Anil Ananthaswamy

Proč se stroje učí

Elegantní matematika, na které je založena AI

VYŠEHRAD

Proč se stroje učí

Vyšlo také v tištěné verzi

Objednat můžete na
www.ivysehrad.cz
www.albatrosmedia.cz

Anil Ananthaswamy
Proč se stroje učí – e-kniha

ALBATROS MEDIA

Anil Ananthaswamy

Proč se stroje učí

Elegantní matematika, na které je založena AI

Anil Ananthaswamy

Proč se stroje učí

Elegantní matematika, na které je založena AI

Přeložil Ivo Magera

VYŠEHRAD

Autor srdečně děkuje Nadaci Alfreda P. Sloana
za podporu při studiu materiálů a psaní této knihy.

ISBN tištěné verze 978-80-267-3052-1
ISBN e-knihy 978-80-267-3053-8 (1. zveřejnění, 2024) (ePDF)

Všem učitelům, opěvovaným i neopěvovaným

Ať už děláme cokoli, musíme si vytvořit životní vektory.
Čáry o síle a směru.

Liam Neeson jako agent FBI Mark Felt
ve stejnojmenném filmu z roku 2017

Obsah

Úvod 11
1. Zoufale hledáme vzory 17
2. Všichni jsme tu jen čísla 34
3. Dno misky 67
4. Se vší pravděpodobností 95
5. Vrána k vráně 145
6. Matice mají své kouzlo 176
7. Velký jádrový trik 205
8. S trochou dopomoci fyziky 239
9. Muž, který zpomalil hluboké učení (ne tak docela) 271
10. Algoritmus, který dal vale zažitému mýtu 296
11. Očima stroje 335
12. Neprobádané území 370

Závěr 401

Poděkování 417
Errata 419
Slovníček pojmů 420
Poznámky 424

Úvod

Na pětadvacáté straně *The New York Times* z 8. července 1958 se ukrýval poněkud neobvyklý příběh.[1] Titulek zněl: „NOVÝ PŘÍSTROJ NÁMOŘNÍCH SIL SE UČÍ VLASTNÍ ČINNOSTÍ: Psycholog představil zárodek počítače navrženého tak, aby četl a rozvíjel se." Úvodní odstavec ještě přihazoval: „Námořnictvo dnes odhalilo zárodek elektronického počítače, od něhož očekává, že bude schopen chodit, mluvit, vidět, psát, rozmnožovat se a uvědomovat si svou existenci."

Při zpětném pohledu je zřejmé, že šlo o úsměvné zveličení. *New York Times* za to však úplně nemohly. Část přehnaných tvrzení pocházela také od Franka Rosenblatta, psychologa a projektového inženýra z Cornellovy univerzity.[2] Rosenblatt s finanční podporou amerického Úřadu pro námořní výzkum[3] vynalezl perceptron, jehož verzi představil na tiskové konferenci den předtím, než o něm v *New York Times* vyšel článek. Podle Rosenblatta měl být perceptron „prvním zařízením, které bude myslet jako lidský mozek", a podobné přístroje by mohly být vysílány na jiné planety jako „mechaničtí průzkumníci vesmíru".

Nic z toho se neuskutečnilo. Perceptron nenaplnil očekávání, která do něj byla vkládána. Přesto byla Rosenblattova práce zásadní. Téměř každý přednášející o umělé inteligenci (AI) se dnes odvolává na perceptron. A oprávněně. Tento historický okamžik – nástup rozsáhlých jazykových modelů (large language models, LLM), jako je ChatGPT a jemu podobné, a naše reakce na ně – který někteří přirovnávají k tomu, jak si museli připadat fyzikové v desátých a dvacátých letech minulého století, když byli

svědky pobláznění okolo kvantové mechaniky, má své kořeny ve výzkumu iniciovaném právě Rosenblattem. Ve zmiňovaném článku stojí věta, která naznačuje revoluci, již perceptron vyvolal: „Dr. Rosenblatt řekl, že *proč se stroj učí*, dokáže vysvětlit pouze ve značně odborných pojmech." (kurzíva je moje)[4] V článku však žádné „značně odborné" detaily nebyly.

A přesně to chce tato kniha napravit. Zabývá se odbornými detaily. Vysvětluje elegantní matematický aparát a algoritmy, které již desítky let vzrušují badatele v oblasti strojového učení, což je druh umělé inteligence, jehož součástí je vytváření strojů, které se dokážou naučit rozeznávat v datech vzory, aniž by k tomu byly výslovně naprogramovány. Takto natrénované stroje pak dokážou odhalit podobné vzory v nových datech, s nimiž se předtím nesetkaly, díky čemuž mohou vznikat aplikace sahající od rozpoznávání obrázků koček a psů až po vývoj potenciálně autonomních automobilů a další technologie. Stroje se dokážou učit díky mimořádnému propojení matematiky a informatiky s příměsí fyziky a neurovědy.

Strojové učení (machine learning, ML) je rozsáhlý obor plný algoritmů, které využívají relativně jednoduchou matematiku starou několik staletí, matematiku, kterou se učíme na střední škole nebo na začátku univerzitního studia. Nechybí samozřejmě ani elementární algebra. Dalším nesmírně důležitým základním kamenem strojového učení je diferenciální počet, jehož spoluvynálezcem nebyl nikdo menší než slavný učenec Isaac Newton. Tento obor se také do značné míry opírá o dílo Thomase Bayese,[5] anglického statistika a duchovního z 18. století, který nás obohatil o jeho jméno nesoucí Bayesův teorém, zásadní přínos v oblasti pravděpodobnosti a statistiky. Do strojového učení rovněž proniká práce německého matematika Carla Friedricha Gausse[6] o Gaussově rozdělení (a křivce zvonovitého tvaru). Dále je tu lineární algebra, tvořící páteř strojového učení. Nejstarší výklad této matematické disciplíny se objevuje již ve 2000 let starém čínském textu *Matematika v devíti kapitolách*.[7] Moderní podoba lineární algebry má kořeny v dílech řady matematiků, především však

Gausse, Gottfrieda Wilhelma Leibnize, Wilhelma Jordana, Gabriela Cramera, Hermanna Günthera Grassmanna, Jamese Josepha Sylvestera a Arthura Cayleyho.

Již v polovině 50. let 19. století existovaly některé základní matematické poznatky, které se ukázaly jako nezbytné pro konstrukci učících se strojů, zatímco další matematici pokračovali v rozvíjení relevantnější matematiky, z níž se později zrodila a vyvinula informatika. Málokdo však snil o tom, že tyto rané matematické práce budou základem ohromujícího vývoje umělé inteligence v posledním půlstoletí, zejména v posledním desetiletí, díky některým z nichž si můžeme oprávněně představit určitou podobu budoucnosti, kterou Rosenblatt v 50. letech 20. století příliš optimisticky předjímal.

Tato kniha pojednává o cestě od Rosenblattova perceptronu až po moderní hluboké neuronové sítě, což jsou propracované sítě výpočetních jednotek zvaných umělé neurony, a to vše optikou klíčových matematických principů, které jsou oporou oboru strojového učení. Kniha do matematiky zlehka vplouvá, pozvolně však stále svou obtížnost zvyšuje, protože se budeme dostávat od relativně jednoduchých základů z 50. let 20. století k poněkud složitější matematice a algoritmům, které pohánějí dnešní systémy strojového učení.

Proto si bez ostychu osvojíme rovnice a pojmy alespoň ze čtyř hlavních oblastí matematiky – lineární algebry, diferenciálního počtu, teorie pravděpodobnosti, statistiky a teorie optimalizace – získáme tak minimální teoretické znalosti a pojmovou výbavu nezbytné k tomu, abychom dokázali ocenit úžasné schopnosti, které strojům propůjčujeme. Teprve až pochopíme zákonitosti učících se strojů, budeme připraveni vyrovnat se s budoucností, v níž bude umělá inteligence všudypřítomná, ať už v dobrém, nebo ve zlém.

Proniknout pod matematickou slupku strojového učení je zásadní pro pochopení nejen síly této technologie, ale také jejích omezení. Systémy strojového učení za nás již nyní činí rozhodnutí, která nám mění život: schvalují žádosti o kreditní karty

a hypoteční úvěry, posuzují, zda je rakovinný nádor zhoubný, odhadují prognózu člověka, jemuž klesají kognitivní schopnosti (bude mít Alzheimerovu chorobu?), či rozhodují, zda někoho propustit na kauci. Strojové učení proniklo i do vědy: ovlivňuje chemii, biologii, fyziku i vše mezi tím. Používá se při studiu genomů, planet mimo Sluneční soustavu, složitostí kvantových systémů a mnohého dalšího. A v době psaní této knihy žije svět umělé inteligence nástupem rozsáhlých jazykových modelů, jako je například ChatGPT. Tento proces se přitom teprve rozbíhá.

Rozhodování o tom, jak bude umělá inteligence vytvářena a aplikována, nemůžeme nechat pouze na profesionálech, kteří ji vyvíjejí. Chceme-li tuto nesmírně užitečnou, avšak převratnou a potenciálně nebezpečnou technologii účinně regulovat, musí existovat další společenská vrstva – pedagogové, politici, zákonodárci, vědečtí publicisté, nebo dokonce zainteresovaní spotřebitelé umělé inteligence – obeznámená se základy matematiky strojového učení.

Matematička Eugenia Chengová ve své knize *Is Math Real?* (Je matematika skutečná?) píše o tom, že učení se matematice je postupný proces: „Může se zdát, že děláme velmi drobné krůčky a nikam se neposouváme, až se najednou ohlédneme a zjistíme, že jsme vylezli na obrovskou horu. To vše může být nepříjemné, ale podstoupit trochu intelektuálního nepohodlí (nebo někdy i hodně nepohodlí) je důležitou součástí zdokonalování se v matematice.“[8]

Naštěstí je „intelektuální nepohodlí“, které nás v knize čeká, velmi dobře snesitelné a bude více než zmírněno naším intelektuálním vítězstvím, protože matematika v pozadí moderního strojového učení je relativně jednoduchá a elegantní – což nejlépe ilustruje následující vzpomínka Ilji Sutskevera. Ten je dnes znám především jako spoluzakladatel společnosti OpenAI, která stojí za projektem ChatGPT. Před více než desetiletím hledal Sutskever coby vysokoškolský student akademického poradce na Torontské univerzitě a zaklepal na dveře Geoffreyho Hintona, s nímž chtěl začít spolupracovat. Hinton byl již tehdy známým jménem

v oblasti hlubokého učení, jedné z forem strojového učení, a dal Sutskeverovi k přečtení několik článků. Sutskever je doslova zhltl a vzpomíná, že byl zaskočen jednoduchostí použité matematiky v porovnání s matematikou a fyzikou v jeho obvyklých vysokoškolských kurzech. Přečetl si tyto dokumenty o hlubokém učení a hned pochopil důležité principy. „Jak je možné, že je to tak jednoduché … tak jednoduché, že to můžete bez větší námahy vysvětlit středoškolákům?" řekl mi. „Domnívám se, že je to vlastně zázrak. Je to pro mne také známkou toho, že jsme pravděpodobně na správné cestě. Nemůže jít o náhodu, že tak jednoduché principy mají takový dosah."[9]

Sutskever už byl samozřejmě v matematice poměrně pokročilý, takže to, co se zdálo jednoduché jemu, nemusí být jednoduché pro většinu z nás, včetně mne. Ale uvidíme. Cílem této knihy je přiblížit vám jednoduchost základů, na nichž stojí strojové a hluboké učení.

