FICHA TÉCNICA

facebook.com/manuscritoeditora

Título original: *The Master Algorithm*
Autor: *Pedro Domingos*

Tradução: *Francisco Silva Pereira*
Revisão: *Gabriela Varino/Editorial Presença*
Capa: *Putt Studio*
Composição, impressão e acabamento: *Multitipo — Artes Gráficas, Lda.*

ISBN: 978-989-8871-18-3
Depósito legal n.º 431 086/17

1.ª edição, Lisboa, outubro, 2017

O grande objetivo da ciência é abranger o maior número de factos experimentais por dedução lógica a partir do menor número de hipóteses ou axiomas.

Albert Einstein

A civilização avança ao alargar o número de operações importantes que podemos efetuar sem pensar nelas.

Alfred North Whitehead

Índice

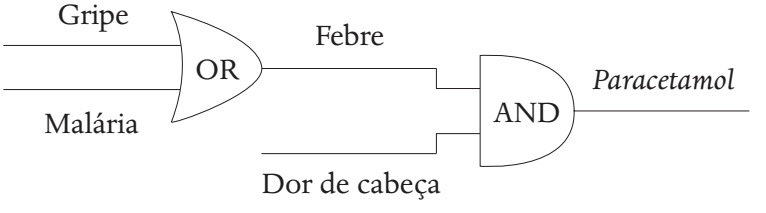
CAPÍTULO UM

A revolução da aprendizagem automática

Vivemos na era dos algoritmos. Há apenas uma ou duas gerações, a referência à palavra «algoritmo» teria deixado a maioria das pessoas completamente indiferente. Hoje, os algoritmos encontram-se em todos os recantos da civilização. Estão inseridos no tecido da vida quotidiana. Não apenas no nosso telemóvel ou computador portátil, mas também em nossa casa, nos nossos eletrodomésticos e brinquedos. O nosso banco é um emaranhado gigantesco de algoritmos, com seres humanos a mexer em botões aqui e acolá. Os algoritmos agendam voos e depois pilotam os aviões. Os algoritmos administram fábricas, comercializam e encaminham mercadorias, contabilizam os lucros e mantêm os registos de tudo isto. Se, de repente, todos os algoritmos deixassem de trabalhar, seria o fim do mundo como o conhecemos.

Um algoritmo é uma sequência de instruções que diz a um computador o que fazer. Os computadores são constituídos por milhares de milhões de pequenos interruptores chamados transístores, e os algoritmos ligam-nos e desligam-nos milhões de vezes por segundo. O algoritmo mais simples é: ligar um interruptor. O estado de um transístor é um *bit* de informação: «1» se o transístor está ligado, «0» se está desligado. Um *bit* algures nos computadores do nosso banco diz se a nossa conta está a descoberto ou não. Outro *bit* algures nos computadores da Segurança Social diz se estamos vivos ou mortos. O segundo algoritmo mais simples é: combinar dois *bits*. Claude Shannon, mais conhecido como o pai da teoria da informação, foi o primeiro a perceber que aquilo que os transístores estão a fazer, ao ligarem-se e desligarem-se em resposta a outros transístores, é raciocinar. (Foi esta a sua

dissertação de mestrado no MIT — a dissertação de mestrado mais importante de todos os tempos.) Se o transístor A se liga apenas quando os transístores B e C estão ambos ligados, então está a proceder a um ínfimo raciocínio lógico. Se A se liga quando B ou C estão ligados, então esta é outra ínfima operação lógica. E se A se liga sempre que B está desligado, e vice-versa, temos uma terceira operação. Acreditemos ou não, todos os algoritmos, por mais complexos que sejam, podem ser reduzidos a apenas estas três operações: AND, OR e NOT. Os algoritmos simples podem ser representados por diagramas, usando diferentes símbolos para as operações AND, OR e NOT. Por exemplo, se uma febre pode ser causada pela gripe ou pela malária, e se devemos tomar paracetamol para uma febre e para uma dor de cabeça, isto pode ser expresso da seguinte forma:



Ao combinar muitas operações como estas, podemos desenvolver cadeias de raciocínio lógico muito elaboradas. É frequente as pessoas pensarem que os computadores só lidam com números, mas não é verdade. Os computadores são lógica. Os números e a aritmética são feitos de lógica, assim como tudo o resto num computador. Queremos somar dois números? Existe uma combinação de transístores que faz isso. Queremos derrotar o campeão humano do concurso *Jeopardy!*? Também existe uma combinação de transístores para isso (muito maior, naturalmente).

Todavia, seria proibitivamente dispendioso se tivéssemos de construir um novo computador para todas as coisas diferentes que queremos fazer. Como tal, um computador moderno é um vasto conjunto de transístores que podem fazer muitas coisas diferentes, dependendo dos que são ativados. Miguel Ângelo disse que não fazia mais do que ver a estátua no interior do bloco de mármore e eliminar o excesso de pedra até que a estátua fosse revelada. Do mesmo modo, um algoritmo

remove os transístores excedentários no computador até que a função pretendida seja revelada, quer se trate do piloto automático de um avião comercial ou de um novo filme da Pixar.

