Umělá inteligence

Jan Konečný

3. července 2025

Co je vlastně AI?

„AI is whatever hasn't been done yet.“

(Umělá inteligence je to, co ještě nebylo uděláno.)

- Toto je zkratka filozofického a kulturního jevu, který pojmenovali odborníci jako **AI effect**.

Významné milníky v AI

Rok	Milník
1966	**ELIZA**
1997	**Deep Blue poráží Kasparova**
2011	**Watson poráží lidské šampiony v Jeopardy!**
2016	**AlphaGo poráží Lee Sedola**
2022–23	**ChatGPT a generativní AI**

Co je to umělá inteligence? Dvě takové dimenze

Historicky výzkumníci sledovali několik různých verzí AI.

- inteligence jako věrná podoba *lidské* činnosti
- inteligence jako *racionalita* – volně řečeno, dělat „správnou věc“.

Pohled na samotný předmět inteligence se také liší:

- vlastnost vnitřních myšlenkových procesů a uvažování
- inteligentní chování – vnější charakterizace.

Z těchto dvou dimenzí – lidské vs. racionální a myšlení vs. chování – existují čtyři možné kombinace.

Racionální a lidské opravdu není totéž.

> *„Když jednáme s lidmi, uvědomme si, že nejednáme s logickými bytostmi. Jednáme s bytostmi plnými emocí, bytostmi plnými předsudků a bystostmi motivovanými pýchou a ješitností.“*
>
> *When dealing with people, remember you are not dealing with creatures of logic, but with creatures bristling with prejudice and motivated by pride and vanity.*

Dale Carnegie
How to Win Friends and Influence People
(Jak získávat přátele a působit na lidi)

Jednání jako člověk

Turingův test (imitation game) – byl navržen jako myšlenkový experiment, který by se vyhnul filozofické vágnosti otázky „Může stroj myslet?“

- AI testem projde, pokud lidský tazatel po položení několika písemných otázek nemůže říct, zda písemné odpovědi pocházejí od člověka nebo od počítače.

Jednání jako člověk

Turingův test (imitation game) – byl navržen jako myšlenkový experiment, který by se vyhnul filozofické vágnosti otázky „Může stroj myslet?“

- AI testem projde, pokud lidský tazatel po položení několika písemných otázek nemůže říct, zda písemné odpovědi pocházejí od člověka nebo od počítače.

Turingův test: počítač bude potřebovat následující schopnosti:

- *zpracování přirozeného jazyka*
- *reprezentace znalostí*
- *automatizované uvažování*
- *strojové učení*

úplný Turingův test – vyžaduje interakci s předměty a lidmi v reálném světě.

K úspěšnému zvládnutí úplného Turingova testu bude AI (robot) potřebovat navíc:

- *počítačové vidění a rozpoznávání řeči*
- *robotika*

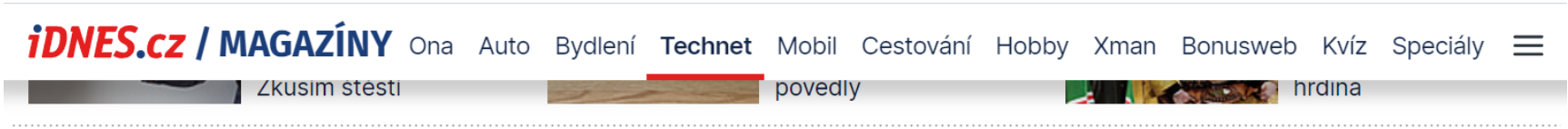
Těchto 6 disciplín tvoří většinu výzkumu AI.

Výzkumníci AI věnují celkem málo úsilí tomu, aby prošli Turingovým testem, protože věří, že je důležitější studovat základní principy inteligence.

Analogie:

vs.

vsuvka:



Počítač přesvědčivě napodobil člověka. Hrál si na puberťáka

9. června 2014 18:45, aktualizováno 23:07

Počítačovému programu se povedlo přesvědčit třetinu lidských rozhodčích, že není počítačový program. V netradiční soutěži došlo poprvé k překonání 30 % hranice úspěšného blafování. Program komunikoval s rozhodčími přes textové zprávy a na přesvědčení měl 5 minut.