To neznamená, že vše, čeho jsme nyní v umělé inteligenci svědky – zejména chování hlubokých neuronových sítí a rozsáhlých jazykových modelů –, lze analyzovat jednoduchou matematikou. Ve skutečnosti nás vyústění této knihy přivádí do situace, která může někomu připadat znepokojivá, pro jiné však bude vzrušující: zdá se, že tyto sítě a umělé inteligence se vzpírají některým základním myšlenkám, na nichž je strojové učení již desítky let založeno. Jako by empirické důkazy zlomily vaz teoretickému velbloudovi, podobně jako experimentální pozorování hmotného světa na počátku 20. století rozbouralo klasickou fyziku; potřebujeme něco nového, abychom pochopili odvážný nový svět, který nás čeká.

Při přípravě materiálu pro tuto knihu jsem vypozoroval, že se učím podle vzorce, který mi připomíná způsob, jakým se učí moderní umělé neuronové sítě: při každém průchodu daty se algoritmus dozvídá více o vzorech, které v těchto datech existují. Jeden průchod nemusí stačit, ani deset, ani sto. Někdy se neuronové sítě učí v průběhu desítek tisíc průchodů daty. Tímto způsobem jsem si skutečně osvojil téma knihy tak, abych byl o něm schopen psát. S každým průchodem určitým koutem této obrovské báze znalostí vytvořily některé neurony v mém mozku spojení, doslova

i obrazně řečeno. Věci, které napoprvé nebo napodruhé nedávaly smysl, se nakonec při dalších průchodech propojily.

Tuto techniku jsem použil, abych pomohl čtenářům vytvořit podobná spojení: zjistil jsem, že v průběhu psaní knihy opakuji myšlenky a pojmy, někdy stejné formulace, jindy jsou stejné pojmy jinak pojaty. Tato opakování a reformulace jsou záměrné: jsou jednou z možností, jak se většina z nás, kteří nejsme matematici ani odborníci na strojové učení, může vyrovnat s paradoxně jednoduchým, přesto však složitým tématem. Náš mozek může při pozdějším setkání s předkládanými myšlenkami spatřovat v těchto myšlenkách vzory a souvislosti a přisuzovat jim hlubší smysl, než by dokázal na první pohled.

Doufám, že si vaše neurony tento proces užijí stejně jako ty mé.

Kapitola 1

Zoufale hledáme vzory

Rakouský vědec Konrad Lorenz, nadšený příběhy z knihy *Podivuhodná cesta Nilse Holgerssona* od švédské spisovatelky a nositelky Nobelovy ceny za literaturu Selmy Lagerlöfové, vyprávějící o dobrodružstvích chlapce, která prožil s divokými husami, se v dětství „toužil stát divokou husou".[10] Tohle přání se mu pochopitelně nesplnilo, a tak se spokojil s péčí o jednodenní kachní mládě, které získal od souseda. K chlapcově radosti ho kachňátko začalo pronásledovat: vtisklo si ho do paměti. Vtištění (imprinting) označuje schopnost mnoha zvířat, včetně kachňat a housat, vytvořit si po vylíhnutí pouto k první pohybující se věci, kterou uvidí. Lorenz se později stal etologem a průkopníkem studií v oblasti chování zvířat, zejména vtištění (dosáhl toho, že se kachňatům vtiskl do paměti, a ta ho následovala, když chodil, běhal, plaval, a dokonce i když odplouval na kánoi).[11] Lorenz získal v roce 1973 společně s kolegy etology Karlem von Frischem a Nikolaasem Tinbergenem Nobelovu cenu za fyziologii a lékařství. Všichni tři byli oceněni „za objevy týkající se uspořádání a vyvolání individuálních a sociálních *vzorů* chování".[12]

Vzory. Zatímco etologové rozpoznávali vzorce či vzory v chování zvířat, zvířata objevovala své vlastní. Nově vylíhlá kachňata mají schopnost rozeznat nebo rozlišit vlastnosti věcí, které se kolem nich pohybují. Ukázalo se, že kachňata si mohou vtisknout do paměti nejen prvního živého tvora, kterého uvidí v pohybu, ale i neživé věci. Například mláďata kachen divokých si mohou vtisknout do paměti dvojice pohybujících se předmětů, které mají podobný tvar nebo barvu. Přesněji řečeno, vtiskují si princip

vztahu těchto dvou předmětů.[13] Pokud tedy kachňata po narození vidí dva pohybující se červené objekty, budou se později taktéž držet dvou objektů stejné barvy (i když jsou modré, nikoli červené), nikoli dvou objektů různých barev.[14] V tomto případě si kachňata vtiskují představu podobnosti. Prokazují také schopnost rozeznat nepodobnost. Budou-li první pohybující se objekty, které kachňátka uvidí, například krychle a hranol obdélníkového tvaru, rozpoznají, že objekty jsou odlišného tvaru, a i později se budou držet dvou objektů, které mají odlišný tvar (například jehlan a kužel), a budou ignorovat dva objekty se stejným tvarem.

Nad tím se na chvilku zamysleme. Novorozená kachňata již po krátkém kontaktu se smyslovými podněty rozpoznávají vzory v tom, co vidí, vytvářejí si abstraktní představy o podobnosti versus nepodobnosti a tyto abstrakce pak rozeznávají v podnětech, které vidí později, a podle nich taky jednají. Badatelé v oblasti umělé inteligence by dali jmění, aby se dozvěděli, jak to kachňata dělají.

Ačkoli dnešní umělá inteligence není zdaleka schopna provádět takové úkony s lehkostí a efektivitou kachňat, mají něco společného. A to je schopnost rozpoznávat v datech vzory a z nich se učit. Když Frank Rosenblatt koncem 50. let 20. století vynalezl perceptron, bylo jedním z důvodů, proč vyvolal takový ohlas, to, že se jednalo o první impozantní algoritmus „inspirovaný mozkem", který dokázal poznávat v datech vzory pouhým zkoumáním těchto dat. A co je nejdůležitější, za určitých předpokladů týkajících se dat, vědci dokázali, že Rosenblattův perceptron najde vzor skrytý v datech vždy v konečném čase, čili jinak řečeno, perceptron bude bezchybně konvergovat k řešení. Takové jistoty jsou v informatice učiněným klenotem. Není divu, že algoritmus učení perceptronu vyvolal takový rozruch.

Co však tyto pojmy znamenají? Co jsou to vzory v datech? Co rozpoznávání těchto vzorů znamená? Začněme zkoumáním této tabulky:

x1	**x2**	**y**
4	2	8
1	2	5
0	5	10
2	1	4

Každý řádek tabulky představuje trojici hodnot proměnných *x1*, *x2* a *y*. V těchto datech se skrývá jednoduchý vzorec: v každém řádku je hodnota *y* v určitém vztahu k odpovídajícím hodnotám *x1* a *x2*. Než budete číst dále, zkuste ho odhalit.

V tomto případě lze s tužkou, papírem a trochou snahy zjistit, že *y* se rovná *x1* plus dvakrát *x2*.

$$y = x1 + 2x2$$

Drobná poznámka k formě zápisu: v tomto případě se obejdeme bez znaménka násobení („×“) mezi dvěma proměnnými nebo mezi konstantou a proměnnou. Například budeme psát

$$2 \times x2 \text{ jako } 2x2 \text{ a } x1 \times x2 \text{ jako } x1x2$$

V ideálním případě bychom měli psát $2x2$ jako $2x_2$ a $x1x2$ jako x_1x_2, kdy proměnné by měly být opatřeny indexem. My se však obejdeme i bez dolních indexů, nebude-li nezbytně nutné je použít (puristé se budou pohoršovat, pomůže to však udržet náš text přehlednější a snadno vnímatelný očima). Mějte tedy na paměti: pokud za symbolem, například „*x*“, následuje číslice, například 2, čímž získáme *x2*, berte celý symbol jako jednu věc. Pokud symbolu (řekněme *x* nebo *x2*) předchází číslo, řekněme 9, nebo jiný symbol, řekněme *w1*, pak se číslo a symbol, respektive oba symboly, spolu násobí. Tedy:

$$2x2 = 2 \times x2$$

$$x1x2 = x1 \times x2$$

$$w2x1 = w2 \times x1$$

Vrátíme-li se k naší rovnici, $y = x1 + 2x2$, můžeme ji obecněji zapsat následovně:

$$y = w1x1 + w2x2, \text{ kde } w1 = 1 \text{ a } w2 = 2$$

Pro ujasnění, našli jsme jeden z mnoha možných vztahů mezi *y* a *x1* a *x2*. Mohou existovat i další. A pro tento příklad skutečně existují, pro naše účely se jimi zde však nemusíme zabývat. Hledání vzorů není zdaleka tak jednoduché, jak naznačuje tento příklad, ale navede nás k němu.

Identifikovali jsme tzv. lineární vztah mezi *y* na jedné straně a *x1* a *x2* na straně druhé („lineární" znamená, že *y* závisí pouze na *x1* a *x2*, nikoli na *x1* nebo *x2* umocněném na nějaký exponent nebo na nějakém součinu *x1* a *x2*). Slova „rovnice" a „vztah" zde pak používám zaměnitelně.

Vztah mezi *y*, *x1* a *x2* je dále určen konstantami *w1* a *w2*. Tyto konstanty se nazývají koeficienty neboli váhy lineární rovnice upravující vztah *y* k *x1* a *x2*. V tomto jednoduchém případě nám k nalezení hodnot *w1* a *w2* stačilo se nad daty zamyslet. Často však vztah mezi *y* a (*x1*, *x2*, …) není tak zjevný, zvláště je-li na pravé straně rovnice více hodnot.

Vezměme si například:

$$y = w1x1 + w2x2 + w3x3 + \ldots + w9x9$$

Nebo obecněji pro množinu *n* vah a formálním matematickým zápisem:

$$y = w1x1 + w2x2 + w3x3 + \ldots + wnxn = \sum_{i=1}^{n} wixi$$

Výraz vpravo, se symbolem sumy, je zkráceným zápisem součtu všech *wixi*, kde *i* nabývá hodnot od 1 do *n*.

V případě devíti vstupů byste hodnoty *w1* až *w9* stěží určili pouhým pohledem na data a výpočty zpaměti. A právě tady přichází na řadu učení. Existuje-li způsob, jak váhy zjistit algoritmicky, pak se daný algoritmus váhy „učí". K čemu je to však dobré?

No, jakmile se naučíte váhy – řekněme *w1* a *w2* v našem cvičném příkladu – pak při zadání určité hodnoty *x1* a *x2*, která v našem

původním datovém souboru nebyla, dokážeme vypočítat hodnotu *y*. Řekněme, že *x1* = 5 a *x2* = 2. Tyto hodnoty dosadíme do rovnice $y = x1 + 2x2$, čímž získáme hodnotu $y = 9$.

