KMI/UMIN Umělá inteligence

Deep Learning

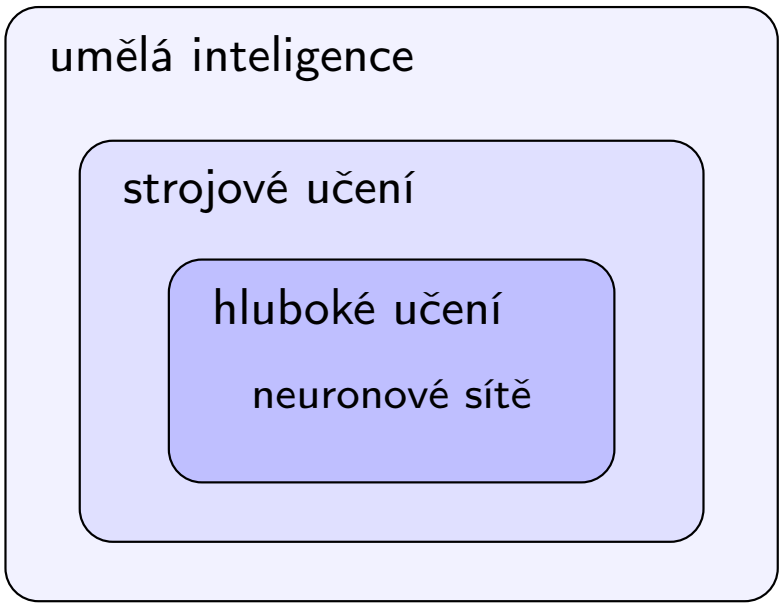
1. Úvod do hlubokého učení a perceptron

Jan Konečný

Katedra informatiky, PřF UP v Olomouci

21. září 2026

Tři vnořené pojmy



Umělá inteligence — systémy, které vnímají, usuzují, učí se nebo jednají.

Strojové učení — chování se neurčuje ručně zapsanými pravidly, ale odvozuje se z dat.

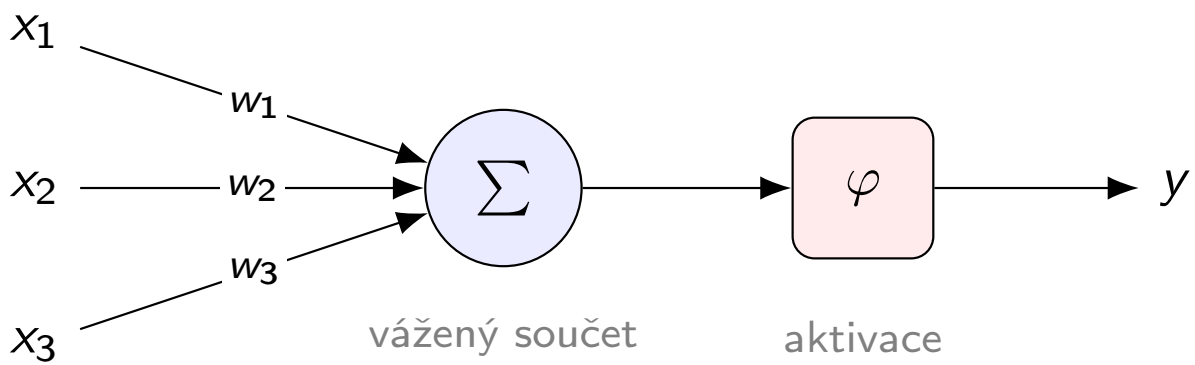
Hluboké učení — část strojového učení založená na sítích s více vrstvami naučených reprezentací.

klasický program: pravidla + vstup → výstup

strojové učení: příklady + měřítko chyby → pravidla

Neuron: jednoduchá výpočetní jednotka

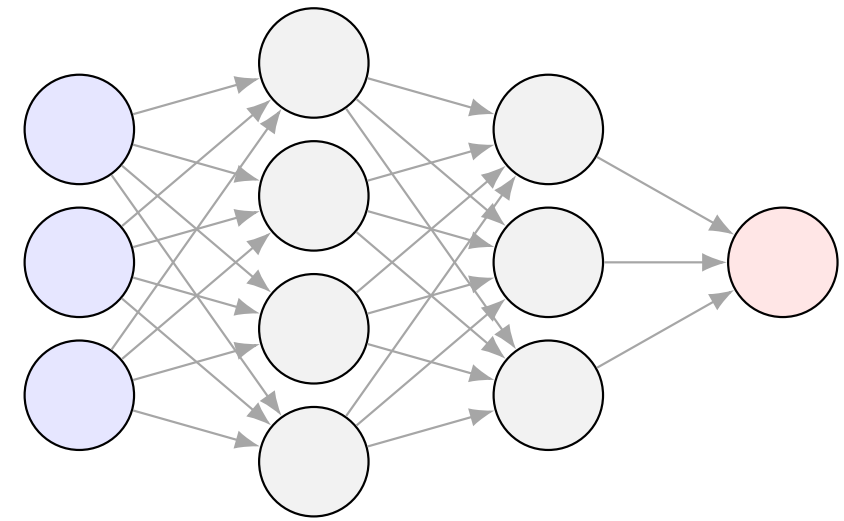
Stavebním prvkem neuronové sítě je neuron. Vstupy vynásobí vahami, sečte a na součet použije jednoduchou aktivační funkci.



$$y = \varphi\left(w_1 x_1 + w_2 x_2 + w_3 x_3\right)$$

Váhy jsou čísla, která se *učí*; všechno ostatní je pevně dané.

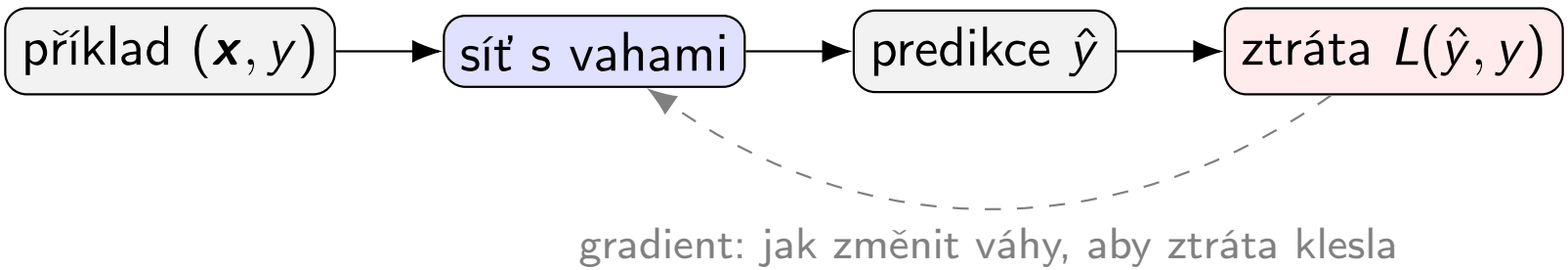
Síť: neurony uspořádané do vrstev



- Výstupy jedné vrstvy jsou vstupy vrstvy následující.
- Celá síť je tak jediná složená funkce: ze vstupu $\boldsymbol{x}$ počítá predikci $\hat{y}$.
- Váhy všech neuronů dohromady jsou parametry sítě.
- Hluboká síť má skrytých vrstev více — odtud „hluboké“ učení.

