhá-la com seus vizinhos. Com o mesmo espírito, no algoritmo dos *k* vizinhos mais próximos, um exemplo de teste é classificado ao localizarmos seus *k* vizinhos mais próximos e os deixarmos votar. Se a imagem mais próxima do novo upload for uma face, mas as outras duas mais próximas não o forem, os três vizinhos mais próximos decidirão se o novo upload não é uma face. O algoritmo do vizinho mais próximo é propenso ao sobreajuste: se a classe estiver errada para um ponto de dados, ela será disseminada para toda a região vizinha. O algoritmo dos *k* vizinhos mais próximos é mais robusto porque só erra quando a maioria dos *k* vizinhos mais próximos apresenta ruído. É claro que o preço é sua visão ser menos clara: detalhes finos da fronteira são deixados de lado pela votação. Quando *k* aumenta, a variância diminui, mas a tendenciosidade é maior.

Não basta usar os *k* vizinhos mais próximos em vez de um único vizinho. Os exemplos mais próximos do exemplo de teste devem intuitivamente valer mais. Isso nos leva ao algoritmo da média ponderada dos *k* vizinhos mais próximos. Em 1994, uma equipe de pesquisadores da Universidade de Minnesota e do MIT construiu um sistema de recomendação baseado no que eles chamaram de “ideia enganosamente simples”: pessoas que concordaram no passado tendem a concordar mais uma vez no futuro. Essa noção levou diretamente aos sistemas de filtragem colaborativa que todos os sites respeitáveis de e-commerce têm. Suponhamos que, como ocorre no Netflix, você reunisse um banco de dados de classificações de filmes, com cada usuário dando uma pontuação de uma a cinco estrelas para os filmes que viu. Você quer decidir se o usuário Ken gostará de *Gravidade*, logo encontra os usuários cujas classificações passadas estejam mais próximas às dele. Se todos tiverem dado pontuações altas a *Gravidade*, provavelmente Ken também dará, e você poderá recomendar o filme. No entanto, se eles discordarem quanto a *Gravidade*, você precisará de um ponto alternativo, que nesse caso será classificar os usuários de acordo com o nível de concordância com Ken. Assim, se o nível de concordância de Lee em relação a Ken for mais alto do que o de Meg, de modo correspondente suas classificações devem valer mais. A classificação prevista de Ken será então a média ponderada

da de seus vizinhos, com o peso de cada vizinho sendo seu coeficiente de concordância com Ken.

Porém, há um truque interessante. Suponhamos que Lee e Ken tivessem gostos muito semelhantes, mas Lee fosse mais exigente do que Ken. Sempre que Ken dá a um filme cinco estrelas, Lee dá três; quando Ken dá três, Lee dá uma, e assim por diante. Gostaríamos de usar as classificações de Lee para prever as de Ken, todavia se o fizermos diretamente, sempre teremos uma discrepância de duas estrelas. Em vez disso, o que precisamos fazer é prever quanto as classificações de Ken estarão acima ou abaixo de sua média, de acordo com quanto as de Lee estiverem. E agora, já que Ken está sempre duas estrelas acima de sua média quando Lee está duas estrelas acima da sua, nossas previsões estarão certas.

A propósito, você não precisa de classificações explícitas para fazer a filtragem colaborativa. Se Ken pediu um filme no Netflix, isso significa que espera gostar dele. Logo, as "classificações" podem apenas ser do tipo pediu/não pediu, e dois usuários serão semelhantes se tiverem pedido muitos filmes iguais. Mesmo clicar em algo demonstra interesse implícito. O vizinho mais próximo opera com todos esses esquemas. Atualmente, todos os tipos de algoritmos são usados para recomendar itens para os usuários, mas o algoritmo da média ponderada dos *k* vizinhos mais próximos foi o primeiro a ser amplamente empregado, e ainda é difícil derrotá-lo.

Os sistemas recomendadores, como também são chamados, constituem um grande negócio: um terço das vendas da Amazon vem de suas recomendações, assim como três quartos das vendas do Netflix. Estamos muito distantes da época da criação do algoritmo do vizinho mais próximo, quando ele foi considerado inviável devido aos seus requisitos de memória. Naquela época, as memórias dos computadores eram feitas de pequenos anéis de ferro, um por bit, e mesmo o armazenamento de alguns milhares de exemplos as sobrecarregavam. Como os tempos mudaram! Entretanto, não é tão inteligente se lembrar de todos os exemplos vistos e então ter de percorrê-los, principalmente porque a maioria deve ser irrelevante. Se você voltar ao mapa de Posistan e Negaland, notará que se Positiville desaparecer nada mudará. As regiões metropolitanas de cidades próximas se expandirão para o terreno anteriormente ocupado por Positiville, mas já que são todas cidades de Posistan, a fronteira com Negaland permanecerá a mesma. As únicas cidades que realmente importam são as localizadas do outro

lado da fronteira com uma cidade do outro país; podemos omitir todas as demais. Logo, uma maneira simples de tornar o vizinho mais próximo mais eficiente é excluir todos os exemplos que estejam classificados corretamente por seus vizinhos. Esse e outros truques permitem que métodos do vizinho mais próximo sejam usados em algumas áreas surpreendentes, como o controle de braços robôs em tempo real. Porém, é evidente que eles não são a primeira opção para coisas como negociações de alta frequência, em que computadores compram e vendem ações em frações de segundo. Em uma corrida entre uma rede neural, que só pode ser aplicada a um exemplo com um número fixo de adições, multiplicações e sigmoides, e um algoritmo que precise procurar em um grande banco de dados os vizinhos mais próximos do exemplo, certamente a rede neural venceria.

Outra razão para inicialmente os pesquisadores não acreditarem no vizinho mais próximo era o fato de não estar claro se ele poderia aprender os limites reais entre os conceitos. Porém, em 1967, Tom Cover e Peter Hart provaram que, se receber dados suficientes, na pior das hipóteses ele é apenas duas vezes mais propenso a erros do que o melhor classificador imaginável. Se, digamos, pelo menos 1% dos exemplos de teste for inevitavelmente classificado de maneira incorreta devido ao ruído nos dados, então o vizinho mais próximo garante estar no máximo 2% errado. Essa foi uma revelação significativa. Até então, todos os classificadores conhecidos presumiam que a fronteira tinha uma forma muito específica, normalmente uma linha reta. Era uma faca de dois gumes: por um lado, dava provas de ser possível obter exatidão, como no caso do perceptron, mas também significava que o classificador era estritamente limitado no que podia aprender. O vizinho mais próximo foi o primeiro algoritmo da história a se beneficiar de quantidades de dados ilimitadas para aprender conceitos arbitrariamente complexos. Nenhum ser humano poderia esperar rastrear as fronteiras que ele forma no hiperespaço a partir de milhões de exemplos, mas graças à prova de Cover e Hart sabemos que elas ficam próximas do correto. De acordo com Ray Kurzweil, a Singularidade começa quando não conseguimos mais entender o que os computadores fazem. Por esse padrão, não é totalmente errado dizer que ela já está ocorrendo – começou ainda em 1951, quando Fix e Hodges inventaram o vizinho mais próximo, o pequeno algoritmo que entende.

A maldição da dimensionalidade

É claro que há uma serpente nesse Éden. Ela se chama maldição da dimensionalidade e, embora afete todos os aprendizes em maior ou menor grau, é particularmente ruim para o vizinho mais próximo. Em poucas dimensões (como duas ou três), o vizinho mais próximo costuma funcionar muito bem. À medida que o número de dimensões aumenta, as coisas começam a falhar rapidamente. Hoje em dia não é raro o aprendizado ocorrer a partir de milhares ou até mesmo milhões de atributos. Para um site de e-commerce tentando aprender nossas preferências, cada clique dado é um atributo. É o que também acontece com cada palavra de uma página web e cada pixel de uma imagem. Contudo, mesmo com apenas dezenas ou centenas de atributos, há chances de que o vizinho mais próximo tenha problemas. O primeiro problema é que a maioria dos atributos é irrelevante: você pode conhecer um milhão de factoides sobre Ken, mas é provável que poucos digam algo sobre o risco (por exemplo) de ele contrair câncer de pulmão. Ainda que saber que ele fuma seja crucial para fazer essa previsão específica, talvez não ajude a decidir se gostará de *Gravidade*. De sua parte, os métodos simbolistas são muito bons em descartar atributos irrelevantes. Se um atributo não tiver informações sobre a classe, ele nunca será incluído na árvore de decisão ou conjunto de regras. Todavia, o vizinho mais próximo fica confuso com atributos irrelevantes porque eles contribuem como um todo para a semelhança entre exemplos. Com atributos irrelevantes, a semelhança acidental nas dimensões inaplicáveis faz desaparecer semelhanças significativas nas importantes, e o vizinho mais próximo torna-se quase o mesmo que uma adivinhação aleatória.

Um problema maior é que, surpreendentemente, a existência de mais atributos pode ser danosa mesmo quando todos são relevantes. Costumamos achar que é sempre melhor haver mais informações – não é esse o lema de nossa época? Mas à medida que o número de dimensões aumenta, o número de exemplos de treinamento necessários para a localização das fronteiras do conceito cresce de maneira exponencial. Com vinte atributos booleanos, há aproximadamente um milhão de diferentes exemplos possíveis. Com vinte e um, há dois milhões e um número correspondente de maneiras pelas quais a fronteira poderia se estender entre eles. Cada atributo adicional torna o problema do aprendizado duas vezes mais difícil, e isso apenas com atributos booleanos. Se o atributo for altamente informativo, o benefício de adicioná-lo pode compensar o custo. Porém, se você tiver apenas atributos pouco informativos, como as palavras de um email ou os

pixels de uma imagem, pode ter problemas, mesmo se coletivamente eles tiverem informações suficientes para prever o desejado.

E fica pior. O vizinho mais próximo se baseia na busca de objetos semelhantes, e com mais dimensões a própria noção de similaridade se perde. O hiperespaço é como a Zona Crepuscular. As intuições que temos ao viver em três dimensões deixam de ser aplicáveis e coisas estranhas começam a ocorrer. Considere uma laranja: uma bola de polpa saborosa rodeada por uma fina casca. Digamos que 90% do raio da laranja fossem ocupados por polpa e os outros 10% por casca. Isso significa que 73% do volume da laranja são de polpa ($0{,}9^3$). Agora considere uma hiperlaranja: ainda com 90% do raio ocupados por polpa, mas, por exemplo, em uma centena de dimensões. A polpa diminui para apenas cerca de três milésimos de um por cento do volume da hiperlaranja ($0{,}9^{100}$). A hiperlaranja é toda casca, e você nunca terminará de descascá-la!

Outro exemplo perturbador é o que ocorre com a nossa boa e velha distribuição normal, também conhecida como curva de sino. O que a distribuição normal diz é que os dados estão localizados em um ponto (a média da distribuição), mas com alguma variância ao redor (gerada pelo desvio-padrão). Certo? Não no hiperespaço. Com uma distribuição normal de muitas dimensões, é mais provável que você obtenha uma amostra longe da média e não perto dela. Uma curva de sino no hiperespaço se parece mais com uma rosquinha do que com um sino. E quando o algoritmo do vizinho mais próximo entra em ação nesse mundo desordenado, ele fica confuso. Todos os exemplos parecem igualmente semelhantes, mas ao mesmo tempo estão distantes demais uns dos outros para fazer previsões úteis. Se você espalhar exemplos de maneira uniforme e aleatória dentro de um hipercubo com muitas dimensões, a maioria ficará mais perto de uma face do cubo e não de seu vizinho mais próximo. Em mapas medievais, áreas inexploradas eram marcadas com dragões, serpentes marinhas e outras criaturas fantásticas, ou apenas com a expressão *aqui há dragões* (here be dragons). No hiperespaço, os dragões estão em todos os lugares, inclusive na porta da frente. Tente ir até a casa de seu vizinho e nunca chegará lá; ficará para sempre perdido em terras estranhas, perguntando-se para onde foi tudo que lhe era familiar.

As árvores de decisão também não estão imunes à maldição da dimensionalidade. Digamos que o conceito que você estivesse tentando aprender fosse uma esfera: pontos dentro dela são positivos e pontos do lado de fora são

negativos. Uma árvore de decisão pode se aproximar de uma esfera pelo menor cubo que conseguir conter. Não é perfeito, mas também não é tão ruim: só os cantos do cubo serão classificados incorretamente. Porém, com mais dimensões, quase todo o volume do hipercubo fica fora da hiperesfera. Para cada exemplo classificado corretamente como positivo, vários exemplos negativos serão classificados incorretamente como positivos, fazendo a precisão diminuir.

Na verdade, nenhum aprendiz está imune à maldição da dimensionalidade. Ela é o segundo pior problema do machine learning, depois do sobreajuste. O termo *maldição da dimensionalidade* foi cunhado por Richard Bellman, estudioso da teoria do controle, nos anos 1950. Ele observou que algoritmos de controle que funcionavam bem em três dimensões se tornavam extremamente ineficientes em espaços com mais dimensões, como quando tentamos controlar cada junta de um braço robô ou cada botão de uma fábrica de produtos químicos. Mas no machine learning o problema não é só o custo computacional – é o fato de o próprio aprendizado se tornar cada vez mais difícil à medida que a dimensionalidade aumenta.

No entanto, nem tudo está perdido. A primeira coisa que podemos fazer é nos livrar das dimensões irrelevantes. As árvores de decisão fazem isso automaticamente calculando o ganho de informações de cada atributo e usando apenas os mais informativos. Com o vizinho mais próximo, podemos usar algo semelhante descartando todos os atributos cujo ganho de informações fique abaixo de um limite e então medindo a similaridade somente no espaço reduzido. Esse esquema é rápido e satisfatório para algumas aplicações, mas infelizmente impede o aprendizado de muitos conceitos, como a função exclusive-OR: se um atributo só informar algo sobre a classe quando combinado com outros, mas não individualmente, ele será descartado. Uma opção mais custosa, porém mais inteligente, é “anexar” a seleção de atributos ao redor do próprio aprendiz, com uma pesquisa de escalada de montanha que se mantenha excluindo atributos contanto que não prejudique a precisão do vizinho mais próximo para dados retidos. Newton fez uso intensivo da seleção de atributos quando decidiu que a única coisa que importava para a previsão da trajetória de um objeto era sua massa – e não sua cor, cheiro, idade ou várias outras propriedades. Na verdade, o mais importante sobre uma equação são as quantidades que não aparecem nela: quando sabemos quais são os elementos básicos, descobrir como eles dependem uns dos outros com frequência é a parte

mais fácil.

Para manipular atributos pouco relevantes, uma opção é aprender seus pesos. Em vez de permitir que a similaridade tenha o mesmo valor ao longo de todas as dimensões, “reduzimos” o das menos importantes. Suponhamos que os exemplos de treinamento fossem pontos em uma sala e a dimensão correspondente à altura não tivesse tanta importância para nossos fins. Seu descarte traria todos os exemplos para o chão. Diminuir o peso seria semelhante a dar à sala um teto mais baixo. O peso de um ponto ainda conta no cálculo de sua distância de outros pontos, mas menos que sua posição horizontal. E como em vários outros aspectos do machine learning, é possível aprender pesos de atributos pela descida de gradiente.

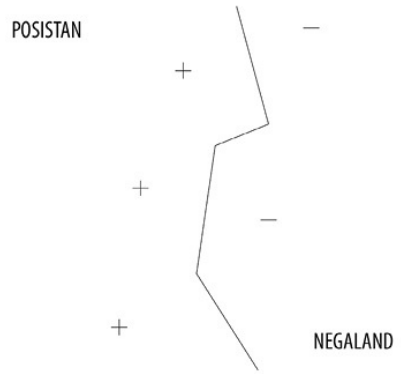
Pode ocorrer de a sala ter um teto alto, mas os pontos de dados estarem todos perto do chão, como uma fina camada de pó sobre o tapete. Se for assim, tivemos sorte: o problema parece tridimensional, mas na verdade está mais para bidimensional. Não precisamos diminuir o peso porque a natureza já o diminuiu para nós. Essa “dádiva da não uniformidade”, pela qual os dados não se disseminam de maneira uniforme no (hiper) espaço, é o que costuma salvar o dia. Os exemplos podem ter mil atributos, porém todos “residem” em um espaço com bem menos dimensões. Por exemplo, é por isso que o vizinho mais próximo pode ser bom para o reconhecimento de dígitos manuscritos: cada pixel é uma dimensão, logo existem várias, mas só uma pequena fração de todas as imagens possíveis são dígitos, e todos eles residem em um pequeno e aconchegante canto do hiperespaço. No entanto, a forma do espaço de menos dimensões em que os dados residem pode ser bastante excêntrica. Por exemplo, se uma sala tiver mobília, o pó não se acumulará apenas no chão; ele se acumulará no tampo das mesas, no assento das cadeiras, nos cobertores das camas e em coisas semelhantes. Se pudermos descobrir a forma aproximada da camada de poeira que cobre a sala, só precisaremos das coordenadas de cada ponto dela. Como veremos no próximo capítulo, há todo um subcampo do machine learning dedicado à, por assim dizer, descoberta das formas de camadas pelo agrupamento na escuridão do hiperespaço.

Cobras em um avião

Até o meio dos anos 1990, o aprendiz analógico mais amplamente usado era o do vizinho mais próximo, mas ele foi ofuscado por seus primos mais glamorosos

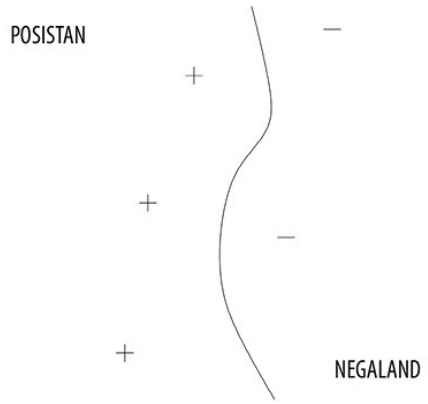
das outras tribos. Então, surgiu um novo algoritmo baseado na similaridade que ofuscou todos os anteriores. Na verdade, podemos considerá-lo como outro "produto da paz" do fim da Guerra Fria. As máquinas de vetores de suporte, ou SVMs (support vector machines), foram inventadas por Vladimir Vapnik, um frequentista soviético. Vapnik passou grande parte de sua carreira no Institute of Control Sciences em Moscou, mas em 1990, quando a União Soviética se desmembrou, emigrou para os Estados Unidos, onde ingressou na lendária Bell Labs. Enquanto estava na Rússia, Vapnik estava satisfeito em lidar com teoria, trabalhando com caneta e papel, porém a atmosfera na Bell Labs era diferente. Os pesquisadores queriam resultados práticos, e Vapnik acabou transformando suas ideias em um algoritmo. Em alguns anos, ele e seus colegas da Bell Labs desenvolveram as SVMs, e não demorou muito para elas estarem em todos os lugares, definindo novos recordes de precisão.

Superficialmente, uma SVM se parece muito com a média ponderada dos *k* vizinhos mais próximos: a fronteira entre as classes positivas e negativas é definida por um conjunto de exemplos e seus pesos, junto com uma medida de similaridade. Um exemplo de teste pertence à classe positiva se, em média, parecer mais com os exemplos positivos do que com os negativos. A média é ponderada e a SVM só considera os exemplos-chave necessários para a definição da fronteira. Se lembrarmos do caso das terras de Posistan/Negaland, quando descartamos todas as cidades que não ficavam na fronteira, tudo que sobrou foi este mapa:

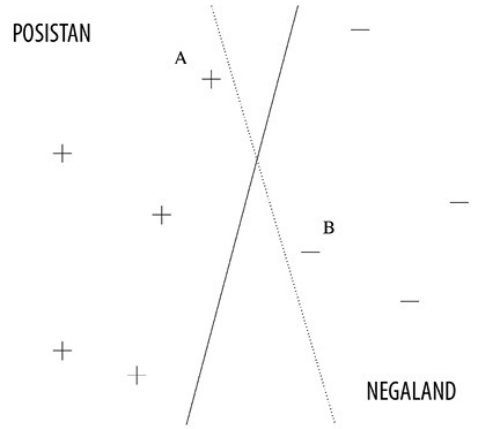


Esses exemplos são chamados de vetores de suporte porque constituem os vetores que "sustentam" a fronteira: remova um, e uma seção da fronteira deslizará para um local diferente. Também é possível notar que a fronteira é uma linha denteada, com viradas repentinas que dependem do local exato dos exemplos. Conceitos reais tendem a ter fronteiras mais suaves, o que significa

que a aproximação obtida com o algoritmo do vizinho mais próximo provavelmente não seja ideal. No entanto, com as SVMs podemos aprender fronteiras mais suaves, mais parecidas com esta:



Para aprender uma SVM, precisamos selecionar os vetores de suporte e seus pesos. Geralmente, a medida de similaridade, que no universo das SVMs se chama kernel, é selecionada *a priori*. Um dos principais insights de Vapnik foi o de que nem todas as fronteiras que separam os exemplos de treinamento positivos dos negativos são criadas da mesma forma. Suponhamos que Posistan e Negaland estivessem em guerra, e um território não ocupado com campos minados dos dois lados os separasse. Sua missão é explorar o território não ocupado, caminhando de um lado a outro sem pisar em nenhuma mina. Felizmente, você tem um mapa de onde elas estão enterradas. É claro que não é possível seguir qualquer caminho: é preciso atribuir às minas o espaço mais amplo possível. É isso que as SVMs fazem, com os exemplos como minas e a fronteira aprendida como o caminho selecionado. A margem de segurança será o mais perto que a fronteira chegar de um exemplo, e a SVM selecionará os vetores de suporte e pesos que gerarem a maior margem possível. Por exemplo, a sólida fronteira em linha reta desta figura é melhor que a pontilhada:



A fronteira pontilhada separa bem os exemplos positivos e negativos, mas fica perigosamente perto das minas terrestres localizadas em A e B. Esses exemplos são vetores de suporte: exclua um deles e a fronteira de margem máxima se moverá para um local diferente. Em geral, a fronteira é curva, tornando a margem mais difícil de divisar, contudo podemos visualizá-la como uma cobra deslizando pelo território não ocupado, sendo a margem o quanto ela é gorda. Se uma cobra muito gorda puder deslizar por toda a extensão sem explodir, a SVM conseguirá separar muito bem os exemplos positivos e negativos, e Vapnik mostrou que nesse caso podemos ter certeza de que ela não cometeu sobreajuste. Intuitivamente, se compararmos com uma cobra fina, há menos caminhos para uma cobra gorda percorrer evitando as minas terrestres; da mesma forma, se compararmos com uma SVM de pequena margem, uma de margem maior terá menos chances de cometer sobreajuste traçando uma fronteira excessivamente intrincada.

A segunda parte da história é como a SVM encontra a cobra mais gorda que caiba entre as minas terrestres positivas e negativas. À primeira vista, pode parecer que aprender um peso para cada exemplo de treinamento pela descida de gradiente seja satisfatório. Tudo que temos de fazer é encontrar os pesos que maximizem a margem, e qualquer exemplo que acabe ficando com peso zero poderá ser descartado. É uma pena, mas isso faria apenas os pesos crescerem sem limite, porque, matematicamente, quanto maiores os pesos, maior a margem. Se você estiver a trinta centímetros e meio de uma mina e dobrar o tamanho de tudo, inclusive o seu, agora estará a sessenta e um centímetros da mina, o que não diminuirá sua probabilidade de pisar nela. O que temos de fazer é maximizar a margem obedecendo à restrição de que os pesos só possam aumentar até algum valor fixo. Ou, igualmente, podemos minimizar os pesos

contanto que todos os exemplos tenham uma margem específica, que poderia ser igual a 1 – o valor preciso é arbitrário. Geralmente é isso que as SVMs fazem.

A otimização restrita é o problema que ocorre quando maximizamos ou minimizamos uma função sujeita a limites. O universo maximiza a entropia contanto que a energia se mantenha constante. Problemas desse tipo se encontram dispersos pelos negócios e pela tecnologia. Por exemplo, podemos maximizar o número de produtos que uma fábrica gera, dependendo do número de ferramentas disponíveis, das especificações dos produtos, e assim por diante. Com o advento das SVMs, a otimização restrita também se tornou crucial para o machine learning. A otimização irrestrita chega ao topo da montanha, e é isso que a descida (ou, nesse caso, a subida) de gradiente faz. A otimização restrita sobe o máximo que pode seguindo a estrada. Quando a estrada segue até o ponto máximo, os problemas restritos e irrestritos têm a mesma solução. No entanto, o que ocorre com mais frequência é a estrada serpentear subindo a montanha e então descer sem alcançar o topo. Você sabe que alcançou o ponto mais alto quando não consegue subir mais sem dirigir fora da estrada; em outras palavras, quando o caminho até o topo está em ângulo reto com a estrada. Se a estrada e o caminho até o topo formarem um ângulo oblíquo, podemos chegar mais alto dirigindo afastados da estrada, mesmo se isso não nos levar tão rapidamente quanto seguir em linha reta até o topo da montanha. Logo, o caminho para resolver um problema de otimização restrita é seguir não o gradiente, mas a parte dele paralela à superfície da restrição – nesse caso a estrada – e parar quando essa parte for zero.

Em geral, temos de lidar com várias restrições ao mesmo tempo (uma para cada exemplo, no caso das SVMs). Suponhamos que você quisesse chegar o mais próximo possível do Polo Norte, mas não pudesse sair de sua sala. Cada uma das quatro paredes da sala é uma restrição, e a solução é seguir a bússola até se deparar com o canto em que as paredes nordeste e noroeste se encontram. Dizemos que essas duas paredes são as restrições ativas por serem elas que o impedem de alcançar o ponto ótimo, a saber, o Polo Norte. Se sua sala tiver uma parede voltada exatamente para o norte, ela será a única restrição ativa, e a solução estará em um ponto no meio dela. Caso você fosse o Papai Noel e a sala já se encontrasse sobre o Polo Norte, todas as restrições seriam inativas, permitindo-lhe pensar apenas no problema de uma distribuição de brinquedos ótima. (Para vendedores ambulantes isso é mais fácil que para o Papai Noel.) Em

uma SVM, as restrições ativas são os vetores de suporte porque sua margem já é a menor permitida; mover a fronteira violaria uma ou mais restrições. Todos os outros exemplos são irrelevantes e seu peso é zero.

Na verdade, normalmente deixamos as SVMs violarem algumas restrições, o que significa classificar exemplos incorretamente ou por um valor menor que a margem, porque de outro modo elas cometeriam sobreajuste. Se houver um exemplo negativo incômodo em algum local no meio da região positiva, não queremos que a fronteira serpenteie dentro dela só para que o exemplo fique correto. Porém, a SVM sofre uma penalidade para cada exemplo que calcula errado, o que a encoraja a mantê-los em um nível mínimo. As SVMs são como os vermes da areia do filme *Duna*: grandes, fortes e podem sobreviver a algumas explosões se passarem sobre minas terrestres, contanto que não sejam muitas.

Tentando encontrar aplicações, Vapnik e seus colaboradores se depararam rapidamente com o reconhecimento de dígitos manuscritos, no qual seus colegas conexionistas da Bell Labs eram os maiores especialistas do mundo. Para a surpresa de todos, as recém-criadas SVMs se saíram tão bem quanto os perceptrons multicamadas elaborados com cuidado durante anos para o reconhecimento de dígitos. Isso preparou o terreno para a longeva e ampla competição entre os dois. As SVMs podem ser vistas como uma generalização do perceptron, porque o que obtemos com o uso de uma medida de similaridade específica (o produto dos pontos entre os vetores) é um limite de hiperplano entre as classes. Mas as SVMs têm uma vantagem importante se comparada com os perceptrons multicamadas: os pesos têm um único optimum em vez de vários local optima (soluções ótimas) e, portanto, é muito mais fácil aprendê-los com segurança. Mesmo assim, elas não são menos expressivas que os perceptrons multicamadas; os vetores de suporte agem como uma camada oculta e sua média ponderada como a camada de saída. Por exemplo, uma SVM pode representar com facilidade a função exclusive-OR com um vetor de suporte para cada uma das quatro configurações possíveis. Porém, os conexionistas não desistiram sem luta. Em 1995, Larry Jackel, chefe do departamento de Vapnik na Bell Labs, apostou um jantar com ele que por volta do ano 2000 as redes neurais seriam tão reconhecidas quanto as SVMs. Ele perdeu. Em uma revanche, Vapnik apostou que em 2005 ninguém usaria mais as redes neurais e também perdeu. (A única pessoa a jantar de graça foi Yann LeCun, a testemunha.) Além disso, com o advento do aprendizado profundo, os conexionistas voltaram a se destacar.

Contanto que possamos aprendê-las, redes multicamadas podem expressar várias funções de forma mais compacta que as SVMs, que sempre têm apenas uma camada, e isso pode fazer toda a diferença.

Outro sucesso inicial notável das SVMs foi na classificação de texto, que provou ser um grande benefício porque naquele momento a web estava começando a se sobressair. Na época, o Naïve Bayes era o classificador de texto de ponta, mas quando cada palavra da linguagem é uma dimensão, até ele pode cometer sobreajuste. É preciso apenas uma palavra que ocorra em, digamos, todas as páginas esportivas nos dados de treinamento e em nenhuma outra, e o Naïve Bayes começa a se confundir achando que todas as páginas que contêm a palavra são esportivas. Contudo, graças à maximização da margem, as SVMs resistem ao sobreajuste mesmo em um grande número de dimensões.

Geralmente, quanto menor o número de vetores de suporte que uma SVM seleciona, melhor ela generaliza. Qualquer exemplo de treinamento que não seja um vetor de suporte será classificado corretamente se aparecer como um exemplo de teste porque a fronteira entre exemplos positivos e negativos continuará no mesmo local. Logo, a taxa de erro esperada de uma SVM é no máximo uma fração dos exemplos que são vetores de suporte. À medida que o número de dimensões aumenta, essa fração também tende a crescer, portanto as SVMs não estão imunes à maldição da dimensionalidade. Mas elas são mais resistentes que a maioria.

Além dos sucessos práticos, as SVMs também renovaram muitas das ideias convencionais que tínhamos sobre o machine learning. Por exemplo, elas desmentiram a noção, às vezes comparada erroneamente com a navalha de Occam, de que modelos mais simples são mais precisos. Pelo contrário, uma SVM pode ter um número infinito de parâmetros e mesmo assim não cometer sobreajuste, contanto que tenha uma margem suficientemente grande.

No entanto, a propriedade mais surpreendente das SVMs é que, não importa quanto as fronteiras formadas sejam curvas, elas são sempre apenas linhas retas (ou, em geral, hiperplanos). A razão para que isso não seja uma contradição é que as linhas retas estão em um espaço diferente. Suponhamos que os exemplos residissem no plano (x,y) e o limite entre as regiões positiva e negativa fosse a parábola $y = x^2$. Não há como representá-la com uma linha reta, mas se adicionarmos uma terceira coordenada z, significando que agora os dados residem no espaço (x,y,z), e a definirmos para cada exemplo com o quadrado de

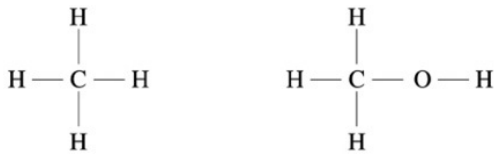
sua coordenada *x*, a fronteira será apenas o plano diagonal definido por $y = z$. Na verdade, os pontos de dados surgem na terceira dimensão, alguns mais que outros, mas na medida certa, e *presto* – nessa nova dimensão os exemplos positivos e negativos podem ser separados por um plano. Podemos então considerar o que as SVMs fazem com os kernels, vetores de suporte e pesos como o mapeamento dos dados para um espaço com mais dimensões e a busca de um hiperplano de margem máxima nesse espaço. Para alguns kernels, o espaço derivado tem dimensões infinitas, mas as SVMs não se incomodam nem um pouco. O hiperespaço pode ser a Zona Crepuscular, mas as SVMs sabem como navegar nela.

Subindo a escada

Duas coisas são semelhantes se concordam uma com a outra em alguns aspectos. Se concordarem em certos aspectos, provavelmente elas também concordarão em outros. Essa é a essência da analogia. Ela também ressalta os dois principais subproblemas do raciocínio analógico: descobrir quanto duas coisas são semelhantes e decidir o que mais pode ser inferido de suas semelhanças. Até agora, examinamos a extremidade de "baixo poder" da analogia, com algoritmos como o do vizinho mais próximo e das SVMs, em que as respostas a essas duas perguntas são muito simples. Eles são os mais usados, mas um capítulo sobre aprendizado analógico não estaria completo sem pelo menos uma rápida visita às partes mais poderosas do espectro.

A pergunta mais importante em qualquer aprendiz analógico é como medir a similaridade. Sua resposta poderia ser tão simples como medir a distância euclidiana entre os pontos de dados, ou tão complexa quanto um programa inteiro com vários níveis de sub-rotinas cuja saída final fosse um valor de similaridade. De qualquer forma, a função da similaridade controla como o aprendiz faz generalizações partindo de exemplos conhecidos e chegando a novos. É por intermédio dela que inserimos no aprendiz nosso conhecimento da área concernente ao problema, o que a torna a resposta dos analogistas para a pergunta de Hume. Podemos aplicar o aprendizado analógico a todos os tipos de objeto, não apenas a vetores de atributos, contanto que tenhamos uma maneira de medir a similaridade entre eles. Por exemplo, podemos medir a similaridade entre duas moléculas pelo número de subestruturas idênticas que elas contêm. O metano e o metanol são similares porque têm três ligações carbono-hidrogênio

em comum e só diferem na substituição de um átomo de hidrogênio por um grupo hidroxila:



Metano Metanol

No entanto, isso não significa que seu comportamento químico é similar. O metano é um gás e o metanol é um álcool. A segunda parte do raciocínio analógico é saber o que podemos inferir sobre o novo objeto com base em objetos similares que encontrarmos. Isso pode ser muito simples ou muito complexo. No vizinho mais próximo ou nas SVMs, só precisamos prever a classe do novo objeto de acordo com as classes dos vizinhos mais próximos ou dos vetores de suporte. Porém, no raciocínio baseado em casos, outro tipo de aprendizado analógico, a saída pode ser uma estrutura complexa formada por partes componentes dos objetos recuperados. Suponhamos que sua impressora HP estivesse exibindo uma linguagem incoerente e você chamasse o help desk. Há chances de que eles tenham visto o problema muitas vezes, logo uma boa estratégia é encontrar esses registros e reunir uma possível solução para o problema a partir deles. Mas não é apenas uma questão de encontrar reclamações com vários atributos similares aos seus: por exemplo, se você estiver usando sua impressora com o Windows ou o Mac OS X, isso pode originar configurações muito diferentes para o sistema e a impressora para serem relevantes. E quando você encontrar os casos mais relevantes, a sequência de etapas necessária à resolução de seu problema pode ser uma combinação de etapas de casos diferentes, com alguns ajustes adicionais específicos para o seu.

Atualmente os help desks são a aplicação mais popular do raciocínio baseado em casos. A maioria ainda emprega um intermediário humano, mas a funcionária virtual Eliza da IPsoft fala diretamente com o cliente. Eliza, que aparece inteira com uma persona em vídeo 3D interativo, resolveu mais de vinte milhões de problemas de clientes até hoje, grande parte para empresas blue-chip norte-americanas. “Saudações da Robotistan, o novo destino mais barato da terceirização”, é como um blog de terceirização se apresentou recentemente. E, assim como a terceirização continua subindo a escada das habilidades, o mesmo ocorre com o aprendizado analógico. Os primeiros advogados-robôs que

defendem uma sentença específica com base em precedentes já foram construídos. Um desses sistemas previu corretamente os resultados de mais de 90% dos casos de segredos comerciais que examinou. Talvez em uma corte cibernética do futuro, em sessão em algum local na nuvem da Amazon, um advogado-robô o livrará da multa por excesso de velocidade que o RoboCop emitiu para seu carro autônomo, tudo isso enquanto você está na praia, e o sonho de Leibniz de reduzir toda a argumentação a cálculos finalmente se tornará realidade.

Talvez em um degrau ainda mais alto da escada de habilidades esteja a composição musical. David Cope, professor emérito de música na Universidade da Califórnia, em Santa Cruz, projetou um algoritmo que cria novas músicas no estilo de compositores famosos selecionando e recombinando passagens curtas de seu trabalho. Em uma conferência a que compareci há alguns anos, ele tocou três peças "de Mozart": uma do Mozart real, outra de um compositor humano imitando Mozart e ainda outra criada por seu sistema. Em seguida, pediu à audiência para votar no Amadeus autêntico. Wolfgang venceu, mas o computador venceu o imitador humano. Já que se tratava de uma conferência de inteligência artificial, a audiência ficou encantada. O público de outros eventos ficaram menos felizes. Um ouvinte acusou Cope raivosamente de ter-lhe retirado o prazer de ouvir música. Se Cope está certo, a criatividade – a imensidão definitiva – se resume a analogia e recombinação. Julgue você mesmo procurando no Google "david cope mp3".