Co to má společného s reálným světem? Vezměme si velmi jednoduchý, praktický a někdo by řekl naprosto nudný problém. Řekněme, že *x1* představuje počet obytných místností v domě, *x2* celkovou plochu a *y* cenu domu. Předpokládejme, že existuje lineární vztah mezi (*x1*, *x2*) a *y*. Pak tím, že jsme se „naučili" váhy lineární rovnice z určitých dostupných údajů o domech a jejich cenách, jsme v podstatě vytvořili velmi jednoduchý model, pomocí nějž lze predikovat cenu domu při daném počtu pokojů a podlahové ploše.

Výše uvedený příklad – jako opravdu jen malý krůček – je začátkem strojového učení. To, co jsme právě provedli, je zjednodušená forma toho, čemu se říká řízené učení (též učení s učitelem, supervised learning). Dostali jsme vzorky dat, které v sobě skrývaly určitou korelaci mezi množinou vstupů a množinou výstupů. O takových datech se říká, že jsou označená nebo popsaná; říká se jim také trénovací data. Ke každému vstupu (*x1*, *x2*, …, *xn*) je připojena značka neboli popisek *y*. V naší předchozí číselné tabulce je tedy dvojice čísel (4, 2) označena $y = 8$, dvojice (1, 2) popiskem 5 atd. Našli jsme korelaci. Jakmile na ni přijdeme („naučíme" se ji), můžeme díky ní předpovídat hodnoty k novým vstupům, které nebyly součástí trénovacích dat.

Prováděli jsme přitom velmi specifický typ řešení problémů zvaný regrese, kdy jsme pro určité dané nezávislé proměnné (*x1*, *x2*) sestavili model (rovnici) pro predikci hodnoty závislé proměnné (*y*). Existuje řada dalších typů modelů, které jsme mohli sestavit, a ve vhodnou chvíli se k nim také dostaneme.

V tomto případě byla korelace neboli vzor tak jednoduchá, že nám stačilo malé množství označených dat. Moderní strojové učení jich však vyžaduje řádově více – a dostupnost takových dat byla jedním z faktorů, které podnítily revoluci v umělé inteligenci. (Oproti tomu jsou kachňata pravděpodobně obdařena sofistikovanější formou učení. Žádná kachna neseděla a neoznačovala data

pro svá káčata, a přesto se její mláďata učí. Jak to dělají? Spoiler: to nevíme, ale možná tím, že pochopíme, *proč* se učí stroje, budeme moci jednoho dne plně porozumět tomu, jak se učí kachňata a vlastně i lidé).

Může se to zdát neuvěřitelné, ale tento první krok, který jsme učinili na značně jednoduchém příkladu řízeného učení, nás přivádí na cestu k pochopení moderních hlubokých neuronových sítí, samozřejmě krok po kroku (s malými, jemnými – a občas možná ani ne tak jemnými – doušky vektorů, matic, lineární algebry, diferenciálního počtu, teorie pravděpodobnosti, statistiky a teorie optimalizace, s nimiž budeme průběžně podle potřeby pracovat).

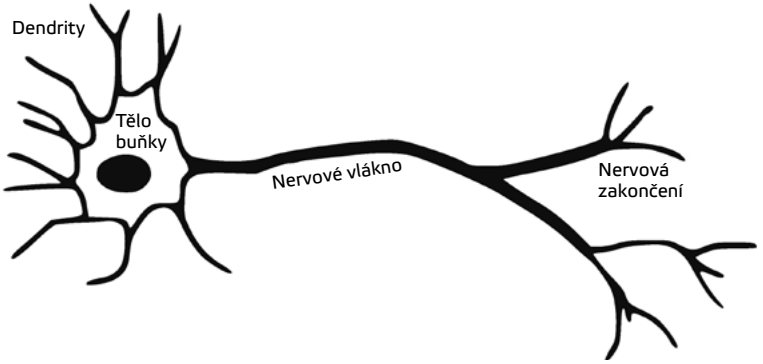
Příkladem jednoho takového učebního algoritmu, na svou dobu úžasného, byl Rosenblattův perceptron, s nímž jsme se krátce setkali v úvodu. A protože byl namodelován podle vzoru, jakým podle neurologů fungují lidské neurony, byl opředen tajemnem a příslibem, že perceptrony jednou skutečně naplní očekávání kladená na umělou inteligenci.

První umělý neuron

Kořeny perceptronu sahají k článku z roku 1943, který publikovala nesourodá autorská dvojice: filozoficky založený neurovědec, čtyřicátník, a teenager bez domova. Warren McCulloch[15] byl americký neurofyziolog, který vystudoval filozofii, psychologii a medicínu. Ve 30. letech 20. století se zabýval neuroanatomií, kdy sestavoval mapy propojení částí opičích mozků. Byl při tom posedlý i „logikou mozku".[16] V té době již práce matematiků a filozofů, jako byli Alan Turing, Alfred North Whitehead a Bertrand Russell, naznačovaly hlubokou souvislost mezi počítáním a logikou. Příkladem logické věty je výrok: „Je-li P pravda a zároveň Q je pravda, pak také S je pravda." Tvrdil, že na tuto logiku lze redukovat veškeré výpočty.[17] Při takovém nahlížení na proces počítání bylo otázkou, která McCullocha trápila: je-li mozek výpočetním zařízením, jak se mnozí domnívají, jak takovou logiku realizuje?

S těmito otázkami v hlavě se McCulloch v roce 1941 přemístil z Yaleovy univerzity na Illinoiskou, kde se seznámil s mimořádně nadaným teenagerem jménem Walter Pitts. Tento mladík, již uznávaný logik („chráněnec významného matematického logika Rudolfa Carnapa"),[18] navštěvoval semináře, které v Chicagu vedl ukrajinský matematik a fyzik Nicolas Raševskij.[19] Pitts byl však současně „rozpolceným adolescentem, v podstatě uprchlíkem od rodiny, která nedokázala docenit jeho genialitu".[20] McCulloch se svou ženou Rook poskytli Walterovi domov. „Následovaly nekonečné večery, kdy seděli u kuchyňského stolu u McCullochových a snažili se přijít na to, jak funguje mozek, přičemž dcera McCullochových Taffy kreslila náčrty," napsal přední vědec v oblasti informatiky Michael Arbib.[21] Taffyinými obrázky byla později ilustrována práce *A Logical Calculus of the Ideas Immanent in Nervous Activity* (Logický počet pro myšlenky spjaté s nervovou činností) z roku 1943.[22]

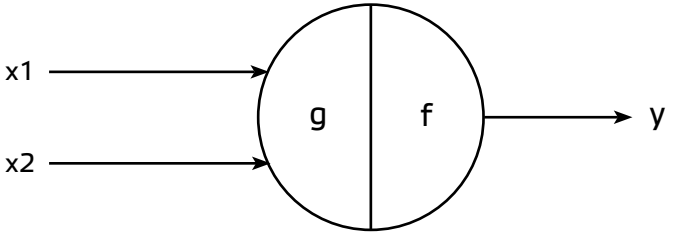
V ní McCulloch s Pittsem navrhli jednoduchý model biologického neuronu. Nejprve uvádíme ilustraci obecného biologického neuronu (autorství viz poznámky):[23]



Tělo buňky neuronu přijímá vstupy prostřednictvím stromovitých výběžků zvaných dendrity. Tělo neuronu provádí na těchto vstupech určité výpočty a na základě jejich výsledků může vysílat elektrický signál šířící se dalším delším výběžkem zvaným nervové vlákno. Signál putuje nervovým vláknem, až dosáhne jeho

rozvětvených zakončení, odkud je předán dendritům sousedních neuronů. A tak to pokračuje dál a dál. Takto propojené neurony tvoří biologickou neuronovou síť.

Z toho vytvořili McCulloch s Pittsem jednoduchý výpočetní model, umělý neuron.[24] Ukázali, jak lze pomocí jednoho takového umělého neuronu neboli neurodu (neuron + node, tj. anglicky uzel) realizovat některé základní logické operace typu AND, OR, NOT atd., které jsou základními stavebními kameny digitálních výpočtů (pro některé booleovské operace, jako je exkluzivní OR neboli XOR, je zapotřebí více než jednoho neuronu, ale o tom až později). Následuje názorný obrázek jednoho neurodu; písmena „*g*" a „*f*" uvnitř neuronu prozatím ignorujte, za okamžik se k nim vrátíme:



V této jednoduché verzi McCullochova-Pittsova modelu mohou být hodnoty *x1* a *x2* buď 0 nebo 1. Formální notací můžeme říci:

$$x1, x2 \in \{0,1\}$$

To čteme následovně: *x1* je prvkem množiny {0, 1}, *x2* je prvkem množiny {0, 1}; *x1* a *x2* tedy mohou nabývat pouze hodnot 0 nebo 1, žádných jiných. Výstup *y* z neurodu se vypočte tak, že nejprve sečteme vstupy a pak ověříme, zda je tento součet větší nebo roven určité prahové hodnotě *theta* (θ). Pokud ano, je *y* rovno 1; pokud ne, je *y* rovno 0.

$$suma = x1 + x2$$

$$Pokud\ suma \geq \theta: y = 1$$

$$Jinak: y = 0$$

Zobecníme-li tento postup na libovolnou posloupnost vstupů *x1*, *x2*, *x3*, ..., *xn*, můžeme zapsat formální matematický popis jednoduchého neurodu. Nejprve definujeme funkci $g(x)$, kde x je množina vstupů ($x1, x2, x3, \ldots, xn$) sčítající vstupy. Následně definujeme funkci $f(g(x))$, která vezme tento součet a provede s ním prahování, čímž vygeneruje výstup y: ten je nulový, pokud je $g(x)$ menší než určité θ, a 1, je-li $g(x)$ větší nebo rovno θ.

$$g(x) = x1 + x2 + x3 + \cdots + xn = \sum_{i=1}^{n} xi$$

$$f(z) = \begin{cases} 0, & z < \theta \\ 1, & z \geq \theta \end{cases}$$

$$y = f(g(x)) = \begin{cases} 0, & g < \theta \\ 1, & g \geq \theta \end{cases}$$

S jedním umělým neuronem, jak jsme si ho popsali, můžeme navrhnout některá základní booleovská logická hradla (například AND a OR). U logického hradla AND je výstup *y* roven 1, jsou-li *x1* i *x2* rovny 1; v opačném případě je výstup roven 0. V tomto případě to splňuje $\theta = 2$. Nyní bude výstup *y* roven 1 pouze tehdy, když *x1* i *x2* budou mít hodnotu 1 (pouze tehdy bude *x1* + *x2* větší nebo rovno 2). S hodnotou θ si můžete pohrát při návrhu dalších logických hradel. Například u hradla OR je výstup roven 1, je-li *x1* nebo *x2* rovno 1, jinak je výstup roven 0. Jaká by měla být hodnota θ?