Um algoritmo não é apenas um conjunto qualquer de instruções: estas têm de ser suficientemente precisas e não ambíguas para serem executadas por um computador. Por exemplo, uma receita de culinária não é um algoritmo porque não especifica exatamente por que ordem devemos fazer as coisas, nem explica exatamente o que é cada um dos passos a seguir. Uma colher cheia é exatamente quanto açúcar? Como o sabe qualquer um que já tenha experimentado uma receita nova, segui-la pode resultar em algo delicioso ou numa catástrofe. Em contraste, um algoritmo produz sempre o mesmo resultado. Mesmo se uma receita especificar precisamente meio quilo de açúcar, ainda não estamos safos porque o computador não sabe o que é o açúcar, nem o que é um quilo. Se quiséssemos programar um robô cozinheiro para fazer um bolo, teríamos de lhe dizer como reconhecer o açúcar a partir de um vídeo, como agarrar numa colher, e assim sucessivamente. (Ainda estamos a trabalhar nisso.) O computador tem de saber como executar o algoritmo durante todo o processo, até ao ponto de ligar e desligar transístores específicos. Como tal, uma receita de cozinha está muito longe de ser um algoritmo.

Por outro lado, temos abaixo um algoritmo para o jogo do galo:

Se o adversário tiver dois em linha, jogar no quadrado restante.
Senão, se existir uma jogada que crie duas linhas de dois, executá-la.
Senão, se o quadrado central estiver livre, ocupá-lo.
Senão, se o adversário ocupou um canto, ocupar o canto oposto.
Senão, se houver um canto livre, ocupá-lo.
Senão, ocupar qualquer quadrado livre.

Este algoritmo tem a simpática propriedade de nunca perder! É claro que ainda lhe faltam muitos pormenores, como a representação do tabuleiro na memória do computador e como esta representação é alterada por cada jogada. Por exemplo, podíamos ter 2 *bits* para cada quadrado, com o valor 00 se o quadrado estiver vazio, que passa a 01 se tiver um «O» e a 10 se tiver um «X». Mas é suficientemente preciso para que qualquer programador competente seja capaz de preencher o que está em falta. O facto de não termos realmente de especificar um

algoritmo até ao nível dos transístores individuais também ajuda: podemos usar algoritmos preexistentes como blocos de construção, e existe uma quantidade imensa por onde escolher.

Os algoritmos são um padrão exigente. É frequente dizer-se que não compreendemos realmente uma coisa antes de a conseguirmos expressar sob a forma de um algoritmo. (Como Richard Feynman disse: «O que não consigo criar, não compreendo.») As equações, o pão com manteiga dos físicos e engenheiros, não passam na realidade de um tipo especial de algoritmo. Por exemplo, a segunda lei de Newton, talvez a equação mais importante de todos os tempos, diz-nos para calcular a força resultante que atua sobre um corpo multiplicando a massa do mesmo pela sua aceleração. Também nos diz implicitamente que a aceleração é a força dividida pela massa, mas explicitar isto é, em si, um passo algorítmico. Em qualquer área da ciência, se uma teoria não pode ser expressa como algoritmo, não é totalmente rigorosa. (E além disso não podemos usar um computador para a resolver, o que limita severamente o que podemos fazer com ela.) Os cientistas criam teorias e os engenheiros criam dispositivos. Os engenheiros informáticos criam algoritmos, que são simultaneamente teorias e dispositivos.

Conceber um algoritmo não é fácil. As armadilhas abundam, e nada pode ser tomado como certo. Algumas das nossas intuições acabam por se revelar erradas, e temos de procurar outro caminho. Além de conceber um algoritmo, temos de o escrever numa linguagem que os computadores consigam entender, como Java ou Python (altura em que passa a chamar-se um programa). A seguir temos de o depurar: descobrir todos os erros e corrigi-los até que o computador corra o nosso programa sem fazer asneira. Mas assim que temos um programa que faz o que queremos, estamos à vontade. Os computadores fazem o que mandamos milhões de vezes, a uma velocidade ultrarrápida, sem se queixar. Qualquer pessoa no mundo pode usar a nossa criação. O custo pode ser nulo, se quisermos, ou o suficiente para nos tornar milionários, se o problema que resolvemos for suficientemente importante. Um programador — alguém que cria algoritmos e os codifica — é um deus menor que cria universos à sua vontade. Até poderíamos dizer que o próprio Deus do Génesis é um programador: a linguagem, e não a manipulação, é a ferramenta da criação. Palavras tornam-se mundos. Hoje em dia, sentados no sofá com o nosso computador portátil, também nós podemos ser deuses. Imagina um universo e torna-o real. As leis da física são opcionais.