Entretanto, o truque mais hábil dos analogistas é o aprendizado que se estende por várias áreas. Os humanos fazem isso o tempo todo: um executivo pode passar de, digamos, uma empresa de mídia para uma de produtos para consumidores sem começar do zero porque habilidades gerenciais semelhantes se aplicam às duas atividades. Wall Street contrata um grande número de físicos porque problemas da física e de finanças, embora superficialmente muito diferentes, com frequência têm uma estrutura matemática similar. Mesmo assim, todos os aprendizes que vimos até agora falhariam se, por exemplo, os treinássemos para prever o movimento browniano e então solicitássemos que previssem o mercado de ações. O preço das ações e a velocidade de partículas suspensas em um fluido são variáveis diferentes, logo o aprendiz não saberia por onde começar. Porém, os analogistas podem fazê-lo usando o mapeamento de estrutura, um algoritmo inventado por Dedre Gentner, um psicólogo da

Universidade Northwestern. O mapeamento de estrutura pega duas descrições, encontra uma correspondência coerente entre algumas de suas partes e relações e, então, com base nessa correspondência, transfere propriedades adicionais de uma estrutura para a outra. Por exemplo, se as estruturas fossem o sistema solar e o átomo, poderíamos mapear planetas para elétrons e o Sol para o núcleo e concluir, como fez Bohr, que os elétrons giram ao redor do núcleo. É claro que a verdade é mais sutil e, em geral, precisamos refinar as analogias após criá-las. Todavia, conseguir aprender a partir de um único exemplo como esse é com certeza um atributo-chave de um aprendiz universal. Quando somos confrontados com um novo tipo de câncer – e isso ocorre o tempo todo porque os cânceres estão sempre em mutação –, os modelos que aprendemos com base em cânceres anteriores não são aplicáveis. Também não temos tempo para coletar dados sobre o novo câncer a partir de vários pacientes; pode existir apenas um, e ele precisará de uma cura urgente. A melhor aposta então é comparar o novo câncer com os já conhecidos e tentar encontrar um cujo comportamento tenha similaridade suficiente para que linhas de ataque semelhantes funcionem.

Há algo que a analogia não possa fazer? Não de acordo com Douglas Hofstadter, cientista cognitivo e autor de *Gödel*, *Escher*, *Bach*: *Laços eternos*. Hofstadter, que se parece um pouco com um gêmeo bom do Grinch, talvez seja o analogista mais conhecido mundialmente. Em seu livro *Surfaces and Essences: Analogy as the Fuel and Fire of Thinking*, ele e seu colaborador Emmanuel Sander argumentam de maneira apaixonada que o comportamento inteligente como um todo se resume à analogia. Tudo que aprendemos ou descobrimos, desde o significado de palavras cotidianas como *mãe* e *jogar* até os insights brilhantes de gênios como Albert Einstein e Évariste Galois, é resultado da analogia em ação. Quando o pequeno Tim vê mulheres cuidando de outras crianças como sua mãe cuida dele, generaliza o conceito “mamãe” como significando a mãe de qualquer pessoa e não só a sua. Isso, por sua vez, é um trampolim para o entendimento de coisas como “nave mãe” e “Mãe Natureza”. O “pensamento mais feliz” de Einstein, a partir do qual evoluiu a teoria geral da relatividade, era uma analogia entre gravidade e aceleração: se você está em um elevador, não tem como saber se seu peso é consequência de uma ou de outra porque seus efeitos são os mesmos. Nadamos em um vasto oceano de analogias, e ao mesmo tempo em que o manipulamos para nossos fins também somos

involuntariamente manipulados por ele. Livros têm analogias em cada página (como o título desta seção ou o da anterior). *Gödel, Escher, Bach* é uma analogia estendida entre o teorema de Gödel, a arte de Escher e a música de Bach. Se o Algoritmo Mestre não é analogia, com certeza deve ser parecido.

Suba e brilhe[1]

A ciência cognitiva tem testemunhado um duradouro debate entre simbolistas e analogistas. Os simbolistas mencionam algo que eles podem modelar, mas os analogistas não; em seguida, os analogistas descobrem como fazê-lo, inventam algo que podem modelar que os simbolistas não podem, e o ciclo se repete. O aprendizado baseado em instâncias, como às vezes é chamado, é supostamente melhor para modelar a maneira como nos lembramos de episódios específicos de nossas vidas; as regras são a opção presumível para o raciocínio com conceitos abstratos como "trabalho" e "amor". Porém, quando aluno de pós-graduação, ocorreu-me que as duas abordagens são apenas pontos em um continuum, e deveríamos poder aprender ao longo de toda a sua extensão. As regras são na verdade instâncias generalizadas para casos em que nos "esquecemos" de alguns atributos porque eles não importavam. Por outro lado, as instâncias são regras muito específicas, com uma condição para cada atributo. À medida que nossa vida avança, episódios semelhantes vão se transformando aos poucos em estruturas baseadas em regras, como "almoçar em um restaurante". Você sabe que ir a um restaurante envolve fazer um pedido a partir de um menu e deixar gorjeta e segue essas "regras de conduta" sempre que almoça fora, mas talvez não se lembre dos restaurantes específicos em que tomou conhecimento delas pela primeira vez.

Em minha tese de PhD, projetei um algoritmo que unifica o aprendizado baseado em instâncias com o aprendizado baseado em regras. Uma regra não encontra correspondência apenas com entidades que satisfaçam a todas as suas pré-condições; encontra qualquer entidade que tenha uma correspondência maior com ela do que com alguma outra regra, no sentido de chegar mais perto de atender suas condições. Por exemplo, alguém com nível de colesterol de 220 mg/dL chega mais perto de satisfazer a regra *Se seu colesterol está acima de 240 mg/dL, você pode ter um ataque cardíaco* do que alguém com 200 mg/dL. RISE, como chamei o algoritmo, começa aprendendo cada exemplo de treinamento como uma regra e generaliza gradualmente cada regra para absorver os

exemplos mais próximos. O resultado costuma ser uma combinação de muitas regras gerais, que entre elas apresentam correspondência com a maioria dos exemplos, com regras mais específicas que se aplicam a exceções às regras anteriores, e assim por diante, se estendendo em uma cauda longa de memórias específicas. RISE fez previsões mais acertadas do que os melhores aprendizes baseados em regras e baseados em instâncias da época, e meus experimentos mostraram que isso ocorre porque ele combinava as melhores características de ambos. Regras podem ser atendidas analogicamente e dessa forma deixam de ser frágeis. Instâncias podem selecionar características distintas em diferentes regiões do espaço e assim combatem a maldição da dimensionalidade muito melhor do que o vizinho mais próximo, que só pode selecionar as mesmas características em todos os locais.

RISE foi um passo em direção ao Algoritmo Mestre, já que combinava o aprendizado simbólico e o analógico. No entanto, foi um pequeno passo, porque não tem o poder completo desses dois paradigmas e ainda faltam os outros três. As regras do RISE não podem ser encadeadas de maneiras diferentes; cada regra prevê apenas a classe de um exemplo diretamente a partir de seus atributos. Além disso, elas não podem se referir a mais de uma entidade ao mesmo tempo; por exemplo, o RISE não pode expressar uma regra como *Se A tem gripe e B teve contato com A, B também pode ter gripe*. Pelo lado analógico, ele somente generaliza o algoritmo simples do vizinho mais próximo; não pode aprender vários assuntos usando o mapeamento de estrutura ou alguma estratégia semelhante. Na época em que terminei meu PhD, não tinha descoberto uma maneira de reunir em um único algoritmo o poder completo de todos os cinco paradigmas e deixei o problema de lado por algum tempo. Porém, ao aplicar o machine learning a problemas como o marketing de boca a boca, a integração de dados, a programação por exemplos e a personalização de sites, via constantemente como cada um dos paradigmas fornecia apenas parte da solução. Tinha de haver uma maneira melhor.

Bem, já exploramos o território das cinco tribos, coletando seus insights, negociando cruzamentos de fronteiras, nos perguntando como as peças podem ser unidas. Sabemos muito mais agora do que quando começamos. Mas ainda falta algo. Há um espaço vazio no centro do quebra-cabeça, dificultando enxergar o padrão. O problema é que todos os aprendizes que vimos precisam de um professor que lhes diga a resposta correta. Eles não podem aprender a

distinguir as células cancerosas das saudáveis a menos que alguém as rotule como “cancerosas” ou “saudáveis”. Contudo, os humanos podem aprender sem um professor; eles o fazem desde que nascem. Como Frodo nos portões de Mordor, nossa longa jornada terá sido em vão se não encontrarmos uma maneira de vencer essa barreira. Mas há um caminho que passa pelas muralhas e pelos guardas, e o prêmio está próximo. Siga-me...

---

1 N.T.: Subir em inglês é “rise”, sendo uma referência ao nome do algoritmo criado pelo autor.

capítulo 8

Aprendendo sem professor

Se você é pai, o mistério do aprendizado se revelou em sua plenitude diante de seus olhos nos três primeiros anos de vida de seu filho. Um recém-nascido não consegue falar, caminhar, reconhecer objetos, ou até mesmo entender que um objeto continua existindo quando o bebê não está olhando para ele. Porém, mês após mês, em passos menores e maiores, por tentativa e erro e grandes saltos conceituais, a criança descobre como o mundo funciona, como as pessoas se comportam e como se comunicar. Em seu terceiro aniversário, todo esse aprendizado foi reunido em uma personalidade estável, um fluxo de consciência que continuará por toda a vida. As crianças mais velhas e os adultos podem viajar no tempo, ou seja, lembrar de coisas passadas, mas apenas até certo ponto. Se pudéssemos revisitar a nós mesmos como bebês e crianças pequenas e ver o mundo mais uma vez por esses olhos recém-abertos, grande parte do que nos intriga com relação ao aprendizado – e até mesmo sobre a própria existência – subitamente pareceria óbvia. Contudo, do modo como ocorre, o maior mistério do universo não é como ele começa ou termina, ou a partir de quais fios infinitesimais foi tecido, mas sim o que acontece na mente de uma criança pequena: como 450 gramas de geleia cinza podem evoluir para formar a base da consciência.

O estudo científico do aprendizado das crianças ainda é jovem, tendo começado de fato há apenas algumas décadas, mas avançou de maneira notável. As crianças não conseguem responder questionários ou seguir protocolos experimentais, porém podemos inferir grande parte do que ocorre em suas mentes gravando e estudando suas reações durante experimentos. Um quadro coerente emerge: a mente de uma criança não é apenas o desabrochar de um programa genético predefinido ou um dispositivo biológico para o registro de

correlações em dados sensíveis; em vez disso, ela sintetiza ativamente sua realidade, e esta muda radicalmente com o tempo.

Cada vez mais, e de maneira muito relevante para nós, os cientistas cognitivos expressam suas teorias sobre o aprendizado das crianças na forma de algoritmos. Um grande número de pesquisadores do machine learning se inspira nisso. Tudo que precisamos está logo ali na mente de uma criança, se pudéssemos de alguma forma capturar sua essência em código de computador. Alguns pesquisadores chegam a argumentar que uma maneira de criar máquinas inteligentes seria construindo um bebê-robô e deixando-o vivenciar o mundo como um bebê humano faria. Nós, pesquisadores, seríamos seus pais (talvez até com a ajuda de crowdsourcing[1], dando um significado totalmente novo para o termo *vila global*). O pequeno Robby – iremos chamá-lo assim, em homenagem ao robô gordo, mas muito maior, de *Planeta proibido* – será o único bebê-robô que teremos de construir. Quando ele tiver aprendido tudo que uma criança de três anos sabe, o problema da inteligência artificial estará resolvido. Poderemos copiar o conteúdo de sua mente em quantos robôs quisermos, e eles assumirão a partir daí, com a parte mais difícil já concluída.

A pergunta, claro, é qual algoritmo deve ser acionado no cérebro de Robby no momento do nascimento. Pesquisadores adeptos da psicologia infantil veem as redes neurais com desconfiança porque o funcionamento microscópico de um neurônio parece estar muito distante da sofisticação até mesmo dos comportamentos mais básicos de uma criança, como estender o braço para alcançar um objeto, pegá-lo e examiná-lo com olhos curiosos e bem abertos. Precisamos modelar o aprendizado da criança em um nível mais alto de abstração, ou podemos nos perder em detalhes deixando de lado a essência. Acima de tudo, ainda que as crianças certamente obtenham muita ajuda de seus pais, elas aprendem em grande parte por conta própria, sem supervisão, e é isso que parece mais surpreendente. Nenhum dos algoritmos que vimos até agora consegue fazer isso, mas estamos prestes a ver vários que conseguem – levando-nos mais um passo em direção ao Algoritmo Mestre.

Agrupando coisas semelhantes

Ao pressionarmos o botão “ativar”, os olhos em vídeo de Robby se abrirão pela primeira vez. Nesse exato momento ele se verá diante do que William James memoravelmente chamou de “exuberante e barulhenta confusão” do mundo.

Com novas imagens fluindo em sua direção a uma velocidade de dezenas por segundo, uma das primeiras coisas que deve fazer é aprender a organizá-las em grupos maiores. O mundo real é composto por objetos que persistem com o tempo e não por pixels aleatórios mudando de maneira arbitrária de um momento para o outro. Mamãe não é substituída por uma mamãe menor quando vai embora. Colocar um prato na mesa não produz um buraco branco nela. Um bebê recém-nascido não ficaria surpreso se um ursinho de pelúcia passasse por trás de um biombo e ressurgisse como um avião, mas se tivesse um ano ficaria. De algum modo, ele aprendeu que ursinhos de pelúcia são diferentes de aviões e não se transmutam espontaneamente. Um pouco depois, ele descobrirá que alguns objetos são mais parecidos que outros e começará a formar categorias. Ao receber uma pilha de cavalinhos de brinquedo e lápis para brincar, um bebê de nove meses não os classificaria em grupos separados de cavalos e lápis, mas um de dezoito meses o faria.

Organizar o mundo em objetos e categorias é algo secundário para um adulto, mas não para uma criança, e menos ainda para Robby, o robô. Poderíamos dotá-lo de um córtex visual na forma de um perceptron multicamadas e lhe mostrar exemplos rotulados de todos os objetos e categorias existentes no mundo – esta é a mamãe de perto e esta é ela de longe –, mas seria uma tarefa interminável. O que precisamos é de um algoritmo que agrupe de maneia espontânea objetos semelhantes, ou diferentes imagens do mesmo objeto. Esse problema se chama clustering (agrupamento) e é um dos mais estudados no machine learning.

Um cluster é um conjunto de entidades semelhantes ou, no mínimo, de entidades mais semelhantes umas às outras que os membros de outros clusters. Agrupar faz parte da natureza humana, e com frequência esse é o primeiro passo no caminho para o conhecimento. Quando olhamos para o céu noturno, não podemos evitar ver grupos de estrelas, e então os nomeamos de maneira imaginária de acordo com as formas que nos lembram. Notar que certos conjuntos de elementos tinham propriedades químicas muito semelhantes foi o primeiro passo para a descoberta da tabela periódica. Agora cada um desses conjuntos é uma coluna da tabela. Tudo que observamos é um cluster, dos rostos dos amigos aos sons da fala. Sem eles, ficaríamos perdidos: as crianças não podem aprender um idioma antes de aprender a identificar os sons característicos dos quais ele é composto, o que fazem no primeiro ano de vida, e todas as palavras que aprendem não significam nada sem os clusters dos objetos reais a

que se referem. Quando confrontados com big data – um número muito grande de objetos –, a primeira coisa em que pensamos é agrupá-los em um número de clusters mais gerenciável. Um mercado completo tem magnitude muito alta, e os clientes são unidades muito pequenas, logo os negociantes dividem o mercado em segmentos, que é a palavra que usam para clusters. Até os próprios objetos são na verdade clusters das observações dos bebês, dos diferentes ângulos projetados pela luz sobre a face de sua mãe às diferentes ondas sonoras que eles ouvem como sendo da palavra *mamãe*. Também não conseguimos pensar sem os objetos, e talvez seja por isso que a mecânica quântica é tão intuitiva: queremos visualizar o mundo subatômico como partículas colidindo ou ondas interagindo, mas ele não é nem um nem outro.

Podemos representar um cluster por seu elemento prototípico: a imagem que formamos mentalmente de nossa mãe ou do típico gato, carro esportivo, casa de campo ou praia tropical. Peoria, Illinois, é a cidade americana média, de acordo com uma visão de marketing. Bob Burns, de 53 anos, supervisor de manutenção predial em Windham, Connecticut, é o cidadão mais comum da América – pelo menos segundo o livro *The Average American* de Kevin O'Keefe. Qualquer coisa que possa ser descrita por atributos numéricos – por exemplo, a altura das pessoas, seu peso, sua cintura, quanto elas calçam, o comprimento do cabelo, e assim por diante – facilita calcular o membro médio: sua altura é a altura média de todos os membros do cluster, seu peso é a média de todos os pesos *etc.* Para atributos categóricos, como gênero, cor do cabelo, código postal ou esporte favorito, a "média" é simplesmente o valor mais frequente. O membro médio descrito por esse conjunto de atributos pode ou não ser uma pessoa real, mas de qualquer forma é uma referência útil: se você estiver tentando solucionar como vender um novo produto, imaginar Peoria como a cidade em que ele está sendo lançado ou Bob Burns como seu cliente-alvo funcionaria melhor que pensar em entidades abstratas como "o mercado" ou "o cliente".

Mesmo com as médias sendo tão úteis, podemos fazer melhor; na verdade, a finalidade do big data e do machine learning é não precisarmos pensar em um nível tão genérico. Nossos clusters podem ser conjuntos muito especializados de pessoas ou até de diferentes aspectos da mesma pessoa: Alice comprando livros para o trabalho, para o lazer ou como presentes de Natal; Alice de bom humor *versus* triste. A Amazon gostaria de diferenciar os livros que Alice compra para ela mesma dos que compra para seu namorado, já que isso lhe permitiria fazer

recomendações apropriadas nos momentos certos. Infelizmente, as compras não vêm rotuladas como “presente para mim” ou “para Bob”, e o serviço precisa descobrir como agrupá-las.

Suponhamos que as entidades do mundo de Robby coubessem em cinco clusters (pessoas, mobília, brinquedos, alimentos e animais), mas não soubéssemos quais elementos pertencem a quais clusters. Esse é o tipo de problema que Robby encontrará quando o ligarmos. Uma opção simples para a classificação de entidades é selecionar cinco objetos aleatórios como os protótipos do cluster e então comparar cada entidade com cada protótipo e atribuí-la ao cluster de protótipo mais semelhante. (Como no aprendizado analógico, a escolha da medida de similaridade é importante. Se os atributos forem numéricos, ela pode ser tão simples como a distância euclidiana, mas há muitas outras opções.) Agora, é preciso atualizar os protótipos. Afinal, o protótipo de um cluster deve ser a média de todos os seus membros, e embora fosse esse o caso quando cada cluster tinha apenas um membro, não será assim após adicionarmos vários membros novos aos clusters. Logo, para cada cluster devemos calcular as propriedades médias de seus membros e torná-las o novo protótipo. Nesse momento, é preciso atualizar novamente os associados do cluster: já que os protótipos mudaram, o protótipo mais aproximado de uma entidade específica também pode ter mudado. Imaginemos que o protótipo de uma categoria fosse um ursinho de pelúcia e o de outra fosse uma banana. Se em nossa primeira classificação agrupamos um biscoito em forma de animal junto com o urso, talvez na segunda o agrupemos com a banana. Inicialmente, o biscoito com forma de animal parecia um brinquedo, mas agora se parece mais com comida. Depois que reclassificarmos os biscoitos com forma de animal no grupo da banana, pode ser que o item prototípico desse grupo também mude de uma banana para um biscoito. Esse ciclo virtuoso, com entidades atribuídas a clusters cada vez melhores, continuaria até a atribuição das entidades a clusters não mudar (e, portanto, também não mudarem os protótipos dos clusters).

Esse algoritmo se chama *k*-means e tem origem ainda nos anos 1950. Ele é preciso, simples e muito popular, mas apresenta várias deficiências, algumas mais fáceis de resolver que outras. Em primeiro lugar, precisamos corrigir o número de clusters antecipadamente, mas no mundo real Robby estará sempre se deparando com novos tipos de objetos. Uma opção é permitir que um objeto inicie um novo cluster se ele for muito diferente dos existentes. Outra é permitir

que os clusters se dividam e se mesclem ao longo do processo. De qualquer forma, é recomendável que o algoritmo inclua a preferência por menos clusters, ou cada objeto pode acabar sendo seu próprio cluster (solução imbatível para clusters compostos por objetos semelhantes, mas obviamente não é o que queremos).

Um grande problema é que o *k*-means só funciona quando os clusters são fáceis de separar: cada cluster tem de ser praticamente uma bolha esférica no hiperespaço, as bolhas devem ficar distantes umas das outras e todas devem ter volumes semelhantes e incluir um número similar de objetos. Se algum desses requisitos não for atendido, podem acontecer coisas indesejadas: um cluster alongado será dividido em dois diferentes, um cluster menor será absorvido por um maior que esteja próximo, e assim por diante. Felizmente, há uma opção melhor.

Suponhamos que achássemos deixar Robby perambular pelo mundo real uma maneira muito lenta e desajeitada de aprender. Em vez disso, como um aspirante a piloto aprendendo em um simulador de voo, o faremos olhar para imagens geradas por computador. Sabemos de quais clusters as imagens estão vindo, mas não dissemos a Robby. Criamos cada imagem primeiro selecionando um cluster aleatório (por exemplo, brinquedos) e então sintetizando um exemplo desse cluster (um pequeno e fofo urso de pelúcia marrom com grandes olhos pretos, orelhas redondas e gravata borboleta). Também selecionamos as propriedades do exemplo de forma aleatória: o tamanho vem de uma distribuição normal com média de 25,40 centímetros, o pelo tem 80% de chance de ser marrom, caso contrário é branco, e assim por diante. Após Robby ter visto várias imagens geradas dessa forma, supõe-se que ele aprenda a agrupá-las em pessoas, mobília, brinquedos etc., porque as pessoas são mais parecidas com pessoas que com mobília, e por aí vai. Porém, a pergunta interessante é: se olharmos para isso pelo ponto de vista de Robby, qual é o melhor algoritmo para a descoberta dos clusters? A resposta é surpreendente: o Naïve Bayes, que vimos pela primeira vez como um algoritmo para o aprendizado supervisionado. A diferença é que agora Robby não conhece as classes, logo temos de adivinhá-las!

É claro que, se Robby as conhecesse, seria fácil: como no Naïve Bayes, cada cluster seria definido por sua probabilidade (17% dos objetos gerados seriam brinquedos) e pela distribuição de probabilidades de cada atributo entre os membros do cluster (por exemplo, 80% dos brinquedos seriam marrons). Robby

poderia estimar essas probabilidades simplesmente contando o número de brinquedos existentes nos dados, o número de brinquedos marrons *etc.* Contudo, para fazê-lo, precisaríamos saber quais objetos são brinquedos. Esse parece um problema difícil de resolver, mas por acaso também já sabemos como. Se Robby tiver um classificador Naïve Bayes e precisar descobrir a classe de um novo objeto, tudo que ele precisa fazer é aplicar o classificador e calcular a probabilidade de cada classe dados os atributos do objeto. (Pequeno, fofo, marrom, com forma de urso, grandes olhos e uma gravata borboleta? Provavelmente um brinquedo, mas também pode ser um animal.)

Robby está diante de um problema do tipo "o ovo ou a galinha": se ele conhecesse as classes dos objetos, poderia aprender os modelos das classes por contagem, e se conhecesse os modelos, poderia inferir as classes. Parece que estamos presos de novo, mas não é bem assim: selecione uma classe para cada objeto da maneira que quiser – até mesmo aleatória – e estará pronto para prosseguir. A partir dessas classes e dos dados, você pode aprender os modelos das classes; com base nesses modelos pode inferir novamente as classes, e assim por diante. À primeira vista parece um esquema maluco: pode nunca terminar, circulando entre inferir as classes a partir dos modelos e os modelos a partir das classes, e mesmo se terminar, não há razão para crermos que se estabelecerá em clusters relevantes. Porém, em 1977, três estatísticos de Harvard (Arthur Dempster, Nan Laird e Donald Rubin) mostraram que o esquema maluco funciona: sempre que percorremos o loop, o modelo do cluster melhora, e o loop termina quando o modelo é um máximo local da probabilidade. Eles chamaram esse esquema de algoritmo EM, em que o E vem de expectativa (inferir as probabilidades esperadas) e o M de maximização (estimar os parâmetros da probabilidade máxima). Também mostraram que muitos algoritmos anteriores eram casos especiais do algoritmo EM. Por exemplo, para aprender modelos ocultos de Markov, nos alternamos entre inferir os estados ocultos e estimar as probabilidades de transição e observação com base neles. Sempre que quisermos aprender um modelo estatístico, mas não tivermos certas informações cruciais (como as classes dos exemplos), basta usar o algoritmo EM. Isso o torna um dos algoritmos mais populares do machine learning.

Você deve ter notado certa semelhança entre o *k*-means e o EM, já que ambos se alternam entre atribuir entidades a clusters e atualizar as descrições desses últimos. Isso não é por acaso: o próprio *k*-means é um tipo especial de EM, que

ocorre quando todos os atributos têm distribuições normais “estreitas”, isto é, com bem pouca variância. Quando os clusters apresentam muita sobreposição, uma entidade pode pertencer, digamos, ao cluster A com probabilidade de 0,7 e ao cluster B com probabilidade de 0,3, e desse modo não temos como decidir que ela pertence ao cluster A sem perder informações. O algoritmo EM leva esse problema em consideração atribuindo a entidade de modo fracionário aos dois clusters e atualizando suas descrições de acordo. No entanto, se as distribuições estiverem muito concentradas, a probabilidade de que uma entidade pertença ao cluster mais próximo será sempre aproximadamente igual a 1, e só teremos de atribuir entidades aos clusters e calcular sua média em cada um deles para obter a média do cluster, o que na verdade é o algoritmo *k*-means.

Até agora vimos como aprender um único nível de clusters, mas é claro que o mundo é muito mais rico, com clusters dentro de clusters até o nível de objetos individuais: cluster de seres vivos se estendendo para plantas e animais, cluster de animais se estendendo para mamíferos, pássaros, peixes etc., descendo de nível até chegar a Fido, o cão da família. Sem problema: depois que tivermos aprendido um conjunto de clusters, poderemos tratá-los como objetos e agrupá-los, subindo até o nível do cluster de todas as coisas. Alternativamente, podemos começar com um clustering genérico e então dividir cada cluster em subclusters: os brinquedos de Robby seriam divididos em animais com enchimento, brinquedos de montar etc.; os animais com enchimento se dividiriam em ursinhos, gatinhos de pelúcia, e outros. As crianças parecem começar pelo meio para então seguir um caminho para cima ou para baixo. Por exemplo, elas aprendem *cão* antes de aprender *animal* ou *beagle*. Essa também pode ser uma boa estratégia para Robby.

Descobrindo a forma dos dados

Sejam dados entrando no cérebro de Robby por intermédio de seus sentidos ou as sequências de cliques de milhões de clientes da Amazon, agrupar um grande número de entidades em um número menor de clusters é só metade do esforço. A outra metade é encurtar a descrição de cada entidade. A primeira foto que Robby viu de uma mãe talvez fosse composta por um milhão de pixels, cada um com sua própria cor, mas é raro precisarmos de um milhão de variáveis para descrever uma face. Da mesma forma, cada item em que clicamos na Amazon fornece uma pequena informação sobre nós, mas o que o serviço gostaria

realmente de conhecer são nossas preferências e aversões, não nossos cliques. As primeiras, que costumam ser bem estáveis, são de alguma forma indicadas pelos últimos, que crescem sem limite à medida que usamos o site. Pouco a pouco, todos esses cliques somados devem levar a um retrato do que gostamos, assim como todos os pixels somados levariam a uma imagem de nosso rosto. O que precisamos é saber como fazer a soma.

Uma face só tem cerca de cinquenta músculos, logo cinquenta números devem ser suficientes para descrever todas as expressões possíveis com folga. O formato dos olhos, nariz, boca, e assim por diante – as características que nos permitem diferenciar duas pessoas – também não devem demandar mais que algumas dezenas de números. Afinal, com apenas dez opções para cada característica facial, um desenhista de retratos falados consegue compor o esboço de um suspeito suficientemente bom para torná-lo reconhecível. Podemos adicionar mais alguns números para especificar a iluminação e a pose, mas isso é tudo. Portanto, se você me der cerca de cem números, isso deve ser o suficiente para que eu recrie a imagem de uma face. Por outro lado, o cérebro de Robby tem de ser capaz de receber a imagem de uma face e reduzi-la rapidamente aos cem números que importam.

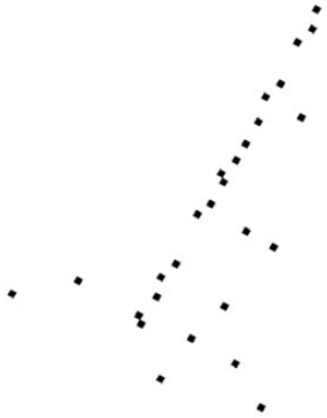
Os machine learners chamam esse processo de redução de dimensionalidade porque ele reduz um grande número de dimensões visíveis (os pixels) a poucas dimensões implícitas (expressão, características faciais). A redução de dimensionalidade é essencial para a manipulação de big data – como os dados que chegam por nossos sentidos a cada segundo. Uma imagem pode substituir mil palavras, mas também é um milhão de vezes mais difícil de processar e lembrar. Mesmo assim, nosso córtex visual faz um bom trabalho quando reduz as informações a uma quantidade gerenciável, suficiente para navegarmos pelo mundo, reconhecermos pessoas e coisas e nos lembrarmos do que vimos. É um dos grandes milagres da cognição e ocorre tão naturalmente que não temos consciência de que está acontecendo.

Quando organizamos uma estante para que livros de tópicos semelhantes fiquem próximos, estamos fazendo um tipo de redução de dimensionalidade, do vasto espaço dos tópicos para a estante unidimensional. Inevitavelmente, alguns livros intimamente relacionados acabarão em locais distantes na estante, mas ainda poderemos organizá-los de maneira a minimizar essas ocorrências. É isso que os algoritmos de redução de dimensionalidade fazem.

Suponhamos que eu lhe desse coordenadas de GPS de todas as lojas de Palo Alto, Califórnia, e você representasse algumas delas em um pedaço de papel:

É possível dizer examinando esse esboço que a rua principal de Palo Alto se estende no sentido sudoeste-nordeste. Você não desenhou uma rua, mas pode intuir que ela está lá pelo fato de que todos os pontos seguem uma linha reta (ou quase – eles podem estar em diferentes lados da rua). Na verdade, a rua se chama University Avenue, e se você quiser fazer compras ou comer em Palo Alto, é para onde deve ir. Como bônus, já que sabe que as lojas ficam na University Avenue, você não precisa de dois números para localizá-las, mas de apenas um: o número do endereço (ou, se quiser ser realmente preciso, a distância da loja à estação do Caltrain, no canto sudoeste, que é onde a University Avenue começa).

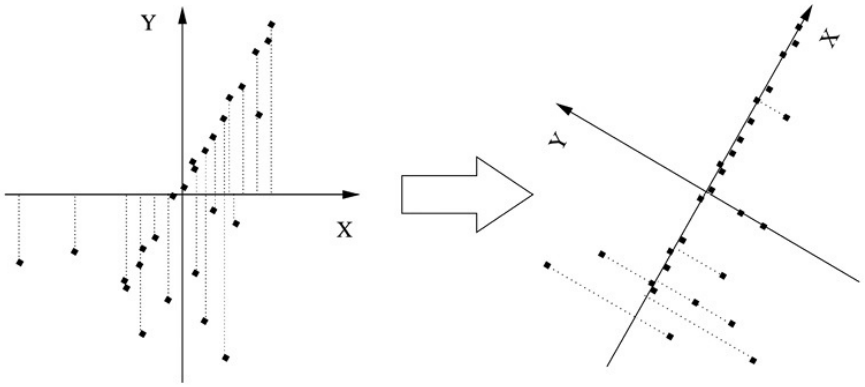
Se você desenhar mais lojas, notará que algumas estão em ruas transversais, um pouco para fora da University Avenue, e outras se encontram em um local totalmente diferente:



No entanto, a maioria das lojas continua bem próxima da University Avenue, e se você só pudesse usar um número como localização de uma loja, sua distância da estação do Caltrain ao longo da avenida seria uma ótima escolha: após

percorrer essa distância, olhar em volta deve ser o suficiente para encontrar a loja. Logo, você acabou de reduzir a dimensionalidade da "localização de lojas em Palo Alto" de duas para uma.

Robby não pode se beneficiar de nosso sistema visual altamente evoluído, de modo que se você quiser que ele pegue suas roupas limpas na Elite Cleaners e só der para seu mapa de Palo Alto uma única coordenada, será preciso um algoritmo que "descubra" a University Avenue a partir das coordenadas de GPS das lojas. A chave para isso é perceber que, se você colocar a origem do plano *x*,*y* na média dos locais onde estão as lojas e girar lentamente os eixos, elas ficarão próximas ao eixo *x* quando ele for girado cerca de 60 graus, isto é, quando ficar alinhado à University Avenue:



Essa direção – conhecida como primeiro componente principal dos dados – também é aquela ao longo da qual a difusão dos dados é maior. (Se você projetar as lojas no eixo *x* verá que na figura da direita elas estão mais distantes que na da esquerda.) Após encontrar o primeiro componente principal, você pode procurar o segundo, que nesse caso é a direção de maior variação em ângulos retos à University Avenue. Em um mapa, há apenas uma direção possível à esquerda (a direção das ruas transversais). Porém, se Palo Alto estivesse em uma ladeira, um ou os dois primeiros componentes principais estariam parcialmente na subida da ladeira, e o terceiro e último se ergueria do chão. Podemos aplicar a mesma ideia aos dados de milhares ou milhões de dimensões, como nas imagens de faces, procurando sucessivamente as direções de maior variação até a variabilidade remanescente ser pequena, ponto em que poderíamos parar. Por exemplo, após girarmos os eixos da figura anterior, a maioria das lojas tem $y = 0$, logo o y médio é muito pequeno e não perderemos muitas informações se ignorarmos a coordenada *y* como um todo. Se decidirmos manter *y*, certamente *z* (que se ergue da superfície) será insignificante. Como podemos ver, o processo inteiro de

encontrar os componentes principais pode ser realizado de uma só vez com um pouco de álgebra linear. E o melhor é que com frequência algumas dimensões assumem grande parte da variação até para dados de muitas dimensões. Mesmo se não for esse o caso, visualizar os dados nas duas ou três dimensões superiores costuma gerar muitos insights porque tira proveito dos surpreendentes poderes de percepção de nosso sistema visual.

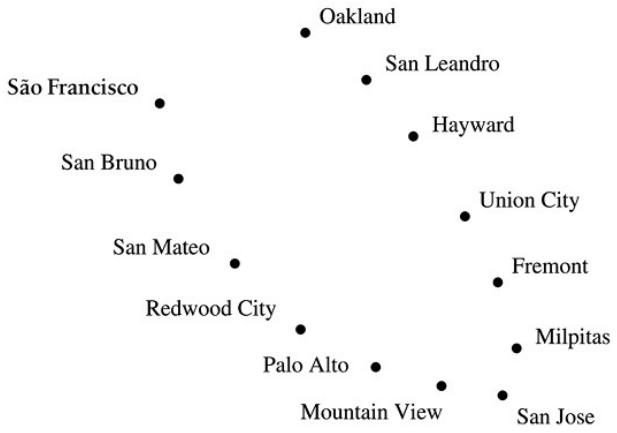
A análise de componentes principais (PCA, principal component analysis), como esse processo é chamado, é uma das ferramentas-chave do kit de ferramentas do cientista. Poderíamos dizer que a PCA é para o aprendizado não supervisionado o que a regressão linear é para a variedade supervisionada. A famosa curva taco de hóquei do aquecimento global, por exemplo, é resultado da descoberta do componente principal de várias séries de dados relacionadas ao clima (anéis de árvores, cores do gelo etc.) e da suposição de que ele é a temperatura. Os biólogos usam a PCA para reduzir os níveis de expressão de milhares de genes diferentes a alguns caminhos. Os psicólogos descobriram que a personalidade se resume a cinco dimensões – extroversão, amabilidade, escrúpulo, neuroticismo e abertura a novas experiências – que eles podem inferir a partir de nossos tuítes e postagens em blogs. (Supostamente os chimpanzés têm mais uma dimensão – reatividade –, mas não há dados disponíveis no Twitter referentes a eles.) A aplicação da PCA a votos congressionais e dados de pesquisa mostra que, ao contrário do que diz a crença popular, a política não se dá basicamente entre liberais *versus* conservadores. Em vez disso, as pessoas se dividem entre duas dimensões principais: uma de questões econômicas e outra de questões sociais. Trazê-las para um único eixo reuniria populistas e libertários, que são polos opostos, e daria a impressão de que entre os extremos há muitos moderados. Tentar atraí-los é uma estratégia de improvável vitória. Por outro lado, se os liberais e os libertários superassem sua aversão mútua, poderiam se aliar em questões sociais, em que ambos são a favor da liberdade individual.

Quando crescer, Robby poderá usar uma variante da PCA para resolver o “problema do coquetel”, que é a detecção de vozes individuais no burburinho da multidão. Um método relacionado pode ajudá-lo a aprender a ler. Se cada palavra for uma dimensão, um texto será um ponto no espaço das palavras, e as direções principais desse espaço acabarão sendo elementos de significado. Por exemplo, *Presidente Obama* e *a Casa Branca* estão distantes no espaço de

palavras, mas próximos no espaço de significado, porque tendem a aparecer em contextos semelhantes. Acredite ou não, esse tipo de análise é tudo o que precisamos para os computadores darem notas a ensaios de exames SAT tão bem quanto os humanos. O Netflix usa uma ideia parecida. Em vez de apenas recomendar filmes que usuários com preferências similares gostaram, primeiro ele projeta tanto os usuários quanto os filmes em um "espaço de preferências" de poucas dimensões e recomenda um filme se ele estiver próximo de você nesse espaço. Dessa forma, consegue encontrar filmes que não sabíamos que agradariam.