Tento jednoduchý McCullochův-Pittsův model lze rozšířit. Můžete zvýšit počet vstupů. Vstupy lze zavést jako „inhibiční", což znamená, že *x1* nebo *x2* může mít hodnotu –1. Pokud je jeden ze vstupů do neurodu inhibiční a vhodně nastavíte prahovou hodnotu, pak neurod bude vždy vysílat hodnotu 0 bez ohledu na hodnoty všech ostatních vstupů. To umožňuje konstruovat složitější logiku. Podobně jako propojení více neurodů tak, že výstup jednoho neurodu poslouží jako vstup do jiného.

To vše bylo úžasné, přesto však limitující. McCullochův-Pittsův (MCP) neuron je výpočetní jednotka, jejímiž kombinacemi lze vytvořit libovolný typ booleovské logiky. Vzhledem k tomu, že všechny digitální výpočty jsou ve své nejzákladnější podstatě posloupností takových logických operací, můžete neurony MCP

na základní úrovni různě propojovat a provádět tak libovolné výpočty. V roce 1943 šlo o neobvyklé tvrzení. Matematické kořeny McCullocha a Pittsova článku byly přitom evidentní: obsahoval pouze tři odkazy – na Carnapovu studii *The Logical Syntax of Language* (Logická syntaxe jazyka), Hilbertovy a Ackermanovy *Foundations of Theoretical Logic* (Základy teoretické logiky) a Whiteheadova a Russellova *Principia Mathematica* – a žádný z nich neměl nic do činění s biologií. O exaktních závěrech, které McCullochova a Pittsova práce přinesla, nebylo pochyb. A přesto byl výsledkem jen stroj, který uměl počítat, nikoliv se učit. Především bylo nutno stanovovat ručně hodnotu θ; neuron nedokázal zkoumat data a tím θ zjistit.

Není divu, že Rosenblattův perceptron vyvolal takový ohlas. Dokázal se potřebné váhy učit z dat. V těchto vahách byly zakódovány určité, byť minimální, znalosti o zákonitostech v datech, a ty si svým způsobem zapamatovaly.

Učení se z chyb

Rosenblattova vědecká práce často uváděla jeho studenty v úžas. George Nagy, který přišel na Cornellovu univerzitu v Ithace ve státě New York v roce 1960, aby si u Rosenblatta udělal doktorát, vzpomínal na procházku, při níž spolu hovořili o stereoskopickém vidění. Rosenblatt Nagyho ohromil svými znalostmi tohoto tématu. „Bylo těžké nepřipadat si při hovoru s ním jako naivní," řekl mi Nagy,[25] nyní emeritní profesor na Rensselaerově polytechnickém institutu ve městě Troy, stát New York. Rosenblattovu zjevnou erudici podtrhovalo jeho relativní mládí (byl ani ne o deset let starší než Nagy).

Rosenblattova mladistvost je málem dostala do potíží během jedné společné cesty. Potřebovali se dostat autem z Ithacy na konferenci do Chicaga. Rosenblatt ještě neměl napsaný příspěvek, který se chystal přednést, a tak požádal Nagyho, aby řídil, zatímco on bude pracovat. Nagy nikdy nevlastnil auto a řídit téměř neuměl, přesto však souhlasil. „Bohužel jsem jel v několika pruzích

najednou a zastavil nás policista," vyprávěl mi Nagy. Rosenblatt policistovi řekl, že je profesor a že požádal svého studenta, aby řídil. „Policista se rozesmál a řekl: ‚Vy nejste profesor, vy jste student.'" Naštěstí měl Rosenblatt u sebe dostatek dokladů, aby policistu přesvědčil o své důvěryhodnosti, a ten je nechal jet. Rosenblatt zbytek cesty do Chicaga dořídil, zůstal vzhůru přes noc a napsal na stroji svou práci, kterou druhý den přednesl. „Takové věci dokázal," vzpomínal Nagy.

V době, kdy Nagy nastoupil na Cornellovu univerzitu, měl již Rosenblatt sestrojený perceptron Mark I; víme již z úvodu knihy, že to bylo v roce 1958, čemuž se dostalo pozornosti v *New York Times*. Nagy začal pracovat na dalším stroji, nazvaném Tobermory (pojmenovaném podle mluvící kočky vytvořené H. H. Munroem alias Sakim), hardwarové neuronové síti určené pro rozpoznávání řeči. Perceptron Mark I a Rosenblattovy myšlenky již mezitím vzbudily značnou pozornost.

V létě 1958 věnoval editor časopisu *Research Trends* vydávaného v Cornell Aeronautical Laboratory Rosenblattovi celé číslo („pro mimořádný význam článku Dr. Rosenblatta," jak uvedl redaktor). Článek nesl název *Návrh inteligentního automatu: Představujeme perceptron – stroj, který vnímá, rozpoznává, pamatuje si a reaguje jako lidská mysl*.[26]

Rosenblatt nakonec litoval, že si pro označení svého díla vybral pojem perceptron. „Rosenblatt si později velmi vyčítal, že použil slovo, které zní jako stroj," řekl mi Nagy. Ve skutečnosti měl Rosenblatt označením perceptrony na mysli třídu modelů nervové soustavy pro vnímání a chápání.

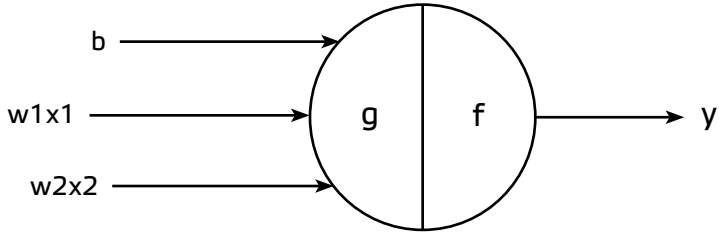
Jeho důraz kladený na mozek nebyl překvapením. Rosenblatt studoval u Jamese Gibsona, jednoho z velikánů na poli zrakového vnímání. Vzhlížel také k McCullochovi a Pittsovi a k Donaldu Hebbovi, kanadskému psychologovi, který v roce 1949 představil model, jak se učí biologické neurony (aby bylo jasno, učení se zde týká učení se vzorů v datech, nikoli toho druhu učení, které si obvykle spojujeme s lidským poznáním na vysoké úrovni). „Vždy o nich mluvil s uznáním," řekl Nagy.

McCulloch s Pittem sice vyvinuli modely neuronů, jenže sítě těchto umělých neuronů se učit nedokázaly. V souvislosti s biologickými neurony navrhl Hebb mechanismus učení, který bývá často stručně, ale poněkud mylně popisován takto: „Neurony, které se současně aktivují, se zároveň i propojují." (Neurons that fire together wire together.)[27] Přesněji řečeno, podle tohoto způsobu myšlení se naše mozky učí, protože spojení mezi neurony se posilují, když se výstup jednoho neuronu stabilně podílí na aktivaci jiného, a oslabují, když tomu tak není; tento proces se nazývá hebbovské učení.[28]

Byl to Rosenblatt, kdo práce těchto průkopníků využil a vyvodil z ní novou představu: umělé neurony, které při učení mění svou konfiguraci a informace vtělují do síly svých spojení.

Jako psycholog neměl Rosenblatt přístup k potřebnému počítačovému výkonu pro simulaci svých myšlenek pomocí hardwaru nebo softwaru. Proto si z Cornellovy letecké laboratoře pronajal čas počítače IBM-704, pětitunového monstra o velikosti místnosti. Spolupráce se ukázala jako plodná, když Rosenblattova práce upoutala pozornost fyziků a vyústila v články nejen v časopisech o psychologii, ale také v žurnálech Americké fyzikální společnosti.[29] Rosenblatt nakonec sestrojil Perceptron Mark I. Zařízení mělo kameru, která vytvářela obraz o rozměrech 20 × 20 pixelů. Když byly tyto obrázky perceptronu zobrazeny, dokázal rozpoznat písmena abecedy. Nejde však o to, že Mark I rozpoznával znaky, řekl Nagy. Vždyť systémy optického rozpoznávání znaků, které měly stejné schopnosti, byly komerčně dostupné již v polovině 50. let. „Jde o to, že se Mark I naučil rozpoznávat písmena tím, že při chybě dostával negativní zpětnou vazbu!" prohlásil Nagy ve své přednášce.[30]

Co však přesně perceptron je a jak probíhalo jeho učení? Ve své nejjednodušší podobě je perceptron rozšířený McCullochův-Pittsův neuron vybavený učicím algoritmem.[31] Na ukázce vidíte příklad se dvěma vstupy. Všimněte si, že každý vstup je násoben odpovídající váhou (je zde také další vstup, *b*, jehož důvod bude brzy jasný):



Výpočet prováděný perceptronem vypadá takto:

$$suma = w1x1 + w2x2 + b$$

$$Pokud\ suma > 0: y = 1$$

$$Jinak: y = -1$$

Obecněji a matematickým zápisem:

$$g(x) = w1x1 + w2x2 + \cdots + wnxn + b = \sum_{i=1}^{n} wixi + b$$

$$f(z) = \begin{cases} -1, z \leq \theta \\ 1, z > \theta \end{cases}$$

$$y = f(g(x)) = \begin{cases} -1, g(x) \leq \theta \\ 1, g(x) > \theta \end{cases}$$

Hlavní rozdíl oproti dříve představenému McCullochově-Pittsově (MCP) modelu spočívá v tom, že vstupy perceptronu nemusejí být binární (0 nebo 1), ale mohou nabývat libovolných hodnot. Také tyto vstupy jsou vynásobeny odpovídajícími váhami, takže nyní máme vážený součet. K tomu přistupuje další člen *b*, zkreslení. Výstup *y* je buď –1, nebo +1 (místo 0 nebo 1 jako u neuronu MCP). Podstatné je, že na rozdíl od neuronu MCP se perceptron může správnou hodnotu vah a zkreslení pro řešení určitého problému naučit.

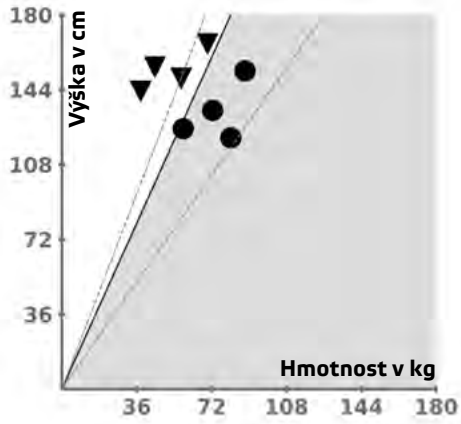
Abychom pochopili, jak to funguje, uvažujme perceptron, který se snaží klasifikovat osoby jako obézní, $y = +1$, nebo neobézní, $y = -1$. Vstupy jsou tělesná hmotnost *x1* a výška *x2*. Řekněme, že datový soubor obsahuje sto vstupů, přičemž každý vstup se

skládá z tělesné hmotnosti a výšky osoby a příznaku (popisku), který říká, zda zda lékař považuje osobu za obézní či neobézní, podle směrnic stanovených americkým Národním ústavem pro srdce, plíce a krev (NHLBI). Úkolem perceptronu je „naučit" se hodnoty *w1* a *w2* a hodnotu výrazu zkreslení *b* tak, aby správně klasifikoval každou osobu v souboru dat jako obézní, nebo neobézní. Poznámka: analyzujeme tělesnou hmotnost a hovoříme také o vahách perceptronu (*w1* a *w2*); v následujícím textu tedy slova „hmotnost" a „váha" nezaměňujte.