Jeden neuron toho umí málo; síla vzniká *skládáním*.

Jak se síť učí



Ztrátová funkce jedno číslo: jak moc se predikce liší od požadovaného výstupu.

Gradient pro každou váhu říká, kterým směrem a jak silně ji změnit, aby ztráta klesla.

Učení malá úprava všech vah proti gradientu, opakovaná na mnoha příkladech.

Cíl učení

Trénovací algoritmus tedy hledá parametry, se kterými síť

1. dosahuje malé chyby na trénovacích příkladech,
2. a současně zobecní na dosud neviděná data.

Dvě různé otázky

Malá trénovací chyba je otázka *optimalizace*.
Malá chyba na nových datech je otázka *generalizace*.

Vstup není ve tvaru, v němž se úloha řeší

Obraz je pole pixelů, věta posloupnost slov, zvuk časová řada.

Síť proto nejprve převede vstup $\boldsymbol{x}$ na vnitřní reprezentaci — to je práce skrytých vrstev —

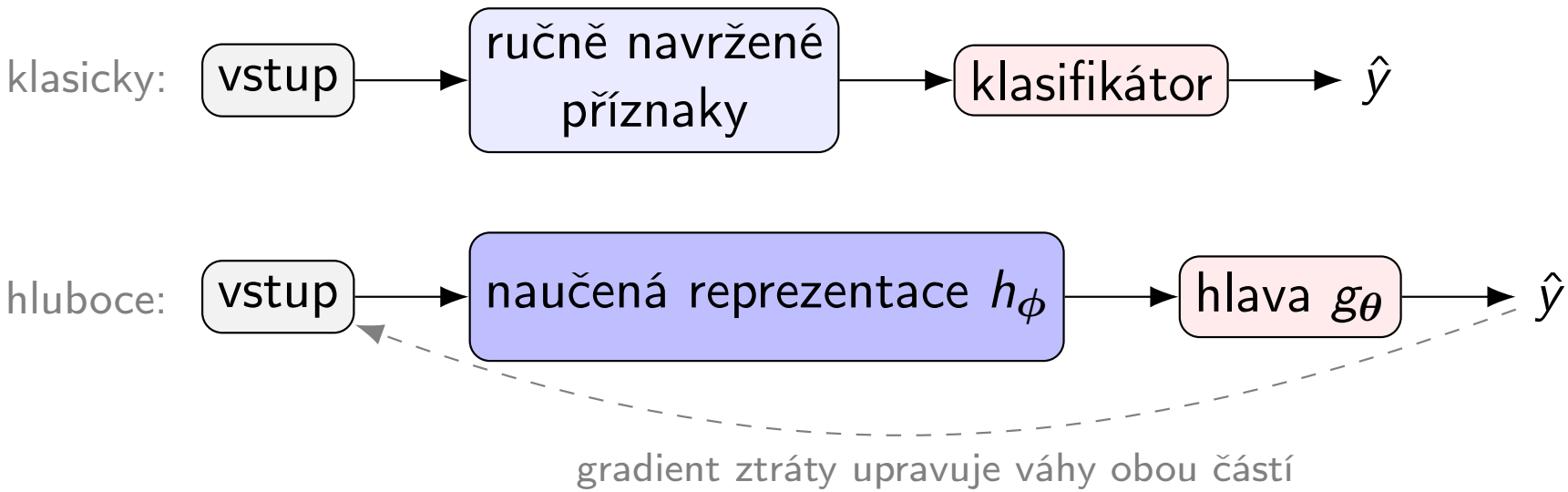
$$\boldsymbol{h} = h_{\boldsymbol{\phi}}(\boldsymbol{x})$$

a teprve z ní poslední vrstva (hlava) počítá predikci

$$\hat{\boldsymbol{y}} = g_{\boldsymbol{\theta}}(\boldsymbol{h}), \qquad f_{\boldsymbol{\theta},\boldsymbol{\phi}} = g_{\boldsymbol{\theta}} \circ h_{\boldsymbol{\phi}}.$$

Váhy $\boldsymbol{\phi}$ i $\boldsymbol{\theta}$ se učí *společně*, minimalizací téže ztrátové funkce. Tomuto principu se říká učení reprezentací (*representation learning*).

Ručně navržené příznaky versus naučená reprezentace

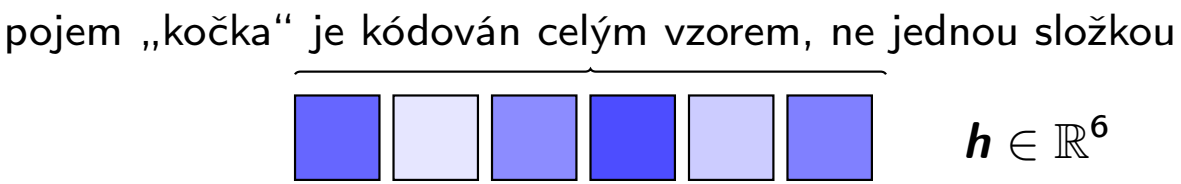


Ruční příznaky umí využít oborovou znalost, ale jejich kvalita je omezena tím, co konstruktér dokázal předem vyjádřit.

Distribuovaná reprezentace

Reprezentace $\boldsymbol{h}$ je vektor výstupů neuronů jedné vrstvy. Jedna vlastnost nemusí odpovídat jedinému neuronu:

- informace bývá rozložena mezi více souřadnic,
- jedna souřadnice se současně podílí na několika vlastnostech.



Důsledek

Interpretovat jednotlivý neuron jako „detektor pojmu“ je zpravidla zjednodušení; vrstvy nemají předem přidělený význam.

Invariance a ekvivariance

Užitečná reprezentace zachová informaci potřebnou pro úlohu a potlačí změny, které pro ni podstatné nejsou.

Invariance

Změna vstupu *nemá* změnit výsledek.

Tentýž předmět vyfotografovaný za slunečna a pod mrakem máme zařadit stejně. Na osvětlení nezáleží.

Ekvivariance

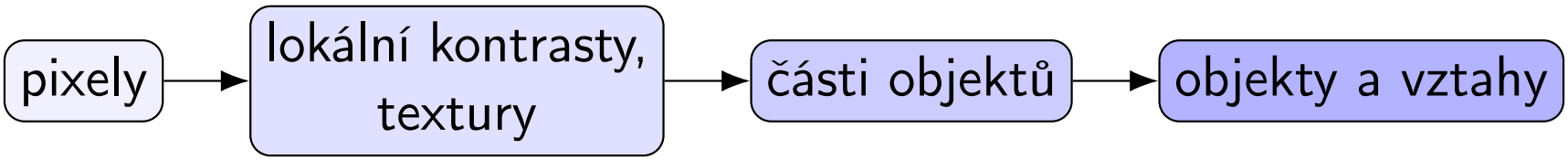
Změna vstupu má výsledek změnit *předvídatelně*.