No entanto, provavelmente você ficaria desapontado se examinasse os componentes principais de um conjunto de dados de faces. Eles não são o que esperávamos, como expressões ou características faciais, e parecem mais com faces fantasmagóricas, embaçadas a ponto de impedir seu reconhecimento. Isso ocorre porque a PCA é um algoritmo linear e, portanto, os componentes principais só podem ser médias ponderadas de cada pixel das faces reais. (Também conhecidas como eigenfaces porque são autovetores da matriz de covariância centralizada dos dados –, mas voltemos ao assunto.) Para entender realmente as faces, e a maioria das formas do mundo, precisamos de algo mais: a redução de dimensionalidade não linear.

Suponhamos que diminuíssemos o zoom aplicado a Palo Alto e eu lhe desse as coordenadas de GPS das principais cidades da Bay Area:



Mais uma vez, olhando esse esboço fica fácil supor que as cidades estão em uma baía, e se você desenhar uma linha ao longo delas, poderá localizar cada cidade usando um único número: a distância que ela está de São Francisco no decorrer da linha. Todavia, a PCA não pode achar essa curva; em vez disso, ela

desenha uma linha reta percorrendo o meio da baía, onde não há nenhuma cidade. Em vez de elucidar a forma dos dados, a PCA a oculta.

Suponhamos por um momento que fôssemos construir a Bay Area a partir do zero. Decidimos onde cada cidade ficará localizada e nosso orçamento nos permite construir uma única estrada conectando-as. Naturalmente, assentamos uma estrada que vai de São Francisco a San Bruno, daí a San Mateo, e assim por diante, seguindo todo o caminho até Oakland. Essa estrada é uma representação unidimensional muito boa da Bay Area e pode ser encontrada por um algoritmo simples: construa uma estrada entre cada par de cidades próximas. É claro que em geral isso resulta em uma rede de estradas e não em uma única estrada passando por cada cidade. Porém, podemos fazer surgir esta última construindo a estrada que melhor se aproxime da rede, no sentido de que as distâncias entre as cidades ao longo da estrada fiquem o mais perto possível das fornecidas pela rede.

Um dos algoritmos mais populares para a redução de dimensionalidade não linear, chamado Isomap, faz exatamente isso. Ele conecta cada ponto de dados de um espaço com várias dimensões (por exemplo, uma face) a todos os pontos próximos (faces muito semelhantes), calcula as distâncias mais curtas entre todos os pares de pontos ao longo da rede resultante e encontra as coordenadas reduzidas que melhor se aproximem dessas distâncias. Ao contrário do que ocorre com a PCA, as coordenadas das faces nesse espaço com frequência são bem significativas: uma pode representar para qual direção a face está voltada (perfil de esquerda, de três quartos, voltado para a frente etc.); a outra qual é a aparência da face (muito triste, um pouco triste, neutra, feliz, muito feliz etc.); e assim por diante. Da compreensão de movimento em vídeo à detecção de emoção na fala, o Isomap tem uma habilidade surpreendente de concentrar-se nas dimensões mais importantes de dados complexos.

Aqui está um experimento interessante. Pegue o streaming de vídeo dos olhos de Robby, trate cada quadro como um ponto no espaço de imagens e reduza esse conjunto de imagens a uma única dimensão. O que descobriu? O tempo. Como um bibliotecário organizando livros em uma estante, o tempo insere cada imagem perto das que forem mais semelhantes a ela. Talvez a percepção que temos disso seja uma consequência natural da destreza de nossos cérebros em reduzir a dimensionalidade. Na rede de estradas da memória, o tempo é a passagem principal, e logo o encontramos. Em outras palavras, o tempo é o

componente principal da memória.

O robô hedonista

O clustering e a redução da dimensionalidade nos levam mais perto do aprendizado humano, mais ainda falta algo importante. As crianças não observam o mundo passivamente; elas fazem coisas. Pegam os objetos que veem, brincam com eles, correm pelas redondezas, comem, choram e fazem perguntas. Mesmo o sistema visual mais avançado não seria útil para Robby se não o ajudasse a interagir com o ambiente. Robby precisa saber não só o que está onde, mas o que fazer em cada momento. Em princípio, poderíamos ensiná-lo a usar instruções passo a passo, associando as leituras dos sensores às ações apropriadas a serem executadas em resposta, porém isso só é viável para tarefas limitadas. As ações que executamos dependem de nossos objetivos, não somente do que quer que estejamos percebendo atualmente, e esses objetivos podem estar em um momento mais distante no futuro. Seja como for, uma supervisão passo a passo não deve ser necessária. Os pais não ensinam seus filhos a engatinhar, caminhar ou correr; eles aprendem sozinhos. Mas nenhum dos algoritmos de aprendizado que vimos até agora consegue fazer isso.

Os humanos têm um guia constante: suas emoções. Procuramos prazer e evitamos dor. Quando você toca em um forno quente, recua instintivamente. Essa é a parte fácil. A difícil é antes de tudo não tocar no forno. Ela requer se mover para evitar uma dor aguda que você ainda não sentiu. Seu cérebro faz isso associando a dor não só ao momento em que você toca o forno, mas às ações que o levam a tocá-lo. Edward Thorndike chamou esse esquema de lei do efeito: ações que levam ao prazer têm mais probabilidade de serem repetidas no futuro; ações que levam à dor têm probabilidade menor. Poderíamos dizer que a lembrança do prazer nos volta do passado, e as ações podem acabar sendo associadas a efeitos que estão muito distantes delas. Os humanos fazem esse tipo de busca de recompensa passada melhor que qualquer outro animal, e ela é crucial para nosso sucesso. Em um experimento famoso, um marshmallow foi mostrado a algumas crianças e elas foram informadas de que, se conseguissem evitar comê-lo por alguns minutos, poderiam comer dois. As que foram bem-sucedidas se saíram melhor na escola e na vida. Talvez seja menos óbvio, mas empresas que usam machine learning para melhorar seus sites e suas práticas de negócios se deparam com um problema semelhante. Uma empresa poderia fazer

uma mudança que trouxesse mais receita no curto prazo – como vender um produto inferior cuja fabricação custasse menos pelo mesmo preço do produto superior original – sem notar que perderia clientes no longo prazo.

Todos os aprendizes que vimos nos capítulos anteriores são guiados pela gratificação instantânea: cada ato, seja a detecção de um spam ou a compra de uma ação, recebe recompensa ou punição imediata do professor. Porém, há um subcampo do machine learning totalmente dedicado a algoritmos que fazem explorações por conta própria, se fixam em algo, reagem quando há recompensa e descobrem como obtê-la de novo no futuro, de forma muito parecida a bebês engatinhando e colocando coisas na boca.

Isso se chama aprendizado por reforço, e provavelmente seu primeiro robô doméstico o usará muito. Se você pedir a Robby para preparar ovos com bacon logo após tê-lo desembalado e ligado, pode demorar um pouco. Mas depois, enquanto você estiver no trabalho, ele explorará a cozinha, observando onde várias coisas estão e que tipo de forno está disponível. Quando você voltar, o jantar estará pronto.

Um programa de jogo de damas criado nos anos 1950 por Arthur Samuel, pesquisador da IBM, foi um precursor importante do aprendizado por reforço. Jogos de tabuleiro são um ótimo exemplo de problema de aprendizado por reforço: é preciso fazer uma longa série de jogadas sem nenhum feedback, e a recompensa ou punição vem apenas no fim, na forma de vitória ou derrota. Mesmo assim, o programa de Samuel conseguiu ensinar a si próprio a jogar tão bem quanto a maioria dos humanos. Ele não aprendeu diretamente qual movimento devia fazer em cada posição do tabuleiro porque seria muito difícil. Em vez disso, aprendeu como avaliar cada posição – qual probabilidade tenho de vencer partindo daqui? – e escolher a jogada que levasse à melhor entre elas. Inicialmente, as únicas posições que ele sabia avaliar eram as finais: uma vitória, um empate ou uma derrota. Mas uma vez que soubesse que uma posição específica significava vitória, também sabia que era preciso seguir as posições que levassem a ela, e assim por diante. Thomas J. Watson Sr., presidente da IBM, previu que quando o programa fosse demonstrado as ações da empresa subiriam quinze pontos. Elas subiram. A lição foi aprendida na IBM, que construiu um campeão de xadrez e um de *Jeopardy!*.

A noção de que nem todos os estados trazem recompensas (positivas ou negativas), mas cada estado tem um valor, é essencial no aprendizado por

reforço. Em jogos de tabuleiro, só as posições finais trazem recompensa (por exemplo, 1, 0 ou -1 para vitória, empate ou derrota). Outras posições não fornecem recompensa imediata, mas têm valor porque podem levar a recompensas posteriores. Uma posição no jogo de xadrez a partir da qual você possa forçar um xeque-mate em algum número de jogadas é quase tão boa quanto uma vitória e, portanto, tem alto valor. Podemos propagar esse tipo de raciocínio até as jogadas de abertura, boas e ruins, mesmo se a essa distância a conexão não for óbvia. Em videogames, geralmente as recompensas são pontos, e o valor de um estado é o número de pontos que podemos acumular a partir dele. Na vida real, uma recompensa imediata é melhor que uma posterior, logo recompensas futuras podem ser obtidas segundo alguma taxa de retorno, como nos investimentos. É claro que as recompensas dependem das ações que executamos, e o objetivo do aprendizado por reforço é sempre selecionar a ação que leve às melhores recompensas. Você deve pegar o telefone e chamar sua amiga para sair? Poderia ser o início de um belo relacionamento ou apenas o caminho para uma dolorosa rejeição. Mesmo se sua amiga aceitar, esse encontro pode ou não acabar bem. De alguma forma, é preciso considerar todos os infinitos caminhos que o futuro pode tomar e decidir agora. O aprendizado por reforço faz isso estimando o valor de cada estado – a soma total das recompensas que você pode obter a partir dele – e selecionando as ações que o maximizem.

Suponhamos que você estivesse caminhando por um túnel, como faria Indiana Jones, e chegasse a uma bifurcação. Seu mapa diz que o túnel da esquerda leva a um tesouro e o da direita a um ninho de serpentes. O valor de onde você se encontra – diante da bifurcação – é o do tesouro porque sua decisão será a de seguir à esquerda. Se você sempre escolher a melhor ação possível, o valor de um estado diferirá do estado sucessor somente de acordo com a recompensa imediata (se houver) obtida caso essa ação seja executada. Se conhecermos a recompensa imediata de cada estado, poderemos usar essa observação para atualizar os valores de estados vizinhos, e assim por diante, até todos os estados terem valores consistentes. O valor do tesouro se propagará para trás ao longo do túnel até alcançar a bifurcação e além dela. Depois que você souber o valor de cada estado, também saberá qual ação selecionar em cada um deles (a que maximize a combinação de recompensa imediata e valor do estado resultante). Esse problema foi formulado nos anos 1950 pelo estudioso da teoria do controle Richard Bellman. Porém, o problema real do aprendizado por reforço é quando

não temos um mapa do território. Então, nossa única chance é explorar e descobrir quais recompensas existem em quais locais. Podemos descobrir um tesouro ou cair em um ninho de serpentes. Sempre que você executar uma ação, notará a recompensa imediata e o estado resultante. Esse cálculo pode ser feito pelo aprendizado supervisionado. Mas você também atualizará o valor do estado do qual saiu para deixá-lo de acordo com o valor que acabou de observar, ou seja, com a recompensa que obteve mais o valor do novo estado em que se encontra. Obviamente, esse valor pode não ser ainda o correto, porém se você continuar fazendo isso por tempo suficiente, acabará obtendo os valores certos para todos os estados e ações correspondentes. Esse é um resumo do aprendizado por reforço.

Observe que o aprendizado por reforço se depara com o mesmo dilema explorar-usufruir que vimos no Capítulo 5: para maximizar nossas recompensas, naturalmente escolhemos sempre a ação que leva ao estado de valor mais alto, mas isso nos impede de descobrir recompensas ainda melhores por outros caminhos. O aprendizado por reforço resolve isso selecionando em alguns casos a melhor ação e em outros uma ação aleatória. (O cérebro parece ter até mesmo um "gerador de ruído" para esse fim.) No início, quando há muito a aprender, faz sentido explorar bastante. Depois que conhecemos o território, é melhor usufruir dele. É isso que os humanos fazem durante toda a sua vida: as crianças exploram e os adultos usufruem (exceto os cientistas, que são eternas crianças). As brincadeiras das crianças são muito mais sérias do que parecem; se a evolução gerou uma criatura que é impotente e dá muito trabalho para os pais nos primeiros anos de vida, esse custo extravagante deve ser compensado por um benefício ainda maior. Na verdade, o aprendizado por reforço é um tipo de evolução acelerada – tentar, descartar e refinar ações dentro de uma única existência em vez de durante gerações –, e por esse padrão ele é muito eficiente.

A pesquisa na área do aprendizado por reforço começou de fato no início dos anos 1980, com o trabalho de Rich Sutton e Andy Barto na Universidade de Massachusetts. Eles perceberam que o aprendizado depende crucialmente da interação com o ambiente, mas os algoritmos supervisionados não capturavam isso, e a inspiração veio então da psicologia do aprendizado animal. Sutton acabou se tornando o principal defensor do aprendizado por reforço. Outra etapa importante ocorreu em 1989, quando Chris Watkins de Cambridge, inicialmente motivado por suas observações experimentais do aprendizado das crianças,

chegou à formulação moderna do aprendizado por reforço como controle ótimo em um ambiente desconhecido.

No entanto, os aprendizes da abordagem por reforço que vimos até agora não são muito realistas, porque não sabem o que fazer em um estado específico, a menos que já tenham passado por ele, e no mundo real duas situações nunca são idênticas. Temos de poder generalizar estados novos a partir de estados já visitados. Felizmente, sabemos como fazê-lo: só precisamos anexar o aprendizado por reforço ao redor de um dos aprendizes supervisionados já vistos, como o perceptron multicamadas. Agora a tarefa da rede neural é prever o valor de um estado, e o sinal de erro para a backpropagation é a diferença entre o valor previsto e o observado. Contudo, há um problema. No aprendizado supervisionado o valor-alvo de um estado é sempre o mesmo, mas no aprendizado por reforço ele muda devido a atualizações em estados próximos. Como resultado, com frequência o aprendizado por reforço com generalização não consegue se fixar em uma solução estável, a menos que o aprendiz interno seja algo muito simples, como uma função linear. Mesmo assim, ele tem apresentado alguns sucessos notáveis. Um dos primeiros sucessos foi um jogador de gamão de nível humano. Mais recentemente, um aprendiz da abordagem por reforço da DeepMind, uma startup com sede em Londres, venceu um jogador humano especialista em Pong e outros jogos simples de arcade. Ele usou uma rede profunda (deep network) para prever os valores das ações a partir dos pixels brutos da tela do console. Com sua visão, aprendizado e controle de ponta a ponta, o sistema exibia pelo menos uma leve semelhança com um cérebro artificial. Isso pode explicar por que o Google pagou meio bilhão de dólares pela DeepMind, uma empresa sem produtos, sem receitas e com poucos funcionários.

Jogos à parte, os pesquisadores já usaram o aprendizado por reforço para estabilizar postes, controlar imagens de ginastas, estacionar carros de ré, pilotar helicópteros de ponta-cabeça, gerenciar diálogos telefônicos automatizados, atribuir canais a redes de celulares, despachar elevadores, programar o envio de carga no ônibus espacial, e muito mais. O aprendizado por reforço também influenciou a psicologia e a neurociência. O cérebro o aplica, usando o neurotransmissor dopamina para propagar diferenças entre recompensas esperadas e reais. O aprendizado por reforço explica o condicionamento pavloviano, mas ao contrário do behaviorismo, ele permite que os animais

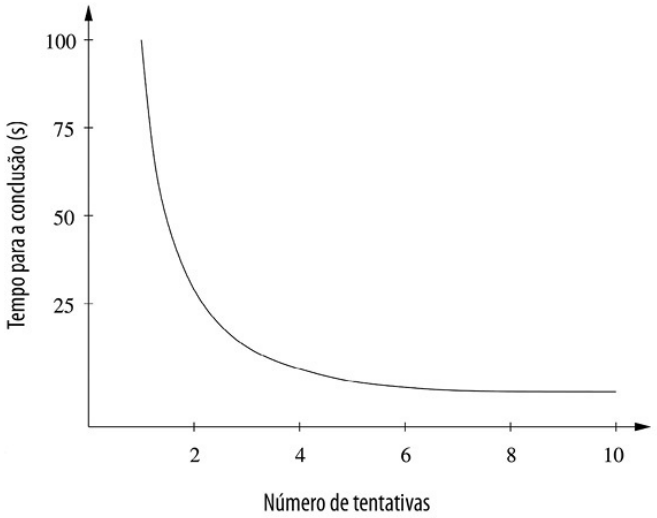
tenham estados mentais internos. Abelhas-operárias o usam, assim como ratos procurando queijo em labirintos. Nossa vida diária é uma sequência de milagres pouco notados, tornados possíveis em parte pelo aprendizado por reforço. Levantamo-nos, nos vestimos, tomamos café e dirigimos até o trabalho, e durante todo esse tempo pensamos em alguma outra coisa. Em segundo plano, o aprendizado por reforço orquestra e afina continuamente essa prodigiosa sinfonia de movimentos. Fragmentos de aprendizado por reforço, também conhecidos como hábitos, compõem grande parte do que fazemos. Você sente fome, vai até a geladeira e faz um lanche. Como Charles Duhigg mostra em *O poder do hábito*, entender e controlar esse ciclo de deixa, rotina e recompensa é a chave para o sucesso, não só para as pessoas, mas para empresas e até sociedades inteiras.

Dos fundadores do aprendizado por reforço, Rich Sutton é o mais empolgado. Para ele, o aprendizado por reforço é o Algoritmo Mestre e resolvê-lo equivale a resolver o problema da inteligência artificial. Chris Watkins, por outro lado, não está satisfeito. Ele vê muitas coisas que as crianças conseguem fazer que os aprendizes da abordagem por reforço não conseguem: resolver problemas, resolvê-los melhor após algumas tentativas, fazer planos, adquirir conhecimento cada vez mais abstrato. Felizmente, também temos algoritmos de aprendizado para essas habilidades de mais alto nível, e o mais importante de todos é o chunking.

A prática leva à perfeição

Aprender é melhorar com a prática. Você pode não se lembrar muito bem agora, mas foi difícil aprender a amarrar o cadarço do sapato. Inicialmente, você não conseguia fazê-lo de forma alguma, apesar de já ter cinco anos. Em seguida, o cadarço desamarrava mais rápido do que você conseguia amarrá-lo. Porém, pouco a pouco foi aprendendo a amarrá-lo melhor e com mais rapidez até se tornar totalmente automático. O mesmo ocorre com várias outras coisas, como engatinhar, andar, correr, andar de bicicleta e dirigir um carro; ler, escrever e fazer cálculos; tocar um instrumento e praticar um esporte; cozinhar e usar um computador. Ironicamente, aprendemos melhor quando é mais difícil: no início, quando cada etapa é difícil, erramos o tempo todo, e mesmo quando acertamos os resultados não são muito bons. Após dominar sua tacada no golfe ou seu saque no tênis, você pode passar anos aperfeiçoando-o, mas todos esses anos

farão menos diferença que as primeiras semanas. Ficamos melhores com a prática, porém não em um ritmo constante: inicialmente melhoramos com rapidez, em seguida não tão rápido, e depois muito lentamente. Seja em um esporte ou com uma guitarra, a curva da melhoria do desempenho com o passar do tempo – a habilidade para fazer algo ou o tempo gasto para fazê-lo – tem uma forma muito específica:



Esse tipo de curva se chama lei de potência, porque o desempenho varia com o tempo elevado a alguma potência negativa. Por exemplo, na figura anterior, o tempo para a conclusão é proporcional ao número de tentativas elevado a menos dois (ou, igualmente, um sobre o número de tentativas ao quadrado). Quase todas as habilidades humanas seguem uma lei de potência, com diferentes potências para habilidades distintas. (Ao contrário do Windows, que não fica mais rápido com a prática – algo que a Microsoft deveria melhorar.)

Em 1979, Allen Newell e Paul Rosenbloom começaram a se perguntar o que causaria a chamada lei de potência aplicada à prática. Newell foi um dos fundadores da inteligência artificial e um dos maiores psicólogos cognitivos, e Rosenbloom foi seu aluno de pós-graduação na Universidade Carnegie Mellon. Na época, nenhum dos modelos de prática existentes explicava a lei de potência. Newell e Rosenbloom suspeitaram que ela pudesse ter algo a ver com chunking, um conceito da psicologia da percepção e da memória. Percebemos e lembramos de coisas em chunks (pedaços), e só podemos reter um número específico de chunks na memória de curto prazo em um dado momento (sete mais ou menos dois, de acordo com o artigo clássico de George Miller). Definitivamente, agrupar coisas em chunks nos permite processar muito mais informações do que

seria possível de outra forma. É por isso que os números de telefone têm hifens: 1–723–458–3897 é mais fácil de lembrar que 17234583897. Herbert Simon, colaborador de longa data de Newell e cofundador da inteligência artificial, já tinha descoberto que a principal diferença entre jogadores de xadrez novatos e experientes é que os novatos percebem as posições do jogo uma peça de cada vez, enquanto os especialistas veem padrões maiores envolvendo várias peças. Melhorar no xadrez envolve basicamente perceber chunks maiores e em maior número. Newell e Rosenbloom deduziram que um processo semelhante está em ação na obtenção de qualquer habilidade, não apenas no xadrez.

Na percepção e na memória, um chunk é apenas um símbolo que representa um padrão de outros símbolos, como IA representa inteligência artificial. Newell e Rosenbloom adaptaram essa noção à teoria da resolução de problemas que Newell e Simon haviam desenvolvido. Newell e Simon pediram para participantes de um experimento resolverem problemas – por exemplo, derivar uma fórmula matemática de outra no quadro-negro – narrando ao mesmo tempo o que estavam fazendo. Eles descobriram que os humanos resolvem problemas decompondo-os em subproblemas, subsubproblemas, e assim por diante, e reduzindo de maneira sistemática as diferenças entre o estado inicial (nesse caso, a primeira fórmula) e o estado desejado (a segunda fórmula). No entanto, isso requer procurar uma sequência de ações que funcione, o que leva tempo. Newell e Rosenbloom formularam a hipótese de que sempre que resolvemos um problema, formamos um chunk que nos permite ir diretamente do estado anterior à solução ao estado posterior a ela. Um chunk nesse caso tem duas partes: o estímulo (padrão que reconhecemos no mundo externo ou na memória de curto prazo) e a resposta (sequência de ações que executamos como resultado). Depois que você aprender um chunk, o armazenará na memória de longo prazo. Na próxima vez que tiver de resolver o mesmo subproblema, poderá simplesmente aplicar o chunk e poupar o tempo gasto na busca. Isso ocorrerá em todos os níveis até você ter um chunk para o problema inteiro e poder resolvê-lo automaticamente. Para amarrar o cadarço, você dá o nó inicial, dá uma volta com uma extremidade, passa a outra extremidade ao redor dela e a puxa pelo buraco no meio. Cada uma dessas ações está longe de ser trivial para uma criança de cinco anos, mas a partir do momento que você detecta os chunks correspondentes, está quase lá.

Rosenbloom e Newell prepararam seu programa de chunking para operar com

uma série de problemas, mediram o tempo que ele levava em cada tentativa, e adivinhem: obtiveram uma série de curvas da lei de potência. Porém, isso foi apenas o começo. Em seguida, eles incorporaram o chunking ao Soar, uma teoria geral da cognição na qual Newell estava trabalhando com John Laird, outro de seus alunos. Em vez de funcionar apenas dentro de uma hierarquia de objetivos predefinida, o programa Soar conseguia definir e resolver um novo subproblema sempre que encontrava um obstáculo. Quando formava um novo chunk, o Soar o generalizava para aplicar a problemas similares, de maneira semelhante à dedução inversa. O chunking do Soar acabou se tornando um bom modelo de muitos fenômenos do aprendizado, não só da lei de potência aplicada à prática. Ele poderia ser aplicado até mesmo ao aprendizado de novos conhecimentos pelo chunking de dados e analogias. Isso levou Newell, Rosenbloom e Laird a deduzir que o chunking é o *único* mecanismo necessário para o aprendizado – em outras palavras, o Algoritmo Mestre.

Sendo defensores clássicos da inteligência artificial, Newell, Simon e seus alunos e seguidores acreditavam veementemente na primazia da resolução de problemas. Se o solucionador de problemas é poderoso, o aprendiz pode contar com o seu suporte e ser simples. Na verdade, o aprendizado é apenas outro tipo de resolução de problemas. Newell e companhia fizeram um esforço conjunto para reduzir todo o aprendizado ao chunking e toda a cognição ao Soar, mas acabaram falhando. Uma das complicações foi que, à medida que o solucionador de problemas aprendia mais chunks, e chunks ainda mais complexos, o custo de testá-los com frequência se tornou tão alto que o programa ficou mais lento em vez de mais veloz. De alguma forma, os humanos evitam isso, mas os pesquisadores da área ainda não descobriram como. Além disso, tentar reduzir o aprendizado por reforço, o aprendizado supervisionado e tudo o mais ao chunking criou mais problemas do que resolveu. Os pesquisadores assumiram a derrota e incorporaram esses outros tipos de aprendizado ao Soar como mecanismos separados. Mesmo assim, o chunking permanece sendo um exemplo de destaque de um algoritmo de aprendizado inspirado pela psicologia, e o verdadeiro Algoritmo Mestre, qualquer que seja ele, certamente deve compartilhar sua habilidade de melhorar com a prática.

O chunking e o aprendizado por reforço não são tão amplamente usados nos negócios como o aprendizado supervisionado, o clustering ou a redução de dimensionalidade, mas um tipo mais simples de aprendizado por interação com o

ambiente é: aprender os efeitos de nossas ações (e agir de acordo). Se a cor do plano de fundo da home page de seu site de e-commerce atualmente é azul e você está se perguntando se mudá-la para vermelho aumentaria as vendas, faça o teste com cem mil clientes selecionados aleatoriamente e compare os resultados com os do site comum. Essa técnica, chamada teste A/B, foi usada pela primeira vez principalmente em testes de medicamentos, mas desde então se propagou para muitas áreas em que dados podem ser coletados sob demanda, do marketing à ajuda internacional. Ela também pode ser generalizada para o teste de várias combinações de mudanças de uma só vez, sem perder o controle de quais mudanças levam a quais ganhos (ou perdas). Empresas como a Amazon e o Google são adeptos; provavelmente você já participou de milhares de testes A/B sem perceber. O teste A/B desmente as críticas frequentemente ouvidas de que o big data só é bom para a busca de correlações e não da causa. Sutilezas filosóficas à parte, aprender a causalidade é o mesmo que aprender os efeitos de nossas ações, e qualquer pessoa com um fluxo de dados sobre os quais possa influir consegue fazê-lo – de uma criança de um ano brincando na banheira a um presidente em campanha para a reeleição.

Aprendendo a se relacionar

Se dermos a Robby, o robô, todas as habilidades de aprendizado que vimos até agora neste livro, ele ficará muito inteligente, mas ainda continuará um pouco autista. Verá o mundo como uma porção de objetos separados, que conseguirá identificar, manipular e sobre os quais fará até mesmo previsões, mas não entenderá que o universo é uma teia de interconexões. Robby, o médico, seria muito bom em diagnosticar alguém com resfriado com base em seus sintomas, porém não suspeitaria que o paciente tem gripe suína porque esteve em contato com uma pessoa infectada. Antes do Google, os mecanismos de busca decidiam se uma página web era relevante para uma consulta examinando seu conteúdo – de que outra forma poderia ser? Brin e Page deduziram que a indicação mais forte de que uma página é relevante é o fato de que outras páginas relevantes levam a ela. Da mesma forma, se você quiser prever se um adolescente está prestes a começar a fumar, o melhor que pode fazer é verificar se seus amigos fumam. A forma de uma enzima é tão inseparável da forma das moléculas que reúne quanto uma fechadura o é de sua chave. Presa e predador têm propriedades profundamente entrelaçadas, desenvolvidas para derrotar as qualidades um do

outro. Em todos esses casos, a melhor maneira de entender uma entidade – seja uma pessoa, um animal, uma página web ou uma molécula – é saber como ela se relaciona com outras entidades. Isso requer um novo tipo de aprendizado que não trata os dados como uma amostra aleatória de objetos não relacionados, mas como um vislumbre de uma rede complexa. Os nós da rede interagem; o que fazemos a uma pessoa afeta as outras e retorna para nos afetar. Os aprendizes relacionais, como são chamados, podem não ter tanta inteligência social, mas constituem o próximo grande passo. No aprendizado estatístico tradicional, todo homem é uma ilha, fechado em si mesmo. No aprendizado relacional, todo homem é um pedaço do continente, uma parte do principal. Os humanos são aprendizes relacionais, feitos para se conectar, e se quisermos que Robby evolua para um robô perceptivo e socialmente adaptado, também temos de fazê-lo se conectar.

A primeira dificuldade que enfrentaremos é que, quando os dados forem uma única e grande rede, não teremos mais muitos exemplos a partir dos quais aprender, apenas um – o que não é suficiente. O Naïve Bayes aprende que a febre é um sintoma da gripe contando o número de pacientes de gripe que tiveram febre. Se ele só pudesse ver um único paciente, concluiria que a gripe sempre causa febre ou que nunca a causa, e as duas conclusões estão erradas. Gostaríamos de aprender que a gripe é contagiosa examinando o padrão de infecções em uma rede social – um grupo de pessoas infectadas aqui, um grupo de pessoas não infectadas acolá –, mas só teremos um único padrão para examinar, mesmo se estivermos em uma rede de sete bilhões de pessoas, logo não será fácil generalizar. A chave é observar que, embutidos nessa grande rede, temos muitos exemplos de *pares* de pessoas. Se pessoas que se conhecem tiverem mais probabilidade de ter gripe que pessoas que nunca se viram, então ser amigo de um paciente com gripe o tornará mais propenso a contraí-la. No entanto, infelizmente não podemos apenas contar quantos pares de conhecidos pertencentes aos dados têm gripe e transformar essas contagens em probabilidades. Isso ocorre porque uma pessoa costuma ter vários conhecidos, e todas as probabilidades em pares não levam a um modelo coerente que nos permita, por exemplo, calcular a probabilidade de alguém ter gripe dado que um de seus conhecidos a contraiu. Não tivemos esse problema quando os exemplos estavam separados e não o teríamos em, digamos, uma sociedade de casais sem filhos, com cada casal vivendo em sua própria ilha deserta. Porém, o mundo real

não é assim, e nem por isso haveria uma epidemia.

A solução seria termos um conjunto de características e aprender seus pesos, como nas redes de Markov. Para cada pessoa X, poderíamos ter a característica *X tem gripe*; para cada par de conhecidos X e Y, teríamos a característica *Tanto X quanto Y têm gripe*; e assim por diante. Como nas redes de Markov, os pesos da probabilidade máxima são os que fazem cada característica ocorrer com a frequência observada nos dados. O peso de *X tem gripe* será alto se muitas pessoas estiverem gripadas. O peso de *Tanto X quanto Y têm gripe* será alto se, quando a pessoa X tiver gripe, as chances do conhecido Y também a contrair forem mais altas que as de um membro da rede selecionado de maneira aleatória. Se 40% das pessoas estiverem gripadas assim como 16% de todos os pares de conhecidos, o peso de *Tanto X quanto Y têm gripe* será zero, porque não precisamos dessa característica para reproduzir corretamente as estatísticas dos dados (0,4 × 0,4 = 0,16). Porém, se a característica tiver peso positivo, haverá mais probabilidade de a gripe ocorrer em grupos do que apenas infectar pessoas aleatoriamente, e você estará mais propenso a pegar gripe se seus conhecidos pegarem.

Observe que a rede tem uma característica separada para cada par de pessoas: *Tanto Alice quanto Bob têm gripe*, *Tanto Alice quanto Chris têm gripe etc.* Contudo, não podemos aprender um peso separado para cada par, porque temos um único ponto de dados por par (infectado ou não), e não poderíamos generalizar para membros da rede que ainda não diagnosticamos (tanto Yvette quanto Zach têm gripe?). O que podemos fazer em vez disso é aprender um único peso para características do mesmo tipo, com base em todas as instâncias dela que foram vistas. Na verdade, *Tanto X quanto Y têm gripe* é um template de características que pode ser instanciado com cada par de conhecidos (Alice e Bob, Alice e Chris, e assim por diante). Os pesos de todas as instâncias de um template são “associados”, no sentido de que todos têm o mesmo valor, e é assim que podemos generalizar apesar de termos somente um exemplo (a rede inteira). No aprendizado não relacional, os parâmetros de um modelo são associados de uma única maneira: ao longo de todos os exemplos independentes (que poderiam ser todos os pacientes que diagnosticamos). No aprendizado relacional, cada template de características que criamos associa os parâmetros de todas as suas instâncias.

Não estamos restritos a características em pares ou individuais. O Facebook

tenta prever quem são nossos amigos para poder recomendá-los a nós. Ele pode usar a regra *Amigos de amigos têm probabilidade de ser amigos* para isso, mas cada instância dela envolve três pessoas: se Alice e Bob são amigos, e Bob e Chris também o são, então Alice e Chris são possíveis amigos. A brincadeira de H. L. Mencken, que diz que um homem rico é aquele que ganha mais que o marido da irmã de sua esposa, envolve quatro pessoas. Cada uma dessas regras pode ser transformada em um template de características de um modelo relacional, e um peso para ele pode ser aprendido com base na frequência com que a característica ocorre nos dados. Como nas redes de Markov, as próprias características podem ser aprendidas a partir dos dados.

Os aprendizes relacionais podem fazer generalizações de uma rede para outra (por exemplo, aprender um modelo de como a gripe se alastra em Atlanta e aplicá-lo a Boston). Eles também podem aprender em mais de uma rede (digamos, em Atlanta e Boston, presumindo-se imaginariamente que ninguém em Atlanta tenha em algum momento estado em contato com alguém de Boston). Porém, ao contrário do aprendizado "comum", em que todos os exemplos devem ter o mesmo número de atributos, no aprendizado relacional as redes podem variar de tamanho; uma rede maior terá mais instâncias dos mesmos templates que uma menor. É claro que a generalização de uma rede menor para uma maior pode não ser precisa, mas o importante é que nada impede que ela ocorra; e as redes grandes com frequência se comportam localmente como se fossem pequenas.

O truque mais inteligente que um aprendiz relacional pode fazer é transformar um professor esporádico em um assíduo. Para um classificador comum, exemplos sem classes são inúteis. Se me fornecerem os sintomas de um paciente, mas não o diagnóstico, isso não me ajudará a aprender a diagnosticar. Porém, se eu souber que alguns amigos do paciente têm gripe, isso será uma evidência indireta de que ele também pode tê-la. Diagnosticar algumas pessoas de uma rede e então propagar esses diagnósticos para seus amigos, e para os amigos destes, é o próximo grande passo para o diagnóstico de todo o grupo. O diagnóstico obtido pode apresentar ruído, mas as estatísticas gerais de como os sintomas se correlacionam com a gripe provavelmente serão mais precisas e completas do que se eu tivesse apenas alguns diagnósticos isolados para me basear. As crianças são muito boas em tirar o máximo da supervisão esporádica que recebem (contanto que não optem por ignorá-la). Os aprendizes relacionais

compartilham em parte essa habilidade.

No entanto, todo esse poder tem um custo. Em um classificador comum, como uma árvore de decisão ou um perceptron, inferir a classe de uma entidade a partir de seus atributos depende apenas de algumas pesquisas e um pouco de aritmética. Em uma rede, a classe de cada nó depende indiretamente da classe de todos os outros nós, e não é possível inferi-la de forma isolada. Podemos recorrer às mesmas técnicas de inferência que usamos para redes bayesianas, como a propagação de crença em loops ou o MCMC, mas a escala é diferente. Uma rede bayesiana típica pode ter milhares de variáveis, mas uma rede social tem milhões de nós ou mais. Felizmente, já que o modelo da rede é composto por muitas repetições das mesmas características com os mesmos pesos, com frequência podemos condensá-la em "supernós", cada um composto por vários nós com as mesmas probabilidades, e resolver um problema bem menor com o mesmo resultado.