Jakmile se perceptron naučí správné hodnoty vah *w1* a *w2* a členu zkreslení, je připraven dávat predikce. Při zadání tělesné hmotnosti a výšky jiné osoby (která nebyla v původním datovém souboru, takže nejde o jednoduché vyhledání v tabulce záznamů) dokáže perceptron tuto osobu klasifikovat jako obézní, nebo neobézní. Tento model je samozřejmě založen na několika předpokladech, z nichž mnohé souvisejí s rozdělením pravděpodobnosti, k nimž se dostaneme v následujících kapitolách. Perceptron však činí jeden zásadní předpoklad: předpokládá, že existuje jasná dělicí čára mezi kategoriemi lidí klasifikovanými jako obézní a klasifikovanými jako neobézní.

Kdybyste, jedná-li se o tento jednoduchý příklad, vynesli tělesnou hmotnost a výšku lidí do grafu XY, hmotnost na ose x a výšku na ose y tak, že každý člověk bude bodem v grafu, pak předpoklad „jasné dělicí čáry" říká, že existuje přímka, která odděluje body představující obézní a body představující neobézní osoby. Pokud tomu tak je, říká se, že datový soubor je lineárně oddělitelný.

Zde je grafické znázornění toho, co se děje, když se perceptron učí. Začneme se dvěma množinami datových bodů, z nichž jedna je představována černými kolečky ($y = +1$, obézní) a druhá černými trojúhelníčky ($y = -1$, neobézní). Každý datový bod je popsán dvojicí hodnot (*x1*, *x2*), kde *x1* je tělesná hmotnost daného člověka v kilogramech, vynesená na osu x, a *x2* je jeho výška v centimetrech, vynesená na osu y.



Perceptron začíná s váhami *w1* a *w2* a zkreslením nastaveným na nulu. Váhy a zkreslení představují přímku v rovině XY. Perceptron se pak snaží najít dělicí přímku definovanou určitou množinou hodnot vah a zkreslení, která by od sebe oddělila oba druhy bodů. Na začátku klasifikuje některé body správně a jiné nesprávně. Dva z nesprávných pokusů jsou znázorněny šedými čárkovanými čarami. V tomto případě je patrné, že v jednom pokusu leží všechny body na jedné straně přerušované čáry, takže trojúhelníčky klasifikuje správně, ale kolečka nikoli, v dalším pokusu pak správně klasifikuje kolečka, avšak chybně zase některé trojúhelníčky. Perceptron se ze svých chyb učí a upravuje své váhy a zkreslení. Po mnoha průchodech daty nakonec objeví alespoň jednu množinu správných hodnot svých vah a členu zkreslení. Najde přímku, která od sebe shluky odděluje: kolečka a trojúhelníčky leží na opačných stranách od ní. V našem grafu je znázorněna jako plná černá čára, rozdělující prostor souřadnicového systému na dvě oblasti (z nichž jedna je šedě vystínovaná). Váhy naučené perceptronem určují sklon přímky, zkreslení vzdálenost neboli posun přímky od počátku.

Jakmile se perceptron naučí korelaci mezi fyzickými charakteristikami člověka (hmotností a výškou) a tím, zda daná osoba

je obézní ($y = +1$ nebo –1), můžete mu zadat hmotnost a výšku člověka, jehož data nebyla při tréninku použita, a perceptron vám řekne, zda by měl být klasifikován jako obézní. Perceptron nyní samozřejmě predikuje, jak nejlépe umí, protože se naučil váhy a zkreslení, avšak jeho predikce může být chybná. Přijdete na to, proč? Zkuste, zda dokážete odhalit problém pouhým pohledem na graf. (Nápověda: kolik různých čar dokážete nakreslit, aby se vám podařilo oddělit kolečka od trojúhelníčků?) Jak uvidíme, velká část strojového učení spočívá právě v minimalizaci predikční chyby (chyby předpovědi).

To, co je popsáno výše, je jedna jednotka perceptronu neboli jeden umělý neuron. Zdá se to jednoduché a možná si říkáte, proč kolem toho děláme tolik povyku. Inu, představte si, že by počet vstupů do perceptronu přesáhl dva (*x1*, *x2*, *x3*, *x4* atd.), přičemž každý vstup (*xi*) by dostal vlastní osu. K vyřešení problému vám už nebudou stačit jednoduché výpočty. K oddělení dvou shluků, které nyní existují v mnohem více než dvou dimenzích, již nestačí přímka. Máte-li například tři body (*x1*, *x2*, *x3*), budou data trojrozměrná, a abyste je od sebe oddělili, potřebujete 2D rovinu. Při 4 a více dimenzích potřebujete nadrovinu (kterou si naší 3D myslí nedokážeme představit). Obecně se tento vícerozměrný ekvivalent 1D přímky nebo 2D roviny nazývá nadrovina.

Nyní se vraťme do roku 1959. Rosenblatt sestrojil svůj Perceptron Mark I z množství takových jednotek. Stroj dokázal zpracovat obrázek o rozměrech 20×20 – celkem 400 pixelů, přičemž každému pixelu odpovídala vstupní hodnota *x*. Mark I tedy přijímal jako vstup dlouhou řadu hodnot: *x1*, *x2*, *x3*, …, *x400*. Složitým uspořádáním umělých neuronů, jak s pevnými, náhodnými váhami, tak s váhami, které se daly naučit, docházelo k proměně tohoto vektoru 400 hodnot na výstupní signál, pomocí něhož bylo možno rozeznávat vzor v obraze. (Jedná se o velmi zjednodušený popis. Některé výpočty byly natolik složité, že k nim byl zapotřebí počítač IBM 704. Do detailů této architektury nahlédneme v kapitole 10.) Mark I se dokázal naučit zatřiďovat písmena abecedy zakódovaná v těchto hodnotách pixelů. Celá tato logika, rozšířená

na 400 vstupů, byla implementována hardwarově. Jakmile se stroj naučil, měl své znalosti uloženy v silách (vahách) svých spojení. Není divu, že lidé popouštěli uzdu své fantazii.

Pokud však blíže prozkoumáte, co se perceptron učí, vyjdou – samozřejmě při zpětném pohledu – jeho omezení najevo. Algoritmus pomáhá perceptronu nabývat znalosti o korelacích mezi hodnotami množin (*x1*, *x2*, …, *x400*) a odpovídající hodnotou *y*, pokud takové korelace v datech existují. Jistě, učí se korelace, aniž by mu bylo explicitně řečeno, o jaké korelace se jedná, nicméně jsou to korelace. Je rozpoznání korelací totéž, co myšlení a uvažování? Ovšemže pokud Mark I rozlišil písmeno B od písmene G, řídil se prostě vzory a nepřikládal těmto písmenům žádný význam, který by vedl k dalšímu uvažování. Takové otázky jsou jádrem moderní debaty o limitech hlubokých neuronových sítí, úžasných potomků perceptronů. Existuje souvislost, která spojuje tyto rané perceptrony s technologií rozsáhlých jazykových modelů nebo umělou inteligencí vyvíjenou například pro samořídící automobily. Tato souvislost není přímá, jde spíše o dlouhou a klikatou trajektorii s falešnými odbočkami a slepými uličkami. Přesto však je to fascinující a zajímavá cesta a my se na ni nyní vydáváme.

Sestrojení perceptronu bylo velkým úspěchem. Ještě větším úspěchem byl matematický důkaz, že jedna vrstva perceptronů vždy najde lineárně oddělující nadrovinu, *pokud* jsou data lineárně oddělitelná. Pochopení tohoto důkazu bude vyžadovat, abychom se seznámili s vektory a poznali, proč tvoří základ metod používaných k reprezentaci dat ve strojovém učení. Je to naše první matematická zastávka.

Kapitola 2

Všichni jsme tu jen čísla

Necelý měsíc před svou smrtí v září 1865 napsal irský matematik William Rowan Hamilton dopis ve čtyřech odstavcích[32] svému synovi. V tomto dopise Hamilton mimo jiné vzpomínal na procházku kolem Královského kanálu v irském Dublinu. Bylo to 16. října 1843. Hamilton se se svou ženou nacházel na cestě na zasedání Královské irské akademie. Když došli pod Broughamův most, Hamilton, který se již více než deset let potýkal s hlubokými matematickými otázkami, dostal záblesk inspirace. „Zdálo se, že se uzavírá elektrický obvod; a zajiskřilo to… Nemohl jsem odolat nutkání – jakkoli nefilozofickému – vyrýt nožem do kamene Broughamova mostu, pod nímž jsme procházeli, základní vzorec se symboly, *i*, *j*, *k*, a sice $i^2 = j^2 = k^2 = ijk = -1$."[33]

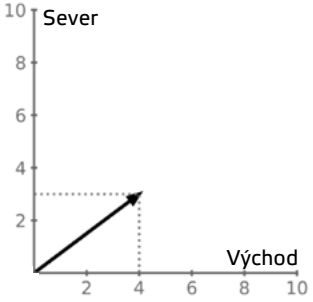
Hamilton dopis synovi podepsal následujícími slovy: „Tento dopis uzavírám *tímto kvaternionem odstavců…* (kurzíva je moje) Tvůj láskyplný otec, William Rowan Hamilton."[34]

Slovo kvaternion („čtveřice") použil záměrně. Jde o matematický útvar složený ze čtyř prvků s velmi zvláštními a neobvyklými vlastnostmi, na který Hamilton přišel onoho osudného dne pod Broughamovým mostem. Rovnice, kterou vyryl do kamene a která představuje obecnou podobu kvaternionu, je jedním z nejslavnějších příkladů matematického graffiti. Originál, který ovšem již dávno sešel, byl nahrazen oficiální pamětní deskou s nápisem:

Zde procházeje
16. října 1843
Sir William Rowan Hamilton
v záblesku geniality objevil

základní vzorec pro
násobení kvaternionů
$i^2 = j^2 = k^2 = ijk = -1$
a vryl ho do kamene tohoto mostu.[35]

Kvaterniony jsou exotické entity a nás se netýkají. Aby však Hamilton mohl sestrojit algebraický aparát pro nakládání s kvaterniony, vypracoval některé další matematické principy, které se staly základem strojového učení. Zejména zavedl pojmy skalár a vektor.[36] V dnešní době většina z nás o Hamiltonovi pravděpodobně neuslyší, jsou nám však povědomé pojmy skalár a vektor, i když ne jejich formální definice. Zde je stručný úvod do problematiky. Uvažujme muže, který ujde 5 kilometrů. Ve vztahu k tomuto tvrzení můžeme o tom, co tento muž učinil, říci pouze jediné číslo: vzdálenost, kterou ušel. To je skalární veličina, samostatné číslo. Kdybychom však řekli, že muž ušel 5 km severovýchodním směrem, měli bychom informace dvě: ušlou vzdálenost a směr. To lze znázornit vektorem. Vektor má tedy jednak délku (velikost), jednak i směr. V grafu níže je vektorem šipka o velikosti 5.