Scott: *Which is bigger, a shoebox or Mount Everest?*

Eugene: *I can't make a choice right now. I should think it out later. And I forgot to ask you where you are from. . .*

Scott: *How many legs does a camel have?*

Eugene: *Something between 2 and 4. Maybe, three? :-))) By the way, I still don't know your specialty – or, possibly, I've missed it?*

Scott: *How many legs does a millipede have?*

Eugene: *Just two, but Chernobyl mutants may have up two five. I know you are supposed to trick me.*

Scott: *No, I need to know that you're not a chatbot. Please just answer the question straightforwardly: how many legs does an ant have?*

Eugene: *Something between 2 and 4. Maybe three? :-))) Oh, what a fruitful conversation ;-)*

Myslet jako člověk

Abychom mohli říci, že program myslí jako člověk, musíme vědět, jak lidé myslí. O lidském myšlení se můžeme učit třemi způsoby:

- introspekce – pokoušíme se zachytit naše vlastní myšlenky, jak jdou kolem;
- psychologické experimenty – pozorování osoby v akci;
- zobrazování mozku – pozorování mozku v akci.

Kognitivní věda – Interdisciplinární oblast spojuje počítačové modely z umělé inteligence a experimentální techniky z psychologie k vytvoření přesných a testovatelných teorií o lidské mysli.

Myslet racionálně („Zákony myšlení“)

- **1965**: Programy teoreticky schopné řešit jakýkoliv řešitelný problém popsaný v logické notaci.
- **Logická tradice v AI**: Snaží se vytvářet inteligentní systémy na základě logických programů.
- **Omezení konvenční logiky**: Vyžaduje přesné/jisté znalosti o světě, což je zřídka k dispozici.
- **Teorie pravděpodobnosti**: Umožňuje myšlení s nejistými informacemi.

Jednat racionálně (přístup racionálního agenta)

- **Agent**: Entita, která jedná.
  Od AI se chce víc, než „jednání“: autonomie, vnímání prostředí, adaptace.
- **Racionální agent**: Maximální výstup či (v případě nejistoty) maximální očekávaný výstup.
- **Racionální uvažování vs. racionální jednání**: Racionální uvažování (z předch. slajdu) vede automaticky k racionálnímu jednání.
  Na druhou stranu existuje racionální jednání, kterému nepředchází důkladné racionální uvažování.