O aprendizado relacional tem uma longa história, que vem desde pelo menos os anos setenta e de técnicas simbolistas como a dedução inversa. Porém, ganhou um novo ímpeto com o advento da internet. Subitamente as redes estavam em todos os lugares, e era urgente modelá-las. Um fenômeno que achei particularmente intrigante foi o boca a boca. Como as informações se propagam em uma rede social? Podemos medir a influência de cada membro e nos concentrar em um número suficiente de membros mais influentes para disparar uma onda de divulgação boca a boca? Com meu aluno Matt Richardson, projetei um algoritmo que fazia exatamente isso. Ele foi aplicado então ao Epinions, um site de avaliação de produtos que permitia que os membros dissessem quem fez as avaliações em que confiavam. Descobrimos, entre outras coisas, que anunciar um produto para o membro mais influente – que tinha a confiança de muitos seguidores, que por sua vez tinham a confiança de outros, e assim por diante – era tão bom quanto anunciar de forma isolada para um terceiro membro entre os seguidores. A partir dessa pesquisa surgiram muitas outras sobre esse problema. Desde então, apliquei o aprendizado relacional a várias pesquisas, inclusive à previsão de quem formará vínculos em uma rede social, à integração de bancos de dados e a permitir que robôs construam mapas de suas redondezas.

Se você quer entender como o mundo funciona, o aprendizado relacional é uma boa ferramenta. Na série *Fundação* de Isaac Asimov, o cientista Hari Seldon consegue prever o futuro da humanidade matematicamente e assim a

salva da decadência. Paul Krugman, entre outros, admitiu que foi esse sonho sedutor que o fez querer ser economista. De acordo com Seldon, as pessoas são como moléculas de um gás, e a lei dos grandes números assegura que mesmo se os indivíduos forem imprevisíveis, sociedades inteiras não o são. O aprendizado relacional revela por que isso não ocorre. Se as pessoas fossem independentes, tomando decisões isoladamente, as sociedades seriam previsíveis, porque todas essas decisões aleatórias produziriam uma média muito constante. Porém, quando as pessoas interagem, grupos maiores podem ser menos, e não mais, previsíveis que os menores. Se a confiança e o medo forem contagiosos, cada um dominará por algum tempo, mas de vez em quando uma sociedade inteira passará de um para o outro. No entanto, isso não é tão ruim. Se pudermos medir com que força as pessoas influenciam umas às outras, poderemos estimar quanto tempo demorará para uma mudança ocorrer, mesmo se for a primeira – outra situação em que os cisnes negros não são necessariamente imprevisíveis.

Uma reclamação comum sobre o big data é a de que quanto mais dados temos, mais fácil encontramos padrões falsos neles. Isso pode ser verdade se os dados forem apenas um imenso conjunto de entidades desconectadas, mas se estiverem correlacionados, o quadro muda. Por exemplo, críticos do uso da mineração de dados na captura de terroristas argumentam que, questões éticas à parte, ela nunca funcionará porque há muitas pessoas inocentes e poucos terroristas e, portanto, a busca de padrões suspeitos causará um excesso de alarmes falsos ou absolutamente ninguém será pego. Alguém filmando o New York City Hall seria um turista ou um terrorista selecionando o local de uma explosão? E alguém comprando grandes quantidades de nitrato de amônia seria um fazendeiro ou um fabricante de bombas? Todas essas pessoas parecem suficientemente inocentes quando sós, mas se o “turista” e o “fazendeiro” fizeram contato telefônico, e o último dirigiu sua caminhonete carregada até Manhattan, talvez seja hora de alguém verificar com mais cuidado. A NSA gosta de minerar registros de quem telefonou para quem não só porque é possivelmente legal, mas porque com frequência eles são mais informativos para os algoritmos de previsão que o conteúdo das chamadas, que teria de ser avaliado por um humano.

Se deixarmos de lado as redes sociais, a aplicação mais interessante do aprendizado relacional é na compreensão de como as células vivas funcionam. Uma célula é uma rede metabólica complexa com genes codificando proteínas que regulam outros genes, longas cadeias de reações químicas interligadas e

produtos migrando de uma organela para a outra. Entidades independentes, fazendo seu trabalho isoladamente, não são encontradas em nenhum local. Uma droga contra o câncer deve interromper o funcionamento das células cancerosas sem danificar as normais. Se tivermos um modelo relacional preciso de ambas, poderemos testar muitas drogas diferentes *in silico*, deixando o modelo inferir seus efeitos bons e ruins e mantendo somente os bons para testar *in vitro* e finalmente *in vivo*.

Como a memória humana, o aprendizado relacional tece uma rica teia de associações. Ele conecta percepções, que um robô como Robby pode adquirir por clustering e redução da dimensionalidade, a habilidades, que ele pode aprender por reforço e chunking, e a conhecimento de nível superior que vem da leitura, de ir à escola e de interagir com humanos. O aprendizado relacional é a última peça do quebra-cabeça, o ingrediente final de nossa alquimia. É hora de voltarmos ao laboratório para transmutar todos esses elementos no Algoritmo Mestre.

---

1 N.T.: Crowdsourcing é o processo de obtenção de serviços, ideias ou conteúdo mediante a solicitação de contribuições de um grande grupo de pessoas.

capítulo 9

Encaixe das peças do quebra-cabeça

O machine learning é ao mesmo tempo ciência e tecnologia, e essas duas características nos dão dicas de como unificá-lo. Pelo lado da ciência, com frequência a unificação de teorias começa com uma observação ilusoriamente simples. Dois fenômenos que não perecem ter relação acabam sendo faces da mesma moeda, e como a primeira peça de dominó que cai, essa é uma percepção que dá início a várias outras. Uma maçã que cai no chão e a Lua surgindo no firmamento: os dois fenômenos são causados pela gravidade, e – seja ou não uma história real – desde que Newton descobriu como, a gravidade também passou a ser responsável pelas marés, pela precessão dos equinócios, pela trajetória dos cometas, e muito mais. Na experiência cotidiana, a eletricidade e o magnetismo nunca são vistos juntos: um relâmpago reluz em um local, uma rocha que atrai objetos de ferro em outro, e ambos são muito raros. Porém, depois que Maxwell descobriu como um campo elétrico variável dá origem ao magnetismo e vice-versa, ficou claro que a luz provém da ocorrência indissociável dos dois, e hoje sabemos que, longe de ser raro, o eletromagnetismo permeia toda a matéria. A tabela periódica de Mendeleev não só organizou todos os elementos conhecidos em apenas duas dimensões, como também previu onde novos elementos seriam encontrados. As observações de Darwin a bordo do *Beagle* subitamente começaram a fazer sentido quando o *Ensaio sobre a população* de Malthus sugeriu a seleção natural como princípio organizador. Quando Crick e Watson encontraram a estrutura de hélice dupla como explicação para as propriedades enigmáticas do DNA, viram imediatamente como ele podia se replicar, e a transição da biologia de uma coleção de selos (nas palavras pejorativas de Rutherford) para uma ciência unificada havia começado. Em todos esses casos, uma desconcertante variedade

de observações teve uma causa comum, e quando os cientistas a identificaram, puderam usá-la para prever muitos fenômenos novos. Da mesma forma, ainda que os aprendizes que vimos neste livro pareçam bem diferentes – alguns baseados no cérebro, outros na evolução e ainda outros em princípios matemáticos abstratos – na verdade eles têm muito em comum, e a teoria do aprendizado resultante produz vários insights novos.

Embora esse seja um fato pouco conhecido, muitas das tecnologias mais importantes do mundo resultaram da invenção de um unificador, um mecanismo único capaz de fazer o que antes demandava um esforço conjunto. A internet, como o nome sugere, é uma rede que interconecta redes. Sem ela, cada tipo de rede precisaria de um protocolo diferente para conversar com os outros tipos, de maneira semelhante a como precisamos de um dicionário diferente para cada par de idiomas existente no mundo. Os protocolos da internet são uma espécie de esperanto que dá a cada computador a impressão de estar conversando diretamente com os outros computadores e que permite que o correio eletrônico e a web ignorem os detalhes da infraestrutura física pela qual eles fluem. Os bancos de dados relacionais fazem algo parecido para aplicações empresariais, permitindo que os desenvolvedores e usuários considerem o modelo relacional abstrato e ignorem as diferentes maneiras pelas quais os computadores respondem consultas. Um microprocessador é um conjunto de componentes eletrônicos digitais que pode simular qualquer conjunto de componentes. As máquinas virtuais permitem que o mesmo computador aja como vários computadores diferentes para muitas pessoas ao mesmo tempo e ajudam a tornar a nuvem possível. As interfaces gráficas de usuário nos permitem editar documentos, planilhas, conjuntos de slides, e muito mais usando uma linguagem comum de janelas, menus e cliques do mouse. O próprio computador é um unificador: um dispositivo capaz de resolver qualquer problema lógico ou matemático, se soubermos como programá-lo. E a eletricidade também é: você pode gerá-la a partir de muitas fontes diferentes – carvão, gás, energia nuclear, água, vento, sol – e consumi-la de várias maneiras. Uma usina elétrica não precisa saber como a eletricidade produzida será consumida, e a lâmpada da varanda, a lavadora de pratos ou um Tesla novo em folha não tem ideia de onde vem a eletricidade que lhes é fornecida. A eletricidade é o esperanto da energia. O Algoritmo Mestre é o unificador do machine learning: ele permite que um aplicativo use qualquer aprendiz, já que condensa os aprendizes em uma forma

comum que é só o que os aplicativos precisam conhecer.

O primeiro passo que daremos em direção ao Algoritmo Mestre é surpreendentemente simples. Não é difícil combinar muitos aprendizes em um com o uso do que é conhecido como meta-aprendizado. O Netflix, o Watson, o Kinect e muitos outros aplicativos o usam, e ele é uma das armas mais poderosas do arsenal do machine learning. Também é um trampolim para a unificação mais profunda que vem a seguir.

A partir de muitos modelos surge um único algoritmo

Vou lançar um desafio: você tem quinze minutos para combinar árvores de decisão, perceptrons multicamadas, sistemas classificadores, o algoritmo Naïve Bayes e as SVMs em um único algoritmo contendo as melhores propriedades de cada um. Seja rápido – o que você pode fazer? É claro que não é possível incluir os detalhes dos algoritmos individuais; não há tempo. E se fizer o seguinte? Considere cada aprendiz como um especialista em um comitê. Cada um deles olha cuidadosamente para a instância a ser classificada – qual é o diagnóstico deste paciente? – e faz sua previsão com confiança. Você não é um especialista, mas é o presidente do comitê, e sua missão é combinar as recomendações em uma decisão final. O que tem em suas mãos é na verdade um novo problema de classificação, em que em vez dos sintomas do paciente, as entradas são as opiniões dos especialistas. Contudo, você pode aplicar o machine learning a esse problema da mesma forma que os especialistas o aplicaram ao original. Chamamos isso de meta-aprendizado porque ele está aprendendo sobre os aprendizes. O meta-aprendiz pode ser qualquer aprendiz, de uma árvore de decisão a uma simples votação ponderada. Para aprender os pesos, ou a árvore de decisão, substituímos os atributos de cada exemplo original pelas previsões dos aprendizes. Os aprendizes que costumam prever a classe correta recebem pesos altos, e os imprecisos tendem a ser ignorados. Com o uso de uma árvore de decisão, decidir se um aprendiz será usado pode depender das previsões de outros aprendizes. Seja como for, para obter a previsão de um aprendiz para um exemplo de treinamento específico, primeiro devemos aplicá-lo ao conjunto de treinamento original *excluindo esse exemplo* e usar o classificador resultante – caso contrário o comitê correrá o risco de ter predominantemente aprendizes que cometem sobreajuste, já que podem prever a classe correta apenas se lembrando dela. O ganhador do Netflix Prize usou o meta-aprendizado para combinar

centenas de aprendizes diferentes. O Watson o usa para selecionar sua resposta final entre as candidatas disponíveis. Nate Silver combina votos de maneira semelhante para prever resultados de eleições.

Esse tipo de meta-aprendizado é chamado de stacking (empilhamento) e foi inventado por David Wolpert, que vimos no Capítulo 3 como sendo o autor do teorema "não existe almoço grátis". Um aprendiz ainda mais simples é o bagging, inventado pelo estatístico Leo Breiman. O bagging gera variações aleatórias do conjunto de treinamento por reamostragem, aplica o mesmo aprendiz a cada variação e combina os resultados por votação. Ele faz isso para reduzir a variância: o modelo combinado é bem menos sensível a imprecisões nos dados que qualquer modelo individual, o que o torna uma maneira muito fácil de melhorar a precisão. Se os modelos forem árvores de decisão e as variarmos ainda mais retendo um subconjunto aleatório dos atributos a partir de considerações em cada nó, o resultado será a chamada floresta aleatória. As florestas aleatórias são alguns dos classificadores mais precisos que existem. O Kinect da Microsoft as usa para descobrir o que estamos fazendo, e elas também costumam vencer competições de machine learning.

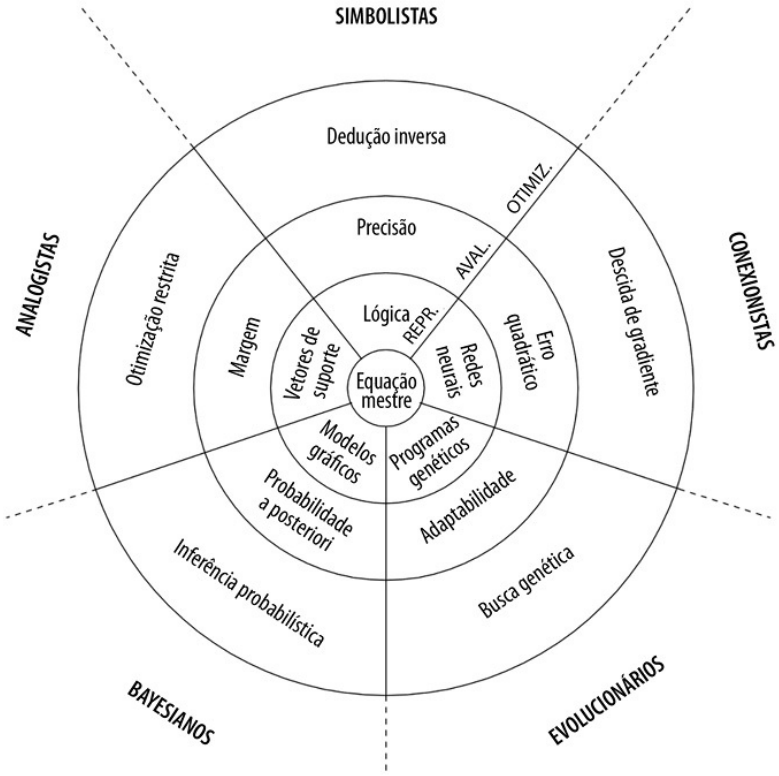
Um dos meta-aprendizes mais inteligentes é o de boosting, criado por dois estudiosos do aprendizado, Yoav Freund e Rob Schapire. Em vez de combinar diferentes aprendizes, o método de boosting aplica repetidamente o mesmo classificador aos dados, usando cada novo modelo para corrigir os erros dos anteriores. Ele faz isso atribuindo pesos aos exemplos de treinamento; o peso de um exemplo classificado de maneira incorreta é aumentado após cada rodada de aprendizado, o que faz as rodadas posteriores se concentrarem mais nele. O nome *boosting* vem da noção de que esse processo pode aprimorar (boost) um classificador que seja apenas um pouco melhor que a adivinhação aleatória, mas de forma tão consistente que ele se torne um classificador quase perfeito.

O meta-aprendizado é notavelmente bem-sucedido, porém não é uma maneira muito profunda de combinar modelos. Ele também é caro, demandando muitas rodadas de aprendizado, e os modelos combinados podem ser bem opacos. ("Acredito que você tenha câncer de próstata porque é o que mostram a árvore de decisão, o algoritmo genético e o Naïve Bayes, embora o perceptron multicamadas e a SVM discordem".) Além disso, todos os modelos combinados são na verdade apenas um grande e confuso modelo. Não podemos ter um único aprendiz que realize a mesma tarefa? Sim, podemos.

O Algoritmo Mestre

Talvez a melhor maneira de introduzir o nosso aprendiz unificado seja por intermédio de uma alegoria estendida. Se o machine learning é um continente dividido nos territórios das cinco tribos, o Algoritmo Mestre é a cidade que representa sua capital e está situado no único local em que os cinco territórios se encontram. À medida que nos aproximamos, podemos ver que a cidade é composta por três círculos concêntricos, cada um cercado por um muro. O círculo mais externo e sem dúvida mais largo é o Município da Otimização. Cada casa contida nele é um algoritmo, que existem em todas as formas e tamanhos. Algumas estão em construção, com os habitantes locais andando ocupadamente ao seu redor, outras são novas em folha e ainda outras parecem antigas e abandonadas. Mais para cima da colina está a Fortaleza da Avaliação. De suas mansões e palácios são emitidas continuamente ordens para os algoritmos abaixo. Acima de tudo, projetadas contra o céu, erguem-se as Torres de Representação. Aqui vivem os legisladores da cidade. Suas leis imutáveis definem o que pode ou não ser feito não só na cidade, mas em todo o continente. Em cima da torre central e mais alta tremula a bandeira do Algoritmo Mestre, vermelha e preta, com uma estrela de cinco pontas ao redor de uma inscrição que ainda não temos como decifrar.

A cidade está dividida em cinco setores, cada um pertencendo a uma das cinco tribos. Cada setor se estende para baixo a partir de sua Torre de Representação e continua até os muros mais externos da cidade, englobando a torre, um conjunto de palácios da Fortaleza da Avaliação e as ruas e casas do Município da Otimização para as quais tem vista. Os cinco setores e os três anéis dividem a cidade em quinze distritos, quinze formas, quinze peças do quebra-cabeça que precisamos montar:



Você examina o mapa atentamente, tentando decifrar seu segredo. Todas as quinze peças se encaixam de maneira quase perfeita, mas é preciso descobrir como elas se combinam para formar apenas três: os componentes do Algoritmo Mestre referentes à representação, avaliação e otimização. Todos os aprendizes têm esses três elementos, porém variam de uma tribo para a outra.

A representação é a linguagem formal em que o aprendiz expressa seus modelos. A linguagem formal dos simbolistas é a lógica, cujas regras e árvores de decisão são casos especiais. A dos conexionistas são as redes neurais. A dos evolucionários são os programas genéticos, inclusive os sistemas classificadores. A dos bayesianos são os modelos gráficos, um termo que abrange as redes bayesianas e de Markov. A dos analogistas são as instâncias específicas, possivelmente com pesos, como em uma SVM.

O componente referente à avaliação é uma função de pontuação que mostra quanto um modelo é bom. Os simbolistas usam a precisão ou o ganho de informações. Os conexionistas empregam uma medida de erro contínuo, como o erro quadrático, que é a soma dos quadrados das diferenças entre os valores previstos e os reais. Os bayesianos usam a probabilidade *a posteriori*. Os analogistas (pelo menos os adeptos da SVM) usam a margem. Além da

adaptabilidade do modelo aos dados, todas as tribos levam em consideração outras propriedades desejáveis, como a simplicidade do modelo.

A otimização é o algoritmo que procura o modelo de pontuação mais alta e o retorna. O algoritmo de busca característico dos simbolistas é a dedução inversa. O dos conexionistas é a descida de gradiente. O dos evolucionários é a busca genética, inclusive o crossover e a mutação. Os bayesianos têm uma maneira própria de tratar disso: não procuram apenas o melhor modelo, mas a média de todos os modelos, ponderada pelo seu nível de probabilidade. Para calcular os pesos com eficiência, eles usam algoritmos de inferência probabilística como o MCMC. Os analogistas (ou mais precisamente, os peritos em SVMs) usam a otimização restrita para encontrar o melhor modelo.

Após a jornada de um longo dia, o Sol se aproxima rapidamente do horizonte, e você precisa se apressar antes que escureça. O muro mais externo da cidade tem cinco portões maciços, cada um controlado por uma das tribos e levando a seu distrito no Município da Otimização. Entraremos pelo Portão da Descida de Gradiente, após sussurrar a senha – "aprendizado profundo" – para o guarda, e seguiremos a espiral em direção às Torres de Representação. A partir do portão, a estrada sobe a colina de modo íngreme até o Portão do Erro Quadrático da fortaleza, mas em vez disso você virará à esquerda em direção ao setor evolucionário. As casas do distrito da descida de gradiente são todas de curvas suaves e padrões densamente entrelaçados, parecendo mais uma floresta que uma cidade. Porém, quando a descida de gradiente dá caminho à busca genética, o quadro muda. Aqui as casas são mais altas, com estruturas empilhadas umas sobre as outras, mas as estruturas são esparsas, quase vazias, como se esperassem ser preenchidas por curvas de descida de gradiente. É isso: a maneira de combinar as duas é usar a busca genética para encontrar a estrutura do modelo e deixar que a descida de gradiente o preencha com seus parâmetros. É o que a natureza faz: a evolução cria estruturas cerebrais, e a experiência individual as modula.

Dado o primeiro passo, você corre para o distrito bayesiano. Mesmo à distância, é possível ver como ele se aglomera ao redor da Catedral do Teorema de Bayes. A Alameda MCMC ziguezagueia aleatoriamente ao longo do caminho. Isso vai demorar um pouco. Você toma um atalho na Rua da Propagação de Crenças, mas ela parece formar um loop infinito. Então a vê: a Avenida da Maior Probabilidade, erguendo-se majestosa em direção ao Portão

da Probabilidade a Posteriori. Em vez de calcular a média de todos os modelos, você pode seguir direto para o mais provável, certo de que as previsões resultantes serão quase iguais. E pode deixar a busca genética selecionar a estrutura do modelo e a descida de gradiente selecionar seus parâmetros. Com um suspiro de alívio, você percebe que essa é toda a inferência probabilística de que precisava, pelo menos até chegar a hora de responder perguntas usando o modelo.

Você continua o trajeto. O distrito da otimização restrita é um labirinto de vielas estreitas e becos sem saída, com exemplos de todos os tipos encostados lado a lado em vários locais e uma ocasional clareira ao redor de um vetor de suporte. Obviamente, tudo que você precisa fazer para não se deparar com exemplos da classe errada é adicionar restrições ao otimizador que compôs. No entanto, pensando bem, nem isso é necessário. Quando aprendemos com SVMs, geralmente deixamos margens serem violadas para evitar o sobreajuste, contanto que cada violação sofra uma penalidade. Nesse caso os pesos de exemplos ótimos podem ser aprendidos de novo por uma espécie de descida de gradiente. Isso foi fácil. Você sente que está pegando o jeito.

As densas graduações das instâncias terminam abruptamente e você se vê no distrito da dedução inversa, um lugar de largas avenidas e antigos prédios de pedra. A arquitetura aqui é geométrica, austera, feita de linhas e ângulos retos. Até as árvores rigorosamente podadas têm troncos retangulares e suas folhas são rotuladas de maneira meticulosa com previsões de classes. Os moradores deste distrito parecem construir suas casas de modo peculiar: começam com o telhado, que chamam de “Conclusões”, e aos poucos preenchem as lacunas entre ele e o chão, que chamam de “Premissas”. Eles encontram cada bloco de pedra que tenha a forma certa para preencher uma lacuna específica e o içam até o seu lugar. Mas você notou que muitas lacunas têm a mesma forma, e seria mais rápido cortar e combinar os blocos até se adequarem a ela e então repetir o processo quantas vezes forem necessárias. Em outras palavras, você poderia usar a busca genética para executar a dedução inversa. Ótimo. Parece que resumimos os cinco otimizadores em uma receita simples: busca genética para a estrutura e descida de gradiente para os parâmetros. E até mesmo isso pode ser um exagero. Para vários problemas, é possível reduzir a busca genética à escalada de montanha se você fizer três coisas: deixar de fora o crossover, testar todos os pontos de mutação possíveis em cada geração e sempre selecionar a melhor

hipótese para alimentar a próxima geração.

Que estátua é aquela ali em frente? Aristóteles, olhando em desaprovação para a bagunça do quarteirão da descida de gradiente. Você deu uma volta completa. Está com o otimizador unificado que precisa dar ao Algoritmo Mestre, mas não é hora de se gabar. A noite chegou e ainda há muito a fazer. Você entra na Fortaleza da Avaliação pelo imponente, mas um pouco estreito, Portão da Precisão. A inscrição acima dele diz "Você que está entrando, abandone toda a esperança de sobreajuste". Ao dar a volta pelos palácios dos avaliadores das cinco tribos, você coloca as peças mentalmente em seus lugares. Usa a precisão para avaliar previsões de tipo sim-ou-não e o erro quadrático para as contínuas. Adaptabilidade é apenas o nome que os evolucionários dão à função de pontuação; você pode usar a que quiser, inclusive a precisão e o erro quadrático. A probabilidade *a posteriori* se reduzirá ao erro quadrático se você ignorar a probabilidade *a priori* e os erros seguirem uma distribuição normal. Caso possa ser violada em troca de um preço, a margem se torna uma versão mais leve da precisão: em vez de não sofrer penalidade por uma previsão correta e estar sujeita a ela por uma previsão incorreta, a penalidade será zero até que você esteja dentro da margem, ponto em que ela começa a subir constantemente. Puxa! Combinar os avaliadores foi muito mais fácil que combinar os otimizadores. Mas as Torres de Representação, que se erguem acima, o enchem de uma sensação de mal-estar.

Você alcançou o estágio final de sua busca. Bate então na porta da Torre de Vetores de Suporte. Um guarda de aparência ameaçadora a abre e de repente você se lembra de que não sabe a senha. Fala "Kernel" sem pensar, tentando não demonstrar pânico. O guarda se curva e se põe de lado. Voltando à calma, você entra, repreendendo-se mentalmente por seu descuido. Todo o andar térreo da torre está tomado por um aposento circular prodigamente mobiliado, com o que parece ser uma representação em mármore de uma SVM ocupando o lugar de honra no centro. Conforme você a circunda, nota uma porta no lado oposto. Ela deve levar à torre central – a Torre do Algoritmo Mestre. A porta parece desguarnecida. Você decide tomar um atalho. Esgueirando-se pela entrada, caminha por um corredor curto e se vê em um aposento pentagonal ainda maior, com uma porta em cada parede. No centro, uma escada em espiral sobe tão alto quanto os olhos podem ver. Você ouve vozes acima e se esconde na entrada oposta. Ela leva à Torre de Redes Neurais. Mais uma vez você está em um

aposento circular, e este tem uma escultura de um perceptron multicamadas como peça central. Suas partes são diferentes das da SVM, mas sua disposição é muito semelhante. Subitamente você entende: uma SVM é apenas um perceptron multicamadas com uma camada oculta composta por kernels, em vez de curvas S, e uma saída, que é uma combinação linear em vez de outra curva S.

As outras representações também teriam uma forma semelhante? Cada vez mais empolgado, você volta pelo aposento pentagonal e entra na Torre da Lógica. Olhando para a representação de um conjunto de regras no centro, tenta descobrir um padrão. Sim! Cada regra é apenas um neurônio altamente estilizado. Por exemplo, a regra *Se é um réptil gigante e lança fogo, então é um dragão* é somente um perceptron com pesos 1 para *é um réptil gigante* e *lança fogo* e um limite de 1,5. Além disso, um conjunto de regras é um perceptron multicamadas com uma camada oculta contendo um único neurônio para cada regra e um neurônio de saída para formar a disjunção das regras. Há uma dúvida lancinante em sua mente, mas não há tempo para ela agora. Na passagem pelo aposento pentagonal até a Torre de Programas Genéticos, você já consegue ver como integrá-los ao conjunto. Os programas genéticos são apenas programas, e programas são somente estruturas lógicas. A escultura de um programa genético no aposento tem a forma de uma árvore, com as sub-rotinas se ramificando em mais sub-rotinas, e quando você olha atentamente para as folhas, vê que elas são apenas regras simples. Logo, os programas se resumem a regras, e se regras podem ser reduzidas a neurônios, os programas também podem.

Hora de ir à Torre de Modelos Gráficos. Infelizmente, a escultura de seu aposento circular não se parece com as outras. Um modelo gráfico é um produto de fatores: probabilidades condicionais, no caso de redes bayesianas, e funções não negativas do estado, no caso de redes de Markov. Mesmo se tentarmos ao máximo, não conseguiremos ver a conexão com redes neurais ou conjuntos de regras. Você se sente, então, frustrado. Porém, coloca seus “loggles”1, que substituem uma função por seu logaritmo. Heureca – agora o produto de fatores é uma soma de termos, como uma SVM, um conjunto de regras de votação ou um perceptron multicamadas sem a curva S da saída. Por exemplo, você pode transformar um classificador Naïve Bayes de dragões em um perceptron cujo peso de *lança fogo* seja o logaritmo de *P*(*lança fogo* | *dragão*) menos o logaritmo de *P*(*lança fogo* | *não dragão*). Contudo, é claro que os modelos gráficos são muito mais gerais que isso porque podem representar distribuições

de probabilidade em muitas variáveis, não apenas a distribuição de uma variável (a classe) dadas as outras (os atributos).

Você conseguiu! Ou não? Absorver SVMs em redes neurais e redes neurais em modelos gráficos funcionou. Também funcionou absorver programas genéticos em lógica. Mas combinar lógica e modelos gráficos? Há algo errado aí. Tardiamente, você vê o problema: a lógica tem uma dimensão que falta aos modelos gráficos e vice-versa. As esculturas dos cinco aposentos coincidiram porque eram simples alegorias, mas a realidade não o é. Os modelos gráficos não nos permitem representar regras que envolvam mais de um objeto, como *Amigos de amigos são amigos*; todas as suas variáveis têm de ser propriedades do mesmo objeto. Eles também não podem representar programas arbitrários, que passem conjuntos de variáveis de uma sub-rotina para outra. A lógica pode fazer as duas coisas facilmente, mas não pode representar a incerteza, a ambiguidade ou graus de similaridade. E sem uma representação que possa fazer tudo isso, não temos um aprendiz universal.

Você se esforça para encontrar uma solução, porém quanto mais tenta, mais difícil lhe parece. Talvez unificar lógica e probabilidade esteja além da habilidade humana. Exausto, você adormece. Um rugido profundo o acorda. O monstro da complexidade com cabeça de Hidra o ataca com as mandíbulas abertas, mas você se esquiva no último momento. Golpeando-o de modo desesperado com a espada do aprendizado, a única que pode matá-lo, finalmente consegue cortar todas as suas cabeças. Antes que novas cresçam, você sobe a escada.

Após uma árdua subida, alcança o topo. Um casamento está ocorrendo. Praedicatus, o Primeiro Lorde da Lógica, legislador do reino simbólico e Protetor dos Programas, diz para Markovia, a Princesa da Probabilidade e Imperatriz das Redes: “Unifiquemos nossos reinos. Às minhas regras tu deves adicionar pesos, criando uma nova representação que se espalhará por toda a terra”. A princesa diz: “E chamaremos nossa criação de redes lógicas de Markov”.

Sua cabeça está girando. Você sai para a sacada. O Sol nasceu sobre a cidade. Você olha por cima dos telhados para o campo. Florestas de servidores se estendem em todas as direções, farfalhando calmamente, esperando pelo Algoritmo Mestre. Comboios se movem ao longo da estrada, carregando ouro das minas de dados. Mais para o oeste, a terra se abre em um mar de

informações, pontilhado com navios. Você olha para cima para a bandeira do Algoritmo Mestre. Agora consegue ver claramente a inscrição contida na estrela de cinco pontas:

$$P = e^{p \cdot n} / Z$$

Qual será o significado, você se pergunta.

Redes lógicas de Markov

Em 2003, comecei a pensar sobre o problema de como unificar lógica e probabilidade, com a ajuda de meu aluno Matt Richardson. Inicialmente fizemos pouco progresso porque estávamos tentando fazê-lo com redes bayesianas, e sua forma rígida – uma ordem rigorosa para as variáveis, distribuições condicionais de filhos dados os pais – é incompatível com a flexibilidade da lógica. Porém, no dia anterior à véspera de Natal, percebi que havia uma maneira muito melhor. Se mudássemos para as redes de Markov, poderíamos usar *qualquer* fórmula lógica como template de características da rede, e isso unificaria lógica e modelos gráficos. Vejamos como.

Você deve se lembrar de que a rede de Markov é definida por uma soma ponderada de características, semelhante ao que ocorre em um perceptron. Suponhamos que tivéssemos um conjunto de fotos de pessoas. Selecionamos uma foto aleatória e avaliamos características como *A pessoa tem cabelo grisalho*, *A pessoa é idosa*, *A pessoa é uma mulher*, e assim por diante. Em um perceptron, compararíamos a soma ponderada dessas características com um limite para decidir se, digamos, a pessoa é ou não nossa avó. Em uma rede de Markov, fazemos algo bem diferente (pelo menos inicialmente): exponenciamos a soma ponderada, a transformamos em um produto de fatores, e esse produto é a probabilidade de seleção dessa foto específica no conjunto, não importando se nossa avó faz ou não parte dele. Se você tiver muitas fotos de pessoas idosas, o peso dessa característica aumentará. Se a maioria delas for de homens, o peso de *A pessoa é uma mulher* diminuirá. Podemos usar as características que quisermos, o que torna as redes de Markov uma maneira incrivelmente flexível de representar distribuições de probabilidade.

Na verdade, não é bem assim: o produto de fatores ainda não é uma probabilidade porque as probabilidades de todas as fotos devem produzir um total igual a 1, e não há garantias de que os produtos de fatores de todas as

imagens totalizem isso. Precisamos normalizá-los, ou seja, dividir cada produto pela soma de todos eles. A soma de todos os produtos normalizados garantirá então o resultado 1 porque será apenas um número dividido por ele mesmo. Logo, a probabilidade de uma foto é a soma ponderada de suas características, exponenciada e normalizada. Se você examinar novamente a equação da estrela de cinco pontas, deve começar a ter uma ideia do que ela significa. *P* é uma probabilidade, *p* é um vetor de pesos (observe que está em negrito), *n* é um vetor de números, e seu produto representado por um ponto é exponenciado e dividido por *Z*, a soma de todos os produtos. Se definirmos que o primeiro componente de *n* será 1 caso a primeira característica da imagem seja verdadeira e que de outra forma será zero, e assim por diante, *w*•*n* é apenas uma abreviação da soma ponderada de características sobre a qual estivemos falando.

Portanto, a equação fornece a probabilidade de uma imagem (ou do que quer que seja) de acordo com uma rede de Markov. Contudo, ela é mais geral que isso porque não é apenas a equação de uma rede de Markov; é a equação de uma rede lógica de Markov, como a chamamos. Em uma rede lógica de Markov, ou MLN (Markov logic network), os números de *n* não têm de ser apenas zero ou um e não se referem a características – referem-se a fórmulas lógicas. No fim do Capítulo 8, vimos como podemos ir além das redes de Markov e chegar aos modelos relacionais, que são definidos em termos de templates de características, não apenas de características. *Tanto Alice quanto Bob têm gripe* é uma característica específica de Alice e Bob. *Tanto X quanto Y têm gripe* é um template de características, que pode ser instanciado com Alice e Bob, Alice e Chris e qualquer outro par de pessoas. Um template de características é algo poderoso porque pode resumir bilhões de características ou mais em uma única e curta expressão. Porém, precisamos de uma linguagem formal para definir templates de características, e já temos uma disponível: a lógica.

Uma MLN é apenas um conjunto de fórmulas lógicas e seus pesos. Quando aplicada a um conjunto de entidades específico, ela define uma rede de Markov com os estados possíveis. Por exemplo, se as entidades forem Alice e Bob, um estado possível seria o de Alice e Bob serem amigos, Alice ter gripe e Bob também. Suponhamos que a MLN tivesse duas fórmulas: *Todos têm gripe* e *Se alguém tem gripe, seus amigos também têm*. Na lógica-padrão, esse seria um par de declarações bastante inútil: a primeira descartaria qualquer estado que tivesse até mesmo uma única pessoa saudável e a segunda seria redundante. Porém, em

uma MLN, a primeira fórmula significa somente que há uma característica *X tem gripe* para cada pessoa X, com o mesmo peso da fórmula. Se as pessoas tiverem probabilidade de contrair gripe, a fórmula terá peso alto e o mesmo ocorrerá com as características correspondentes. Um estado com muitas pessoas saudáveis apresenta menos probabilidade de ocorrência de gripe que um com poucas pessoas saudáveis, mas isso não é impossível. E por causa da segunda fórmula, um estado em que alguém tenha gripe e seus amigos não é menos provável que um em que pessoas saudáveis e pessoas infectadas estejam em grupos de amigos separados.

A essa altura você já deve ter adivinhado o que é o *n* da equação mestre: seu primeiro componente é o número de instâncias verdadeiras da primeira fórmula existentes no estado, o segundo é o número de instâncias verdadeiras da segunda fórmula *etc.* Se estivéssemos examinando um grupo de dez amigos e sete deles tivessem gripe, o primeiro componente de *n* seria sete, e assim por diante. (A probabilidade não deveria ser diferente se sete entre vinte em vez de sete entre dez amigos tivessem gripe? Sim, e ela é, por causa de *Z*.) Em um caso extremo, se deixarmos todos os pesos alcançarem o infinito, a lógica de Markov será reduzida à lógica-padrão porque violar uma única instância de uma fórmula fará a probabilidade cair para zero, tornando o estado impossível. Pelo lado probabilístico, uma MLN é reduzida a uma rede de Markov quando todas as fórmulas se referem ao mesmo objeto. Logo, a lógica de Markov inclui tanto a lógica quanto as redes de Markov como casos especiais e é a unificação que procurávamos.

Aprender uma MLN significa descobrir fórmulas que sejam verdadeiras um número maior de vezes que a probabilidade aleatória preveria e calcular pesos para essas fórmulas que façam as probabilidades previstas coincidirem com as frequências observadas. Depois que aprendermos uma MLN, poderemos usá-la para responder perguntas como “Qual a probabilidade de Bob ter gripe, já que é amigo de Alice e ela a contraiu?”. O interessante é que a probabilidade é dada por uma curva S aplicada à soma ponderada das características, de maneira semelhante a como ocorre em um perceptron multicamadas. E uma MLN com longas cadeias de regras pode representar uma rede neural profunda, com uma camada por elo da cadeia.