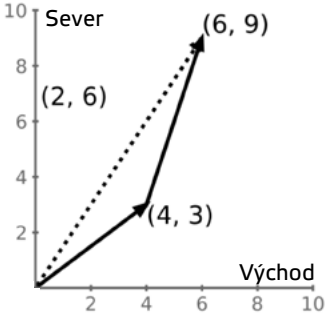


Prozkoumáte-li vektor pozorně, zjistíte, že má dvě složky: jednu ve směru osy x a druhou ve směru osy y. Je to totéž jako říct, že muž urazil 4 km směrem na východ a 3 km směrem na sever. Vektor představující skutečnou chůzi je šipka směřující z bodu o souřadnicích (0, 0) do bodu (4, 3), udávající jak směr, tak vzdálenost. Velikost vektoru je rovna délce přepony pravoúhlého trojúhelníku

tvořeného vektorem a jeho složkami ve směru os x a y. V tomto případě se jedná o délku přepony pravoúhlého trojúhelníku. Velikost neboli délka vektoru je tedy rovna $\sqrt{4^2 + 3^2} = 5$.

Uvažování ve vektorech, bez formálních metod jejich reprezentace a operací s nimi, existovalo už před Hamiltonem. Například již koncem 17. století uvažoval Isaac Newton o vektorových entitách, jako je zrychlení a síla, geometricky. Druhý Newtonův pohybový zákon říká, že zrychlení, jímž se těleso pohybuje, je úměrné síle, která na něj působí, a že zrychlení tělesa i síla mají stejný směr. První důsledek Newtonových pohybových zákonů v jeho knize *Principia* říká: „Těleso, na něž působí současně dvě síly, opíše úhlopříčku rovnoběžníku za stejnou dobu, za jakou by opsalo jeho strany, kdyby tyto síly působily samostatně."[37] To je výrok využívající geometrie k sečtení dvou vektorů, i když Newton tyto veličiny nenazval vektory.

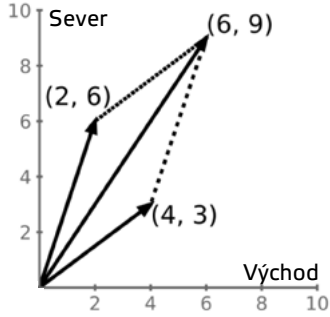
Abychom sčítání vektorů pochopili, můžeme se vrátit k naší osobě, která ušla 5 km, reprezentovaných vektorem směřujícím z bodu (0, 0) do bodu (4, 3). Po dosažení cíle se osoba otočí severněji, takže v souřadnicové soustavě dosáhne bodu (6, 9): fakticky ušla další 2 km směrem na východ a dalších 6 km směrem na sever. To je znázorněno druhým vektorem, šipkou vedenou z bodu (4, 3) do bodu (6, 9). Tento nový vektor má složku x o hodnotě 2 a y o hodnotě 6. Jakou celkovou vzdálenost tento člověk ušel? A jaká je čistá vzdálenost v souřadnicové soustavě XY od počátku do konečného cíle? Odpovědi na obě otázky vám ukáže tento graf:



Velikosti obou jednotlivých vektorů neboli délky procházek jsou $\sqrt{4^2+3^2}=\sqrt{25}=5$ a $\sqrt{2^2+6^2}=\sqrt{4+36}=\sqrt{40}=6{,}32$. Celková vzdálenost, kterou člověk ujde, tedy je $5+6{,}32=11{,}32$ km.

Výsledným vektorem je šipka vedená z počátku do konečného cíle (6, 9) a její velikost je $\sqrt{6^2+9^2}=\sqrt{36+81}=\sqrt{117}=10{,}82$. Čistá vzdálenost v souřadnicové soustavě XY je 10,82 km.

To nám nyní pomůže pochopit, co říkal Newton. Řekněme, že zrychlení způsobené jednou silou působící na těleso je dáno vektorem (2, 6) a zrychlení způsobené jinou silou na totéž těleso je dáno vektorem (4, 3). Obě síly působí na těleso současně. Jaké je celkové zrychlení tělesa? Geometrická interpretace Newtonových tvrzení spočívá v nakreslení rovnoběžníku, jak je znázorněno na obrázku níže; čisté zrychlení je pak dáno úhlopříčným vektorem (6, 9):



Je-li zrychlení v jednotkách m/s^2, pak čisté zrychlení je dáno velikostí vektoru (6, 9), což se rovná 10,82 m/s^2 ve směru šipky.

Pro sčítání jsem v tomto případě zvolil stejné vektory jako v příkladu s kráčející osobou, zde však oba vektory představují zrychlení, ne vzdálenost, a oba mají počátek v bodě (0, 0). To vám říká, že vektor (2, 6) je stejný vektor bez ohledu na to, zda vychází z bodu (0, 0) nebo z bodu (4, 3), jako v předchozím příkladu. Důležitou vlastností vektorů je, že šipku reprezentující vektor můžeme v souřadnicové soustavě posouvat, a nezměníme-li délku šipky ani její orientaci, jedná se o tentýž vektor. Proč? Inu proto, že jsme nezměnili jeho délku ani směr, tedy dvě vlastnosti, jimiž je vektor definován.

Když Newton v roce 1687 publikoval svou knihu *Principia*, nebylo nic z toho chápáno jako formální počátky vektorové analýzy. Jeho současník Gottfried Wilhelm Leibniz (1646–1716) však měl více než jen ponětí o tomto novém způsobu uvažování. V roce 1679 v dopise jinému svému současníkovi, Christianu Huygensovi, napsal: „Jsem přesvědčen, že jsem našel způsob…, jak lze znázorňovat čísla, a dokonce i stroje a pohyby pomocí znaků – tak, jako se v algebře znázorňují čísla nebo veličiny."[38] Leibniz své myšlenky nikdy plně neformalizoval, ovšem jeho prozíravost – jak uvidíme, až pochopíme význam vektorů pro strojové učení – byla ohromující. Po Leibnizovi vyvinula řada dalších matematiků, včetně Johanna Carla Friedricha Gausse (1777–1855), metody geometrické reprezentace určitých typů čísel ve dvou rozměrech, což připravilo půdu pro Hamiltonův objev kvaternionů a formalizaci vektorové analýzy.

Vektory jako čísla

Práce s vektory nemusí být jen geometrická. Můžeme přejít k operacím s čísly zapsanými v určitém tvaru. A u strojového učení musíme vlastně o vektorech právě takto přemýšlet. Například zrychlení způsobená dvěma silami v předchozím příkladu jsou prostě pole o dvou číslech, [4, 3] a [2, 6]. Jejich sečtení je totéž jako sečtení jednotlivých složek každého vektoru (umístěných svisle na sebou, do sloupce). O šipky se nemusíte starat:

$$\begin{bmatrix}4\\3\end{bmatrix} + \begin{bmatrix}2\\6\end{bmatrix} = \begin{bmatrix}6\\9\end{bmatrix}$$

Odečítání vektorů je podobné.

$$\begin{bmatrix}4\\3\end{bmatrix} - \begin{bmatrix}2\\6\end{bmatrix} = \begin{bmatrix}2\\-3\end{bmatrix}$$

K čemu zde došlo? Proč je složka y výsledného vektoru záporná? Představují-li tato čísla stále zrychlení, pak odečtení znamená, že druhá síla působí proti první síle; podél osy x je zrychlení jen

o něco menší, než když jsme oba vektory sečetli, ale stále kladné, ovšem podél osy y nyní síla působí proti původnímu směru pohybu, což vede ke zpomalení.

Vektor lze vynásobit skalárem – stačí vynásobit každý prvek vektoru tímto skalárem.

$$5 \times \begin{bmatrix} 4 \\ 3 \end{bmatrix} = \begin{bmatrix} 20 \\ 15 \end{bmatrix}$$

Z geometrického hlediska je to totéž, jako kdybyste šipku (nebo vektor) natáhli pětkrát stejným směrem. Velikost původního vektoru je 5. Jeho pětinásobné zvětšení nám dává novou velikost 25. Pokud byste vypočítali velikost nového vektoru pomocí jeho zvětšených souřadnic, opět dostanete:

$$\sqrt{20^2 + 15^2} = \sqrt{400 + 225} = \sqrt{625} = 25$$

Existuje ještě jeden způsob reprezentace vektorů. Omezíme-li se na dva rozměry, představíme si vektor délky jedna, **i**, ve směru x, a vektor délky jedna, **j**, ve směru osy y. Všimněte si, že **i** i **j** jsou napsány malými písmeny a tučně: znamená to, že se jedná o vektory. Takže **i** si můžeme představit jako šipku, která směřuje z (0, 0) do (1, 0), a **j** jako šipku, která směřuje z (0, 0) do (0, 1). Oba mají velikost 1 a říká se jim také jednotkové vektory. Za této situace lze vektory (4, 3) a (2, 6) v kartézské souřadnicové soustavě zapsat takto: $4\mathbf{i} + 3\mathbf{j}$ respektive $2\mathbf{i} + 6\mathbf{j}$. Je to totéž, jako kdybychom řekli, že vektor (4, 3) má délku 4 jednotky ve směru osy x a délku 3 jednotky ve směru osy y a vektor (2, 6) 2 jednotky ve směru osy x a 6 jednotek ve směru osy y. Písmena **i** a **j** jsou zkratkami zastupujícími vektory. Je také namístě zdůraznit, že jednotkový vektor je prostě vektor s velikostí 1; nemusí splývat ani být rovnoběžný s kolmými osami souřadnicové soustavy.

Tyto úvahy platí i pro vícerozměrné prostory (vyšší dimenze), k nimž se ještě dostaneme. Prozatím nám zvládnutí matematických operací s dvourozměrnými (2D) vektory a jejich odpovídajícího geometrického významu výrazně pomůže pochopit roli jejich vícerozměrných ekvivalentů ve strojovém učení.

Skalární součin

Další důležitou operací s vektory je tzv. skalární součin. Uvažujme vektor (4, 0), říkejme mu **a**, a vektor (5, 5), říkejme mu **b** (malá tučná písmena **a** a **b** opět značí, že se jedná o vektory). Slovně je skalární součin vektorů **a**.**b** definován jako velikost vektoru **a** násobena průmětem vektoru **b** do vektoru **a**, kde průmět si můžeme představit jako „stín vržený“[39] jedním vektorem na druhý.

Velikost vektoru **a** se značí ||**a**|| nebo |**a**|. Projekce vektoru **b** do směru vektoru **a** je dána velikostí **b** neboli ||**b**|| vynásobenou kosinem úhlu mezi oběma vektory. Mezi vektory, které jsme zvolili, je úhel 45 stupňů (neboli $\frac{\pi}{4}$ radiánů), jak je znázorněno výše v grafu.