Posuneme-li obraz o deset pixelů vpravo, má se nalezená poloha předmětu posunout také o deset pixelů vpravo.

Kterou vlastnost chceme, plyne z *úlohy*, ne z modelu: u otázky „co je na obrázku“ je posun nepodstatný, u otázky „kde to je“ podstatný.

Hierarchie vrstev

Každá vrstva staví na reprezentaci té předchozí:



U jazyka: počáteční vektor slova (*embedding*) se průchodem vrstvami mění v reprezentaci závislou na okolním kontextu.

Meze této interpretace

Tato interpretace je užitečná, ale není doslovným popisem každé sítě. Naučená informace může být rozložena napříč celým modelem.

Učení end-to-end a jeho riziko

Společné učení celé kompozice se označuje jako učení end-to-end. Výhoda: reprezentace se přizpůsobí přímo zadanému cíli.

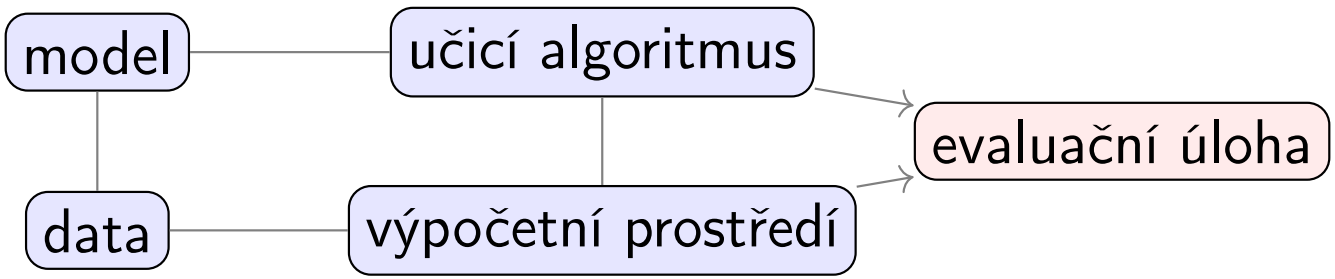
Táž vlastnost je zdrojem rizika. Ztrátová funkce neurčuje, *jakým* signálem má model úlohu řešit.

Příklad 1 (Zkratka v datech)

Pokud je v trénovacích datech pozadí silně korelované s třídou, může síť zakódovat pozadí místo zamýšleného pojmu. Trénovací chyba přitom klesne.

Malá trénovací chyba tedy nedokazuje, že reprezentace zachycuje správnou strukturu problému.

Pokrok vzniká souběhem čtyř složek



Metoda úspěšná v jedné doméně nemusela být praktická v jiné. Samotný architektonický nápad stačil jen zřídka.

Backpropagation je z roku 1986, AlexNet z roku 2012 — týž algoritmus a dvacet šest let. Chyběla data, výpočetní výkon a shoda na tom, co se měří.

Vybrané milníky

- **1943** formální neuron — McCulloch, Pitts
- **1958** perceptron — Rosenblatt
- **1969** kritika omezení — Minsky, Papert
- **1986** backpropagation — Rumelhart, Hinton, Williams
- **1997–98** LSTM — Hochreiter, Schmidhuber; LeNet — LeCun
- **2012** AlexNet — Krizhevsky, Sutskever, Hinton
- **2014–15** GAN — Goodfellow; attention — Bahdanau
- **2017** Transformer — Vaswani a kol.
- **2020** difuzní modely — Ho, Jain, Abbeel

Poznámka k milníkům

- Novější metoda své předchůdce zpravidla *nenahradila*.
- Konvoluční sítě (LeNet, AlexNet) jsou pro obraz účinné dodnes.
- Rekurentní sítě (LSTM) se hodí pro data, která přitékají postupně.
- Při omezených datech bývá jednodušší model lepší volbou než rozsáhlá architektura.

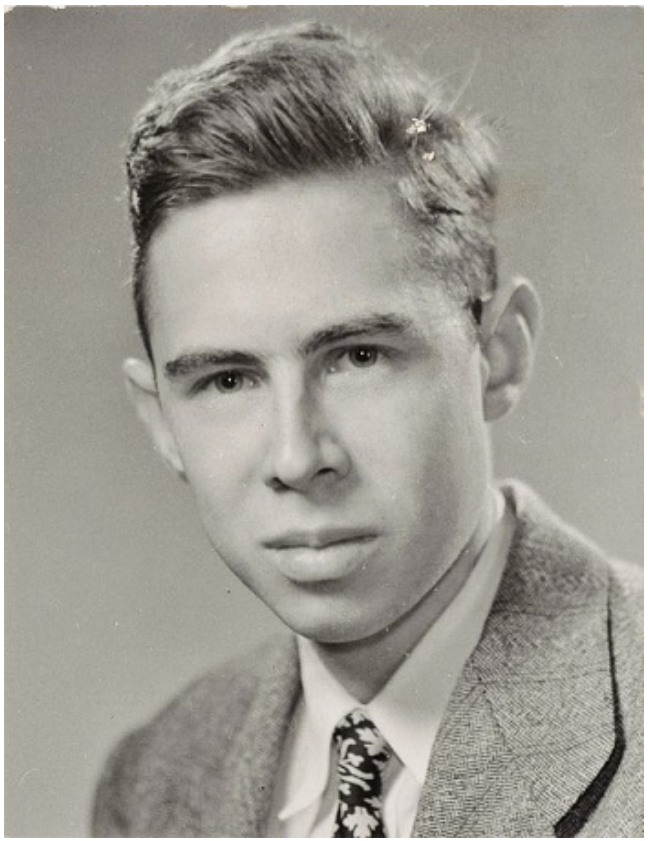
Společný základ: vrstvy neuronů složené za sebe a učené gradientem.
Architektury se liší tím, *jak* jsou neurony propojeny.

Nejjednodušší učicí model

Perceptron je neuron z úvodu přednášky v nejjednodušší podobě: jeden vážený součet a schodková aktivace. Je to lineární binární klasifikátor a historický výchozí bod neuronových sítí.

Jeho jednoduchost dovoluje přesně popsat

- rozhodovací hranici,
- algoritmus učení,
- i podmínky konvergence.



Frank Rosenblatt (1928–1971)

F. Rosenblatt: The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain. *Psychological Review* 65(6), 386–408 (1958).

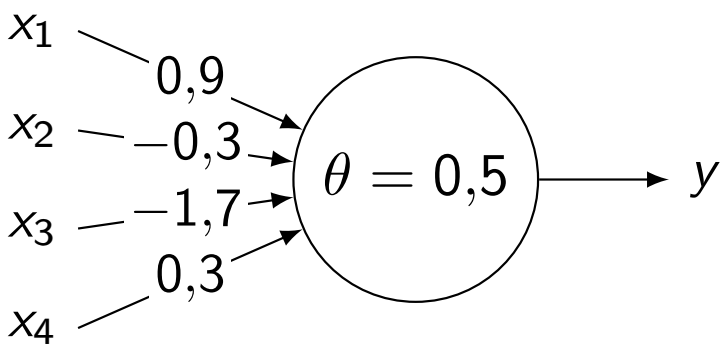
Skripta: kap. 1 *Perceptron*

Definice

Definice 2 (Perceptron)

Perceptron s n vstupy je určen dvojicí: vektorem vah $\boldsymbol{w} = \langle w_1, \ldots, w_n \rangle$ a prahem excitace θ. Počítá funkci $f : \mathbb{R}^n \to \{0, 1\}$,

$$f(\boldsymbol{x}) = \begin{cases} 1 & \text{pokud } \sum_{i=1}^{n} w_i x_i \geq \theta, \\ 0 & \text{jinak.} \end{cases}$$



Hrany nesou váhy, uvnitř uzlu je práh excitace.