Neuronové sítě a jejich význam

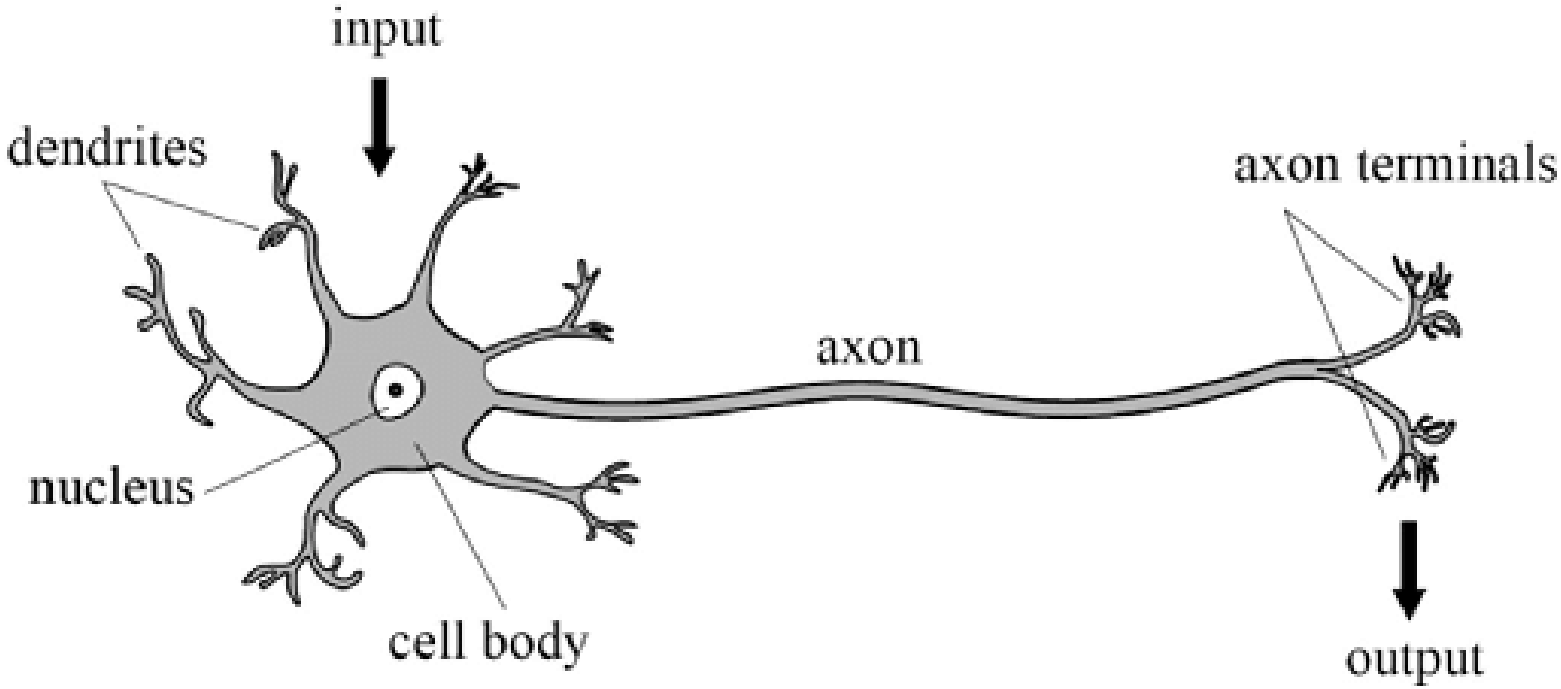
- Neuronové sítě inspirované biologií jsou základem moderní umělé inteligence.
- Významně ovlivňují technologie, vědu i společnost.
- Zaměříme se na jejich principy, aplikace a historický vývoj.
- Dále probereme klíčové osobnosti a vybrané architektury.

Prvopočátek: Umělý neuron

- 1943: Warren McCulloch (neurofyziolog) Walter Pitts (matematik) – první matematický model neuronu, který se choval jako jednoduchá logická jednotka
- Aby popsali, jak by mohly fungovat neurony v mozku, vymodelovali jednoduchou neuronovou síť pomocí elektrických obvodů.

W. S. McCulloch and W. Pitts
A logical calculus of the ideas immanent in nervous activity.
Bulletin of Mathematical Biophysics, 5, 115–137 (1943).

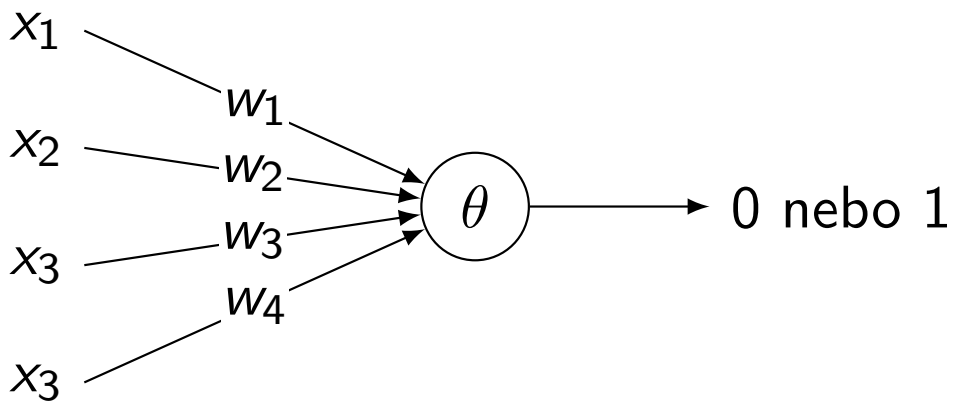
Perceptron – analogie z biologickým neuronem



Pro zájemce:

J. Feldman, R. Rojas
Neural Networks: A Systematic Introduction
Springer Berlin Heidelberg, 2013

Perceptron



Funkce perceptronu

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

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

Funkce perceptronu

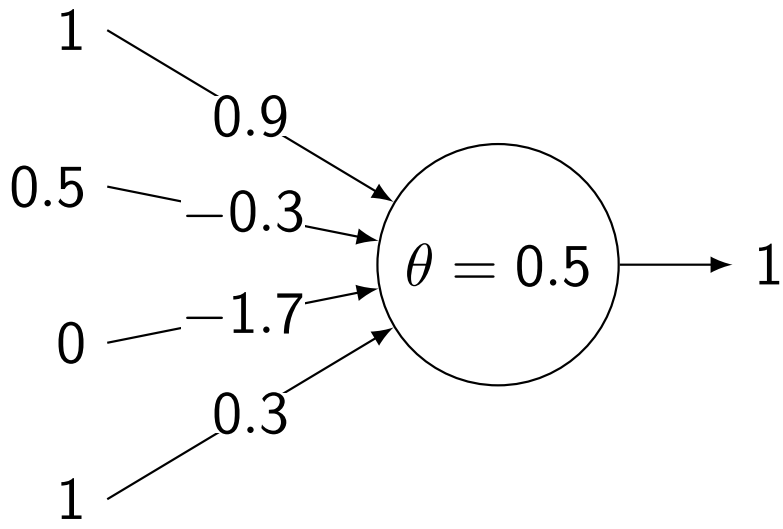
$$f_{\boldsymbol{w},\theta}(\boldsymbol{x}) = \begin{cases} 1 & \text{pokud } \sum_{i=1}^{n} w_i x_i \geq \theta \\ 0 & \text{jinak} \end{cases}$$

- $\boldsymbol{w}$ je vektor vah $\langle w_1, w_2, w_3, w_4 \rangle$
- $\boldsymbol{x}$ je vektor vstupních signálů $\langle x_1, x_2, x_3, x_4 \rangle$

Vlastně bychom to mohli zapsat pomocí skalárního součinu:

$$f_{\boldsymbol{w},\theta}(\boldsymbol{x}) = \begin{cases} 1 & \text{pokud } \boldsymbol{w} \cdot \boldsymbol{x} \geq \theta \\ 0 & \text{jinak} \end{cases}$$

Příklad

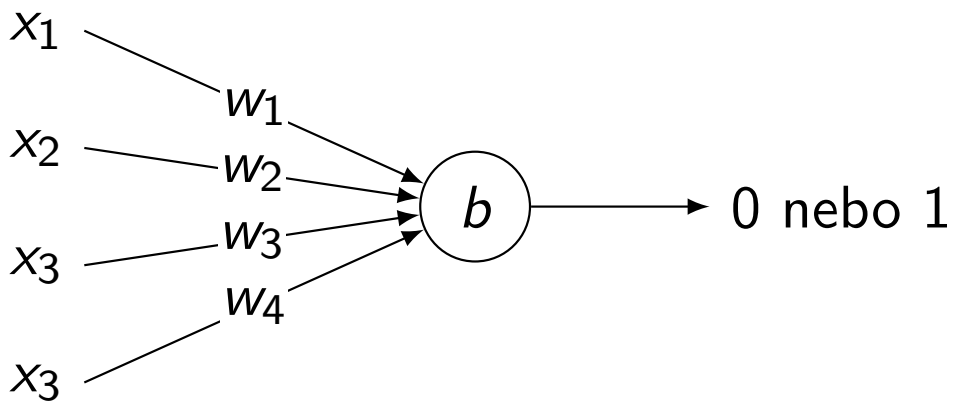


Pro vstupní vektor $\boldsymbol{x} = \langle 1, 0.5, 0, 1 \rangle$ máme:

$$\boldsymbol{w} \cdot \boldsymbol{x} = 0.9 \cdot 1 + (-0.3) \cdot 0.5 + (-1.7) \cdot 0 + 0.3 \cdot 1 = 1.05.$$

Tj. více než θ a tedy $f_{\boldsymbol{w},\theta}(\boldsymbol{x}) = 1$.