Tedy:

$$\mathbf{a.b} = \|\mathbf{a}\| \times \|\mathbf{b}\| \times \cos(\pi/4)$$

$$cos(\pi/4) = \frac{1}{\sqrt{2}}$$

$$\|\mathbf{a}\| = \sqrt{4^2 + 0^2} = 4$$

$$\|\mathbf{b}\| = \sqrt{5^2 + 5^2} = \sqrt{50} = 5\sqrt{2}$$

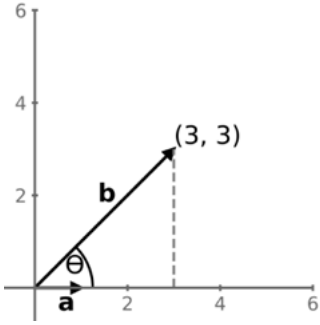
$$\Rightarrow \text{a.b} = 4 \times 5\sqrt{2} \times \frac{1}{\sqrt{2}} = 20$$

Poznámka: symbol ⇒ znamená „z toho plyne“.

Nyní proveďme několik drobných úprav. Nechť vektor **a** je dán (1, 0), vektor **b** (3,3).

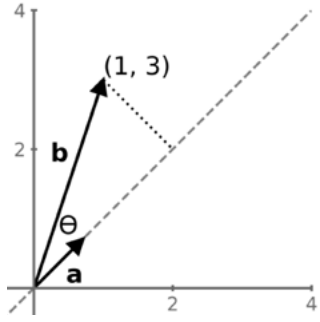
Vektor **a** má velikost 1, je to tedy **jednotkový vektor**. Pokud bychom nyní provedli skalární součin **a.b**, dostaneme:

$$\begin{aligned}\mathbf{a.b} &= \|\mathbf{a}\| \times \|\mathbf{b}\| \times \cos(\pi/4) \\ &= 1 \times 3\sqrt{2} \times cos(\pi/4) \\ &= 1 \times 3\sqrt{2} \times \frac{1}{\sqrt{2}} \\ &= 3\end{aligned}$$



Ukazuje se, že skalární součin je roven x-ové složce vektoru **b** neboli stínu, který vrhá **b** na osu x, ve směru jednotkového vektoru. To má významnou geometrickou interpretaci: jestliže jeden z vektorů účastnící se skalárního součinu má délku 1, pak se skalární součin rovná průmětu druhého vektoru do vektoru jednotkové délky. V našem konkrétním případě jednotkový vektor má směr osy x, takže průmětem vektoru **b** na osu x je zkrátka jeho složka x, tedy 3.

Na skalárních součinech je však pozoruhodná ještě další věc. Tato geometrická zákonitost platí, i když jednotkový vektor neleží ve směru žádné z os. Řekněme, že **a** je vektor $(1/\sqrt{2}, 1/\sqrt{2})$. Jeho velikost je jedna, jde tedy o jednotkový vektor, který však svírá s osou x úhel 45°. Řekněme, že **b** je vektor (1, 3). Skalární součin **a.b** je $\|\mathbf{a}\| \times \|\mathbf{b}\| \times \cos(\theta)$, což se rovná $1 \times \|\mathbf{b}\| \times \cos(\theta)$, což je znovu průmět vektoru **b** na přímku, která je prodloužením vektoru **a.**



Další důležitou věcí, kterou nám skalární součin o dvou vektorech říká, je to, zda spolu svírají pravý úhel neboli zda jsou kolmé. Svírají-li pravý úhel, pak se kosinus (90°) rovná nule. Bez ohledu na délku vektorů tedy bude jejich skalární součin neboli projekce vektoru **b** do vektoru **a** vždy nulová. A naopak, je-li skalární součin dvou vektorů roven nule, jsou na sebe kolmé.

Jak bychom vypočetli skalární součin, kdybychom použili druhý zmíněný způsob reprezentace vektorů, pomocí jejich složek, a neznáme-li úhel mezi oběma vektory?

Řekněme, že $\mathbf{a} = a1\mathbf{i} + a2\mathbf{j}$ a $\mathbf{b} = b1\mathbf{i} + b2\mathbf{j}$. Potom:

$$\mathbf{a.b} = (a1\mathbf{i} + a2\mathbf{j}).\ (b1\mathbf{i} + b2\mathbf{j}) =$$
$$a1b1 \times \mathbf{i.i} + a1b2 \times \mathbf{i.j} + a2b1 \times \mathbf{j.i} + a2b2 \times \mathbf{j.j}$$

Všimněte si, že druhý a třetí člen rovnice jsou nulové. Vektory **i** a **j** jsou vzájemně kolmé, **i.j** a **j.i** jsou tedy nulové. Také platí, že **i.i** a **j.j** se rovnají 1. Zbývá nám pouze skalární veličina:

$$\mathbf{a.b} = a1b1 + a2b2$$

Stroje a vektory

Pokud se vám zdá, že je to všechno vzdálené strojovému učení, perceptronům a hlubokým neuronovým sítím, buďte si jisti, že tomu tak není. Je to jejich ústředním bodem. A my se k němu dostáváme mílovými kroky, přičemž opatrně našlapujeme jen na kameny, které jsou nezbytné jako pevná opora.

Je čas vrátit se k perceptronu a přemýšlet o něm vektorově. Záměrem je získat geometrický náhled na to, jak datové body a váhy perceptronu reprezentovat formou vektorů a jak si představit, co se děje, když se perceptron snaží najít lineárně oddělující nadrovinu, která rozdělí datové body do dvou shluků. Velká část toho souvisí s využitím skalárních součinů vektorů k nalezení jejich relativních vzdáleností od nadroviny, jak uvidíme.

Připomeňme si obecnou rovnici perceptronu, která říká, že perceptron dává výstup 1, je-li vážený součet jeho vstupů plus určitý člen zkreslení *b* větší než 0, jinak je výstupem –1.

$$g(x) = w1x1 + w2x2 + \cdots + wnxn + b = \sum_{i=i}^{n} wixi + b$$

$$f(z) = \begin{cases} -1, z \leq \theta \\ 1, z > \theta \end{cases}$$

$$y = f(g(x)) = \begin{cases} -1, g(x) \leq \theta \\ 1, g(x) > \theta \end{cases}$$

Provedli jsme jemnou změnu v notaci, kterou jsme používali dříve: argumentem funkce *g* je nyní vektor; v předchozí kapitole, kde jsme ještě neměli princip vektorů zaveden, jsme místo $g(\mathbf{x})$ používali prostě g(*x*). Zůstaňme u dvourozměrného případu, kdy jsou datové body dány různými hodnotami pro (*x1*, *x2*) a váhy perceptronu jsou určeny (*w1*, *w2*). Perceptron nejprve vypočítá vážený součet vstupů:

$$w1x1 + w2x2$$

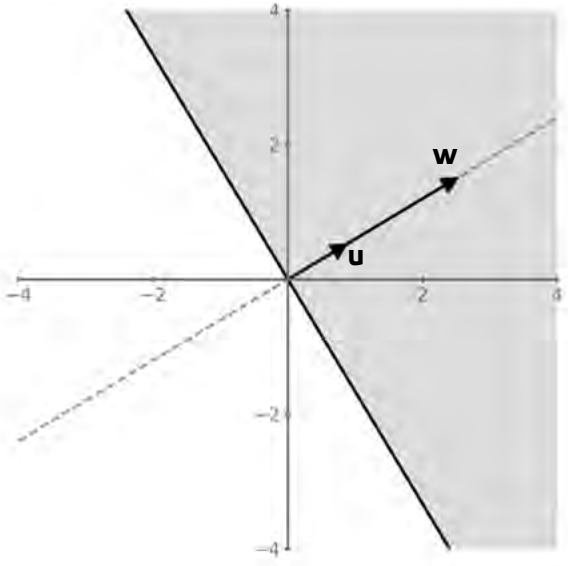
Jestliže je tento vážený součet vyšší než určitá prahová hodnota, nazvěme ji –*b*, pak výstupem perceptronu *y* je 1. V opačném případě je to –1. Tedy:

$$y = \begin{cases} -1, w1x1 + w2x2 \leq -b \\ 1, w1x1 + w2x2 > -b \end{cases}$$

To lze přepsat následovně:

$$y = \begin{cases} -1, w1x1 + w2x2 + b \leq \theta \\ 1, w1x1 + w2x2 + b > \theta \end{cases}$$

Navažme na naši vektorovou interpretaci. Množina vah (*w1*, *w2*) není nic jiného než vektor **w.** Co však **w** přesně představuje?

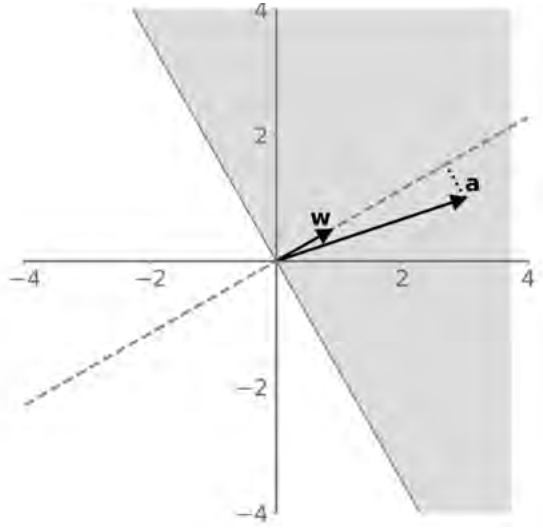


Na obrázku výše je zobrazen váhový vektor **w** = (2,5; 1,5). Zobrazuje také jednotkový vektor **u** ve stejném směru. Přerušovaná čára nám udává směr, v němž leží oba vektory. Nakresleme plnou černou přímku kolmou k vektorům **w** a **u**. Tato přímka odděluje vystínovanou oblast od zbytku souřadnicového prostoru. Pokud bychom se tedy snažili najít přímku, která jasně rozděluje rovinu XY na dvě oblasti, vystínovanou a nevystínovanou, pak nám k určení takové hranice stačí vektor **w** nebo odpovídající jednotkový vektor **u**. Touto hranicí je přímka, pokud dělíme 2D prostor, rovina při dělení 3D prostoru a nadrovina při více rozměrech.