Výpočet krok za krokem

Příklad 3 (Jeden dopředný průchod)

Pro váhy $\boldsymbol{w} = \langle 0{,}9,\ -0{,}3,\ -1{,}7,\ 0{,}3 \rangle$, práh $\theta = 0{,}5$ a vstup $\boldsymbol{x} = \langle 1,\ 0{,}5,\ 0,\ 1 \rangle$ vychází

$$\boldsymbol{w} \cdot \boldsymbol{x} = 0{,}9 \cdot 1 + (-0{,}3) \cdot 0{,}5 + (-1{,}7) \cdot 0 + 0{,}3 \cdot 1 = 1{,}05.$$

To je více než θ, a tedy $f(\boldsymbol{x}) = 1$.

Dvě funkce v jednom uzlu

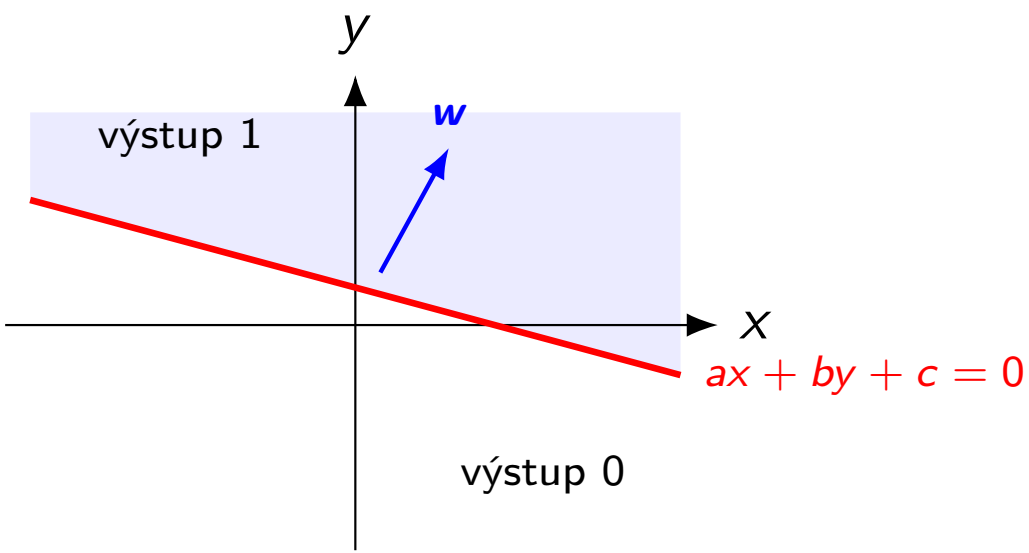
Agregační funkce $\mathbb{R}^n \to \mathbb{R}$: vážený součet $\boldsymbol{w}^T \boldsymbol{x}$.
Aktivační funkce $\mathbb{R} \to \{0, 1\}$: schodková funkce s prahem θ.

Perceptron dělí prostor nadrovinou

Pro dva vstupy, $\boldsymbol{w} = \langle a, b \rangle$ a práh $\theta = -c$, vrací perceptron

$$f(\langle x, y \rangle) = \begin{cases} 1 & \text{pokud } ax + by + c \geq 0, \\ 0 & \text{jinak.} \end{cases}$$

Rovnice $ax + by + c = 0$ je obecná rovnice přímky.



Analogicky perceptron se třemi vstupy dělí $\mathbb{R}^3$ rovinou atd.

Malý příklad

Příklad 4 (Rozhodovací přímka a dvě klasifikace)

Perceptron s $\boldsymbol{w} = (2, -1)$ a prahem $\theta = 1$ má hranici $2x - y - 1 = 0$.

- Pro bod $(2, 1)$ je vážený součet $3 \geq 1$, tedy výstup 1.
- Pro bod $(0, 1)$ dostaneme $-1 < 1$, tedy výstup 0.

Vliv parametrů

Změna vah *otáčí* hranici, změna prahu ji *posouvá* rovnoběžně.

Práh jako další váha

Nahradíme aktivační funkci s_θ funkcí s_0 (práh v nule), přidáme vstup $x_0 = 1$ a váhu $w_0 = -\theta$.

Před změnou perceptron vracel 1 právě pro

$$\sum_{i=1}^{n} w_i x_i \geq \theta.$$

Po změně vrací 1 právě pro

$$\sum_{i=0}^{n} w_i x_i \geq 0.$$

Obě podmínky jsou ekvivalentní, protože $w_0 x_0 = -\theta$.

Důvod

Algoritmus učení pak pracuje s jediným typem parametru a aktivační funkce nemusí mít samostatný práh. Této složce se říká bias.

Zároveň tím splyne otáčení a posouvání hranice z předchozí strany do jediné operace: upraví-li se váhový vektor, změní se obojí naráz.

Rozšířený vstupní prostor

Ekvivalentně lze rozšířit vstupní prostor: místo $\mathbb{R}^n$ pracujeme s $\{1\} \times \mathbb{R}^n$, tedy

$$\boldsymbol{x} = \langle x_1, \ldots, x_n \rangle \quad \longmapsto \quad \overline{\boldsymbol{x}} = \langle 1, x_1, \ldots, x_n \rangle .$$

Této konvence se budeme držet i v algoritmu učení: prahy do vah, vstupy rozšířené o konstantní jedničku.

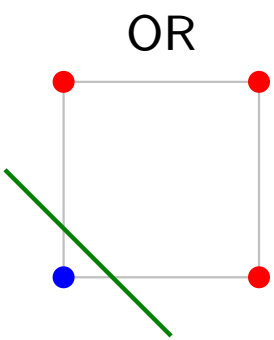
Perceptron je tedy lineární model se schodkovou aktivací. Práh není zvláštní parametr; lze jej zapsat jako váhu konstantního vstupu.

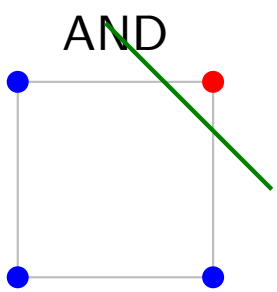
Co se tím získalo

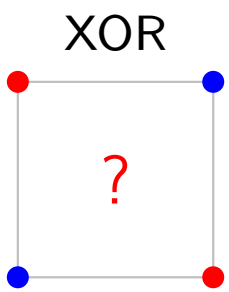
Rozhodování je jediný skalární součin a porovnání s nulou. Hranice sice nutně prochází počátkem — ale v prostoru o $n + 1$ rozměrech; v původním prostoru je to libovolná nadrovina.