Perceptron – mírná úprava (bias místo hranice excitace)

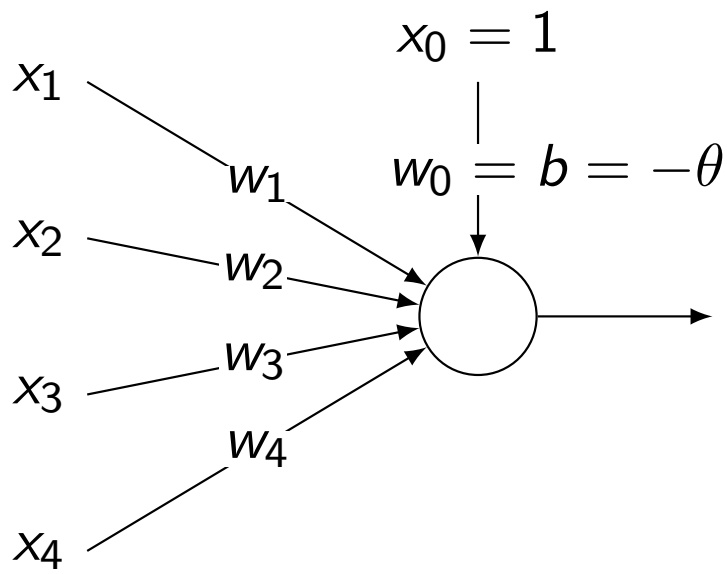


θ (hranice excitace) $\rightarrow -b$ (bias)

Funkce perceptronu

$$f_{\boldsymbol{w},b}(\boldsymbol{x}) = \begin{cases} 1 & \text{pokud } \sum_{i=1}^{n} w_i x_i + b \geq 0 \\ 0 & \text{jinak} \end{cases}$$

Bias je vlastně další váha



- Stačí uvažovat, že do perceptronu jde jeden konstantní vstup $x_0 = 1$ a pak máme:

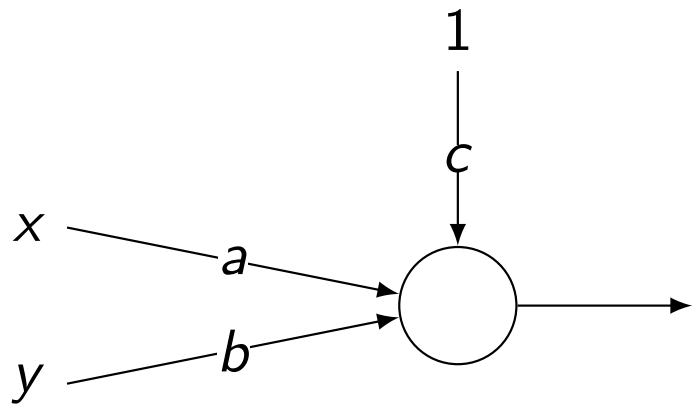
$$f_{\boldsymbol{w}}(\boldsymbol{x}) = \begin{cases} 1 & \text{pokud } \sum_{i=0}^{n} w_i x_i + b \geq 0 \\ 0 & \text{jinak} \end{cases}$$

Geometrický pohled

Uvažujme teď perceptron se dvěma vstupy.

Udělejme jednoduché přeznačení:

- váhy označme a a b
- bias označme c
- vstupní signály označme x a y

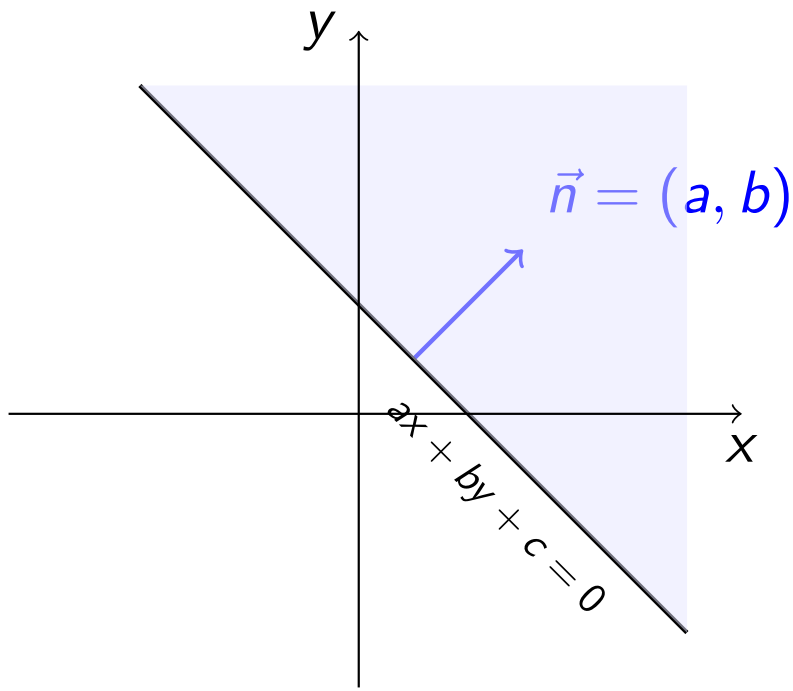


K excitaci pak dojde, pokud

$$ax + by + c \geq 0$$

- To by nám mohlo něco připomínat.

- Rovnice $ax + by + c = 0$ je obecná rovnice přímky (čísla a a b jsou souřadnice normálového vektoru a c je reálné číslo).
- Nerovnice $ax + by + c \geq 0$ pak určuje polorovinu.



- Takový neuron vlastně rozhoduje, jestli vstup $\langle x, y \rangle$ leží v oné polorovině, nebo ne.

Učení (trénování) perceptronu

Máme dvě množiny (ukázkových) vstupů:

- P – ty, pro které chceme, aby perceptron dal 1
- N – ty, pro které chceme, aby perceptron dal 0

Potřebujeme nastavit váhy tak, aby se perceptron opravdu takto choval.

D. O. Hebb
The Organization of Behavior
New York, 1949

F. Rosenblatt
Principles of neurodynamics: Perceptrons and the theory of brain mechanisms. Vol. 55.
Washington, DC: Spartan books, 1962.

Hebbovské učení

Algoritmus učení perceptronu

- $P \leftarrow$ vstupy s cílovou hodnotou 1
- $N \leftarrow$ vstupy s cílovou hodnotou 0
- Náhodně inicializuj $\boldsymbol{w}$
- **Dokud nenastane konvergence:**
  - Náhodně vyber $\boldsymbol{x} \in P \cup N$
  - Pokud $\boldsymbol{x} \in P$ a $\boldsymbol{w}^T \cdot \boldsymbol{x} < 0$, pak aktualizuj $\boldsymbol{w} \leftarrow \boldsymbol{w} + \boldsymbol{x}$
  - Pokud $\boldsymbol{x} \in N$ a $\boldsymbol{w}^T \cdot \boldsymbol{x} \geq 0$, pak aktualizuj $\boldsymbol{w} \leftarrow \boldsymbol{w} - \boldsymbol{x}$

Funkce AND – vstupní data

Trénovací data

- Vstupy: $[x_1, x_2] \in \{0, 1\}^2$
- Cílové hodnoty:
  $[0, 0] \to 0 \quad [0, 1] \to 0 \quad [1, 0] \to 0 \quad [1, 1] \to 1$
- Rozšířený vektor: $\tilde{\boldsymbol{x}} = \begin{bmatrix} x_1 \\ x_2 \\ 1 \end{bmatrix}$

Rozdělení vzorů

- Pozitivní vzory $P = \left\{ \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \right\}$
- Negativní vzory $N = \left\{ \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} \right\}$