É claro que não devemos ficar desapontados com a MLN simples mostrada anteriormente que apenas prevê a disseminação da gripe. No lugar dela,

devemos pensar em uma MLN para o diagnóstico e a cura do câncer. Essa MLN representa uma distribuição de probabilidade pelos estados de uma célula. Cada parte da célula, cada caminho metabólico, cada gene e proteína são entidades da rede, e suas fórmulas codificam as dependências entre elas. Podemos perguntar à MLN: “Esta célula é cancerosa?” e testá-la com diferentes drogas para ver o que acontece. Ainda não temos uma MLN assim, mas posteriormente neste capítulo veremos como ela poderia ser criada.

Recapitulando: o aprendiz unificado a que chegamos usa MLNs como representação, probabilidade *a posteriori* como função de avaliação, e busca genética associada à descida de gradiente como otimizador. Se quisermos, podemos substituir facilmente a probabilidade *a posteriori* por alguma outra medida de precisão, ou a busca genética pela escalada de montanha. Subimos em um pico alto e agora podemos apreciar a vista. No entanto, eu não seria tão precipitado a ponto de chamar esse aprendiz de Algoritmo Mestre. Em primeiro lugar, só podemos avaliar o pudim comendo-o, e embora na última década esse algoritmo (ou variações dele) tenha sido aplicado com sucesso em muitas áreas, há várias outras em que não houve sucesso, portanto ainda não está claro seu nível de generalização. Em segundo lugar, há alguns problemas importantes que ele não resolve. Porém, antes de os examinarmos, veremos o que ele pode fazer.

De Hume ao seu robô doméstico

Você pode baixar o aprendiz que acabei de descrever a partir de alchemy.cs.washington.edu. Nós o batizamos de Alchemy para nos lembrar de que, apesar de todos os seus sucessos, o machine learning ainda está no estágio alquímico da ciência. Se você o baixar, verá que ele inclui muito mais que apenas o algoritmo básico que descrevi, mas também que ainda faltam algumas coisas que mencionei que o aprendiz universal deveria ter, como o crossover. Mesmo assim, para simplificar usaremos o nome Alchemy para nos referir ao nosso candidato a aprendiz universal.

O Alchemy responde à pergunta original de Hume usando outra entrada além dos dados: nosso conhecimento inicial, na forma de um conjunto de fórmulas lógicas, com ou sem pesos. As fórmulas podem ser inconsistentes, incompletas, ou até estar erradas; o aprendizado e o raciocínio probabilístico resolvem esse problema. O ponto-chave é que o Alchemy não tem de aprender a partir do zero. Na verdade, podemos pedir que mantenha as fórmulas inalteradas e aprenda

somente os pesos. Nesse caso, o fornecimento das fórmulas apropriadas pode transformá-lo em uma máquina de Boltzmann, em uma rede bayesiana, em um aprendiz baseado em instâncias e em muitos outros modelos. Isso explica por que podemos ter um aprendiz universal apesar do teorema "não existe almoço grátis". O Alchemy seria mais como uma máquina de Turing indutiva, que podemos programar para que se comporte como um aprendiz muito poderoso ou muito restrito; depende de nós. Ele fornece um unificador para o machine learning da mesma forma que a internet fornece um para as redes de computadores, o modelo relacional para os bancos de dados, ou a interface gráfica de usuário para os aplicativos comuns.

É claro que mesmo se você usar o Alchemy sem fórmulas iniciais (e é possível fazê-lo), isso não o torna destituído de conhecimento. A escolha da linguagem formal, da função de pontuação e do otimizador codifica implicitamente suposições sobre o mundo. Logo, é natural alguém perguntar se podemos ter um aprendiz ainda mais genérico que o Alchemy. O que a evolução supôs quando começou sua longa jornada das primeiras bactérias a todas as formas de vida que existem hoje? Acho que há uma suposição simples a partir da qual surge todo o resto: o aprendiz faz parte do mundo. Ou seja, o aprendiz como sistema físico obedece às mesmas leis de seu ambiente, quaisquer que sejam e, portanto, já as "conhece" de maneira implícita e é instruído a descobri-las. Na próxima seção, veremos o que essa suposição pode significar concretamente e como incorporá-la no Alchemy. Por enquanto, devemos notar que talvez essa seja a melhor resposta que poderíamos dar para a pergunta de Hume. Por um lado, presumir que o aprendiz faz parte do mundo *é* uma suposição – em princípio, o aprendiz poderia obedecer a leis diferentes das que o mundo obedece –, de modo que satisfaz a máxima de Hume de que o aprendizado só é possível com conhecimento prévio. Por outro, é uma suposição tão básica e difícil de desaprovar que pode ser tudo o que precisamos para este mundo.

No outro extremo, os engenheiros do conhecimento – os críticos mais determinados do machine learning – têm uma boa razão para gostar do Alchemy. Em vez de uma estrutura de modelo básica ou alguns palpites aproximados, o Alchemy pode usar uma grande e cuidadosamente reunida base de conhecimento, se ela estiver disponível. Já que as regras probabilísticas conseguem interagir de maneiras muito mais sofisticadas que as determinísticas, codificar conhecimento de forma manual abre mais portas na lógica de Markov.

E já que as bases de conhecimento da lógica de Markov não precisam ser autoconsistentes, elas podem ser muito extensas e acomodar vários colaboradores diferentes sem se fragmentar – um objetivo que até o momento frustrou os engenheiros do conhecimento.

No entanto, mais que qualquer outra coisa, o Alchemy resolve os problemas nos quais cada uma das cinco tribos de machine learning trabalhou por tanto tempo. Vamos examiná-los um de cada vez.

Os simbolistas combinam conhecimentos diferentes dinamicamente, da mesma forma que os matemáticos combinam axiomas para provar teoremas. Isso é o oposto das redes neurais e outros modelos de estrutura fixa. O Alchemy o faz usando lógica, como os simbolistas, mas com uma peculiaridade. Para provar um teorema em lógica, precisamos encontrar uma única sequência de aplicações do axioma que o produza. Já que o Alchemy raciocina de maneira probabilística, ele faz mais: encontra várias sequências de fórmulas que levam ao teorema ou à sua negação e fornece pesos a elas para calcular a probabilidade de o teorema ser verdadeiro. Dessa forma pode raciocinar não só sobre proposições matemáticas universais, mas sobre se o termo “o presidente” citado em uma notícia significa “Barack Obama”, ou em qual pasta um email deve ser arquivado. O algoritmo mestre dos simbolistas, a dedução inversa, postula novas regras lógicas necessárias para servir como etapas entre os dados e uma conclusão desejada. O Alchemy introduz novas regras por escalada de montanha, começando com as regras iniciais e construindo regras que, combinadas com as iniciais e os dados, tornam as conclusões mais prováveis.

Os modelos dos conexionistas são inspirados no cérebro, com redes de curvas S que correspondem a neurônios e conexões ponderadas entre elas correspondentes a sinapses. No Alchemy, duas variáveis estão conectadas quando aparecem juntas em alguma fórmula, e a probabilidade de uma variável ocorrer dados seus vizinhos é uma curva S. (Não vou mostrar por que, mas isso é consequência direta da equação mestre que vimos na seção anterior.) O algoritmo mestre dos conexionistas é a backpropagation, que eles usam para descobrir quais neurônios são responsáveis por quais erros e para ajustar seus pesos de acordo. A backpropagation é um tipo de descida de gradiente, que o Alchemy usa para otimizar os pesos de uma rede lógica de Markov.

Os evolucionários usam algoritmos genéticos para simular a seleção natural. Um algoritmo genético mantém uma população de hipóteses e a cada geração

faz o cruzamento das mais aptas e as modifica para produzir a geração seguinte. O Alchemy mantém uma população de hipóteses na forma de fórmulas ponderadas, modifica-as de várias maneiras a cada etapa e retém as variações que mais aumentam a probabilidade *a posteriori* dos dados (ou alguma outra função de pontuação). Se a população tiver uma única hipótese, essa abordagem será reduzida à escalada de montanha. A implementação open source atual do Alchemy não inclui o crossover, mas é fácil adicioná-lo. O algoritmo mestre dos evolucionários é a programação genética, que aplica o crossover e a mutação a programas de computador representados como árvores de sub-rotinas. Árvores de sub-rotinas podem ser representadas por conjuntos de regras lógicas, e a linguagem de programação Prolog faz exatamente isso. No Prolog, cada regra corresponde a uma sub-rotina, e seus antecedentes são as sub-rotinas que ela chama. Logo, podemos pensar no Alchemy com crossover como programação genética usando uma linguagem de programação como o Prolog, com a vantagem adicional de que as regras podem ser probabilísticas.

Os bayesianos acreditam que modelar a incerteza é a chave para o aprendizado e usam representações formais como as redes bayesianas e as redes de Markov para fazê-lo. Como já vimos, as redes de Markov são um tipo especial de MLN. As redes bayesianas também são facilmente representadas com o uso da equação mestre da MLN, com uma característica para cada estado possível de uma variável e seus pais, e o logaritmo da probabilidade condicional correspondente como seu peso. (A constante de normalização Z é então convenientemente reduzida a 1, o que significa que podemos ignorá-la.) O algoritmo mestre dos bayesianos é o teorema de Bayes, implementado com o uso de algoritmos de inferência probabilística como a propagação de crenças e o MCMC. Como você deve ter notado, o teorema de Bayes é um caso especial da equação mestre, com $P = P(A|B)$, $Z = P(B)$ e as características e pesos correspondentes a $P(A)$ e $P(B|A)$. O sistema Alchemy inclui tanto a propagação de crenças quanto o MCMC para a inferência, generalizados para manipular fórmulas lógicas ponderadas. Usando a inferência probabilística nos caminhos de comprovação fornecidos pela lógica, o Alchemy pesa as evidências a favor e contra uma conclusão e fornece a probabilidade de que esta ocorra. Isso é o contrário da lógica básica usada pelos simbolistas, que produz tudo ou nada e, portanto, falha quando recebe evidências contraditórias.

Os analogistas aprendem supondo que entidades com propriedades conhecidas

semelhantes têm as mesmas propriedades desconhecidas: pacientes com sintomas semelhantes têm diagnósticos similares, leitores que compraram os mesmos livros no passado farão a mesma coisa no futuro, e assim por diante. As MLNs podem representar a similaridade entre entidades com fórmulas como *Pessoas com os mesmos gostos compram os mesmos livros*. Logo, quanto maior o número de livros iguais que Alice e Bob compraram, maior a probabilidade de que eles tenham os mesmos gostos e (com a aplicação da mesma fórmula na direção oposta) maior a probabilidade de Alice comprar um livro se Bob também o fizer. Sua similaridade é representada pela probabilidade de terem os mesmos gostos. Para tornar essa abordagem útil, podemos ter diferentes pesos para instâncias distintas da mesma regra: se tanto Alice quanto Bob comprassem um livro raro específico, provavelmente isso seria mais informativo do que se ambos comprassem um best-seller e, portanto, deve ter peso mais alto. Nesse caso as propriedades cuja similaridade estamos calculando são discretas (comprar/não comprar), mas também podemos representar a similaridade entre propriedades contínuas, como a distância entre duas cidades, deixando que uma MLN tenha essas similaridades como características. Se a função de avaliação for uma função de pontuação que use margens em vez da probabilidade *a posteriori*, o resultado será uma generalização das SVMs, o algoritmo mestre dos analogistas. Um desafio maior para nosso aprendiz mestre seria reproduzir o mapeamento de estrutura, o mais poderoso tipo de analogia que faz inferências a partir de uma área (digamos, o sistema solar) para outra (o átomo). Podemos fazer isso aprendendo fórmulas que não referenciem nenhuma das relações específicas da área de origem. Por exemplo, *Amigos de fumantes também fumam* é sobre amizade e fumar, mas *Entidades relacionadas têm propriedades semelhantes* é aplicável a qualquer relação e propriedade. É possível aprender generalizando a partir de *Amigos de amigos também fumam*, *Colaboradores de especialistas também são especialistas* e outros padrões desse tipo encontrados em uma rede social e, então, aplicando talvez à web, com instâncias como *Páginas interessantes conduzem a páginas interessantes*, ou à biologia molecular, com instâncias como *Proteínas que interagem com proteínas reguladoras de genes também regulam genes*. Os pesquisadores de meu grupo e outros têm feito tudo isso e muito mais.

O Alchemy também permite o uso dos cinco tipos de aprendizado não supervisionado que vimos no capítulo anterior. Ele emprega o aprendizado

relacional, claro, e na verdade foi nessa área que a maioria de suas aplicações ocorreu até agora. O Alchemy usa a lógica para representar relações entre entidades e redes de Markov para permitir que elas sejam incertas. Podemos transformá-lo em um aprendiz por reforço anexando ao seu redor recompensas posteriores e usando-o no aprendizado do valor de cada estado da mesma forma que os aprendizes por reforço tradicionais usam, digamos, uma rede neural. Podemos executar chunking no Alchemy adicionando uma nova operação que condense cadeias de regras em regras individuais. (Por exemplo, *Se A então B* e *Se B então C* em *Se A então C*.) Uma MLN com uma única variável não observada conectada a todas as variáveis observáveis executa clustering. (Uma variável não observada é aquela cujos valores nunca vemos nos dados; é "oculta", por assim dizer, e só pode ser inferida.) MLNs com mais de uma variável não observada faz um tipo de redução de dimensionalidade discreta inferindo os valores dessas variáveis (em menor número) a partir das observáveis (mais numerosas). O Alchemy também pode manipular MLNs com variáveis não observadas contínuas, o que seria necessário para fazermos a análise de componentes principais e a execução do Isomap, por exemplo. Logo, em princípio o Alchemy faz tudo que queremos que Robby, o robô, faça, ou pelo menos tudo que discutimos neste livro. Na verdade, usamos o Alchemy para permitir que um robô aprendesse um mapa de seu ambiente, descobrindo a partir de seus sensores onde estão as paredes e portas, seus ângulos e distâncias, e assim por diante, o que é o primeiro passo para a construção de um robô doméstico completo.

Para concluir, podemos transformar o Alchemy em um meta-aprendiz como o de stacking, codificando os classificadores individuais como MLNs e adicionando ou aprendendo fórmulas para combiná-los. Foi isso que a Darpa fez em seu projeto PAL. O PAL, Personalized Assistant that Learns (Assistente Personalizável que Aprende), foi o maior projeto de inteligência artificial da história da Darpa e é o progenitor do Siri. O objetivo do projeto era a construção de uma secretária automatizada. Ele usava a lógica de Markov como sua representação abrangente, transformando as saídas de diferentes módulos nas decisões finais sobre o que fazer. Isso também permitia que os módulos do PAL aprendessem uns com os outros para chegar a um consenso.

Uma das aplicações mais amplas do Alchemy até hoje foi aprender uma rede semântica (ou grafo do conhecimento, como o Google a chama) a partir da web.

Uma rede semântica é um conjunto de conceitos (como planetas e estrelas) e das relações entre eles (planetas orbitam estrelas). O Alchemy aprendeu mais de um milhão desses padrões a partir de fatos extraídos da web (por exemplo, a Terra orbita o Sol). Ele descobriu conceitos como o de planeta por conta própria. A versão que usamos era mais avançada que a básica que descrevi aqui, mas as ideias principais são as mesmas. Vários grupos de pesquisa usaram o Alchemy ou suas próprias implementações de MLN para resolver problemas de processamento de linguagem natural, visão computacional, reconhecimento de atividades, análise de redes sociais, biologia molecular, e muitas outras áreas.

Apesar de seus sucessos, o Alchemy apresenta algumas deficiências significativas. Ele não está preparado para uma escala de volume de dados realmente grande (big data), e alguém sem um PhD em machine learning o achará difícil de usar. Devido a esses problemas, ainda não pode ter uma aplicação ampla. Mas examinemos o que pode ser feito para resolvê-los.

Machine learning em escala planetária

Na ciência da computação, um problema não está resolvido até ser resolvido eficientemente. Não adianta saber como fazer algo se não pudermos fazê-lo com o tempo e a memória disponíveis, e estes podem acabar com muita rapidez quando lidamos com uma MLN. É rotineiro aprendermos MLNs com milhões de variáveis e bilhões de características, mas essa escala não é tão grande quanto parece porque o número de variáveis cresce muito rapidamente com o número de entidades da MLN: em uma rede social com mil pessoas, já teremos um milhão de pares de amigos possíveis e um bilhão de instâncias da fórmula *Amigos de amigos são amigos*.

A inferência no Alchemy é uma combinação de inferência lógica e probabilística. A primeira é realizada pela comprovação de teoremas e a última pela propagação de crenças, pelo MCMC e por outros métodos que vimos no Capítulo 6. Combinamos as duas na comprovação probabilística de teoremas, e o algoritmo de inferência unificado, capaz de calcular a probabilidade de qualquer fórmula lógica, é uma parte-chave do sistema Alchemy atual. Porém, pode ser muito caro computacionalmente. Se nosso cérebro usasse a comprovação probabilística de teoremas e nos deparássemos com um tigre, ele nos devoraria antes de pensarmos em fugir. Esse é um preço alto a se pagar pela generalidade da lógica de Markov. Tendo evoluído no mundo real, nosso cérebro deve

codificar suposições adicionais que lhe permitam fazer inferências com muita eficiência. Nos últimos anos, começamos a descobrir quais seriam essas suposições e as codificamos no Alchemy.

O mundo não é uma confusão de interações aleatória; ele tem uma estrutura hierárquica: galáxias, planetas, continentes, países, cidades, vizinhanças, sua casa, você, sua cabeça, seu nariz, uma célula na ponta de seu nariz, as organelas que ela contém, moléculas, átomos, partículas subatômicas. Logo, a maneira de modelá-lo seria com uma MLN que também tivesse uma estrutura hierárquica. Esse é um exemplo da suposição que diz que o aprendiz e seu ambiente são semelhantes. A MLN não precisa saber *a priori* de quais partes o mundo é composto; tudo que o Alchemy precisa fazer é supor que o mundo *tem* partes e procurá-las, um pouco como uma prateleira recém-criada supõe que existam livros, sem saber ainda quais serão colocados nela. A estrutura hierárquica ajuda a tornar a inferência tratável porque as subpartes do mundo quase sempre interagem com outras subpartes da mesma parte: vizinhos conversam mais uns com os outros que com pessoas de outro país, moléculas produzidas em uma célula costumam reagir com outras moléculas dessa mesma célula, e assim por diante.

Outra propriedade do mundo que facilita o aprendizado e a inferência é que as entidades existentes nele não têm formas arbitrárias. Em vez disso, elas se enquadram em classes e subclasses, com membros da mesma classe sendo mais semelhantes que os de classes diferentes. Vivo ou inanimado, animal ou planta, pássaro ou mamífero, humano ou não: se soubermos todas as diferenças relevantes para a questão a ser resolvida, basta agrupar as entidades que não as contenham e economizaremos muito tempo. Como antes, a MLN não precisa saber *a priori* quais são as classes que existem no mundo; ela pode aprendê-las a partir de dados pelo clustering hierárquico.

O mundo é composto por partes, e partes pertencem a classes: combinar as duas fornece uma grande parcela do que precisamos para tornar as inferências tratáveis no Alchemy. Podemos aprender a MLN do mundo dividindo-o em partes e subpartes, de modo que a maioria das interações seja entre subpartes da mesma parte, e então agrupando as partes em classes e subclasses. Se o mundo fosse um brinquedo da Lego, poderíamos dividi-lo em tijolos individuais, lembrando quais tijolos se encaixam em quais outros tijolos, e agrupá-los por forma e cor. Se fosse a Wikipédia, extrairíamos as entidades sobre as quais ela

fala, as agruparíamos em classes e aprenderíamos como as classes se relacionam umas com as outras. Então, se alguém nos perguntasse “Arnold Schwarzenegger é uma estrela do cinema?”, poderíamos responder sim, porque ele é uma estrela e trabalha em filmes de ação. Aos poucos, poderíamos aprender MLNs cada vez maiores, até chegar a um ponto que um amigo no Google chama de “machine learning em escala planetária”: modelar todas as pessoas do mundo de uma só vez, com dados entrando e repostas saindo continuamente.

É claro que um aprendizado nessa escala requer muito mais que uma implementação direta dos algoritmos que vimos. Em primeiro lugar, além de um certo ponto um único processador não é suficiente; temos de distribuir o aprendizado por muitos servidores. Os pesquisadores tanto da indústria quanto das universidades têm investigado intensamente como, por exemplo, executar a descida de gradiente usando muitos computadores em paralelo. Uma opção é dividir os dados entre os processadores; outra é dividir os parâmetros do modelo. Após cada etapa, combinamos os resultados e redistribuímos o trabalho. Seja como for, não é fácil fazer isso sem deixar o custo da comunicação nos prejudicar ou a qualidade dos resultados ser afetada. Outro problema é que, se tivermos um fluxo interminável de dados chegando, não poderemos esperar para vê-lo em sua totalidade antes de tomar algumas decisões. Uma solução é usar o princípio da amostragem: se quisermos prever quem ganhará a próxima eleição presidencial, não precisamos perguntar a cada eleitor em quem ele votará; uma amostragem de alguns milhares será suficiente, se estivermos dispostos a aceitar um pouco de incerteza. O truque é generalizar isso para modelos complexos com milhões de parâmetros. Porém, podemos fazê-lo pegando a cada etapa a quantidade de exemplos que forem necessários para ficarmos bastante seguros de que estamos tomando a decisão certa e que a incerteza total ao longo de todas as decisões permanece dentro dos limites. Dessa forma poderemos aprender efetivamente a partir de dados em um tempo finito, como mencionei em um artigo mais antigo propondo essa abordagem.

Os sistemas de big data são as produções de Cecil B. DeMille do machine learning, com milhares de servidores em vez de milhares de coadjuvantes. Nos projetos maiores, reunir todos os dados, verificá-los, limpá-los e alterá-los para uma forma que os aprendizes possam assimilar faz a construção das pirâmides parecer um passeio no parque. Como objetivo final grandioso, o projeto europeu FuturICT visa construir um modelo – literalmente – do mundo todo. Sociedades,

governos, cultura, tecnologia, agricultura, doenças, a economia global: nada deve ser deixado de fora. Com certeza essa é uma iniciativa prematura, mas prenuncia o que está por vir. Enquanto isso, projetos como esse podem nos ajudar a descobrir quais são os limites da escalabilidade e como superá-los.

Complexidade computacional é uma coisa e complexidade humana é outra. Se os computadores são como sábios idiotas, os algoritmos de aprendizado podem às vezes parecer crianças prodígio propensas a acessos de raiva. Essa é uma das razões de os humanos que conseguem controlá-los ganharem tão bem. Se você souber como girar de maneira correta os botões de controle até eles chegarem ao ponto exato, algo mágico pode ocorrer na forma de um fluxo de insights avançados para a idade do aprendiz. E, semelhante ao que acontece no oráculo de Delfos, interpretar os pronunciamentos de um aprendiz pode requerer uma habilidade considerável. No entanto, gire os botões de maneira errada e ele pode emitir uma grande quantidade de informações sem sentido ou calar-se por pirraça. Infelizmente, nesse aspecto o Alchemy não é melhor que a maioria. Escrever o que sabemos quanto à lógica, fornecer os dados e pressionar o botão é a parte divertida. Quando o Alchemy retorna uma MLN eficiente e elegantemente precisa, uma ida ao pub é merecida. Quando ele não o faz – o que ocorre quase sempre –, a batalha começa. O problema está no conhecimento, no aprendizado ou na inferência? Por um lado, por causa do aprendizado e da inferência probabilística, uma MLN simples pode executar a tarefa de um programa complexo. Por outro, quando ele não funciona, é muito mais difícil depurar. A solução é torná-lo mais interativo, capaz de olhar para dentro e explicar seu raciocínio. Isso nos leva um passo mais perto do Algoritmo Mestre.

O doutor o verá agora

A cura do câncer é um programa que recebe o genoma da doença e fornece a droga que a matará. Já podemos imaginar como seria esse programa – vamos chamá-lo de CanceRx. Apesar de sua aparente simplicidade, o CanceRx é um dos maiores e mais complexos programas já construídos – na verdade, é tão grande e complexo que só poderia ser construído com a ajuda do machine learning. Ele é baseado em um modelo detalhado de como as células vivas funcionam, com uma subclasse para cada tipo de célula do corpo humano e um modelo abrangente de como elas interagem. Esse modelo, na forma de uma MLN ou algo semelhante a ela, combina conhecimento de biologia molecular

com grandes quantidades de dados provenientes de sequenciadores de DNA, microarranjos e muitas outras fontes. Parte do conhecimento foi codificada manualmente, mas a maioria foi extraída de maneia automática da literatura biomédica. O modelo está sempre evoluindo, incorporando os resultados de novos experimentos, fontes de dados e históricos de pacientes. No fim do processo, ele conhecerá todos os caminhos, mecanismos reguladores e reações químicas de cada tipo de célula humana – tudo o que se conhece de biologia molecular humana.

O CanceRx passa grande parte de seu tempo consultando o modelo com drogas candidatas. Dada uma nova droga, o modelo prevê seu efeito tanto sobre as células cancerosas quanto sobre as normais. Quando Alice foi diagnosticada com câncer, o CanceRx instanciou seu modelo com as células normais e com as do tumor e testou todas as drogas disponíveis até encontrar uma que matasse as células cancerosas sem danificar as saudáveis. Se ele não puder encontrar uma droga ou uma combinação de drogas que funcione, tentará projetar uma que o faça, talvez desenvolvendo-a a partir das existentes usando a escalada de montanha ou o crossover. A cada etapa da busca, ele testa as drogas candidatas no modelo. Se uma droga interromper o câncer, mas produzir algum efeito colateral danoso, o CanceRx tentará ajustá-la para eliminar o efeito colateral. Quando o câncer de Alice sofreu mutação, ele repetiu o processo inteiro. Mesmo antes de o câncer mudar, o modelo previu possíveis mutações, e o CanceRx prescreveu drogas que interromperiam sua evolução. No jogo de xadrez entre a humanidade e o câncer, o CanceRx significa xeque-mate.

É bom ressaltar que o machine learning não nos dará o CanceRx sozinho. Não é como se tivéssemos um vasto banco de dados de biologia molecular pronto para ser usado, o inseríssemos no Algoritmo Mestre e ele fornecesse o modelo perfeito de uma célula viva. O CanceRx seria o resultado final, após muitas iterações, de uma colaboração mundial entre centenas de milhares de biólogos, oncologistas e cientistas de dados. No entanto, o mais importante é que ele incorporaria dados de milhões de pacientes com câncer, com a ajuda de seus médicos e hospitais. Sem esses dados, não podemos curar o câncer; com eles, podemos. Contribuir com esse crescente banco de dados seria não só um benefício para cada paciente com câncer; trata-se de um dever ético. No universo do CanceRx, tentativas clínicas discretas são coisa do passado; novos tratamentos propostos pelo CanceRx estão continuamente sendo produzidos, e se

funcionarem, serão ministrados a um amplo grupo de pacientes. Tanto os sucessos quanto as falhas fornecem dados valiosos para o aprendizado do CanceRx, em um círculo virtuoso de melhoria. Se olharmos para isso de um ponto de vista, o machine learning é apenas uma pequena parte do projeto CanceRx, que fica bem atrás da coleta de dados e de contribuições humanas. Porém, se olharmos de outro, o machine learning é a chave de todo o esforço. Sem ele, só teríamos um conhecimento fragmentário da biologia do câncer, espalhado entre milhares de bancos de dados e milhões de artigos científicos, cada médico conhecendo apenas uma pequena parte. Reunir esse conhecimento em um todo coerente está além da habilidade dos humanos se não houver ajuda, não importa quanto eles sejam inteligentes; só o machine learning pode fazer isso. Já que cada caso de câncer é diferente, precisamos do machine learning para encontrar os padrões comuns. E já que um único tecido pode gerar bilhões de pontos de dados, precisamos do machine learning para descobrir o que fazer para cada novo paciente.

A tentativa de construir o que no fim das contas será o CanceRx já está em andamento. Pesquisadores da nova área de biologia de sistemas modelam redes metabólicas inteiras em vez de genes e proteínas individuais. Um grupo de Stanford construiu um modelo de uma célula inteira. A Aliança Global para Genômica e Saúde (Global Alliance for Genomics and Health) promove compartilhamento de dados entre pesquisadores e oncologistas, visando a análise de larga escala. A CancerCommons.org monta modelos de câncer e permite que os pacientes comparem suas histórias e aprendam com casos semelhantes. A Foundation Medicine detecta as mutações nas células cancerosas de um paciente e sugere as drogas mais apropriadas. Uma década atrás, não estava claro se, ou como, o câncer seria curado. Agora podemos ver como chegar lá. A estrada é longa, mas a encontramos.

---

1 N.T.: Goggles, em português, são óculos de proteção. O autor cria fantasiosamente óculos que substituem a função por seu logaritmo e então os chama de loggles.

capítulo 10

Como fica o mundo com o machine learning

Agora que você conheceu o mundo maravilhoso do machine learning, mudaremos de enfoque e veremos o que tudo isso significa para nós. Como a pílula vermelha do filme *Matrix*, o Algoritmo Mestre é o portal para uma realidade diferente: a realidade na qual você vive, mas que ainda não sabia que existia. De namorar a trabalhar, do autoconhecimento ao futuro da sociedade, do compartilhamento de dados à guerra, e dos perigos da inteligência artificial à próxima etapa da evolução, um novo mundo está tomando forma e o machine learning é a chave que o revela. Este capítulo o ajudará a aproveitá-lo ao máximo em sua vida e a estar pronto para o que vem a seguir. O machine learning não determinará o futuro sozinho, assim como nenhuma outra tecnologia; o que importa é o que resolvermos fazer com ele, e já temos as ferramentas para decidir.

A principal delas é o Algoritmo Mestre. O fato de surgir ou não agora e de se parecer ou não com o Alchemy é menos importante que o que ele traz: os recursos básicos de um algoritmo de aprendizado e onde eles nos levarão. Também poderíamos considerar o Algoritmo Mestre uma combinação de aprendizes atuais e futuros, para usarmos conforme seja conveniente em nossos testes de raciocínio em vez do algoritmo específico existente no produto X ou site Y, que provavelmente as respectivas empresas não compartilharão conosco. Vistos por esse prisma, os aprendizes com os quais interagimos diariamente são versões embrionárias do Algoritmo Mestre, e nossa missão é entendê-los e modelar seu crescimento de modo a servir melhor às nossas necessidades.

Nas próximas décadas, o machine learning afetará uma parcela muito grande

da vida humana para que possamos abordar o assunto em um único capítulo de um livro. Contudo, já podemos ver vários temas recorrentes e é neles que nos concentraremos, começando com o que os psicólogos chamam de teoria da mente – ou seja, a teoria do computador que opera em nossa mente.

Sexo, mentiras e machine learning

Nosso futuro digital começa com uma percepção: sempre que interagimos com um computador – seja o smartphone ou um servidor a milhares de quilômetros de distância – o fazemos em dois níveis. O primeiro é obter o que queremos: uma resposta, um produto para comprar, um novo cartão de crédito. O segundo nível, que em longo prazo é o mais importante, é ensinar ao computador quem somos. Quanto mais o ensinarmos, melhor ele poderá nos servir – ou nos manipular. A vida é um jogo entre nós e os aprendizes que nos rodeiam. Podemos nos recusar a jogar, mas então teremos de viver uma existência do século 20 no século 21. Ou podemos jogar para vencer. Qual modelo queremos que o computador tenha? E quais dados podemos dar a ele para que produza esse modelo? Devemos sempre manter essas duas perguntas em mente ao interagir com um algoritmo de aprendizado – como ocorre quando interagimos com outras pessoas. Alice sabe que Bob tem um modelo mental dela e tenta modelá-la por intermédio de seu comportamento. Se Bob for seu chefe, ela tentará parecer competente, leal e trabalhadora. Se em vez disso Bob for alguém que a atraia, ela será a mais sedutora possível. Seria quase inconcebível viver em sociedade sem essa habilidade de intuir e responder ao que as outras pessoas pensam. A novidade no mundo de hoje é que os computadores, não apenas as pessoas, estão começando a ter teorias da mente. Suas teorias ainda são primitivas, mas estão evoluindo com rapidez, e são o elemento com o qual temos de trabalhar para conseguir o que queremos – não menos que com outras pessoas. Portanto, *você* precisa de uma teoria da mente *do computador*, e é isso que o Algoritmo Mestre fornece, após a ativação da função de pontuação (quais achamos que são os objetivos do aprendiz, ou mais precisamente os objetivos de seu proprietário) e dos dados (o que achamos que ele sabe).

Veja o namoro online. Quando você usa o Match.com, o eHarmony ou o OkCupid (se necessário, esqueça por um momento que não acredita nessas iniciativas), seu objetivo é simples: encontrar a melhor parceira possível. Porém, o mais provável é que tenha muito trabalho e vários encontros frustrantes antes

de achar alguém de quem realmente goste. Um geek persistente extraiu vinte mil perfis do OkCupid, fez sua própria mineração de dados, encontrou a mulher de seus sonhos no octogésimo oitavo encontro e contou sua odisseia para a revista *Wired*. Para você ser bem-sucedido com menos trabalho, as duas principais ferramentas são seu perfil e as respostas às correspondências sugeridas. Uma opção popular é mentir (sobre sua idade, por exemplo). Isso pode parecer antiético, sem mencionar que pode se voltar contra você quando a candidata descobrir a verdade, mas há um truque. Usuárias experientes dos namoros online já sabem que as pessoas mentem sobre sua idade em seu perfil e agem de acordo, logo, se você informar sua idade real, estará dizendo a elas que é mais velho do que realmente é! Por sua vez, o aprendiz que faz as comparações intui que as pessoas preferem parceiros mais jovens do que elas realmente preferem. A próxima etapa lógica é as pessoas mentirem ainda mais sobre sua idade, o que acaba tornando esse atributo inútil.

Uma maneira melhor para quem estiver interessado é enaltecer atributos incomuns que tragam mais possibilidades de formar um casal, no sentido de atrair pessoas que o agradariam e que não agradariam a mais ninguém e, portanto, gerem menos competição. O que você deve fazer (e também sua possível parceira) é fornecer esses atributos. E o mecanismo de busca deve aprender a partir deles, da mesma forma que um casamenteiro de outra época o faria. Comparado com um casamenteiro de uma pequena vila, o algoritmo do site Match.com tem a vantagem de conhecer um número bem maior de pessoas, mas a desvantagem é que ele as conhece muito superficialmente. Um aprendiz simples, como o perceptron, se satisfaria com generalizações como “os homens preferem as loiras”. Um mais sofisticado usará padrões como “pessoas com os mesmos gostos musicais incomuns costumam compor bons casais”. Se tanto Alice quanto Bob gostarem de Beyoncé, só esse atributo não os unirá. Porém, se ambos gostarem de Bishop Allen, isso os tornará pelo menos um pouco mais propensos a serem almas gêmeas. Se os dois forem fãs de uma banda que o aprendiz não conhece, isso será ainda melhor, mas só um algoritmo relacional como o Alchemy conseguirá detectar. Quanto melhor o aprendiz, mais vale a pena ensiná-lo quem você é. Porém, como regra geral, é preciso que você se diferencie o suficiente para que ele não o confunda com a “pessoa média” (lembra-se de Bob Burns do Capítulo 8?), mas sem ser tão incomum a ponto de não ser encontrado.

O namoro online é na verdade um exemplo não muito adequado porque é difícil prever a química existente entre um casal. Duas pessoas que tiverem empatia imediata em um encontro podem acabar se apaixonando e acreditando veementemente que foram feitas uma para a outra, mas se sua conversa inicial tomar um rumo diferente, podem achar a outra maçante e nunca mais querer se encontrar. Um aprendiz sofisticado executaria muitas simulações de Monte Carlo de um encontro entre cada par de possíveis casais e os classificaria pela fração de encontros que deram certo. Quando não possuem esse recurso, os sites de encontros organizam festas e convidam pessoas que possam dar certo com outros convidados, deixando que façam em algumas horas o que de outra forma levaria semanas.

Para aqueles pouco afeitos ao namoro online, uma noção mais útil seria a seleção de quais interações devem ser registradas e em qual local. Se não quiser que suas compras de Natal deixem a Amazon confusa sobre seus gostos, faça-as em outro site. (Desculpe, Amazon.) Se costuma assistir a tipos de vídeos diferentes em casa e no trabalho, abra duas contas no YouTube, uma para cada tipo, e o site aprenderá a fazer as recomendações correspondentes. E se for assistir a vídeos de um tipo no qual normalmente não tem interesse, faça logout antes. Use o modo incógnito do Chrome não para uma navegação pela qual se sinta culpado (o que, claro, você nunca faria), mas quando não quiser que a sessão atual influencie uma futura personalização. Na Netflix, adicionar perfis para as diferentes pessoas que usam sua conta o poupará de recomendações de classificação restrita para uma noite de filmes com a família. Se não gostar de uma empresa, clique em seus anúncios: além de desperdiçar seu dinheiro nesse momento, isso ensinará o Google a desperdiçá-lo de novo no futuro exibindo os anúncios para pessoas que não comprarão os produtos. E se tiver consultas muito específicas que quiser que o Google responda corretamente no futuro, procure os links relevantes nas páginas de resultados posteriores e clique neles. Se um sistema em geral continuar recomendando as coisas erradas, tente ensiná-lo encontrando e clicando em opções corretas e volte depois para ver se ele aprendeu.