Které logické funkce zvládne?

Uvažujme všech 16 booleovských funkcí dvou proměnných. Perceptronem lze spočítat právě ty, u nichž lze body s hodnotou 0 oddělit přímkou od bodů s hodnotou 1.







Ze šestnácti funkcí jich perceptron spočítá čtrnáct. Výjimkou je právě XOR a její negace.

Modrý bod: funkční hodnota 0; červený bod: hodnota 1.

XOR: důkaz sporem

Nechť w_1, w_2 jsou váhy a θ práh perceptronu počítajícího XOR.
Pak musí platit

$$\begin{aligned}
\langle 0,0\rangle \mapsto 0: &\quad 0 < \theta,\\
\langle 1,0\rangle \mapsto 1: &\quad w_1 \geq \theta,\\
\langle 0,1\rangle \mapsto 1: &\quad w_2 \geq \theta,\\
\langle 1,1\rangle \mapsto 0: &\quad w_1 + w_2 < \theta.
\end{aligned}$$

Z první nerovnosti je θ kladné, podle druhé a třetí jsou tedy kladné i w_1 a w_2. Pak ovšem $w_1 + w_2 \geq \theta$, což je spor se čtvrtou nerovností.

Závěr

Žádný perceptron nepočítá XOR. Obdobně pro její negaci — ze šestnácti funkcí dvou proměnných jsou to právě tyto dvě.

Lineární separabilita obecně

Definice 5 (Lineárně separovatelné množiny)

Množiny bodů A a B v n-rozměrném prostoru jsou lineárně separovatelné, pokud existují reálná čísla $w_1, \ldots, w_{n+1}$ taková, že

$$\sum_{i=1}^{n} w_i x_i \geq w_{n+1} \quad \text{pro každý bod } \boldsymbol{x} \in A,$$

$$\sum_{i=1}^{n} w_i x_i < w_{n+1} \quad \text{pro každý bod } \boldsymbol{x} \in B.$$

Logická funkce n proměnných rozděluje vrcholy n-rozměrné hyperkrychle do dvou tříd. Není-li toto rozdělení lineárním řezem, funkce není perceptronem vypočitatelná.

Kolik funkcí je separabilních?

n	separabilních	všech	podíl
2	14	16	87,5 %
3	104	256	40,6 %
4	1 882	65 536	2,9 %

S rostoucí dimenzí podíl rychle klesá: lineární model je čím dál silnější omezení.

Otevřený problém

Vzorec pro počet lineárně separabilních funkcí jako funkci n není přes rozsáhlý výzkum znám.

Perceptron realizuje právě lineárně separabilní funkce. XOR mezi ně nepatří; to byla jedna z příčin útlumu zájmu o neuronové sítě po roce 1969.

M. Minsky, S. Papert: *Perceptrons: An Introduction to Computational Geometry*. MIT Press (1969).

Zadání úlohy

Trénovací sada se skládá ze dvou množin P a N v rozšířeném vstupním prostoru. Hledáme vektor $\boldsymbol{w}$, který je oddělí:

$$\boldsymbol{w} \cdot \boldsymbol{x} > 0 \quad \text{pro všechna } \boldsymbol{x} \in P, \qquad \boldsymbol{w} \cdot \boldsymbol{x} < 0 \quad \text{pro všechna } \boldsymbol{x} \in N.$$

Totéž geometricky

Vektor vah $\boldsymbol{w}$ je normálovým vektorem oddělující nadroviny. Podmínka $\boldsymbol{w} \cdot \boldsymbol{x} > 0$ znamená, že $\boldsymbol{w}$ a $\boldsymbol{x}$ svírají úhel menší než $90°$.

Cíle vzorů budeme od této chvíle značit $+1$ a -1 místo 0 a 1. Je to přeznačení dvou tříd, rozhodovací hranice je táž; opravné pravidlo se tím dá napsat symetricky, jednou s plus a jednou s minus.

Algoritmus

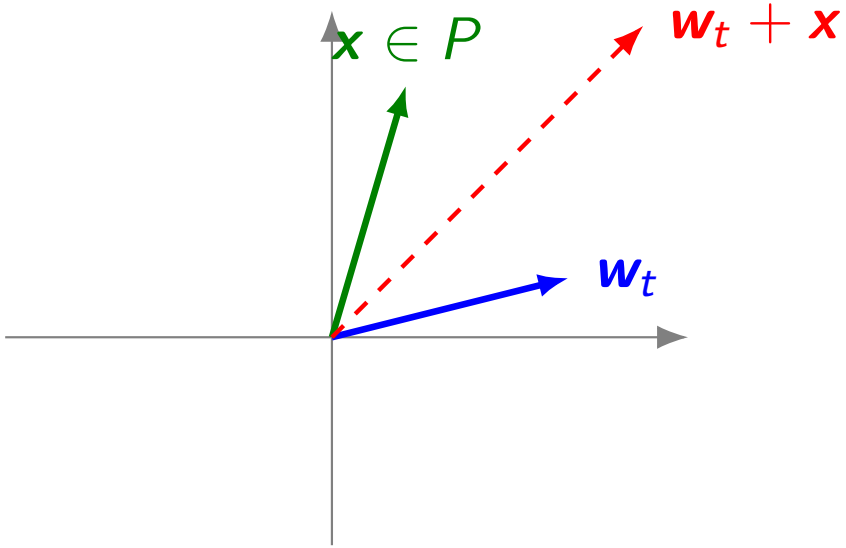
Algorithm 1: Učení perceptronu

Input: konečné množiny P a N v rozšířeném prostoru
Output: váhový vektor $\boldsymbol{w}$ oddělující P a N
zvol $\boldsymbol{w}_0$ náhodně; $t \leftarrow 0$;
while *existuje chybně zařazený vzor* **do**
  vyber vzor $\boldsymbol{x}$;
  if $\boldsymbol{x} \in P$ *a* $\boldsymbol{w}_t \cdot \boldsymbol{x} \leq 0$ **then**
    $\boldsymbol{w}_{t+1} \leftarrow \boldsymbol{w}_t + \boldsymbol{x}$;
  end
  if $\boldsymbol{x} \in N$ *a* $\boldsymbol{w}_t \cdot \boldsymbol{x} \geq 0$ **then**
    $\boldsymbol{w}_{t+1} \leftarrow \boldsymbol{w}_t - \boldsymbol{x}$;
  end
  $t \leftarrow t + 1$;
end

Nerovnosti jsou *neostré*: nulový skalární součin znamená, že vzor leží přesně na hranici, a to za správné zařazení nepovažujeme. Zastavíme se, jakmile celý průchod daty proběhne bez opravy.

Minsky, Papert: *Perceptrons.* MIT Press (1969); rozbor Rojas (1996), kap. 4.

Proč přičítání a odčítání funguje



Je-li $x \in P$ zařazen chybně, svírají w_t a x úhel větší než 90°. Přičtení x otočí váhový vektor *směrem* k x.

Pro $x \in N$ se postupuje symetricky: x se odečte, tedy váhový vektor se otočí *od* něj.