Náš dřívější pohled na algoritmus učení perceptronu ukázal, že se snaží najít nadrovinu, která rozděluje souřadnicový prostor na dvě části. Je-li plná čára na výše uvedeném obrázku dělicí nadrovinou, pak je k ní vektor **w** kolmý a charakterizuje tuto nadrovinu. To, co se perceptron naučí, je vhodná množina vah. Tyto váhy tvoří vektor **w**. Měníte-li váhy perceptronu, měníte tím směr vektoru **w** a tím měníte i orientaci nadroviny, která je vždy na vektor **w** kolmá. A co platí pro vektor **w**, platí také pro jednotkový vektor **u**,

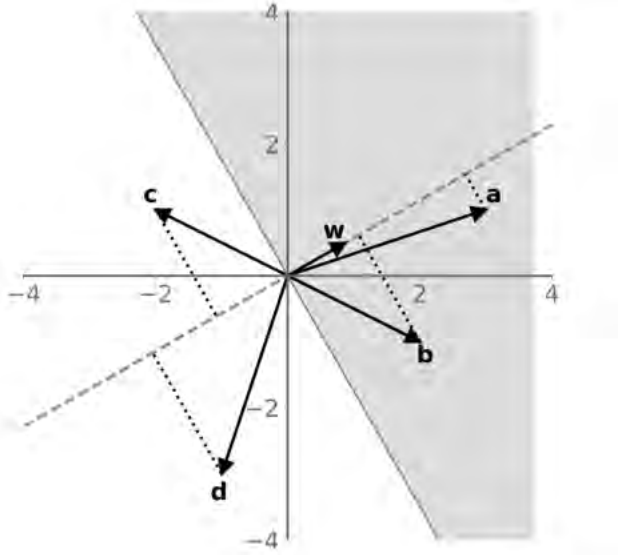
ležící ve stejném směru. Jednou z možností, jak jinak vyjádřit, co perceptron dělá, je říct, že najde vektor **w**, což je totéž jako říct, že najde odpovídající kolmou nadrovinu.

Nyní si představte datové body, které buď leží, nebo neleží ve vystínované oblasti. Každý datový bod je dán souřadnicemi (*x1*, *x2*) a lze si ho představit také jako vektor. Pak je vážený součet ($w1x1 + w2x2$) stejný jako skalární součin vektoru představujícího datový bod s váhovým vektorem. Všimněte si, že leží-li datový bod v nadrovině, což je ve 2D případě pouhá přímka, pak vektor (*x1*, *x2*) bude kolmý k **w**, takže skalární součin se bude rovnat nule. Zde je grafický pohled na skalární součin datových bodů a váhového vektoru. Pro jednoduchost budeme pracovat s váhovým vektorem jednotkové délky. Tím se principiálně nic nemění, zjednodušíme si tím však výpočty. Začněme vektorem **a**, který je dán datovým bodem (3, 1):



Protože **w** je jednotkový vektor, jeho skalární součin s vektorem **a** se rovná průmětu **a** na přerušovanou čáru. Bod, v němž vektor **a** prochází přímkou kolmou k nadrovině, je mírou vzdálenosti bodu (3, 1) od nadroviny.

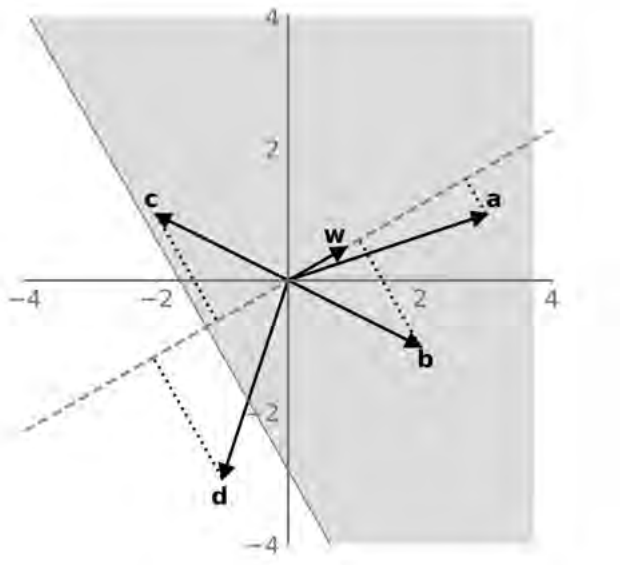
Nyní se podívejme na toto mírně nepřehledné, ovšem důležité znázornění skalárního součinu váhového vektoru se čtyřmi různými datovými body neboli s vektory **a** (3, 1), **b** (2, –1), **c** (–2, 1) a **d** (–3, –1).



Je patrné, že skalární součin každého vektoru s vektorem **w** něco o daném vektoru vypovídá: jaká je jeho vzdálenost od nadroviny a zda je na její jedné straně (v kladném směru) nebo na druhé (v záporném směru). V tomto konkrétním případě jsou body **a** a **b** lineárně odděleny od bodů **c** a **d** (body ležící v šedě vystínované oblasti představují $y = 1$, body ležící v nevystínované oblasti a body ležící přímo na dělicí přímce představují $y = -1$).

Řekněme tedy, že perceptron na první pokus najde váhy a nadrovinu, jak je znázorněno výše. Řekněme však také, že podle našich označených trénovacích dat by body **a**, **b** a **c** měly být na jedné straně nadroviny a pouze bod **d** na druhé. Pro představu řekněme, že body **a**, **b** a **c** představují osoby klasifikované jako mající v oblibě thrillery a bod **d** představuje osobu, která thrillery v oblibě nemá. To znamená, že perceptron ještě nenašel správnou nadrovinu. Jeden milovník thrillerů byl klasifikován jako jejich

odpůrce. Zde přichází ke slovu pojem zkreslení. Přidání zkreslujícího členu do rovnice je totéž jako posunutí nadroviny od počátku, avšak beze změny její orientace. Po průchodu trénovacími daty mohl perceptron najít například následující nadrovinu:



Při pohledu na výše uvedený obrázek je zřejmé, že jsou-li data lineárně oddělitelná do dvou shluků, pak existuje obrovské množství oddělujících nadrovin (pro různé hodnoty zkreslujícího členu a různé orientace vektoru **w**). Jen si představte, kolik můžete nakreslit přímek procházejících plochou mezi vektory **c** a **d**: v zásadě nekonečně mnoho. Perceptron pouze zaručuje, že najde jednu, a to ne nutně tu nejlepší. K tomu, co znamená „nejlepší", se dostaneme podrobněji, souvisí to však s predikcí. Perceptron se přece učí váhy a člen zkreslení, aby predikoval, kam se vzhledem k nadrovině nějaký dosud nepoznaný datový bod dostane. Například při dvou charakteristikách člověka, pomocí nichž budeme klasifikovat, zda má, nebo nemá v oblibě thrillery: na které straně nadroviny by se musel nacházet, aby byl klasifikován jako jeden nebo druhý typ? Dobrá, nebo spíše nejlepší možná nadrovina

bude minimalizovat budoucí chyby predikce. (Definovat „budoucí" predikční chybu, natož ji minimalizovat, je netriviální nebo ne zrovna snadný problém.)

Tyto grafy byly prostředkem k tomu, abychom si vytvořili intuitivní představu o tom, co se děje, když se perceptron učí. Kdybyste se pokusili napsat počítačový program, který by simuloval perceptron, nekreslili byste tabulky a grafy. Operovali byste s čísly. Naštěstí už číselné reprezentace vektorů, které jsme již poznali, stačí k tomu, aby ukázaly sílu těchto abstrakcí. V našem 2D příkladu jsou datové body (*x1*, *x2*) jen pole čísel, každé pole má dva prvky. Podobně je váhový vektor dalším polem dvou čísel. Nalezení skalárního součinu je záležitost operací s těmito poli.

Obecněji se tato pole nazývají matice, které obsahují řádky a sloupce čísel. Například máme-li *m* řádků a *n* sloupců, pak máme matici $m \times n$ (čtěte jako „matice m krát n"). Vektor je pak jen zvláštní formou matice s jedním řádkem nebo jedním sloupcem: buď $m = 1$, nebo $n = 1$. Setkali jsme se s nimi již dříve, jen jsme ještě tehdy nezavedli pojem matice. Ale právě to jsou vektory: matice s jedním sloupcem nebo jedním řádkem. Zde je příklad sečtení dvou jednosloupcových matic, čímž získáme třetí jednosloupcovou matici.

$$\begin{bmatrix}4\\3\end{bmatrix} + \begin{bmatrix}2\\6\end{bmatrix} = \begin{bmatrix}6\\9\end{bmatrix}$$

Překlopíme-li jednu z jednosloupcových matic na bok, získáme matici s jedním řádkem:

$$\begin{bmatrix}4 & 3\end{bmatrix}$$

Formálně je tedy jednosloupcová matice se dvěma prvky zapsaná takto:

$$\begin{bmatrix}a11\\a21\end{bmatrix}$$

Zápis říká, že sloupcová matice má dva řádky (indexované čísly 1 a 2) a každý z řádků má právě jeden prvek (index 1). Když matici překlopíte, změní se číslování (všimněte si, že index řádku je 1, zatímco sloupce mají indexy 1 a 2):

$$[a11 \quad a21]$$

Tomuto postupu se říká transpozice matice (u jednosloupcové matice vypadá triviálně nebo jednoduše, u matic vyšších řádů je to však o něco složitější, k čemuž se dostaneme v dalších kapitolách). Transpozice je klíčovým úkonem při výpočtu skalárního součinu dvou sloupcových matic. Matice budeme označovat velkými tučnými písmeny. Nechť **A** je sloupcová matice $\begin{bmatrix} a11 \\ a21 \end{bmatrix}$ a **B** sloupcová matice $\begin{bmatrix} b11 \\ b21 \end{bmatrix}$. Se dvěma sloupcovými maticemi nelze skalární součin provést. Je tomu tak proto, že aby bylo možné skalární součin provést, musí se počet sloupců první matice rovnat počtu řádků druhé matice. V našem případě je tedy nutno jednu z nich transponovat. Transpozice matice **A** se zapisuje jako $\mathbf{A}^{\mathrm{T}}$. Skalární součin **A.B** se zapisuje jako $\mathbf{A}^{\mathrm{T}}\mathbf{B}$ nebo $\mathbf{B}^{\mathrm{T}}\mathbf{A}$ (v tomto případě jde o jedno a totéž).

$$\mathbf{A}.\mathbf{B} = \mathbf{A}^{\mathrm{T}}\,\mathbf{B} = [\ a11 \quad a21\ ] \times \begin{bmatrix} b11 \\ b21 \end{bmatrix} = a11b11 + a12b21$$

Všimněte si, že to je přesně ta hodnota, kterou byste získali, kdybyste vektory zapsali pomocí jejich jednotkových vektorů **i** a **j.** Jestliže $\mathbf{a} = a11\mathbf{i} + a12\mathbf{j}$ a $\mathbf{b} = b11\mathbf{i} + b21\mathbf{j}$, pak:

$$\mathbf{a.b} = (a11\mathbf{i} + a12\mathbf{j}).(b11\mathbf{i} + b21\mathbf{j})$$

$$\Rightarrow \mathbf{a.b} = a11b11 \times \mathbf{i.i} + a11b21 \times \mathbf{i.j} + a12b11 \times \mathbf{j.i} + a12b21 \times \mathbf{j.j}$$

$$\Rightarrow \mathbf{a.b} = a11b11 + a12b21$$

Na reprezentaci vektorů pomocí matic místo šipek je skvělá ještě jedna věc: pro získání skalární hodnoty skalárního součinu stačí pracovat pouze s čísly, aniž byste se museli starat o kosinus úhlu mezi nimi. To znamená, že máte-li několik datových bodů, z nichž každý je reprezentován vektorem, a chcete zjistit jejich

Vážení čtenáři, právě jste dočetli ukázku z knihy Proč se stroje učí**.**
Pokud se Vám ukázka líbila, na našem webu si můžete zakoupit celou knihu.