Todavia, essas intervenções podem dar trabalho. Infelizmente, o que todas essas ações ilustram é como a comunicação entre o usuário e o aprendiz é precária hoje. Deveria ser possível dizer a ele o máximo que quiséssemos sobre nossa pessoa, em vez de deixá-lo aprender de maneira indireta a partir do que

fazemos. Mais que isso, seria preciso inspecionar o modelo que o aprendiz tem de nós e corrigi-lo conforme desejado. O aprendiz teria a opção de nos ignorar, se achasse que estamos mentindo ou não nos conhecemos bem, mas pelo menos poderia levar nossa opinião em conta. Para que esse esquema funcione, o modelo precisa estar em um formato que os humanos consigam entender, como um conjunto de regras e não uma rede neural, e aceitar declarações gerais como entrada além de dados brutos, como faz o Alchemy. Todos esses pontos nos levam à ponderação de quanto o modelo do aprendiz se assemelharia a nós e o que gostaríamos de fazer com ele.

O espelho digital

Considere por um momento os dados sobre você que estão registrados nos computadores do mundo todo: seus emails, documentos do Office, textos, tuítes e contas do Facebook e LinkedIn; suas buscas na web, cliques, downloads e compras; seus registros de crédito, impostos, telefone e saúde; suas estatísticas no Fitbit; seus dados de direção conforme registrados pelos microprocessadores de seu carro; suas andanças conforme registradas pelo seu celular; todas as fotos que já tirou; aparições breves em câmeras de segurança; seus snippets do Google Glass – e assim por diante. Se um futuro biógrafo não tivesse acesso a nada a não ser a esse "exaustivo conjunto de dados" seus, que imagem de você ele formaria? Provavelmente uma imagem bem precisa e detalhada em muitos aspectos, mas sem alguns pontos essenciais. Por que em determinado momento você decidiu mudar de carreira? O biógrafo teria previsto isso? E quanto àquela pessoa que você encontrou um dia e secretamente nunca esqueceu? O biógrafo poderia retroceder o registro encontrado e dizer "Ah, aí está"?

O que acontece na realidade (e que talvez seja tranquilizador) é que nenhum aprendiz existente no mundo atualmente tem acesso a todos esses dados (nem mesmo a NSA), e mesmo se tivesse não saberia como transformá-los em uma imagem real nossa. Porém, suponhamos que você pegasse todos os seus dados e os passasse para o – real e futuro – Algoritmo Mestre, já contendo tudo que poderíamos ensinar sobre a vida humana. Ele aprenderia um modelo seu, e você poderia carregar esse modelo em um pen drive no bolso, inspecioná-lo quando quisesse e usá-lo para tudo que lhe agradasse. Certamente seria uma ferramenta de introspecção maravilhosa, como se você se olhasse no espelho, mas constituiria um espelho digital que exibiria não só sua aparência como também

todas as coisas observáveis sobre a sua pessoa – um espelho que poderia ganhar vida e conversar com você. O que perguntar a ele? Você pode não gostar de algumas respostas, contudo essa seria mais uma razão para ponderá-las. E algumas lhe dariam novas ideias, novas direções. O modelo que o Algoritmo Mestre construir de você o ajudará até mesmo a se tornar uma pessoa melhor.

Deixando o autoaprimoramento de lado, provavelmente a primeira coisa que você vai querer que seu modelo faça é negociar tudo que há no mundo em seu nome: solte-o no ciberespaço, procurando tudo que lhe interessar. De todos os livros do mundo, ele vai sugerir alguns que talvez você queira ler em seguida, com uma exatidão com a qual a Amazon não pode nem mesmo sonhar. O mesmo podemos dizer de filmes, música, games, roupas, equipamentos eletrônicos – o que você quiser. Ele manteria sua geladeira sempre abastecida. Filtraria seus emails, correio de voz, notícias do Facebook e novidades do Twitter e, quando apropriado, responderia em seu nome. Cuidaria de todos os pequenos aborrecimentos da vida moderna para você, como verificar contas de cartão de crédito, reclamar de cobranças indevidas, fazer negociações, renovar inscrições e preencher declarações de renda. Encontraria um remédio para sua indisposição, conversaria com seu médico e o compraria na Walgreens. Chamaria sua atenção para propostas de emprego interessantes, indicaria locais para férias, sugeriria candidatos em eleições e procuraria possíveis parceiras para um encontro. Após encontrar a parceira, ele associaria seu modelo com o dela para selecionar restaurantes que agradassem a ambos. E é aí que as coisas começam a ficar *realmente* interessantes.

Sociedade de modelos

Nesse futuro que se aproxima rapidamente, você não será o único a ter uma "metade digital" seguindo suas ordens vinte e quatro horas por dia. As outras pessoas também terão um modelo detalhado de si mesmas, e esses modelos conversarão uns com os outros o tempo todo. Se você estiver procurando um emprego e a empresa X estiver contratando, o modelo deles entrevistará o seu. Será muito parecido com uma entrevista real com pessoas de carne e osso – seu modelo terá sido aconselhado a não fornecer informações negativas sobre você, e assim por diante –, mas levará apenas uma fração de segundo. Você clicará em "Encontrar emprego" em sua futura conta do LinkedIn e será candidato a todos os empregos do mundo que estiverem em conformidade com seus parâmetros

(profissão, localização, salário etc.). O LinkedIn responderá com prontidão com uma lista ordenada pelas melhores oportunidades, e dentre elas você selecionará a primeira empresa com a qual deseja conversar. O mesmo ocorrerá com encontros: seu modelo conversará com milhões de parceiras para poupá-lo disso, mas no sábado você encontrará com as que lhe agradaram mais em uma festa organizada pelo OkCupid, sabendo que também é um dos escolhidos *delas* – e sabendo, claro, que *seus* outros escolhidos também estarão presentes. Tenho certeza de que será uma noite interessante.

No mundo do Algoritmo Mestre, "meu pessoal entrará em contato com o seu" se tornará "meu programa chamará o seu". Todos terão uma equipe de robôs para facilitar sua vida. Negócios serão fechados, contratos negociados, acordos feitos, tudo sem movermos um só dedo. Hoje, empresas farmacêuticas conversam com os médicos, porque eles decidem quais medicamentos nos prescreverão. No futuro, os fornecedores dos produtos e serviços que usarmos, ou que viermos a usar, conversarão com nossos modelos, porque eles farão a triagem automaticamente. A meta do robô deles será fazer nosso robô comprar. A meta de nosso robô será ponderar suas reivindicações, como fazemos com os anúncios na TV, mas com um nível de detalhes muito mais preciso, para o qual nunca teríamos paciência ou tempo. Antes de você comprar um carro, seu eu digital examinará cada uma de suas especificações, as discutirá com o fabricante e pesquisará tudo o que já foi dito sobre o carro e suas alternativas. Sua metade digital será como uma direção hidráulica para sua vida: ela irá onde você quiser ir, poupando-o de fazer o esforço. Isso não significa que você acabará confinado em uma "bolha filtro", vendo apenas aquilo de que realmente goste, sem espaço para o inesperado; seu eu digital não agirá assim. Parte de suas instruções incluirá deixar algumas portas abertas ao acaso, para expô-lo a novas experiências e para a descoberta de boas surpresas.

O mais interessante é que o processo não termina ao encontrarmos um carro, uma casa, um médico, uma parceira ou um emprego. Nossa metade digital estará sempre aprendendo a partir de suas experiências, como ocorreria conosco. Ela descobrirá o que funciona ou não, seja em entrevistas de emprego, em encontros ou na busca de um imóvel. Aprenderá sobre as pessoas e empresas com as quais interagir em nosso nome e também (o mais importante) a partir de nossas interações com elas no mundo real. Ela previu que Alice seria uma ótima parceira, mas você teve momentos embaraçosos, logo sua metade digital tentará

descobrir possíveis razões para isso, que testará em sua próxima tentativa de encontrar uma companheira. Ela compartilhará suas descobertas mais importantes com você. (“Você acha que gosta de X, mas na verdade tende a ficar com Y”.) Comparando a experiência que você teve em hotéis com as críticas do TripAdvisor, descobrirá quais são as vantagens reais e as procurará no futuro. Aprenderá não só que fornecedores online são mais confiáveis, mas também como decodificar o que os menos confiáveis dizem. Sua metade digital terá um modelo do mundo: não só do mundo em geral, mas do mundo em relação a você. É claro que, ao mesmo tempo, todas as outras pessoas também terão um modelo de seus mundos em contínua evolução. Cada participante de uma interação aprenderá a partir dela e aplicará o que aprendeu em suas novas interações. Você terá seu modelo de cada pessoa e empresa com a qual interagir, e elas terão modelos seus. À medida que os modelos se aperfeiçoarem, suas interações se tornarão cada vez mais parecidas com as que você teria no mundo real – exceto por serem milhões de vezes mais rápidas e *in silicon*. O ciberespaço de amanhã será um vasto mundo paralelo que selecionará apenas as coisas mais promissoras para testar no mundo real. Será como uma nova subconsciência global, a identidade coletiva da raça humana.

Compartilhar ou não compartilhar, como e onde

Obviamente, aprendermos tudo sobre o mundo por nossa própria conta será um processo lento, mesmo se nossa metade digital o fizer em ordens de magnitude mais rápidas que as de nosso eu de carne e osso. Se outras pessoas aprenderem sobre nós mais rápido que aprendermos sobre elas, teremos problemas. A resposta é compartilhar: um grupo de um milhão de pessoas aprenderá sobre uma empresa ou produto com muito mais rapidez que alguém sozinho, se compartilhar suas experiências. Porém, com quem devemos compartilhar dados? Talvez essa seja a pergunta mais importante do século 21.

Atualmente nossos dados podem ser de quatro tipos: dados que compartilhamos com todas as pessoas, dados que compartilhamos com amigos ou colaboradores, dados que compartilhamos com várias empresas (querendo ou não) e dados que não compartilhamos. O primeiro tipo inclui coisas como críticas do Yelp, da Amazon e do TripAdvisor, pontuações de feedback do eBay, currículos do LinkedIn, blogs, tuítes, e assim por diante. Esses dados têm muito valor e são os menos problemáticos dos quatro. Nós os tornamos disponíveis

para todas as pessoas porque queremos, e todos se beneficiam. O único problema é que as empresas que hospedam os dados nem sempre permitem que eles sejam baixados em massa para a construção de modelos. Deveriam permitir. Hoje podemos acessar o TripAdvisor e ver as críticas e as classificações em estrelas de hotéis específicos em que estamos interessados, mas e quanto a um modelo do que em geral torna um hotel bom ou ruim, que poderíamos usar para classificar hotéis que no momento tivessem poucas críticas ou não apresentassem opiniões confiáveis? O TripAdvisor poderia aprender um modelo assim, mas e um modelo do que fizesse *você* achar que um hotel é bom ou ruim? Isso demandaria informações suas que talvez você não queira compartilhar com o TripAdvisor. A solução seria uma fonte confiável que combinasse os dois tipos de dados e lhe fornecesse os resultados.

O segundo tipo de dado também não deve ser problemático, mas não o é porque se sobrepõe ao terceiro. Compartilhamos atualizações e fotos com nossos amigos no Facebook, e eles compartilham conosco. Porém, todas as pessoas também compartilham suas atualizações e fotos com o Facebook. Bom para ele, que tem um bilhão de amigos. Dia após dia, o Facebook aprende muito mais sobre o mundo que qualquer pessoa. Aprenderia ainda mais se tivesse algoritmos melhores, e eles estão melhorando diariamente, graças a nós, cientistas de dados. O Facebook usa todo esse conhecimento principalmente para destinar propaganda para nós. Em troca, fornece a infraestrutura para nossos compartilhamentos. Esse é o intercâmbio que fazemos ao usar o Facebook. À medida que seus algoritmos de aprendizado melhoram, ele obtém cada vez mais valor dos dados, e parte desse valor retorna para nós na forma de anúncios mais relevantes e um serviço melhor. O único problema é que o Facebook também pode fazer coisas com os dados e os modelos que não são de nosso interesse, e não temos como evitar.

Esse problema chega a um nível extremo com dados que compartilhamos com empresas, que nos dias de hoje incluem quase tudo que fazemos online assim como muito do que fazemos offline. Caso não tenha notado, há uma corrida insana para a coleta de dados sobre você. Todos adoram os seus dados, e não é de surpreender: eles são a porta para o seu mundo, o seu dinheiro, o seu voto e até o seu coração. Contudo, eles só obtêm uma parte. O Google vê suas buscas, a Amazon suas compras online, a AT&T seus telefonemas, a Apple seus downloads de música, a Safeway seus mantimentos, a Capital One suas

transações com cartão de crédito. Empresas como a Acxiom recolhem e vendem informações nossas, mas se dermos uma olhada (o que no caso da Acxiom é possível fazer, em *aboutthedata.com*), não é muita coisa, e alguns dados estão errados. Ninguém tem nada que se aproxime de um retrato completo nosso. Isso é bom e ruim. Bom porque se alguém tivesse, essa pessoa ou empresa teria muito poder. Ruim porque se fosse o caso não haveria como criá-lo. O que queremos é um eu digital cujo único proprietário sejamos nós e que as outras pessoas só possam acessar de acordo com nossa vontade.

O último tipo de dado – aquele não compartilhado – também apresenta um problema, que é o de que talvez devêssemos compartilhá-lo. Pode não nos ter ocorrido fazê-lo, pode não haver uma maneira fácil, ou talvez simplesmente não tenhamos interesse em compartilhar. No último caso, deveríamos considerar se temos a responsabilidade ética de compartilhar. Um exemplo já visto é o de pacientes com câncer, que poderiam contribuir com a cura da doença compartilhando o genoma e o histórico de tratamento de seus tumores. Porém, há muito mais. Todos os tipos de perguntas sobre a sociedade e políticas podem ser respondidos pelo aprendizado a partir dos dados que geramos em nossa vida diária. As ciências sociais estão entrando em uma era de ouro, em que finalmente elas têm dados proporcionais à complexidade dos fenômenos que estudam, e os benefícios para todos nós podem ser enormes – contanto que os dados possam ser acessados por pesquisadores, legisladores e cidadãos. Isso não significa deixar outras pessoas bisbilhotarem nossa vida privada; significa deixá-las ver os modelos aprendidos, que devem conter apenas informações estatísticas. Logo, entre nós e eles é preciso que haja um data broker honesto que garanta que nossos dados não serão usados incorretamente, mas também que portadores de livre trânsito não compartilhem os benefícios sem compartilhar os dados.

Resumindo, todos os quatro tipos de compartilhamento de dados apresentam problemas. Esses problemas têm a mesma solução: um novo tipo de empresa que seja para nossos dados o que o banco é para nosso dinheiro. Os bancos não roubam nosso dinheiro (com raras exceções). Sua missão é investi-lo com prudência, e nossos depósitos são protegidos pelo governo. Atualmente muitas empresas se oferecem para consolidar nossos dados em algum local na nuvem, mas elas ainda estão longe de se parecer com um banco de dados pessoais. Quando são provedores de nuvem, tentam nos confinar – um grande erro.

(Imagine depositar seu dinheiro no Bank of America e não saber se poderá transferi-lo para a Wells Fargo em algum momento futuro.) Algumas startups se oferecem para acumular nossos dados e então distribuir para anunciantes em troca de descontos, mas acho que não é isso que queremos. Podemos querer dar informações para os anunciantes gratuitamente por ser de nosso interesse, podemos não querer dar informações de forma alguma, e o que compartilhar em qual momento é um problema que só um bom modelo nosso pode resolver.

O tipo de empresa em que estou pensando faria várias coisas em troca de uma taxa de inscrição. Ela manteria anônimas nossas interações online, roteando-as por seus servidores e agregando-as às de seus outros usuários. Armazenaria todos os dados relacionados à nossa vida em um único local – em nosso streaming de vídeo 24/7[1] do Google Glass, se tivermos um. Aprenderia um modelo completo sobre nós e o nosso mundo e o atualizaria continuamente. E usaria o modelo em nosso nome, fazendo sempre o que faríamos, da melhor forma possível. O compromisso básico da empresa conosco seria o de que nossos dados e nosso modelo nunca serão usados contra nossos interesses. Uma garantia assim é difícil de ser mantida – afinal, nós mesmos não temos como garantir que nunca faremos nada contra nossos interesses. Porém, a existência da empresa dependeria disso assim como um banco depende da garantia de que não perderá nosso dinheiro, logo teríamos de poder confiar nela como confiamos no banco.

Uma empresa como essa poderia se tornar rapidamente uma das mais valiosas do mundo. Como Alexis Madrigal da revista *Atlantic* aponta: hoje nosso perfil pode ser comprado por meio cent ou menos, mas o valor de um usuário para a indústria de propaganda da internet estaria mais para 1.200 dólares por ano. A parcela de nossos dados que o Google possui deve valer cerca de 20 dólares, a do Facebook 5 dólares, e assim por diante. Adicione a isso todas as parcelas que ninguém tem ainda e o fato de que o todo vale mais que a soma das partes – um modelo nosso baseado em todos os nossos dados é muito melhor que mil modelos baseados em mil parcelas –, e estamos nos referindo a mais de um trilhão de dólares por ano para uma economia do tamanho da dos Estados Unidos. Não é preciso ter uma grande parte desse quinhão para uma empresa estar entre as 500 principais da Fortune. Se decidir aceitar o desafio e acabar se tornando um bilionário, lembre-se de onde tirou a ideia.

É claro que algumas empresas existentes adorariam hospedar nosso eu digital.

O Google, por exemplo. Sergey Brin diz “queremos que o Google seja a terceira metade de seu cérebro”, e algumas aquisições da empresa podem estar relacionadas a quanto seus fluxos de dados de usuários complementam os seus próprios dados. Porém, apesar de terem uma vantagem inicial, empresas como o Google e o Facebook não podem ser consideradas nossa casa digital porque têm um conflito de interesses. Elas ganham seu sustento com a destinação de anúncios e, portanto, têm de equilibrar seus interesses com os dos anunciantes. Você não deixaria que a primeira ou a segunda metade de seu cérebro compartilhasse sua lealdade, então por que deixaria que a terceira o fizesse?

Algo problemático seria o governo citar nossos dados judicialmente e até mesmo nos submeter à prisão preventiva, como no filme *Minority Report*, se o modelo for semelhante ao de um criminoso. Para evitar isso, a empresa que armazena os dados pode mantê-los criptografados, e a chave ficaria conosco. (Hoje é possível executar processamentos com dados criptografados sem ser preciso descriptografá-los.) Ou poderíamos mantê-los em um disco rígido em casa, e a empresa apenas nos alugaria o software.

Se não gosta da ideia de uma entidade com fins lucrativos conter as chaves que revelam o seu mundo, associe-se a um sindicato de dados. (Se ainda não há um em sua parcela da selva cibernética, talvez você possa fundá-lo.) O século 20 precisava de sindicatos para balancear o poder de patrões e trabalhadores. O século 21 precisa de sindicatos de dados por uma razão semelhante. As corporações têm uma capacidade muito maior de coletar e usar dados do que as pessoas. Isso leva a uma assimetria no poder, e quanto mais valiosos os dados – quanto melhores e mais úteis os modelos que puderem ser aprendidos a partir deles –, maior a assimetria. Um sindicato de dados permite que seus membros negociem em igualdade com as empresas sobre o uso de seus dados. Talvez os sindicatos de trabalhadores possam assumir a tarefa e reforçar seus associados, iniciando sindicatos de dados para seus membros. Porém, eles estão organizados por ocupação e localização; os sindicatos de dados podem ser mais flexíveis. Associe-se a pessoas com as quais tenha muitas coisas em comum; os modelos aprendidos serão mais úteis para você dessa forma. É bom ressaltar que pertencer a um sindicato de dados não significa deixar que outros membros vejam os seus dados; significa apenas deixar que os modelos aprendidos a partir do pool de dados sejam usados por todos. Os sindicatos de dados também podem ser um veículo para você dizer aos políticos o que deseja. Seus dados podem

influenciar o mundo tanto quanto seu voto – ou mais – porque só vamos ao local de votação no dia da eleição. Em todos os outros dias, seus dados são seu voto. Manifeste-se!

Até agora não mencionei a palavra *privacidade*. Não foi por acaso. A privacidade é apenas um aspecto da questão maior do compartilhamento de dados, e se nos concentrarmos nela em detrimento do todo, como grande parte dos debates têm feito, corremos o risco de chegar a conclusões erradas. Por exemplo, leis que proíbem o uso de dados para um fim que não seja o originalmente pretendido são muito limitadas. (Nem mesmo um único capítulo de *Freakonomics* poderia ter sido escrito sob o controle de uma lei assim.) Quando as pessoas têm de conciliar privacidade e outros benefícios, como ao criar um perfil em um site, o valor implícito da privacidade acaba sendo menor do que se fizéssemos perguntas abstratas para elas como "Vocês se importam com sua privacidade?". Porém, debates sobre privacidade geralmente são guiados por essa última abordagem. O Tribunal de Justiça da União Europeia decretou que as pessoas têm o direito de ser esquecidas, mas também têm o direito de se lembrar, seja com seus neurônios ou um disco rígido. O mesmo ocorre com as empresas, e até certo ponto os interesses dos usuários, coletores de dados e anunciantes estão alinhados. Uma atenção desperdiçada não beneficia ninguém, e melhores dados geram melhores produtos. A privacidade não é um jogo de soma zero, ainda que com frequência seja tratada dessa forma.

As empresas de hospedagem do eu digital e os sindicatos de dados são o que considero um futuro maduro para os dados na sociedade. Se chegaremos lá é uma questão em aberto. Atualmente, a maioria das pessoas não sabe quantos dados sobre elas estão sendo coletados nem quais são os possíveis custos e benefícios. As empresas parecem satisfeitas em continuar a agir em segredo, com medo de uma admoestação. Porém, cedo ou tarde, a censura ocorrerá, e na balbúrdia resultante serão promulgadas leis draconianas que no fim das contas não beneficiarão ninguém. Seria melhor promover uma conscientização agora e nos deixar fazer nossas escolhas individuais sobre o que compartilhar, o que não compartilhar, como e onde.

Uma rede neural roubou meu emprego

Quanto de seu cérebro seu trabalho usa? Quanto mais ele usar, mais seguro você estará. Nos primórdios da inteligência artificial, a percepção comum era a de que

os computadores substituiriam os operários antes dos executivos, porque o trabalho destes requer mais esforço intelectual. Porém, não foi bem assim. Os robôs montam carros, mas não substituíram operários da construção civil. Por outro lado, os algoritmos de machine learning substituíram os analistas de crédito e os divulgadores de marketing direto. Como ficou comprovado, é mais fácil para as máquinas fazer a avaliação de pedidos de crédito que andar pelo local de uma construção sem tropeçar, ainda que para os humanos seja o contrário. A alegação comum é a de que tarefas definidas de maneira minuciosa são facilmente aprendidas a partir de dados, enquanto tarefas que demandem uma ampla combinação de habilidades e conhecimentos não o são. Grande parte de nosso cérebro é dedicada à visão e ao movimento, o que é um sinal de que caminhar é muito mais complexo do que parece; não nos preocupamos com isso porque, tendo sido aprimorada pela evolução ao nível da perfeição, essa habilidade ocorre de forma quase subconsciente. A empresa Narrative Science tem um sistema de inteligência artificial que pode escrever resumos muito bons de jogos de beisebol, mas não romances, e – que me desculpe George F. Will – a vida é muito mais que apenas jogos de beisebol. O reconhecimento de voz é complexo para os computadores porque é difícil preencher as lacunas – literalmente, os sons que quem fala costuma omitir – quando não temos ideia do assunto sobre o qual a pessoa está falando. Os algoritmos podem prever flutuações de ações, mas não sabem como elas estão relacionadas à política. Quanto maior for o contexto que um trabalho demandar, menos chances haverá de que um computador possa executá-lo em breve. O bom senso é importante não só porque nossa mãe nos ensinou, mas porque os computadores não o têm.

A melhor maneira de não perder seu emprego é você mesmo automatizá-lo. Assim terá tempo para se dedicar a todas as partes dele para as quais antes não tinha tempo e que tão cedo um computador não poderá executar. (Se não houver nenhuma, antecipe-se e procure um novo emprego.) Se um computador aprender a fazer seu trabalho, não dispute; beneficie-se dele. A H&R Block ainda está em atividade, mas empregos de cálculo de impostos são muito menos fatigantes que costumavam ser, agora que os computadores fazem a parte tediosa do trabalho. (Certo, talvez esse não seja o melhor exemplo, dado que o crescimento exponencial de códigos de imposto é um dos poucos que pode se comparar ao do poder de computação.) Pense no big data como uma extensão de seus sentidos e os algoritmos de aprendizado como uma extensão de seu cérebro. Atualmente os

melhores jogadores de xadrez são chamados de centauros, metade homem e metade programa. O mesmo ocorre em muitas outras ocupações, do analista de ações aos olheiros do beisebol. Não é homem contra máquina; é homem com máquina *versus* homem sem ela. Os dados e a intuição são como o cavalo e o cavaleiro, e não tentamos vencer um cavalo; nós o cavalgamos.

À medida que a tecnologia avança, uma combinação cada vez mais íntima de humano e máquina toma forma. Você está com fome; o Yelp sugere bons restaurantes. Você seleciona um; o GPS lhe dá a direção. Você dirige; a parte eletrônica do carro se encarrega do controle de baixo nível. Já somos todos ciborgues. A história real da automação não está relacionada ao que ela substitui, mas sim ao que permite. Algumas profissões desaparecem, mas muitas outras surgem. Acima de tudo, a automação torna possível qualquer coisa que seria cara demais se feita por humanos. Os caixas eletrônicos substituíram alguns atendentes bancários, mas principalmente nos permitem sacar dinheiro a qualquer momento e em qualquer local. Não haveria *Toy Story* e videogames se animadores humanos tivessem de colorir um pixel de cada vez.

Mesmo assim, poderíamos perguntar se os empregos para humanos deixarão de existir. Acho que não. Mesmo se chegar um dia – e não será logo – em que os computadores e robôs consigam fazer tudo melhor, ainda haverá empregos para pelo menos alguns de nós. Um robô pode conseguir personificar um garçom perfeitamente, até ao nível da conversa fiada, mas talvez os clientes continuem preferindo um garçom que saibam que é humano, só por ele o sê-lo. Restaurantes com garçons humanos terão uma marca característica, assim como hoje o têm as mercadorias artesanais. As pessoas continuam indo ao teatro, andando a cavalo e velejando, ainda que existam filmes, carros e lanchas. O mais importante é que alguns profissionais são realmente insubstituíveis porque seu trabalho requer a única coisa que os computadores e robôs por definição não podem ter: a experiência humana. Com esse termo não me refiro a empregos que demandem empatia, já que não é difícil simulá-la; veja o sucesso dos robo-pets. Estou me referindo às humanidades, cujo domínio envolve tudo que não conseguimos entender sem a experiência de ser humano. Preocupamo-nos com o fato das humanidades estarem em uma espiral descendente, mas elas se erguerão das cinzas quando outros profissionais forem automatizados. À medida que cada vez mais coisas forem feitas de forma econômica pelas máquinas, mais valiosa será a contribuição dos humanistas.

Por outro lado, infelizmente as expectativas de longo prazo dos cientistas não são as mais favoráveis. No futuro, talvez os únicos cientistas restantes sejam cientistas computadores, ou seja, computadores fazendo ciência. As pessoas antes conhecidas como cientistas (como eu) dedicarão suas vidas à compreensão dos avanços científicos realizados pelos computadores. Elas não ficarão menos felizes do que antes; afinal, sempre consideraram a ciência como um hobby. E uma ocupação muito importante para os de inclinação técnica continuará existindo: ficar de olho nos computadores. Na verdade, isso demandará mais do que engenheiros: pode acabar sendo a ocupação full-time de toda a humanidade para descobrirmos o que queremos das máquinas e nos certificarmos se é o que estamos obtendo – veremos mais sobre esse assunto posteriormente neste capítulo.

Enquanto isso, à medida que a fronteira entre empregos automatizáveis e não automatizáveis avançar no cenário econômico, o que veremos provavelmente será um crescente desemprego, a diminuição do salário de cada vez mais profissões e o aumento das recompensas para uma quantidade cada vez menor de ocupações que ainda não puderem ser automatizadas. É claro que esse movimento já está em curso, porém vai aumentar muito. A transição será tumultuosa, mas graças à democracia, terá um final feliz. (Cuidado com seu voto – ele pode ser a coisa mais importante que você possui.) Quando a taxa de desemprego subir acima dos 50%, ou mesmo antes, as atitudes quanto à redistribuição mudarão radicalmente. A maioria recém-desempregada votará em quem oferecer generosos benefícios de desemprego vitalícios com as elevadas taxas de impostos necessárias para patrociná-los. Isso não quebrará os bancos porque as máquinas gerarão a produção necessária. Começaremos a falar sobre taxa de emprego e não de desemprego e sua redução será vista como um sinal de progresso. (“O Brasil está ficando para trás. Nossa taxa de emprego ainda é de 23%”.) Os benefícios pagos para o desempregado serão substituídos por uma renda básica para todos. Aqueles que não estiverem satisfeitos com ela poderão ganhar mais, muito mais, nas poucas ocupações humanas restantes. Liberais e conservadores continuarão brigando pela taxa de impostos, mas as metas terão mudado permanentemente. Com o valor total do trabalho tendo diminuído de maneira dramática, as nações mais ricas serão as que tiverem a proporção mais alta de recursos naturais para a população. (Mude-se para o Canadá já.) Para aqueles que não estiverem trabalhando, a vida não será sem sentido, não mais do

que a vida em uma ilha tropical em que a generosidade da natureza atende todas as necessidades. Uma economia de oferta (gift economy) se desenvolverá, da qual o movimento do software open source é um exemplo. As pessoas procurarão significado nos relacionamentos humanos, na autorrealização e na espiritualidade, de forma muito semelhante a como fazem agora. A necessidade de ganhar a vida será uma memória distante, outro fragmento do passado bárbaro da humanidade já superado.

A guerra não é para humanos

O confronto armado é mais difícil de automatizar que a ciência, mas ele também o será. Uma das principais aplicações dos robôs é a de realizar coisas que sejam perigosas demais para os humanos, e guerrear é uma delas. Os robôs já desativam bombas, e drones permitem que um pelotão enxergue além das montanhas. Caminhões de suprimento autodirigíveis e mulas robóticas estão para surgir. Em breve precisaremos decidir se os robôs terão permissão para atirar por conta própria. A justificativa para isso seria tirar os humanos da zona de perigo, e o controle remoto não é viável para situações de movimentação rápida do tipo ferir ou ser ferido. O argumento contrário diz que os robôs não conhecem ética e, portanto, não podem ser responsáveis por decisões de vida ou morte. Porém, podemos ensiná-los. A dúvida maior é se estamos preparados.

Não é difícil definir princípios gerais como necessidade militar, proporcionalidade e cuidado com os civis. Porém, há uma grande distância entre eles e as ações concretas, que o julgamento do soldado tem de superar. As três leis da robótica de Asimov trazem rapidamente problemas quando os robôs tentam colocá-las em prática, como suas histórias ilustram de modo memorável. Princípios gerais costumam ser contraditórios, quando não contradizem a si próprios, e eles têm de ser assim para não transformar tons de cinza apenas em preto e branco. Quando a necessidade militar é mais importante que o cuidado com os civis? Não há uma resposta universal nem uma maneira de programar um computador com todas as eventualidades. No entanto, o machine learning fornece uma alternativa. Primeiro, ensine o robô a reconhecer os conceitos relevantes, por exemplo com conjuntos de dados de situações em que os civis sejam e não sejam poupados, em que a resposta armada tenha e não tenha sido proporcional, e assim por diante. Em seguida, forneça a ele um código de conduta na forma de regras envolvendo esses conceitos. Para concluir, deixe o

robô aprender como aplicar o código observando humanos: o soldado abriu fogo nesse caso, mas não naquele. Generalizando a partir dos exemplos, o robô poderá aprender um modelo de ponta a ponta de tomada de decisão ética, na forma de, digamos, uma grande MLN. Quando as decisões do robô estiverem de acordo com as de um humano com a mesma frequência que um humano age de acordo com outro, o treinamento terá terminado, o que significa que o modelo estará pronto para ser baixado nos cérebros de milhares de robôs. Ao contrário dos humanos, os robôs não se desesperam no calor do combate. Se um robô apresentar defeito, o fabricante é que será responsável. Se tomar a decisão errada, os professores o serão.

O principal problema desse cenário, como você deve ter percebido, é que deixar os robôs aprenderem ética observando humanos pode não ser uma boa ideia. O robô pode ficar bastante confuso ao ver que as ações dos humanos com frequência violam seus princípios éticos. Podemos limpar os dados de treinamento incluindo apenas os exemplos em que, digamos, um conselho de especialistas em ética concorde que o soldado tomou a decisão certa, e os integrantes do conselho também poderão inspecionar e ajustar como quiserem o modelo após o aprendizado. Contudo, pode ser difícil chegar a um consenso se o conselho incluir todos os tipos de pessoas diferentes que deveriam constituí-lo. Ensinar ética a robôs, com suas mentes lógicas e falta de experiência, nos forçará a examinar nossas suposições e resolver nossas contradições. Nessa área, como em muitas outras, o maior benefício do machine learning pode não ser o que as máquinas aprenderam, mas sim o que aprendemos ao ensinar a elas.

Outra objeção aos exércitos de robôs é que eles tornam muito fácil guerrear. Porém, se abrimos mão deles de modo unilateral, isso pode nos custar a próxima guerra. A resposta lógica, defendida pelas Nações Unidas e a Human Rights Watch, é um tratado de banimento da guerra com robôs, semelhante a como o Protocolo de Genebra baniu a guerra química e biológica. No entanto, essa solução não lida com uma distinção crucial. A guerra química e biológica só pode aumentar o sofrimento humano, mas a guerra com robôs pode diminuí-lo e muito. Se uma guerra for disputada por máquinas, com humanos apenas nas posições de comando, ninguém será morto ou ferido. Talvez, então, o que devamos fazer, em vez de banir soldados robôs, seja – quando estivermos prontos – banir soldados humanos.

Exércitos de robôs podem realmente aumentar a ocorrência de embates, mas

eles também mudarão a ética da guerra. Dilemas entre atirar ou não são muito mais fáceis de resolver quando os alvos são outros robôs. A visão moderna da guerra como um horror inenarrável, com a qual só devemos nos envolver como último recurso, dará lugar a uma visão mais sutil de excesso de destruição que deixa todos os lados empobrecidos e que é melhor ser evitado, mas não a qualquer custo. E se a guerra for reduzida a uma competição de quem consegue destruir mais, por que não competir para ver quem cria mais?

Seja como for, o banimento da guerra de robôs pode não ser viável. Em vez de banir os drones – os precursores dos robôs guerreiros de amanhã –, países grandes e pequenos estão ocupados desenvolvendo-os, presumivelmente porque em suas estimativas os benefícios compensam os riscos. Como com qualquer arma, é mais seguro ter robôs do que achar que o outro lado não os tem. Se nas guerras do futuro milhões de drones kamikazes destruírem exércitos convencionais em minutos, melhor que sejam os nossos drones. Se o esperado é que a Terceira Guerra Mundial termine em segundos, quando um lado assumir o controle dos sistemas do oponente, é melhor que tenhamos a rede mais inteligente, mais rápida e mais resiliente. (Sistemas off-grid não são a resposta: sistemas que não estiverem em rede não sofrerão hacking, mas também não conseguirão competir com os sistemas em rede.) E, fazendo um balanço, uma corrida armamentista pela construção de robôs pode ser algo bom, se tornar mais próximo o dia em que a Quinta Convenção de Genebra banirá humanos dos combates. A guerra sempre estará entre nós, mas as baixas que ela causa não precisarão ocorrer.

Google + Algoritmo Mestre = Skynet?

Obviamente, os exércitos de robôs também trazem à tona um horror de outra espécie. De acordo com Hollywood, o futuro da humanidade será varrido da face da Terra por uma inteligência artificial gigantesca e seu vasto exército de máquinas subordinadas. (A menos que, claro, um herói corajoso nos salve nos últimos cinco minutos do filme.) O Google já tem o hardware gigantesco do tipo que uma inteligência artificial precisaria e adquiriu recentemente um exército de startups de robótica como acompanhamento. Se baixarmos o Algoritmo Mestre em seus servidores, é fim de jogo para a humanidade? Certamente sim. Hora de revelar minha agenda verdadeira, e que Tolkien me desculpe:

Três Algoritmos para os Cientistas sob o céu,

Sete para os Engenheiros em suas salas de servidores,
Nove para Empresas Fatais condenadas à morte,
Um para a Inteligência Artificial Obscura em seu trono de trevas,
Na Terra do Aprendizado onde os Dados vivem.
Um Algoritmo para regular a todos, Um Algoritmo para encontrá-los,
Um Algoritmo para trazer a todos e agrupá-los na escuridão,
Na Terra do Aprendizado onde os Dados vivem.