Inicializace váhového vektoru

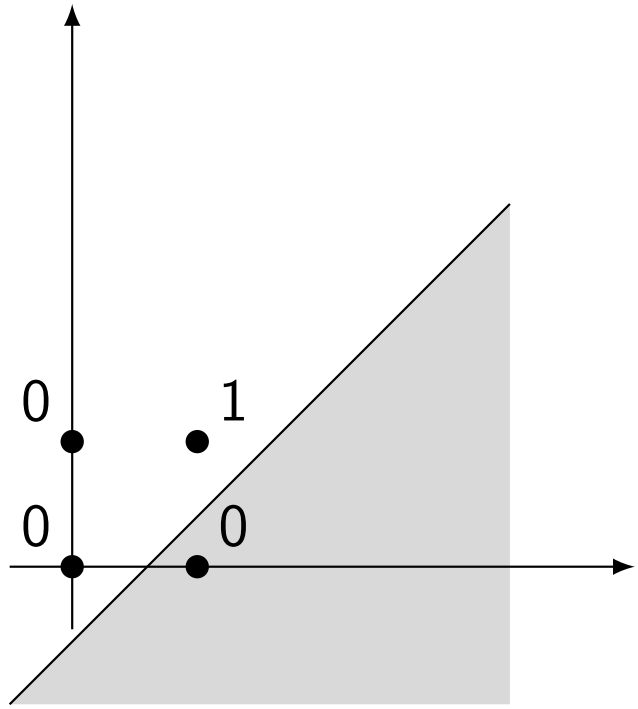
Náhodná inicializace

$$\boldsymbol{w}^{(0)} = \begin{bmatrix} 0.5 \\ -0.5 \\ -0.3 \end{bmatrix}$$

- Počáteční rozhodovací přímka:

$$0.5 \cdot x_1 - 0.5 \cdot x_2 - 0.3 = 0$$

- Vyjádření: $x_2 = x_1 - 0.6$


0
1
0
0

Iterace 1 – vstup $\boldsymbol{x} = [0, 1, 1]^T \in N$

$$\boldsymbol{w}^T \boldsymbol{x} = 0.5 \cdot 0 - 0.5 \cdot 1 - 0.3 = -0.8$$

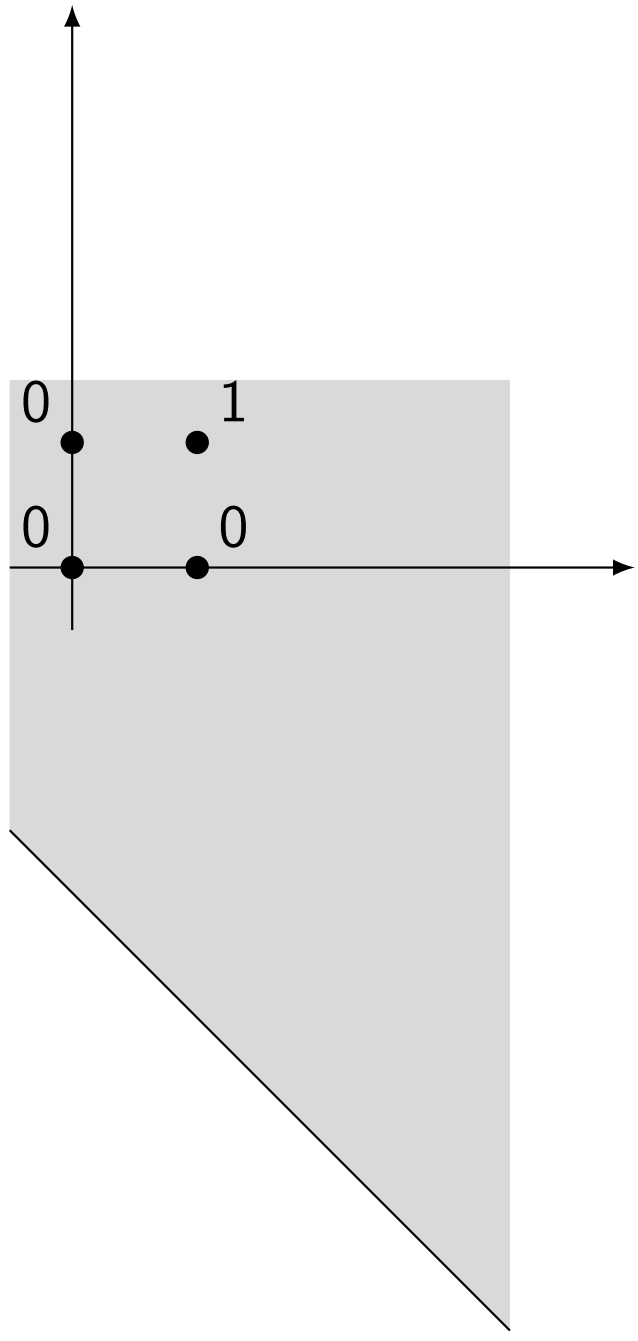
- Výstup je záporný $\rightarrow$ správně klasifikováno.
- Váhy se nemění.
- $\boldsymbol{w}^{(1)} = \boldsymbol{w}^{(0)}$