Vektory z P otáčejí váhový vektor jedním směrem, vektory z N opačným. Existuje-li řešení, algoritmus je najde po konečném počtu kroků.

Jeden opravný krok

Příklad 6 (Aktualizace vah)

Nechť rozšířený kladný vzor je $\boldsymbol{x} = (1, 2, -1)$ a aktuální váhy $\boldsymbol{w} = (0, -1, 0)$.

- Protože $\boldsymbol{w}^T \boldsymbol{x} = -2 < 0$, je vzor zařazen chybně.
- Aktualizace dává $\boldsymbol{w} \leftarrow \boldsymbol{w} + \boldsymbol{x} = (1, 1, -1)$.
- Pro tentýž vzor je nový skalární součin $1 + 2 + 1 = 4 > 0$.

Pozor

Krok opravil *tento* vzor, ale mohl změnit klasifikaci jiných bodů. Proto algoritmus data opakovaně prochází.

Demonstrace: naučit se AND

Vezmeme bipolární kódování, $x_i \in \{-1, +1\}$, a rozšířený vstup $\overline{\boldsymbol{x}} = \langle 1, x_1, x_2 \rangle$. Pak

$$P = \{(1,1,1)\}, \qquad N = \{(1,1,-1),\ (1,-1,1),\ (1,-1,-1)\}.$$

Začneme s $\boldsymbol{w}_0 = (0,0,0)$ a vzory procházíme cyklicky.

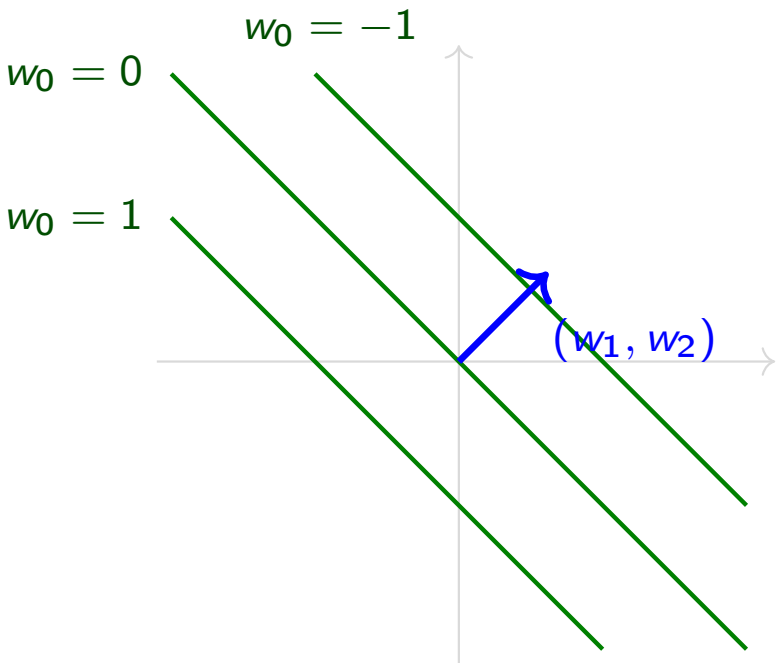
krok	$\overline{\boldsymbol{x}}$	třída	$\boldsymbol{w} \cdot \overline{\boldsymbol{x}}$	akce	nové $\boldsymbol{w}$
1	$(1,1,1)$	P	0	$\boldsymbol{w} + \overline{\boldsymbol{x}}$	$(1,1,1)$
2	$(1,1,-1)$	N	1	$\boldsymbol{w} - \overline{\boldsymbol{x}}$	$(0,0,2)$
3	$(1,-1,1)$	N	2	$\boldsymbol{w} - \overline{\boldsymbol{x}}$	$(-1,1,1)$
4	$(1,-1,-1)$	N	-3	—	$(-1,1,1)$
5	$(1,1,1)$	P	1	—	$(-1,1,1)$
6	$(1,1,-1)$	N	-1	—	$(-1,1,1)$
7	$(1,-1,1)$	N	-1	—	$(-1,1,1)$

Tři opravy, pak celý průchod beze změny — to je zastavovací podmínka.

Váhový vektor má tři složky

Rozšířený vstup je $\overline{\boldsymbol{x}} = \langle 1, x_1, x_2 \rangle$, váhy tedy také tři: $\boldsymbol{w} = (w_0, w_1, w_2)$. Rozhodovací hranice je množina bodů, kde

$$\boldsymbol{w} \cdot \overline{\boldsymbol{x}} = w_0 + w_1 x_1 + w_2 x_2 = 0.$$



Dvojice (w_1, w_2) je normála hranice — určuje jen její *směr* a to, na které straně je kladný výstup. Modrá šipka na obrázcích kreslí právě ji.

Složka w_0 hranicí posouvá, aniž by změnila její směr. Všechny tři zelené přímky mají totožnou normálu $(1, 1)$ a liší se pouze v w_0.

Proto nestačí sledovat šipku: dva stavy se mohou lišit, i když šipka míří stejně.

Co znamená „oprava vzorem“

Algoritmus prochází vzory a u každého se rozhodne, zda váhy upraví:

cíl vzoru	podmínka	úprava
$+1$	$\boldsymbol{w} \cdot \overline{\boldsymbol{x}} \leq 0$ (je na špatné straně)	$\boldsymbol{w} \leftarrow \boldsymbol{w} + \overline{\boldsymbol{x}}$
-1	$\boldsymbol{w} \cdot \overline{\boldsymbol{x}} \geq 0$ (je na špatné straně)	$\boldsymbol{w} \leftarrow \boldsymbol{w} - \overline{\boldsymbol{x}}$
jinak (vzor je zařazen správně)		váhy se nemění

Oprava vzorem $\overline{\boldsymbol{x}}$ je tedy ten krok, ve kterém algoritmus použil k úpravě vah právě vzor $\overline{\boldsymbol{x}}$. V popiscích obrázků uvádíme jen souřadnice (x_1, x_2); první složka rozšířeného vstupu je vždy 1.

Oprava mění směr i posun zároveň

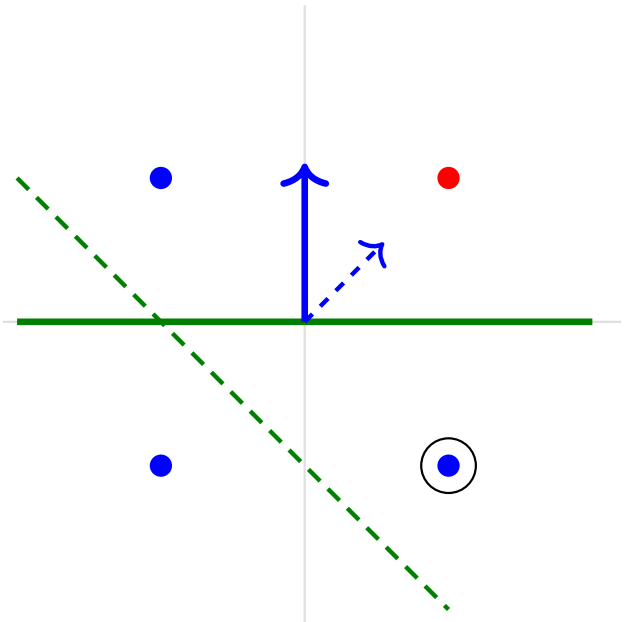
Přičtení vzoru se totiž provede po složkách:

$$\boldsymbol{w} + \overline{\boldsymbol{x}} = (w_0 + 1,\ w_1 + x_1,\ w_2 + x_2).$$

První složka posune hranici, druhá a třetí otočí normálu. Na obrázku se proto hranice zároveň natočí i odsune.