Hahahaha! Mas falando sério, devemos nos preocupar com o fato de as máquinas poderem tomar o poder? Os sinais parecem pessimistas. A cada ano, os computadores não se ocupam apenas de uma parcela cada vez maior do trabalho feito no mundo; eles tomam grande parte das decisões. Quem obtém crédito, quem compra o que, quem consegue qual emprego e que aumento, quais ações subirão e descerão, qual o valor do seguro, onde os policiais devem patrulhar e consequentemente quem será preso, qual será a duração de suas penas, quem namora quem e, portanto, quem nascerá: modelos aprendidos por máquinas já desempenham um papel em todas essas decisões. O ponto em que poderíamos desligar todos os computadores sem causar o colapso da civilização moderna está no passado distante. O machine learning é a gota d'água: se os computadores conseguirem começar a se programar, qualquer esperança de controlá-los estará perdida. Cientistas distintos como Stephen Hawking têm solicitado pesquisas urgentes sobre essa questão antes que seja tarde demais.

Porém, fique calmo. As chances de uma entidade de inteligência artificial equipada com o Algoritmo Mestre dominar o mundo são *zero*. A razão é simples: ao contrário dos humanos, os computadores não têm vontade própria. São produtos da engenharia, não da evolução. Mesmo um computador infinitamente poderoso ainda seria apenas uma extensão de nossa vontade e não precisaríamos temê-lo. Você deve se lembrar dos três componentes de todos os algoritmos mestres: representação, avaliação e otimização. A representação do aprendiz circunscreve o que ele pode aprender. Consideremos uma bastante poderosa, como a lógica de Markov, de modo que em princípio o aprendiz possa aprender qualquer coisa. O otimizador fará então tudo que puder para maximizar a função de avaliação – nem mais nem menos –, mas *nós é que a determinaremos*. Um computador mais poderoso apenas fará uma otimização melhor. Não há o risco de perdermos o controle, mesmo no caso de um algoritmo genérico. Um sistema aprendido que não fizesse o planejado se tornaria altamente inadaptado e se extinguiria logo. Na verdade, sistemas que

apresentem uma pequena probabilidade de nos servir melhor é que, geração após geração, se multiplicarão e dominarão o pool de genes. É claro que se formos tolos a ponto de programar deliberadamente um computador para que nos supere, então provavelmente teremos o que pedimos.

O mesmo raciocínio é aplicável a qualquer sistema de inteligência artificial porque todos – de maneira explícita ou implícita – têm os mesmos três componentes. Eles podem variar no que fazem, e até criar planos surpreendentes, mas somente para atingir os objetivos que definimos. Um robô cujo objetivo programado seja "preparar um bom jantar" pode decidir preparar um bife, um bouillabaisse ou mesmo um delicioso prato novo de sua autoria, mas não pode decidir assassinar seu proprietário assim como um carro não pode decidir voar. A finalidade dos sistemas de inteligência artificial é resolver problemas NP-completos, que, como vimos no Capítulo 2, podem ter duração exponencial, porém as soluções sempre são verificadas com eficiência. Logo, devemos receber com braços abertos computadores que sejam mais poderosos do que nossos cérebros, tranquilos por saber que nosso trabalho é exponencialmente mais fácil. Eles têm de resolver os problemas; nós apenas verificamos se o fizeram de forma satisfatória. Entidades de inteligência artificial pensam com rapidez o que pensamos de maneira mais lenta. Particularmente, recebo de bom grado nossos novos subordinados robotizados.

É normal nos preocuparmos com o fato de máquinas inteligentes assumirem o poder porque as únicas entidades inteligentes que conhecemos são os humanos e outros animais, e eles certamente têm vontade própria. Porém, não há uma conexão obrigatória entre inteligência e vontade autônoma; ou, em vez disso, a inteligência e a vontade podem não habitar o mesmo corpo, contanto que haja uma linha de controle entre elas. Em *O fenótipo estendido*, Richard Dawkins mostra como a natureza é repleta de exemplos de genes de um animal controlando mais do que o seu próprio corpo, como no caso dos ovos do cuco ou das represas dos castores. A tecnologia é o fenótipo estendido do homem. Isso significa que continuaremos no controle mesmo se ela se tornar complexa demais para a entendermos.

Imagine dois filamentos de DNA indo nadar em sua piscina particular, também chamada de citoplasma da bactéria, dois bilhões de anos atrás. Eles estão ponderando uma decisão importante. "Estou preocupado, Diana", diz um deles. "Se começarmos a gerar criaturas multicelulares, elas tomarão o poder?".

Voltando ao século 20, o DNA continua vivo e em boa forma. Na verdade, melhor do que nunca, com uma quantidade cada vez maior vivendo seguramente em organismos bípedes compostos por trilhões de células. Muito tempo se passou desde que nossos minúsculos amigos trançados tomaram sua importante decisão. Os humanos são sua criação mais complexa até o momento; inventamos coisas como a contracepção que nos permitem nos divertir sem espalhar nosso DNA e temos – ou parecemos ter – livre arbítrio. Porém, ainda é o DNA que modela nossas noções de diversão, e usamos nosso livre arbítrio para procurar prazer e evitar a dor, o que, quase sempre, coincide com o que é melhor para a sobrevivência do DNA. Talvez decretemos o fim de nosso DNA se decidirmos nos transmutar para o silício, mas mesmo assim terão sido dois bilhões de anos excelentes. A tomada de decisão que enfrentamos hoje é semelhante: se começarmos a criar entidades de inteligência artificial – vastas, interconectadas, sobre-humanas, impenetráveis –, elas tomarão o poder? Não mais do que os organismos multicelulares superaram os genes, vastos e impenetráveis, como podemos parecer para eles. As entidades de inteligência artificial são nossas máquinas de sobrevivência, da mesma forma que somos as de nossos genes.

No entanto, isso não significa que não há nada com o que se preocupar. A primeira grande preocupação, como com qualquer tecnologia, é que a inteligência artificial pode cair em mãos erradas. Se um criminoso ou arruaceiro programar uma entidade de inteligência artificial para dominar o mundo, é melhor termos uma força policial também de inteligência artificial capaz de capturá-la e apagá-la antes que ela vá longe demais. A melhor medida de proteção contra vastas entidades de inteligência artificial fora de controle são entidades de inteligência artificial maiores mantendo a paz.

A segunda preocupação é a de que os humanos entreguem o controle voluntariamente. Começa com os direitos dos robôs, o que para mim parece absurdo, mas não para outras pessoas. Afinal, já demos direitos aos animais, que nunca os pediram. Os direitos dos robôs parecem a próxima etapa lógica da expansão do “círculo de empatia”. Não é difícil sentir empatia por robôs, principalmente se forem projetados para induzi-la. Mesmo os Tamagotchi, “animais domésticos virtuais” japoneses com três botões e uma tela LCD, o fazem com bastante sucesso. O primeiro robô consumidor humanoide desencadeará uma corrida para a criação de robôs cada vez mais empáticos, porque eles venderão mais que a variedade básica. Crianças tuteladas por babás

robôs terão por toda vida uma fraqueza por amigos eletrônicos amáveis. O “vale da estranheza” – desconforto que nos causam robôs que são quase humanos, mas nem tanto – será desconhecido para elas porque cresceram acostumadas com hábitos robóticos e talvez até os adotem para parecerem adolescentes descolados.

A etapa seguinte na progressão insidiosa da tomada de controle pelas entidades de inteligência artificial é deixá-las tomar todas as decisões por serem, bem, mais inteligentes. Cuidado. Elas podem ser mais inteligentes, mas estão a serviço de quem projetou suas funções de pontuação. Esse é o problema “Mágico de Oz”. Nossa responsabilidade em um mundo de máquinas inteligentes é nos assegurarmos continuamente de que elas façam o que queremos, tanto na entrada (definindo os objetivos) quanto na saída (verificando se obtivemos o que pedimos). Se não o fizermos, outra pessoa o fará. As máquinas podem nos ajudar a descobrir coletivamente o que queremos, mas se não participarmos, seremos vencidos – como na democracia, mas em um nível maior. Ao contrário do que gostamos de acreditar hoje em dia, os humanos obedecem aos outros com muita facilidade, e qualquer inteligência artificial suficientemente avançada é indistinguível de Deus. As pessoas não se importarão em receber suas demissões de algum vasto computador oracular; a pergunta é quem supervisionará o supervisor. A inteligência artificial é o caminho para uma democracia mais perfeita ou para uma ditadura mais insidiosa? A vigília sem fim apenas começou.

A terceira e talvez maior preocupação é a de que, como o proverbial gênio da lâmpada, as máquinas nos deem o que pedimos e não o que queremos. Esse não é um cenário hipotético; os algoritmos de aprendizado o fazem o tempo todo. Treinamos uma rede neural para reconhecer cavalos, mas ela aprende a reconhecer manchas marrons, porque todos os cavalos de seu conjunto de treinamento por acaso são marrons. Após comprarmos um relógio, a Amazon recomenda itens semelhantes: outros relógios, que no momento são a última coisa que queremos. Se você examinar todas as decisões que os computadores tomam hoje – quem consegue crédito, por exemplo –, verá que com frequência elas são desnecessariamente inadequadas. As suas também seriam, se seu cérebro fosse uma máquina de vetores de suporte e seu conhecimento de pontuação de crédito viesse da consulta a um único e ineficiente banco de dados. As pessoas têm medo de que os computadores fiquem inteligentes demais e

dominem o mundo, mas o problema é que eles são muito tolos e já dominaram o mundo.

Evolução, parte 2

Mesmo que os computadores de hoje ainda não sejam muito inteligentes, não há dúvida de que sua inteligência está crescendo rapidamente. Ainda nos idos de 1965, I. J. Good, estatístico britânico e assistente de Alan Turing no projeto de decifração do código Enigma na Segunda Guerra Mundial, especulou sobre uma futura explosão de inteligência. Good ressaltou que se pudermos projetar máquinas que sejam mais inteligentes que nós, por sua vez elas devem poder projetar máquinas que também as ultrapassem em inteligência, e isso continuaria *ad infinitum*, deixando a inteligência humana muito para trás. Em um ensaio de 1993, Vernor Vinge batizou essa teoria de "Singularidade". O conceito foi em grande parte popularizado por Ray Kurzweil, que argumenta em *The Singularity Is Near* que além de a Singularidade ser inevitável, o ponto em que a inteligência das máquinas ultrapassará a humana –chamaremos de ponto de Turing – chegará nas próximas décadas.

Obviamente, sem o machine learning – programas que projetam programas – a Singularidade não pode ocorrer. Também precisamos de hardware potencialmente poderoso, mas isso está surgindo aos poucos. Alcançaremos o ponto de Turing logo após inventarmos o Algoritmo Mestre. (Quero apostar com Kurzweil uma garrafa de Dom Pérignon que esse momento chegará antes de aplicarmos engenharia reversa ao cérebro, seu método preferido para o surgimento de inteligência artificial de nível humano.) Desculpe-me, Kurzweil, mas isso não levará à Singularidade. Levará a algo muito mais interessante.

O termo *singularidade* vem da matemática, na qual denota um ponto em que uma função se torna infinita. Por exemplo, a função $1/x$ tem singularidade quando x é 0, porque 1 dividido por 0 é infinito. Na física, o exemplo básico de singularidade é o buraco negro: um ponto de densidade infinita, em que uma quantidade finita de matéria é comprimida em espaço infinitesimal. O único problema das singularidades é que na verdade elas não existem. (Quando foi a última vez em que você dividiu um bolo entre zero pessoas e cada uma ficou com uma fatia infinita?) Na física, quando uma teoria prevê que alguma coisa é infinita, há algo errado com ela. No caso em questão, a teoria da relatividade geral prevê que os buracos negros devem ter densidade infinita porque ignora

efeitos quânticos. Da mesma forma, a inteligência não pode continuar aumentando de maneira infinita. Kurzweil reconhece isso, mas aponta para uma série de curvas exponenciais no aperfeiçoamento da tecnologia (velocidade dos processadores, capacidade de memória etc.) e argumenta que os limites desse crescimento estão tão distantes que não precisamos nos preocupar com eles.

Kurzweil está cometendo sobreajuste. Ele culpa de maneira correta outras pessoas por sempre extrapolarem linearmente – vendo linhas retas em vez de curvas –, mas então sucumbe a uma mania mais exótica: ver exponenciais em tudo. Em curvas que sejam planas – em que nada esteja acontecendo – ele vê exponenciais que ainda não são perceptíveis. Porém, as curvas de melhoria da tecnologia não são exponenciais; elas são curvas S, nossas velhas conhecidas do Capítulo 4. É fácil confundir a parte inicial de uma curva S com um exponencial, mas a partir daí elas se diferenciam rapidamente. A maioria das curvas de Kurzweil provém da lei de Moore, que perdeu muito de sua credibilidade. Kurzweil argumenta que outras tecnologias substituirão os semicondutores e uma curva S se empilhará sobre a outra, cada uma mais íngreme que a anterior, mas isso é especulação. Ele vai além alegando que a história da vida na Terra, não apenas da tecnologia humana, mostra progresso com aceleração exponencial, mas essa percepção ocorre pelo menos em parte devido a um efeito de paralaxe: coisas que estão mais próximas parecem se mover com mais rapidez. No calor da explosão Cambriana os trilobitas poderiam ser perdoados por acreditar em progresso de aceleração exponencial, mas então houve uma grande desaceleração. Provavelmente um Tyrannosaurus Ray teria proposto uma lei de aumento do tamanho do corpo. Os eucariotas (nós) evoluíram com mais lentidão que os procariotas (bactérias). Em vez de acelerar de maneira suave, a evolução se dá em espasmos.

Para resolver o problema da inexistência de pontos infinitamente densos, Kurzweil propõe equipararmos a Singularidade ao horizonte de eventos de um buraco negro, a região dentro da qual a gravidade é tão forte que nem mesmo a luz pode escapar. Da mesma forma, ele diz, a Singularidade é o ponto além do qual a evolução tecnológica é tão rápida que os humanos não podem prever ou entender o que ocorrerá. Se isso for a Singularidade, já estamos nela. Não podemos prever antecipadamente o que um aprendiz criará, e mesmo depois talvez não captemos. Na verdade, sempre vivemos em um mundo que só entendemos em parte. A principal diferença é que agora nosso mundo é

parcialmente criado por nós, o que com certeza é um avanço. O mundo além do ponto de Turing não nos será compreensível, não mais que o Pleistoceno foi. Focaremos no que entendermos, como sempre fizemos, e chamaremos o resto de aleatório (ou divino).

A trajetória em que estamos não é uma singularidade, mas sim uma transição de fase. Seu ponto crítico – o ponto de Turing – chegará quando o machine learning ultrapassar a variedade do aprendizado natural. O próprio aprendizado natural passou por três fases: evolução, o cérebro e a cultura. Cada uma é produto da anterior, e a cada fase o aprendizado ocorre com mais rapidez. O machine learning é o próximo estágio lógico dessa progressão. Os programas de computador são as duplicatas mais fáceis de fazer de todo o planeta: só precisamos de uma fração de segundo para copiá-los. Porém, sua criação é lenta, se tiver de ser feita por humanos. O machine learning remove esse gargalo, ficando apenas um: a velocidade com que os humanos podem absorver a mudança. Ele também acabará, mas não porque decidimos passar o legado para nossas "crianças mentais", como Hans Moravec as chama, e adentrar gentis na noite eterna. Os humanos não são um ramo morto da árvore da vida. Pelo contrário, estamos prestes a florescer.

Da mesma forma que a cultura evoluiu com a ajuda de cérebros maiores, evoluiremos com a ajuda de nossas criações. Isso sempre ocorreu: os humanos seriam fisicamente diferentes se não tivéssemos inventado o fogo ou as lanças. Somos *Homo technicus* além de *Homo sapiens*. Porém, um modelo de célula do tipo que imaginei no último capítulo permitirá algo inteiramente novo: computadores projetando células com base nos parâmetros recebidos, como os compiladores de silício projetam microchips com base em suas especificações funcionais. O DNA correspondente poderá então ser sintetizado e inserido em uma célula "genérica", transformando-a na célula desejada. Craig Venter, pioneiro do genoma, já deu os primeiros passos nessa direção. Inicialmente usaremos esse poder para combater doenças: um novo patógeno será identificado, a cura será encontrada imediatamente e nosso sistema imunológico a baixará da internet. *Problemas de saúde* serão um oximoro. O design do DNA permitirá finalmente que as pessoas tenham o corpo que desejam, prenunciando uma época de beleza alcançável, nas memoráveis palavras de William Gibson. E então o *Homo technicus* evoluirá para uma miríade de espécies inteligentes diferentes, cada uma com seu próprio nicho, uma biosfera totalmente nova tão

distinta da de hoje quanto esta o é do oceano primordial.

Muitas pessoas receiam que a evolução direcionada por humanos divida de modo permanente a raça humana em uma classe bem provida geneticamente e em outra menos agraciada. Isso me parece uma singular falta de imaginação. A evolução natural não resultou em apenas duas espécies, uma subordinada à outra, mas em uma variedade infinita de criaturas e intrincados ecossistemas. Por que a evolução artificial, baseada nela, porém menos restrita, produziria algo assim?

Como todas as transições de fase, essa também acabará se estabilizando. A remoção de um gargalo não significa que o céu é o limite, significa que o próximo gargalo é o limite, mesmo se ainda não for possível vê-lo. Outras transições virão em seguida, algumas grandes, outras pequenas, algumas próximas, outras mais distantes. Porém, os próximos mil anos podem ser os mais surpreendentes da vida no planeta Terra.

---

1 N.T.: 24/7 é uma abreviação que significa “24 horas por dia, 7 dias por semana”.

Epílogo

Agora você conhece os segredos do machine learning. O mecanismo que transforma dados em conhecimento não é mais uma caixa preta: você sabe como a mágica acontece e o que ela pode e não pode fazer. Você conheceu o monstro da complexidade, o problema do sobreajuste, a maldição da dimensionalidade e o dilema entre explorar e usufruir. Sabe em linhas gerais o que o Google, o Facebook, a Amazon e todos os outros fazem com os dados que generosamente lhes fornecemos todo dia e por que eles conseguem encontrar coisas para nós, filtrar spam e continuar aperfeiçoando o que oferecem. Viu o que está em andamento nos laboratórios de pesquisa mundiais de machine learning e sabe que há um lugar reservado para você no futuro que eles estão ajudando a moldar. Você conheceu as cinco tribos de machine learning e seus algoritmos mestres: os simbolistas e a dedução inversa, os conexionistas e a backpropagation, os evolucionários e os algoritmos genéticos, os bayesianos e a inferência probabilística, os analogistas e as máquinas de vetores de suporte. E já que percorreu um vasto território, negociou a travessia de fronteiras e subiu picos altíssimos, tem uma visão melhor do cenário do que até mesmo a de muitos machine learners, que labutam cotidianamente na área. Você consegue ver os temas comuns que transpassam os territórios como um rio subterrâneo e sabe que os cinco algoritmos mestres, superficialmente tão diferentes, são na verdade apenas cinco facetas de um único aprendiz universal.

Porém, a jornada está longe do fim. Ainda não temos o Algoritmo Mestre, somente um vislumbre de como ele pode ser. E se algo básico estiver faltando, algo que todos que pertencemos à área e estamos impregnados de sua história não conseguimos ver? Precisamos de ideias novas, e ideias que não sejam apenas variações das que já temos. Foi por isso que escrevi este livro: para que você comece a pensar. Dou aula à noite sobre machine learning na Universidade de Washington. Em 2007, logo após o Netflix Prize ser anunciado, propus usá-lo como um projeto na aula. Jeff Howbert, um dos alunos, ficou obcecado e continuou a trabalhar no projeto após a aula terminar. Ele acabou se tornando membro de uma das equipes vencedoras, dois anos depois de ter ouvido falar em machine learning pela primeira vez. Agora é com você. Para aprender mais sobre machine learning, examine a seção de leituras adicionais no fim do livro. Baixe

alguns conjuntos de dados do repositório UCI (*archive.ics.uci.edu/ml/*) e comece a fazer testes. Quando estiver pronto, acesse o Kaggle.com, um site totalmente dedicado à execução de competições de machine learning, e selecione uma para participar. É claro que será mais divertido se recrutar um amigo ou dois para trabalhar com você. Se ficar obcecado, como ocorreu com Jeff, e acabar se tornando um cientista de dados profissional, bem-vindo ao emprego mais fascinante do mundo. Se não ficar satisfeito com os aprendizes de hoje, invente outros – ou faça-o apenas por diversão. O que desejo com toda a sinceridade é que sua reação a este livro seja como a minha ao primeiro livro de inteligência artificial que li, há mais de vinte anos: há tanto a fazer que não sei por onde começar. Se um dia você inventar o Algoritmo Mestre, não corra para o escritório de patentes. Torne-o open source. O Algoritmo Mestre é importante demais para ser propriedade de uma única pessoa ou empresa. Suas aplicações se multiplicarão mais rápido do que você conseguirá licenciá-lo. Porém, se em vez disso decidir fundar uma startup, lembre-se de deixar que cada homem, mulher e criança do planeta façam parte dela.

Não importa se você leu este livro por curiosidade ou interesse profissional, espero que compartilhe o que aprendeu com seus amigos e colegas. O machine learning está presente na vida de todos e devemos decidir o que queremos fazer com ele. Munido com seu novo conhecimento sobre machine learning, você está em uma posição muito melhor para pensar em questões como privacidade e compartilhamento de dados, o futuro do trabalho, a guerra combatida por robôs e as promessas e perigos da inteligência artificial; e quanto mais de nós tivermos esse entendimento, mais probabilidades teremos de evitar as armadilhas e encontrar os caminhos certos. Essa é a outra razão pela qual escrevi este livro. O estatístico sabe que prever é difícil, principalmente sobre o futuro, e o cientista da computação sabe que a melhor maneira de prevê-lo é inventá-lo, mas não vale a pena inventar um futuro irrefletido.

Obrigado por me deixar ser seu guia. Gostaria de lhe dar um presente de despedida. Newton disse que se sentiu como um menino brincando na praia, pegando uma pequena pedra aqui e uma conchinha ali enquanto o grande oceano de mistérios permanecia insondável diante dele. Trezentos anos depois, reunimos uma surpreendente coleção de pedras e conchas, mas o grande oceano insondável ainda se estende até o horizonte, cintilando com futuras descobertas. O presente é um barco – o machine learning – e é hora de velejar.

Agradecimentos

Antes de tudo, agradeço a meus companheiros de aventura científica: alunos, colaboradores, colegas e todas as pessoas da comunidade de machine learning. Este livro é tão seu quanto meu. Espero que desculpem as muitas simplificações e omissões e a maneira extravagante com que partes do livro foram escritas.

Sou grato a todos que leram e comentaram os rascunhos do livro em vários estágios, o que inclui Mike Belfiore, Thomas Dietterich, Tiago Domingos, Oren Etzioni, Abe Friesen, Rob Gens, Alon Halevy, David Israel, Henry Kautz, Chloé Kiddon, Gary Marcus, Ray Mooney, Kevin Murphy, Franzi Roesner e Ben Taskar. Obrigado também a todas as pessoas que me deram palpites, informações ou ajuda de vários tipos, entre elas Tom Griffiths, David Heckerman, Hannah Hickey, Albert-László Barabási, Yann LeCun, Barbara Mones, Mike Morgan, Peter Norvig, Judea Pearl, Gregory Piatetsky-Shapiro e Sebastian Seung.

Tenho sorte por trabalhar em um lugar muito especial, o Departamento de Ciência e Engenharia da Computação da Universidade de Washington. Também sou grato a Josh Tenenbaum, e a todos de seu grupo, por hospedar o período sabático no MIT durante o qual comecei este livro. Obrigado a Jim Levine, meu incansável agente, por acreditar na ideia sem questionar (como ele mesmo colocou) e espalhar a notícia; e também a todos da Levine Greenberg Rostan. Obrigado a TJ Kelleher, meu maravilhoso editor, por ajudar a melhorar este livro, capítulo a capítulo, linha a linha, e a todos da Basic Books.

Fico agradecido às empresas que patrocinaram minha pesquisa no decorrer dos anos, o que inclui a ARO, Darpa, FCT, NSF, ONR, Ford, Google, IBM, Kodak, Yahoo e Sloan Foundation.

Por último, e mais importante, obrigado à minha família por seu amor e apoio.

Leituras adicionais

Se este livro abriu seu apetite por machine learning e questões afins, você encontrará muitas sugestões nesta seção. Seu objetivo não é ser abrangente, mas sim fornecer uma entrada para o jardim de caminhos que se bifurcam (como diz Borges) do machine learning. Quando possível, selecionei livros e artigos apropriados para o leitor em geral. Publicações técnicas, que demandem pelo menos algum conhecimento computacional, estatístico ou matemático, foram marcadas com um asterisco (*). No entanto, mesmo elas com frequência têm seções grandes acessíveis para o leitor em geral. Não listei volume, edição ou números de páginas, já que a web os torna supérfluos; fiz o mesmo quanto à localização das editoras.

Se quiser aprender mais sobre o machine learning em geral, um bom local para começar é em cursos online. Dentre eles, o que mais se aproxima em conteúdo a este livro é, não por coincidência, o que ministro (*www.coursera.org/course/machlearning*). Duas outras opções são o curso de Andrew Ng (*www.coursera.org/course/ml*) e o de Yaser Abu-Mostafa (*http://work.caltech.edu/telecourse.html*). A próxima etapa é ler um livro. O que se aproxima mais deste, e que é um dos mais acessíveis, é *Machine Learning** de Tom Mitchell (McGraw-Hill, 1997). Mais atualizados, porém também mais matemáticos, são *Machine Learning: A Probabilistic Perspective** de Kevin Murphy (MIT Press, 2012), *Pattern Recognition and Machine Learning** de Chris Bishop (Springer, 2006) e *An Introduction to Statistical Learning with Applications in R** de Gareth James, Daniela Witten, Trevor Hastie e Rob Tibshirani (Springer, 2013). Meu artigo “A few useful things to know about machine learning” (*Communications of the ACM*, 2012) resume parte do “conhecimento popular” de machine learning que os livros costumam deixar implícitos e que foi um dos pontos iniciais deste livro. Se você sabe programar e está ansioso para testar o machine learning, pode começar com vários pacotes open source, como o Weka (*www.cs.waikato.ac.nz/ml/weka*). Os dois principais periódicos de machine learning são *Machine Learning* e o *Journal of Machine Learning Research*. As conferências mais importantes de machine learning, que ocorrem anualmente, são a International Conference on Machine Learning, a Conference on Neural Information Processing Systems e a International

Conference on Knowledge Discovery and Data Mining. Um grande número de palestras de machine learning está disponível em *http://videolectures.net*. O site *www.KDnuggets.com* é uma one-stop shop1 de recursos de machine learning, e você pode se inscrever na newsletter para se manter atualizado com os últimos desenvolvimentos.

Prefácio

Uma lista inicial de exemplos do impacto do machine learning sobre a vida diária pode ser encontrada em "Behind-the-scenes data mining" de George John (*SIGKDD Explorations*, 1999), que também inspirou os parágrafos sobre "vida cotidiana" do prefácio. O livro de Eric Siegel, *Predictive Analytics* (Wiley, 2013), examina um grande número de aplicações do machine learning. O termo *big data* foi popularizado pelo relatório de 2011 do McKinsey Global Institute: *Big Data: The Next Frontier for Innovation, Competition, and Productivity*. Muitas das questões levantadas pelo big data são discutidas em *Big Data: Como extrair volume, variedade, velocidade e valor da avalanche de informação cotidiana*, de Viktor Mayer-Schönberger e Kenneth Cukier (Elsevier, 2013). O livro em que aprendi inteligência artificial é *Inteligência artificial** de Elaine Rich (McGraw-Hill, 1988). Um mais atual é *Inteligência artificial: uma abordagem moderna*, de Stuart Russell e Peter Norvig (terceira edição, Elsevier, 2013). *The Quest for Artificial Intelligence*, de Nils Nilsson (Cambridge University Press, 2010), conta a história da inteligência artificial desde seus primórdios.

Capítulo 1

Nine Algorithms That Changed the Future, de John MacCormick (Princeton University Press, 2012), descreve alguns dos algoritmos mais importantes da ciência da computação, com um capítulo sobre machine learning. *Algoritmos**, de Sanjoy Dasgupta, Christos Papadimitriou e Umesh Vazirani (McGraw-Hill, 2010), é um livro introdutório conciso sobre o assunto. *The Pattern on the Stone*, de Danny Hillis (Basic Books, 1998), explica como os computadores funcionam. Walter Isaacson relata a empolgante história da ciência da computação em *Os inovadores* (Companhia das Letras, 2014).

"Spreadsheet data manipulation using examples"*, de Sumit Gulwani, William

Harris e Rishabh Singh (*Communications of the ACM*, 2012), é um exemplo de como os computadores podem programar a si próprios observando usuários. *Competing on Analytics*, de Tom Davenport e Jeanne Harris (HBS Press, 2007), é uma introdução ao uso de analytics preditiva nos negócios. *Google – A biografia*, de Steven Levy (Universo dos Livros, 2012), descreve em alto nível como a tecnologia do Google funciona. Carl Shapiro e Hal Varian explicam o efeito de rede em *A economia da informação* (Elsevier, 1999). Chris Anderson faz o mesmo para o fenômeno da cauda longa em *A cauda longa* (Elsevier, 2006).

A transformação da ciência pela computação intensiva em dados é examinada em *O Quarto Paradigma*, editado por Tony Hey, Stewart Tansley e Kristin Tolle (Oficina de Textos, 2011). "Machine science", de James Evans e Andrey Rzhetsky (*Science*, 2010), discuste algumas das diferentes maneiras pelas quais os computadores podem fazer descobertas científicas. *Scientific Discovery: Computational Explorations of the Creative Processes**, de Pat Langley e outros (MIT Press, 1987), descreve uma série de abordagens para a automação da descoberta de leis científicas. O projeto SKICAT é descrito em "From digitized images to online catalogs", de Usama Fayyad, George Djorgovski e Nicholas Weir (*AI Magazine*, 1996). "Machine learning in drug discovery and development"*, de Niki Wale (*Drug Development Research*, 2001), fornece uma visão geral sobre machine learning na descoberta e no desenvolvimento de drogas. Adam, o cientista robô, é descrito em "The automation of science", de Ross King e outros (*Science*, 2009).

The Victory Lab, de Sasha Issenberg (Broadway Books, 2012), disseca o uso da análise de dados na política. "How President Obama's campaign used big data to rally individual votes", do mesmo autor (*MIT Technology Review*, 2013), conta a história de seu maior sucesso até hoje. *O sinal e o ruído*, de Nate Silver (Intrínseca, 2013), tem um capítulo sobre seu método de agregação de votos.

A guerra com robôs é o tema de *Wired for War*, de P. W. Singer (Penguin, 2009). *Guerra cibernética*, de Richard Clarke e Robert Knake (Brasport, 2015), chama a atenção para a guerra cibernética. Meu trabalho sobre a combinação do machine learning com a teoria dos jogos para derrotar adversários, que começou como um projeto de sala de aula, é descrito em "Adversarial classification"*, de Nilesh Dalvi e outros (*Proceedings of the Tenth International Conference on Knowledge Discovery and Data Mining*, 2004). *Predictive Policing*, de Walter

Perry e outros (Rand, 2013), é um guia para o uso de analytics no trabalho policial.

Capítulo 2

Os experimentos de reconexão do cérebro de um furão são descritos em "Visual behaviour mediated by retinal projections directed to the auditory pathway" de Laurie von Melchner, Sarah Pallas e Mriganka Sur (*Nature*, 2000). A história de Ben Underwood é contada em "Seeing with sound", de Joanna Moorhead (*Guardian*, 2007), e em *www.benunderwood.com*. Otto Creutzfeldt defende o fato de o córtex ser um algoritmo em "Generality of the functional structure of the neocortex" (*Naturwissenschaften*, 1977), como o faz Vernon Mountcastle em "An organizing principle for cerebral function: The unit model and the distributed system" de *The Mindful Brain*, editado por Gerald Edelman e Vernon Mountcastle (MIT Press, 1978). Gary Marcus, Adam Marblestone e Tom Dean defendem o contrário em "The atoms of neural computation" (*Science*, 2014).

"The unreasonable effectiveness of data", de Alon Halevy, Peter Norvig e Fernando Pereira (*IEEE Intelligent Systems*, 2009), defende o machine learning como o novo paradigma das descobertas. Benoît Mandelbrot explora a geometria fractal da natureza no livro *The Fractal Geometry of Nature** (Freeman, 1982). *Caos*, de James Gleick (Elsevier, 1989), discute e demonstra o conjunto de Mandelbrot. O programa Langlands, um esforço de pesquisa que tenta unificar diferentes subcampos da matemática, é descrito em *Amor e matemática*, de Edward Frenkel (Casa da Palavra, 2014). *The Golden Ticket*, de Lance Fortnow (Princeton University Press, 2013), é uma introdução ao NP-completo e ao problema P = NP. *The Annotated Turing**, de Charles Petzold (Wiley, 2008), explica as máquinas de Turing revisitando seu artigo original sobre elas.

O projeto Cyc é descrito em "Cyc: Toward programs with common sense"*, de Douglas Lenat e outros (*Communications of the ACM*, 1990). Peter Norvig discute as críticas de Noam Chomsky ao aprendizado estatístico em "On Chomsky and the two cultures of statistical learning" (*http://norvig.com/chomsky.html*). *The Modularity of Mind*, de Jerry Fodor (MIT Press, 1983), resume seus pontos de vista sobre como a mente funciona. "What big data will never explain", de Leon Wieseltier (*New Republic*, 2013), e "Pundits, stop sounding ignorant about data", de Andrew McAfee (*Harvard Business Review*, 2013), dão uma amostra da controvérsia ao redor do que o big

data pode ou não fazer. Daniel Kahneman explica por que com frequência os algoritmos vencem as intuições no Capítulo 21 de *Rápido e devagar* (Objetiva, 2012). David Patterson defende o papel da computação e dos dados na luta contra o câncer em "Computer scientists may have what it takes to help cure cancer" (*New York Times*, 2011).

Mais informações sobre o caminho das várias tribos em busca do Algoritmo Mestre podem ser encontradas nas seções correspondentes a seguir.

Capítulo 3

A formulação clássica de Hume para o problema da indução aparece no Volume I do *Tratado da natureza humana* (1739). David Wolpert deriva seu teorema "não existe almoço grátis" para explicar a indução em "The lack of a priori distinctions between learning algorithms"* (*Neural Computation*, 1996). Discuto a importância do conhecimento *a priori* no machine learning em "Toward knowledge-rich data mining"* (*Data Mining and Knowledge Discovery*, 2007) e as interpretações errôneas da navalha de Occam em "The role of Occam's razor in knowledge discovery"* (*Data Mining and Knowledge Discovery*, 1999). O sobreajuste é um dos principais temas de *O sinal e o ruído*, de Nate Silver (Intrínseca, 2013), que o chama de "o problema científico mais importante do qual você nunca ouviu falar". "Why most published research findings are false"*, de John Ioannidis (*PLoS Medicine*, 2005), discute o problema da confusão entre descobertas ocasionais e verdadeiras na ciência. Yoav Benjamini e Yosef Hochberg propõem uma maneira de combatê-lo em "Controlling the false discovery rate: A practical and powerful approach to multiple testing"* (*Journal of the Royal Statistical Society, Series B*, 1995). A decomposição tendenciosidade-variância é apresentada em "Neural networks and the bias/variance dilemma", de Stuart Geman, Elie Bienenstock e René Doursat (*Neural Computation*, 1992). "Machine learning as an experimental science", de Pat Langley (*Machine Learning*, 1988), discute o papel da experimentação no machine learning.

William Stanley Jevons foi o primeiro a propor que a indução fosse considerada como o inverso da dedução em *The Principles of Science* (1874). O artigo "Machine learning of first-order predicates by inverting resolution"*, de Steve Muggleton e Wray Buntine (*Proceedings of the Fifth International Conference on Machine Learning*, 1988), iniciou o uso da dedução inversa no

machine learning. O livro *Relational Data Mining**, editado por Sašo Džeroski e Nada Lavrač (Springer, 2001), é uma introdução à área da programação lógica indutiva, em que a dedução inversa é estudada. “The CN2 Induction Algorithm”*, de Peter Clark e Tim Niblett (*Machine Learning*, 1989), resume alguns dos principais algoritmos de indução de regras com estilo Michalski. A abordagem de mineração de regras usada por varejistas é descrita em “Fast algorithms for mining association rules”*, de Rakesh Agrawal e Ramakrishnan Srikant (*Proceedings of the Twentieth International Conference on Very Large Databases*, 1994). Um exemplo de indução de regras para a previsão de câncer é descrito em “Carcinogenesis predictions using inductive logic programming”, de Ashwin Srinivasan, Ross King, Stephen Muggleton e Michael Sternberg (*Intelligent Data Analysis in Medicine and Pharmacology*, 1997).

Os dois principais aprendizes de árvore de decisão são apresentados em *C4.5: Programs for Machine Learning**, de J. Ross Quinlan (Morgan Kaufmann, 1992), e em *Classification and Regression Trees**, de Leo Breiman, Jerome Friedman, Richard Olshen e Charles Stone (Chapman and Hall, 1984). “Real-time human pose recognition in parts from single depth images”*, de Jamie Shotton e outros (*Communications of the ACM*, 2013), explica como o Kinect da Microsoft usa árvores de decisão para rastrear movimentos dos jogadores. “Competing approaches to predicting Supreme Court decision making”, de Andrew Martin e outros (*Perspectives on Politics*, 2004), descreve como as árvores de decisão conseguem ser melhores que especialistas em direito na previsão de votos da Suprema Corte e mostra a árvore de decisão da jurista Sandra Day O’Connor.