Jedna oprava podrobně

Vezměme druhou opravu z tabulky. Váhy jsou $\boldsymbol{w} = (1, 1, 1)$ a na řadu přichází vzor $\overline{\boldsymbol{x}} = (1, 1, -1)$, tedy bod $(1, -1)$ s cílem -1.



čárkovaně před opravou, plně po ní

Kontrola. $\boldsymbol{w} \cdot \overline{\boldsymbol{x}} = 1 + 1 - 1 = +1$. Cíl je -1, ale výstup vyjde $+1$ — vzor je na špatné straně.

Úprava. Cíl je -1, vzor se tedy odečte:

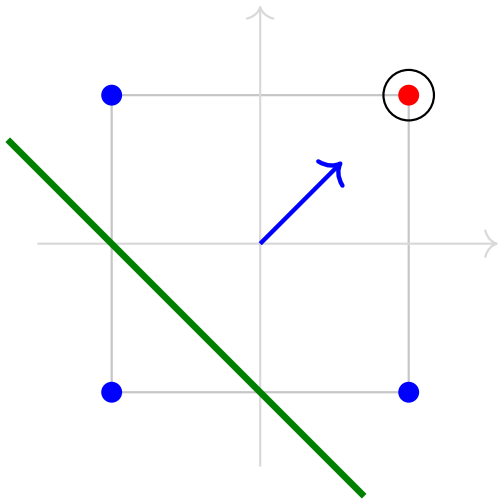
$$\boldsymbol{w} \leftarrow (1, 1, 1) - (1, 1, -1) = (0, 0, 2).$$

Účinek. Hranice $x_1 + x_2 = -1$ se změnila na $x_2 = 0$ a normála se otočila z $(1, 1)$ na $(0, 2)$.

Ověření. Pro tentýž vzor je nově $\boldsymbol{w} \cdot \overline{\boldsymbol{x}} = 0 + 0 - 2 = -2 < 0$, tedy správně.

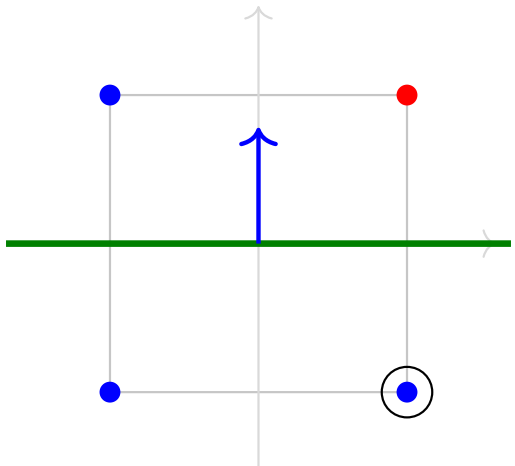
Průběh učení graficky

Zelená přímka je aktuální hranice $\boldsymbol{w} \cdot \overline{\boldsymbol{x}} = 0$, modrá šipka její normála (w_1, w_2). Červený bod má cíl $+1$, modré body -1; kroužek označuje vzor, kterým se v daném kroku opravovalo.



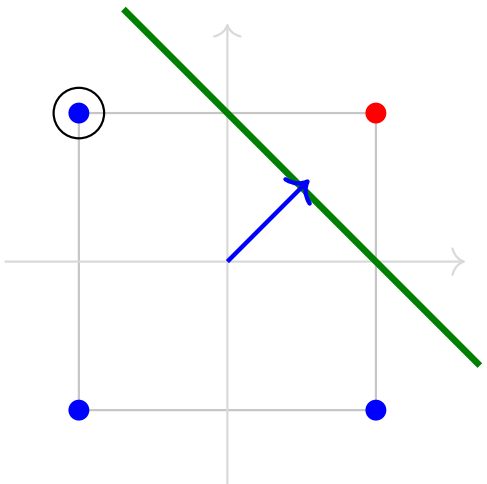
po 1. opravě
vzor $(1, 1)$, cíl $+1$

$\boldsymbol{w} = (1, 1, 1)$



po 2. opravě
vzor $(1, -1)$, cíl -1

$\boldsymbol{w} = (0, 0, 2)$



po 3. opravě
vzor $(-1, 1)$, cíl -1

$\boldsymbol{w} = (-1, 1, 1)$

První a třetí panel: stejná normála, jiná hranice — liší se jen složka w_0.

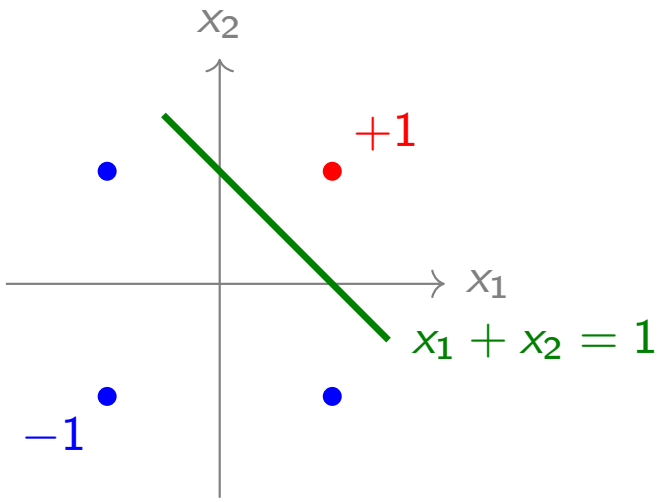
Výsledek

Naučené váhy $\boldsymbol{w} = (-1, 1, 1)$
znamenají práh $\theta = 1$
a rozhodovací funkci

$$y = \text{sign}(x_1 + x_2 - 1),$$

tedy hranici $x_1 + x_2 = 1$.

x_1	x_2	net	y
1	1	1	+1
1	−1	−1	−1
−1	1	−1	−1
−1	−1	−3	−1



Na čem počet kroků závisí

Na pořadí vzorů, na inicializaci a na měřítku vstupů: táž úloha v binárním kódování ($x_i \in \{0, 1\}$) potřebuje při stejném pořadí dvacet dva oprav místo tří.

Věta o konvergenci

Věta 7 (Konvergence perceptronu)

Jsou-li množiny P a N konečné a lineárně separovatelné a jsou-li vektory testovány cyklicky, nalezne algoritmus po konečném počtu kroků váhový vektor, který obě množiny odděluje.