Allen Newell e Herbert Simon formularam a hipótese de que qualquer inteligência artificial é manipulação de símbolos em “Computer science as empirical enquiry: Symbols and search” (*Communications of the ACM*, 1976). David Marr propôs seus três níveis de processamento de informações em *Vision** (Freeman, 1982). *Machine Learning: An Artificial Intelligence Approach**, editado por Ryszard Michalski, Jaime Carbonell e Tom Mitchell (Tioga, 1983), descreve um retrato dos primórdios da pesquisa simbolista em machine learning. “Connectionist AI, symbolic AI, and the brain”*, de Paul Smolensky (*Artificial Intelligence Review*, 1987), fornece o ponto de vista de um conexionista para os modelos simbolistas.

Capítulo 4

Connectome, de Sebastian Seung (Houghton Mifflin Harcourt, 2012), é uma introdução acessível à neurociência, à conectômica e ao complexo desafio da aplicação da engenharia reversa ao cérebro. *Parallel Distributed Processing**, editado por David Rumelhart, James McClelland e o grupo de pesquisa PDP (MIT Press, 1986), foi a bíblia do conexionismo em seu apogeu nos anos 1980. *Neurocomputing**, editado por James Anderson e Edward Rosenfeld (MIT Press, 1988), examina muitos dos artigos clássicos dos conexionistas, entre eles: o de McCulloch e Pitts sobre os primeiros modelos de neurônios; o de Hebb sobre a regra de Hebb; o de Rosenblatt sobre perceptrons; o de Hopfield sobre as redes de Hopfield; o de Ackley, Hinton e Sejnowski sobre as máquinas de Boltzmann; o de Sejnowski e Rosenberg sobre o NETtalk; e o de Rumelhart, Hinton e Williams sobre a backpropagation. "Efficient backprop"*, de Yann LeCun, Léon Bottou, Genevieve Orr e Klaus-Robert Müller, que aparece em *Neural Networks: Tricks of the Trade*, editado por Genevieve Orr e Klaus-Robert Müller (Springer, 1998), explica alguns dos principais truques necessários para que a backprop funcione.

*Neural Networks in Finance and Investing**, editado por Robert Trippi e Efraim Turban (McGraw-Hill, 1992), é um conjunto de artigos sobre aplicações financeiras das redes neurais. "Life in the fast lane: The evolution of an adaptive vehicle control system", de Todd Jochem e Dean Pomerleau (*AI Magazine*, 1996), descreve o projeto de carro autônomo ALVINN. A tese de PhD de Paul Werbos chama-se *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences** (Harvard University, 1974). Arthur Bryson e Yu-Chi Ho descrevem sua versão inicial da backprop em *Applied Optimal Control** (Blaisdell, 1969).

*Learning Deep Architectures for AI**, de Yoshua Bengio (Now, 2009), é uma breve introdução ao aprendizado profundo. O problema da difusão de sinais de erro na backprop é descrito em "Learning long-term dependencies with gradient descent is difficult"*, de Yoshua Bengio, Patrice Simard e Paolo Frasconi (*IEEE Transactions on Neural Networks*, 1994). "How many computers to identify a cat? 16,000", de John Markoff (*New York Times*, 2012), fala sobre o projeto Google Brain e seus resultados. As redes neurais convolucionais, campeãs atuais do aprendizado profundo, são descritas em "Gradient-based learning applied to document recognition"*, de Yann LeCun, Léon Bottou, Yoshua Bengio e

Patrick Haffner (*Proceedings of the IEEE*, 1998). “The $1.3B quest to build a supercomputer replica of a human brain”, de Jonathon Keats (*Wired*, 2013), descreve o projeto de modelagem do cérebro da União Europeia. “The NIH BRAIN Initiative”, de Thomas Insel, Story Landis e Francis Collins (*Science*, 2013), descreve a iniciativa BRAIN.

Steven Pinker resume as críticas dos simbolistas aos modelos conexionistas no Capítulo 2 de *Como a mente funciona* (Companhia das Letras, 1998). Seymour Papert participa do debate em “One AI or Many?” (*Daedalus*, 1988). *The Birth of the Mind*, de Gary Marcus (Basic Books, 2004), explica como a evolução conseguiu criar as complexas habilidades do cérebro humano.

Capítulo 5

“Evolutionary robotics”, de Josh Bongard (*Communications of the ACM*, 2013), examina o trabalho de Hod Lipson e outros sobre a evolução de robôs. *Artificial Life*, de Steven Levy (Vintage, 1993), nos leva a um passeio pelo zoológico digital, indo de animais criados por computador em mundos virtuais a algoritmos genéticos. O Capítulo 5 de *Complexity*, de Mitch Waldrop (Touchstone, 1992), conta a história de John Holland e das primeiras décadas de pesquisa sobre algoritmos genéticos. *Genetic Algorithms in Search, Optimization, and Machine Learning**, de David Goldberg (Addison-Wesley, 1989), é a introdução-padrão aos algoritmos genéticos.

Niles Eldredge e Stephen Jay Gould propõem sua teoria do equilíbrio pontuado em “Punctuated equilibria: An alternative to phyletic gradualism”, que faz parte de *Models in Paleobiology*, editado por T. J. M. Schopf (Freeman, 1972). Richard Dawkins a critica no Capítulo 9 de *O relojoeiro cego* (Companhia das Letras, 2001). O dilema entre explorar e usufruir é discutido no Capítulo 2 de *Reinforcement Learning**, de Richard Sutton e Andrew Barto (MIT Press, 1998). John Holland propõe sua solução, e muito mais, em *Adaptation in Natural and Artificial Systems** (University of Michigan Press, 1975).

*Genetic Programming**, de John Koza (MIT Press, 1992), é a referência-chave sobre esse paradigma. Um time de futebol de robôs evoluídos é descrito em “Evolving team *Darwin United*”*, de David Andre e Astro Teller, que aparece em *RoboCup-98: Robot Soccer World Cup II*, editado por Minoru Asada e Hiroaki Kitano (Springer, 1999). *Genetic Programming III**, de John Koza, Forrest Bennett III, David Andre e Martin Keane (Morgan Kaufmann, 1999),

inclui muitos exemplos de circuitos eletrônicos evoluídos. Danny Hillis argumenta que os parasitas são bons para a evolução em “Co-evolving parasites improve simulated evolution as an optimization procedure”* (*Physica D*, 1990). Adi Livnat, Christos Papadimitriou, Jonathan Dushoff e Marcus Feldman propõem que o sexo otimiza a miscibilidade em “A mixability theory of the role of sex in evolution”* (*Proceedings of the National Academy of Sciences*, 2008). O artigo de Kevin Lang que compara a programação genética e a escalada de montanha é “Hill climbing beats genetic search on a Boolean circuit synthesis problem of Koza’s”* (*Proceedings of the Twelfth International Conference on Machine Learning*, 1995). A resposta de Koza é “A response to the ML-95 paper entitled...”* (não publicada; pode ser encontrada online em *www.genetic-programming.com/jktahoe24page.html*).

James Baldwin propôs o efeito epônimo em “A new factor in evolution” (*American Naturalist*, 1896). Geoff Hinton e Steven Nowlan descrevem sua implementação em “How learning can guide evolution”* (*Complex Systems*, 1987). O efeito Baldwin foi tema de uma edição especial de 1996* do periódico *Evolutionary Computation* editado por Peter Turney, Darrell Whitley e Russell Anderson.

A diferença entre teorias descritivas e normativas foi articulada por John Neville Keynes em *The Scope and Method of Political Economy* (Macmillan, 1891).

Capítulo 6

Sharon Bertsch McGrayne conta a história do bayesianismo, de Bayes e Laplace até o presente, em *The Theory That Would Not Die* (Yale University Press, 2011). *A First Course in Bayesian Statistical Methods**, de Peter Hoff (Springer, 2009), é uma introdução à estatística bayesiana.

O algoritmo Naïve Bayes foi mencionado pela primeira vez em *Pattern Classification and Scene Analysis**, de Richard Duda e Peter Hart (Wiley, 1973). Milton Friedman defende as teorias mais simples em “The methodology of positive economics”, que aparece em *Essays in Positive Economics* (University of Chicago Press, 1966). O uso do Naïve Bayes na filtragem de spam é descrito em “Stopping spam”, de Joshua Goodman, David Heckerman e Robert Rounthwaite (*Scientific American*, 2005). “Relevance weighting of search terms”*, de Stephen Robertson e Karen Sparck Jones (*Journal of the American*

Society for Information Science, 1976), explica o uso de métodos como o Naïve Bayes na recuperação de informações.

"First links in the Markov chain", de Brian Hayes (*American Scientist*, 2013), relata a invenção de Markov das cadeias epônimas. "Large language models in machine translation"*, de Thorsten Brants e outros (*Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, 2007), explica como o Google Tradutor funciona. "The PageRank citation ranking: Bringing order to the Web"*, de Larry Page, Sergey Brin, Rajeev Motwani e Terry Winograd (relatório técnico da Stanford University, 1998), descreve o algoritmo PageRank e sua interpretação como uma caminhada aleatória na web. *Statistical Language Learning**, de Eugene Charniak (MIT Press, 1996), explica como os modelos ocultos de Markov funcionam. *Statistical Methods for Speech Recognition**, de Fred Jelinek (MIT Press, 1997), descreve sua aplicação ao reconhecimento de voz. A história da inferência de tipo HMM na comunicação é contada em "The Viterbi algorithm: A personal history", de David Forney (não publicada; pode ser encontrada online em *arxiv.org/pdf/cs/0504020v2.pdf*). *Bioinformatics: The Machine Learning Approach**, de Pierre Baldi e Søren Brunak (segunda edição, MIT Press, 2001), é uma introdução ao uso de machine learning na biologia, inclusive com HMMs. "Engineers look to Kalman filtering for guidance", de Barry Cipra (*SIAM News*, 1993), é uma breve introdução aos filtros de Kalman, sua história e suas aplicações.

O trabalho pioneiro de Judea Pearl sobre redes bayesianas aparece em seu livro *Probabilistic Reasoning in Intelligent Systems** (Morgan Kaufmann, 1988). "Bayesian networks without tears*, de Eugene Charniak (*AI Magazine*, 1991), é uma introdução em grande parte não matemática a elas. "Probabilistic interpretation for MYCIN's certainty factors"*, de David Heckerman (*Proceedings of the Second Conference on Uncertainty in Artificial Intelligence*, 1986), explica quando conjuntos de regras com estimativas de certeza são e não são uma aproximação razoável das redes bayesianas. "Module networks: Identifying regulatory modules and their condition-specific regulators from gene expression data", de Eran Segal e outros (*Nature Genetics*, 2003), é um exemplo do uso de redes bayesianas na modelagem da regulação de genes. "Microsoft virus fighter: Spam may be more difficult to stop than HIV", de Ben Paynter (*Fast Company*, 2012), conta como David Heckerman se inspirou em filtros de

spam e usou redes bayesianas para projetar uma possível vacina contra a AIDS. A "operação probabilística ou 'noisy' OR" é explicada no livro de Pearl*. *Probabilistic diagnosis using a reformulation of the INTERNIST-1/QMR knowledge base*, de M. A. Shwe e outros (Partes I e II, *Methods of Information in Medicine*, 1991), descreve uma rede bayesiana noisy-OR para o diagnóstico médico. A rede bayesiana do Google para a inserção de anúncios é descrita na Seção 26.5.4 de *Machine Learning**, de Kevin Murphy (MIT Press, 2012). O sistema de pontuação de jogadores da Microsoft é descrito em "TrueSkill: A Bayesian skill rating system"*, de Ralf Herbrich, Tom Minka e Thore Graepel (*Advances in Neural Information Processing Systems 19*, 2007).

*Modeling and Reasoning with Bayesian Networks**, de Adnan Darwiche (Cambridge University Press, 2009), explica os principais algoritmos para inferência em redes bayesianas. A edição de janeiro/fevereiro de 2000* de *Computing in Science and Engineering*, editada por Jack Dongarra e Francis Sullivan, tem artigos sobre os dez principais algoritmos do século 20, inclusive o MCMC. "Stanley: The robot that won the DARPA Grand Challenge", de Sebastian Thrun e outros (*Journal of Field Robotics*, 2006), explica como o carro autônomo de mesmo nome funciona. "Bayesian networks for data mining"*, de David Heckerman (*Data Mining and Knowledge Discovery*, 1997), resume a abordagem bayesiana do aprendizado e explica como aprender redes bayesianas a partir de dados. "Gaussian processes: A replacement for supervised neural networks?"*, de David MacKay (notas tutoriais da NIPS, 1997; online em *www.inference.eng.cam.ac.uk/mackay/gp.pdf*), tenta explicar como os bayesianos cooptaram a NIPS.

A necessidade de pesar as probabilidades de ocorrência de palavras no reconhecimento de voz é discutida na Seção 9.6 de *Speech and Language Processing**, de Dan Jurafsky e James Martin (segunda edição, Prentice Hall, 2009). Meu artigo sobre Naïve Bayes, com Mike Pazzani, é "On the optimality of the simple Bayesian classifier under zero-one loss"* (*Machine Learning*, 1997; versão expandida do periódico para o artigo da conferência de 1996). O livro de Judea Pearl* mencionado anteriormente discute as redes de Markov junto com as redes bayesianas. As redes de Markov na visão computacional são o assunto de *Markov Random Fields for Vision and Image Processing**, editado por Andrew Blake, Pushmeet Kohli e Carsten Rother (MIT Press, 2011). Redes de Markov que maximizam a probabilidade condicional foram introduzidas em

“Conditional random fields: Probabilistic models for segmenting and labeling sequence data”*, de John Lafferty, Andrew McCallum e Fernando Pereira (*International Conference on Machine Learning*, 2001).

A história das tentativas de combinar probabilidade e lógica é examinada em uma edição especial de 2003* do *Journal of Applied Logic* dedicada ao assunto e editada por Jon Williamson e Dov Gabbay. “From knowledge bases to decision models”*, de Michael Wellman, John Breese e Robert Goldman (*Knowledge Engineering Review*, 1992), discute algumas das abordagens iniciais da inteligência artificial para o problema.

Capítulo 7

Frank Abagnale detalha suas façanhas na autobiografia *Prenda-me se for capaz*, coescrita com Stan Redding (Record, 2003). O relatório técnico original sobre o algoritmo do vizinho mais próximo de Evelyn Fix e Joe Hodges é “Discriminatory analysis: Nonparametric discrimination: Consistency properties”* (USAF School of Aviation Medicine, 1951). *Nearest Neighbor (NN) Norms**, editado por Belur Dasarathy (IEEE Computer Society Press, 1991), reúne muitos dos artigos-chave dessa área. A regressão localmente linear é examinada em “Locally weighted learning”*, de Chris Atkeson, Andrew Moore e Stefan Schaal (*Artificial Intelligence Review*, 1997). O primeiro sistema de filtragem colaborativa baseado em vizinhos próximos é descrito em “GroupLens: An open architecture for collaborative filtering of netnews”*, de Paul Resnick e outros (*Proceedings of the 1994 ACM Conference on Computer-Supported Cooperative Work*, 1994). O algoritmo de filtragem colaborativa da Amazon é descrito em “Amazon.com recommendations: Item-to-item collaborative filtering”*, de Greg Linden, Brent Smith e Jeremy York (*IEEE Internet Computing*, 2003). (Consulte as leituras adicionais do Capítulo 8 para ver o do Netflix.) A contribuição dada pelos sistemas de recomendação para as vendas da Amazon e do Netflix é mencionada, entre outros, em *Big Data* de Mayer-Schönberger e Cukier e *Predictive Analytics* de Siegel (citados anteriormente). O artigo de 1967 de Tom Cover e Peter Hart sobre a taxa de erros do vizinho mais próximo é “Nearest neighbor pattern classification”* (*IEEE Transactions on Information Theory*).

A maldição da dimensionalidade é discutida na Seção 2.5 de *The Elements of Statistical Learning**, de Trevor Hastie, Rob Tibshirani e Jerry Friedman

(segunda edição, Springer, 2009). “Wrappers for feature subset selection”*, de Ron Kohavi e George John (*Artificial Intelligence*, 1997), compara métodos de seleção de atributos. “Similarity metric learning for a variable-kernel classifier”*, de David Lowe (*Neural Computation*, 1995), é um exemplo de um algoritmo de avaliação de características.

“Support vector machines and kernel methods: The new generation of learning machines”*, de Nello Cristianini e Bernhard Schölkopf (*AI Magazine*, 2002), é uma introdução em grande parte não matemática às SVMs. O artigo que começou a revolução das SVMs foi “A training algorithm for optimal margin classifiers”*, de Bernhard Boser, Isabel Guyon e Vladimir Vapnik (*Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, 1992). O primeiro artigo que aplicou as SVMs à classificação de texto foi “Text categorization with support vector machines”*, de Thorsten Joachims (*Proceedings of the Tenth European Conference on Machine Learning*, 1998). O Capítulo 5 de *An Introduction to Support Vector Machines*, de Nello Cristianini e John Shawe-Taylor (Cambridge University Press, 2000), é uma breve introdução à otimização restrita no contexto das SVMs.

Case-Based Reasoning, de Janet Kolodner (Morgan Kaufmann, 1993), é um livro didático sobre o assunto. “Using case-based retrieval for customer technical support”*, de Evangelos Simoudis (*IEEE Expert*, 1992), explica sua aplicação aos help desks. A funcionária virtual Eliza da IPsoft é descrita em “Rise of the software machines” (*Economist*, 2013) e no site da empresa. Kevin Ashley examina a raciocínio legal baseado em casos em *Modeling Legal Arguments** (MIT Press, 1991). David Cope resume sua abordagem da composição musical automatizada em “Recombinant music: Using the computer to explore musical style” (*IEEE Computer*, 1991). Dedre Gentner propôs o mapeamento de estruturas em “Structure mapping: A theoretical framework for analogy”* (*Cognitive Science*, 1983). “The man who would teach machines to think”, de James Somers (*Atlantic*, 2013), discute os pontos de vista de Douglas Hofstadter sobre inteligência artificial.

O algoritmo RISE é descrito em meu artigo “Unifying instance-based and rule-based induction”* (*Machine Learning*, 1996).

Capítulo 8

The Scientist in the Crib, de Alison Gopnik, Andy Meltzoff e Pat Kuhl (Harper,

1999), resume as descobertas dos psicólogos de como os bebês e as crianças mais jovens aprendem.

O algoritmo *k*-means foi originalmene proposto por Stuart Lloyd na Bell Labs em 1957, em um relatório técnico intitulado “Least squares quantization in PCM”* (que depois apareceu como artigo na *IEEE Transactions on Information Theory* em 1982). O artigo original sobre o algoritmo EM é “Maximum likelihood from incomplete data via the EM algorithm”*, de Arthur Dempster, Nan Laird e Donald Rubin (*Journal of the Royal Statistical Society B*, 1977). O clustering hierárquico e outros métodos são descritos em *Finding Groups in Data: An Introduction to Cluster Analysis*, de Leonard Kaufman e Peter Rousseeuw (Wiley, 1990).

A análise de componentes principais é uma das técnicas mais antigas de machine learning e estatística, tendo sido proposta por Karl Pearson em 1901 no artigo “On lines and planes of closest fit to systems of points in space”* (*Philosophical Magazine*). O tipo de redução de dimensionalidade usado para avaliar ensaios de exames SAT foi introduzido por Scott Deerwester e outros no artigo “Indexing by latent semantic analysis”* (*Journal of the American Society for Information Science*, 1990). Yehuda Koren, Robert Bell e Chris Volinsky explicam como funciona a filtragem colaborativa do tipo usado pelo Netflix em “Matrix factorization techniques for recommender systems”* (*IEEE Computer*, 2009). O algoritmo Isomap foi introduzido em “A global geometric framework for nonlinear dimensionality reduction”*, de Josh Tenenbaum, Vin de Silva e John Langford (*Science*, 2000).

Reinforcement Learning: An Introduction, de Rich Sutton e Andy Barto (MIT Press, 1998), é o livro-padrão sobre o assunto. *Universal Artificial Intelligence*, de Marcus Hutter (Springer, 2005), é uma tentativa de formular uma teoria geral do aprendizado por reforço. A pesquisa pioneira de Arthur Samuel sobre aprender a jogar damas é descrita em seu artigo “Some studies in machine learning using the game of checkers”* (*IBM Journal of Research and Development*, 1959). Esse artigo também marca uma das mais antigas aparições impressas do termo *machine learning*. A formulação de Chris Watkins do problema do aprendizado por reforço apareceu em sua tese de PhD *Learning from Delayed Rewards** (Cambridge University, 1989). O aprendiz da abordagem por reforço da DeepMind para videogames é descrito em “Human-level control through deep reinforcement learning”*, de Volodymyr Mnih e

outros (*Nature*, 2015).

Paul Rosenbloom reconta a história do desenvolvimento da técnica de chunking em "A cognitive odyssey: From the power law of practice to a general learning mechanism and beyond" (*Tutorials in Quantitative Methods for Psychology*, 2006). O teste A/B e outras técnicas de experimentação online são explicados em "Practical guide to controlled experiments on the Web: Listen to your customers not to the HiPPO"*, de by Ron Kohavi, Randal Henne e Dan Sommerfield (*Proceedings of the Thirteenth International Conference on Knowledge Discovery and Data Mining*, 2007). A uplift modeling, uma generalização multidimensional do teste A/B, é o assunto do Capítulo 7 de *Predictive Analytics* de Eric Siegel (Wiley, 2013).

Introduction to Statistical Relational Learning, editado por Lise Getoor e Ben Taskar (MIT Press, 2007), examina as principais abordagens dessa área. Meu trabalho com Matt Richardson sobre a modelagem boca a boca é resumido em "Mining social networks for viral marketing" (*IEEE Intelligent Systems*, 2005).

Capítulo 9

Ensemble Methods: Foundations and Algorithms, de Zhi-Hua Zhou (Chapman and Hall, 2012), é uma introdução ao meta-aprendizado. O artigo original sobre stacking é "Stacked generalization"*, de David Wolpert (*Neural Networks*, 1992). Leo Breiman introduziu o bagging em "Bagging predictors"* (*Machine Learning*, 1996) e as florestas aleatórias em "Random forests"* (*Machine Learning*, 2001). A técnica de boosting é descrita em "Experiments with a new boosting algorithm", de Yoav Freund e Rob Schapire (*Proceedings of the Thirteenth International Conference on Machine Learning*, 1996).

"I, Algorithm", de Anil Ananthaswamy (*New Scientist*, 2011), descreve o caminho para a combinação de lógica e probabilidade em inteligência artificial. *Markov Logic: An Interface Layer for Artificial Intelligence*, que escrevi em colaboração com Daniel Lowd (Morgan & Claypool, 2009), é uma introdução às redes lógicas de Markov. O site do Alchemy, http://alchemy.cs.washington.edu, também inclui tutoriais, vídeos, MLNs, conjuntos de dados, publicações, indicações de outros sistemas, e assim por diante. Uma MLN para o mapeamento de robôs é descrita em "Hybrid Markov logic networks"*, de Jue Wang e Pedro Domingos (*Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence*, 2008). Thomas Dietterich e Xinlong Bao descrevem o uso

de MLNs no projeto PAL da Darpa em “Integrating multiple learning components through Markov logic”* (*Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence*, 2008). “Extracting semantic networks from text via relational clustering”*, de Stanley Kok e Pedro Domingos (*Proceedings of the Nineteenth European Conference on Machine Learning*, 2008), descreve como usamos MLNs para aprender uma rede semântica a partir da web.

MLNs eficientes com classe hierárquica e estrutura de partes são descritas em “Learning and inference in tractable probabilistic knowledge bases”*, de Mathias Niepert e Pedro Domingos (*Proceedings of the Thirty-First Conference on Uncertainty in Artificial Intelligence*, 2015). A abordagem do Google para a descida de gradiente paralela é descrita em “Large-scale distributed deep networks”*, de Jeff Dean e outros (*Advances in Neural Information Processing Systems 25*, 2012). “A general framework for mining massive data streams”*, de Pedro Domingos e Geoff Hulten (*Journal of Computational and Graphical Statistics*, 2003), resume nosso método baseado em amostragem para o aprendizado a partir de fluxos de dados ilimitados. O projeto FuturICT é o assunto de “The machine that would predict the future”, de David Weinberger (*Scientific American*, 2011).

“Cancer: The march on malignancy” (suplemento da *Nature*, 2014) examina o estado atual da guerra contra o câncer. “Using patient data for personalized cancer treatments”, de Chris Edwards (*Communications of the ACM*, 2014), descreve os estágios iniciais do que poderia evoluir para o CanceRx. “Simulating a living cell”, de Markus Covert (*Scientific American*, 2014), explica como seu grupo construiu um modelo de computador de uma bactéria infecciosa inteira. “Breakthrough Technologies 2015: Internet of DNA”, de Antonio Regalado (*MIT Technology Review*, 2015), relata o trabalho da Aliança Global para Genômica e Saúde (Global Alliance for Genomics and Health). A Cancer Commons é descrita em “Cancer: A Computational Disease that AI Can Cure”, de Jay Tenenbaum e Jeff Shrager (*AI Magazine*, 2011).

Capítulo 10

“Love, actuarially”, de Kevin Poulsen (*Wired*, 2014), conta a história de como um homem usou o machine learning para encontrar uma companheira no site de encontros OkCupid. *Dataclisma*, de Christian Rudder (Best Seller, 2015),

minera dados do OkCupid em busca de diversos tipos de insights. *Total Recall*, de Gordon Moore e Jim Gemmell (Dutton, 2009), examina as implicações de gravarmos digitalmente tudo que fazemos. *The Naked Future*, de Patrick Tucker (Current, 2014), descreve o uso e o abuso de dados para fazermos previsões em nosso mundo. Craig Mundie defende uma abordagem balanceada para a coleta e o uso de dados em "Privacy pragmatism" (*Foreign Affairs*, 2014). *A segunda era das máquinas*, de Erik Brynjolfsson e Andrew McAfee (Alta Books, 2015), discute como o progresso em inteligência artificial moldará o futuro do trabalho e a economia. "World War R" de Chris Baraniuk (*New Scientist*, 2014) dá informações sobre o debate existente ao redor do uso de robôs em combates. "Transcending complacency on superintelligent machines", de Stephen Hawking e outros (*Huffington Post*, 2014), argumenta que é hora de nos preocuparmos com os riscos da inteligência artificial. *Superintelligence*, de Nick Bostrom (Oxford University Press, 2014), considera esses perigos e o que fazer com relação a eles.

A Brief History of Life, de Richard Hawking (Random Penguin, 1982), resume os saltos quânticos da evolução nas eras a.C. (Antes dos Computadores. Brincadeirinha.) *The Singularity Is Near*, de Ray Kurzweil (Penguin, 2005), é um guia para o futuro transumano. Joel Garreau considera três cenários diferentes para como a evolução direcionada por humanos se desvelará em *Radical Evolution* (Broadway Books, 2005). Em *What Technology Wants* (Penguin, 2010), Kevin Kelly argumenta que a tecnologia é a continuação da evolução por outros meios. *Darwin Among the Machines*, de George Dyson (Basic Books, 1997), descreve a evolução da tecnologia e especula para onde ela nos levará. Craig Venter explica como sua equipe sintetizou uma célula viva em *Life at the Speed of Light* (Viking, 2013).

---

1 N.T.: O conceito *one-stop shop* surgiu nos anos 1920 nos Estados Unidos e é utilizado para negócios que oferecem as mais diversas soluções para seus clientes.

PEDRO DOMINGOS é professor de ciência da computação na Universidade de Washington, em Seattle. Ganhou o SIGKDD Innovation Award, a maior honraria da área de ciência de dados. Ele vive em Seattle e é membro da Association for the Advancement of Artificial Intelligence.


CONSTRUINDO UMA
LOJA VIRTUAL
A jornada de uma empreendedora
em seu primeiro negócio online
novatec
André Gugliotti

Construindo uma loja virtual

Gugliotti, André 9788575224953
217 páginas Compre agora e leia

Juliana é uma arquiteta de sucesso, com estabilidade e um emprego com que sempre sonhou. No entanto, algo não vai bem e, em um dia de fúria, ela decide sacudir seu mundo, largar o emprego e abrir sua própria empresa.

No entanto, ela logo descobre que as coisas são mais complicadas do que parecem. Em sua jornada para ter sua primeira loja virtual, Juliana enfrenta diversos problemas e recebe ajuda de personagens emblemáticos como o Professor, um mestre na arte de empreender, e Pedro, o proprietário de uma loja virtual de calçados.

Em sua caminhada, ela aprende a planejar uma empresa, construir catálogos de produtos, cuidar das finanças e do marketing. Ela também descobre as matrioscas, bonecas russas de longa tradição e que serão vendidas em sua loja.

O que Juliana não sabe é que uma das matrioscas esconde um segredo que pode colocar sua vida e a de Pedro em perigo. Embarque nessa aventura, monte sua loja virtual e descubra o mistério da matriosca.


JOVEM E
BEM-SUCEDIDO
Um guia para a realização
profissional e financeira
TRABALHO
DINHEIRO
ESTUDOS
novatec
Juliano Niederauer

Jovem e Bem-sucedido

Niederauer, Juliano 9788575225325
192 páginas Compre agora e leia

Jovem e Bem-sucedido é um verdadeiro guia para quem deseja alcançar a realização profissional e a financeira o mais rápido possível. Repleto de dicas e histórias interessantes vivenciadas pelo autor, o livro desmistifica uma série de crenças relativas aos estudos, ao trabalho e ao dinheiro.

Tem como objetivo orientar o leitor a planejar sua vida desde cedo, possibilitando que se torne bem-sucedido em pouco tempo e consiga manter essa realização no decorrer dos anos. As três perspectivas abordadas são:

ESTUDOS: mostra que os estudos vão muito além da escola ou faculdade. Aborda as melhores práticas de estudo e a aquisição dos conhecimentos ideais e nos momentos certos.

TRABALHO: explica como você pode se tornar um profissional moderno, identificando oportunidades e aumentando cada vez mais suas fontes de renda. Fornece ainda dicas valiosas para desenvolver as habilidades mais valorizadas no mercado de trabalho.

DINHEIRO: explica como assumir o controle de suas finanças, para, então, começar a investir e multiplicar seu patrimônio. Apresenta estratégias de investimentos de acordo com o momento de vida de cada um, abordando as vantagens e desvantagens de cada tipo de investimento.

Jovem e Bem-sucedido apresenta ideias que o acompanharão a vida toda, realizando importantes mudanças no modo como você planeja estudar, trabalhar e lidar com o dinheiro.

e lidar com o dinheiro.


Tudo que você precisa saber para gerar negócios na maior rede social do mundo
Facebook
MARKETING
novatec
Camila Porto

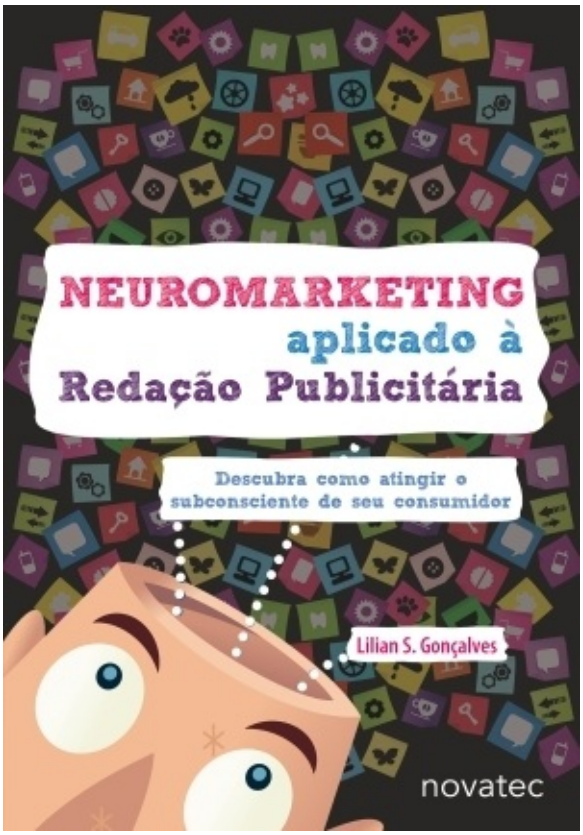
Facebook Marketing

Porto, Camila 9788575224977
360 páginas Compre agora e leia

Como a maior rede social do mundo pode se tornar uma peça-chave para o seu negócio? Como aumentar suas vendas, atrair e fidelizar clientes e conquistar fãs?

Estas e outras respostas você encontrará neste livro, além de tudo que precisa saber para utilizar o Facebook a fim de potencializar seu negócio e ter muito mais resultados. A partir de um conteúdo rico e objetivo, acompanhado de entrevistas exclusivas com profissionais renomados, você saberá como utilizar o Facebook desde a construção da sua estratégia, o que postar, como anunciar, como vender, como mensurar e como estar sempre onde seus clientes estão. Com um bilhão de usuários, o Facebook se tornou um dos canais mais importantes para os negócios, e você precisa saber o que ele é capaz de fazer pelo seu.

Neste livro você aprenderá:
- Como usar o Facebook para alavancar sua carreira;
- Como potencializar sua marca na maior rede social do mundo;
- Como produzir conteúdo e gerar engajamento;
- Como usar os anúncios no Facebook e conseguir mais visibilidade;
- Como integrar o Facebook ao seu site, e-commerce, blog de forma estratégica para vender mais;
- Como saber se suas ações estão gerando resultados, a partir da mensuração da sua presença no Facebook;
- Como gerenciar crises e transformá-las em oportunidades para o seu negócio.


NEUROMARKETING
aplicado à
Redação Publicitária
Descubra como atingir o subconsciente de seu consumidor
Lilian S. Gonçalves
novatec

Neuromarketing Aplicado à Redação Publicitária

Gonçalves, Lilian S.
9788575224991
192 páginas

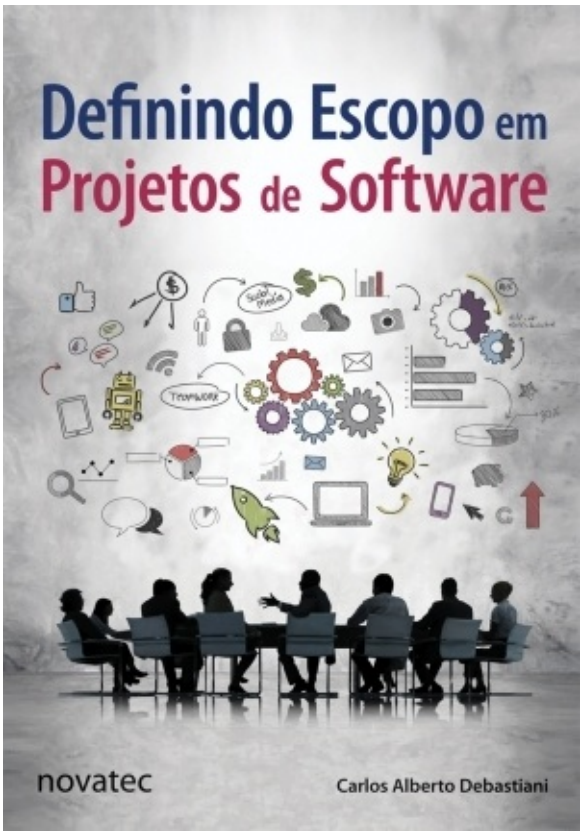
O neuromarketing é uma nova visão para a publicidade mundial e pode ser a chave para uma campanha de sucesso.

A versatilidade proporcionada pelo casamento entre as tradicionais pesquisas de mercado e as descobertas da neurociência se mostra um prato cheio para os redatores de plantão.

Esta obra reúne as principais descobertas feitas pelo neuromarketing nas últimas décadas e suas possíveis aplicações como argumentação para os anúncios publicitários.

Mais do que atingir seu público-alvo, chegou a hora de desenvolver campanhas de marketing com foco no subconsciente de seu consumidor, a fim de ampliar sua lista de clientes fidelizados.

Bom humor, criatividade, técnica e ciência estão de mãos dadas rumo a seu sucesso.


Definindo Escopo em
Projetos de Software
novatec
Carlos Alberto Debastiani

Definindo Escopo em Projetos de Software

Debastiani, Carlos Alberto 9788575224960
144 páginas Compre agora e leia

Definindo Escopo em Projetos de Software é uma obra que pretende tratar, de forma clara e direta, a definição de escopo como o fator mais influente no sucesso dos projetos de desenvolvimento de sistemas, uma vez que exerce forte impacto sobre seus custos. Abrange diversas áreas do conhecimento ligadas ao tema, abordando desde questões teóricas como a normatização e a definição das características de engenharia de software, até questões práticas como métodos para coleta de requisitos e ferramentas para desenho e projeto de soluções sistêmicas.

Utilizando uma linguagem acessível, diversas ilustrações e citações de casos vividos em sua própria experiência profissional, o autor explora, de forma abrangente, os detalhes que envolvem a definição de escopo, desde a identificação das melhores fontes de informação e dos envolvidos na tomada de decisão, até as técnicas e ferramentas usadas no levantamento de requisitos, no projeto da solução e nos testes de aplicação.