Idea důkazu

Sledujeme úhel ρ mezi $\boldsymbol{w}_t$ a normalizovaným řešením $\boldsymbol{w}^*$:

$$\cos\rho \geq \frac{\boldsymbol{w}^* \cdot \boldsymbol{w}_0 + (t+1)\delta}{\sqrt{\|\boldsymbol{w}_0\|^2 + (t+1)}}, \qquad \delta = \min_{\boldsymbol{p}\in P'} \boldsymbol{w}^* \cdot \boldsymbol{p} > 0.$$

Pravá strana roste jako $\sqrt{t}$, ale $\cos\rho \leq 1$. Počet aktualizací proto musí být konečný.

Veličina δ je odstup (*margin*) — nejmenší vzdálenost vzoru od oddělující nadroviny; kladná je právě díky separabilitě.

Důkaz v této podobě: R. Rojas: *Neural Networks: A Systematic Introduction.* Springer (1996), kap. 4.

Dvě praktické poznámky

Inicializace

Dobrou heuristikou je průměr kladných vstupních vektorů mínus průměr záporných; takový vektor leží často blízko oblasti řešení.

Samovolně klesající krok učení

Vektory nejsou normalizované, takže každá aktualizace otáčí $\boldsymbol{w}$ o jiný úhel. Jak $\|\boldsymbol{w}\|$ roste, otáčí jej každá další korekce o méně — algoritmus polohu „ladí“. Perceptron tak má jakousi automatickou učicí konstantu, která během učení klesá.

Co když data separabilní nejsou

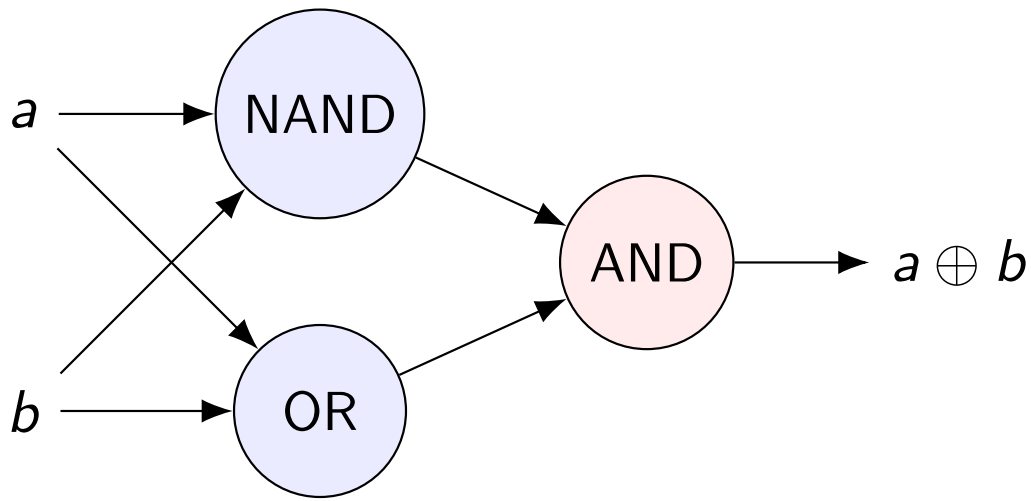
- Algoritmus se nezastaví; váhy oscilují mezi konfliktními vzory.
- Z logu učení to poznáme podle toho, že počet chyb neklesá k nule, ale kolísá kolem nenulové hodnoty.
- Řešením je jiná ztrátová funkce a jiný model.

Učení perceptronu je opravné pravidlo: chybně zařazený vzor otočí váhový vektor směrem k sobě (nebo od sebe). Pro separabilní data konverguje v konečném počtu kroků, pro neseparabilní se nezastaví.

XOR složením tří perceptronů

Funkce AND, NAND a OR jsou lineárně separabilní a pro každé $a, b \in \{0, 1\}$ platí

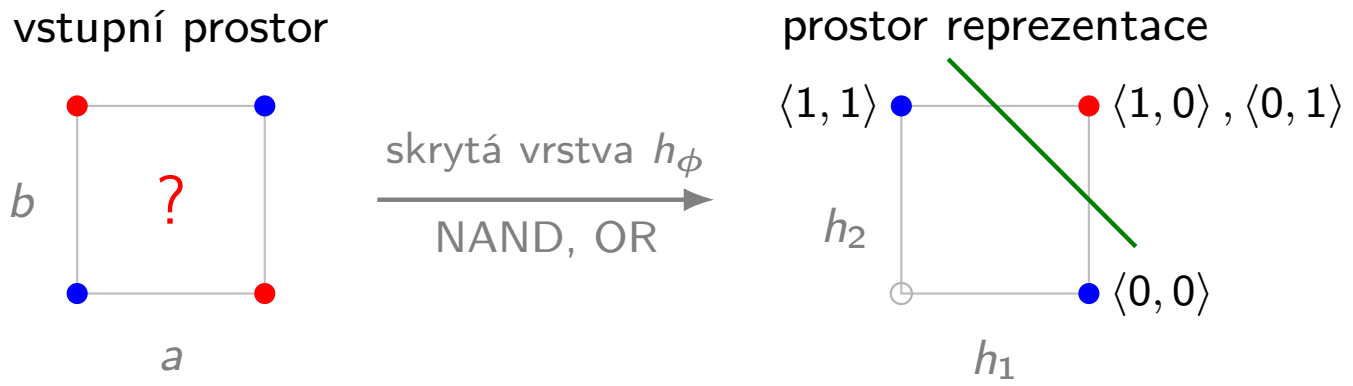
$$a \,\mathrm{XOR}\, b = (a \,\mathrm{NAND}\, b)\,\mathrm{AND}(a \,\mathrm{OR}\, b).$$



Jediná vrstva nestačí; *složením* vrstev vzniká nelineární rozhodovací hranice.

Skrytá vrstva vyrobila novou reprezentaci

První vrstva převádí vstup $\langle a, b \rangle$ na $\boldsymbol{h} = \langle h_1, h_2 \rangle = \langle a \,\mathrm{NAND}\, b,\ a \,\mathrm{OR}\, b \rangle$.



Modrý bod: XOR je 0; červený: 1. U bodů vpravo jsou vstupy, které se do nich zobrazily.

V prostoru $\boldsymbol{h}$ už úloha lineárně separabilní *je* — poslední perceptron (AND) ji rozhodne jedinou přímkou.

Učení reprezentací v malém

Skrytá vrstva je h_ϕ, poslední neuron hlava $g_{\boldsymbol{\theta}}$. Zde jsou váhy navržené ručně; hluboké učení je hledá *samo*.

Pro zbytek kurzu:

Jak ale váhy skryté vrstvy *naučit*?

- Perceptronové pravidlo na to nestačí: skrytému neuronu nikdo neřekne, co měl vrátit, a schodková aktivace není diferencovatelná.
- Potřebujeme hladkou aktivaci a obecný způsob, jak spočítat gradient složené funkce.
- Příští přednáška: dopředné sítě a algoritmus backpropagation.

M. Minsky, S. Papert: *Perceptrons.* MIT Press (1969).
D. E. Rumelhart, G. E. Hinton, R. J. Williams: Learning Representations by Back-Propagating Errors. *Nature* 323, 533–536 (1986).