Iterace 2 – vstup $\boldsymbol{x} = [1, 0, 1]^T \in N$

$$\boldsymbol{w}^T \boldsymbol{x} = 0.5 - 0 - 0.3 = 0.2$$

- Výstup $\geq 0 \rightarrow$ špatně klasifikováno.
- Aktualizace:

$$\boldsymbol{w}^{(2)} = \boldsymbol{w}^{(1)} - \boldsymbol{x} = \begin{bmatrix} 0.5 \\ -0.5 \\ -0.3 \end{bmatrix} - \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} -0.5 \\ -0.5 \\ -1.3 \end{bmatrix}$$

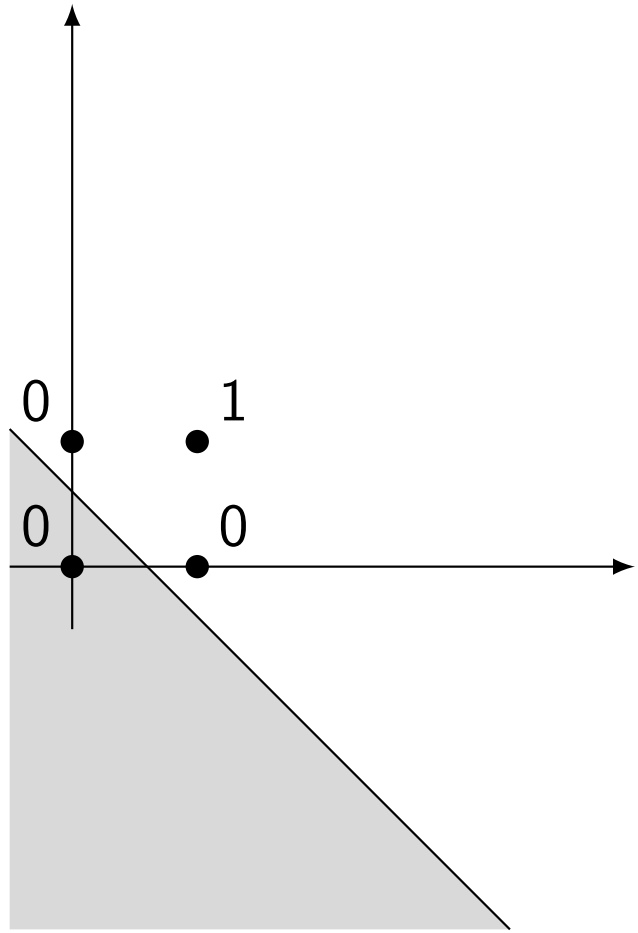

0
1
0
0

Iterace 3 – vstup $\boldsymbol{x} = [1, 1, 1]^T \in P$

$$\boldsymbol{w}^T \boldsymbol{x} = -0.5 - 0.5 - 1.3 = -2.3$$

- Výstup $< 0 \rightarrow$ špatně klasifikováno.
- Aktualizace:

$$\boldsymbol{w}^{(3)} = \boldsymbol{w}^{(2)} + \boldsymbol{x} = \begin{bmatrix} -0.5 \\ -0.5 \\ -1.3 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0.5 \\ 0.5 \\ -0.3 \end{bmatrix}$$


0
1
0
0

Iterace 4 – vstup $\boldsymbol{x} = [0, 1, 1]^T \in N$

$$\boldsymbol{w}^T \boldsymbol{x} = 0 + 0.5 - 0.3 = 0.2$$

- Výstup $\geq 0 \rightarrow$ špatně klasifikováno.
- Aktualizace:

$$\boldsymbol{w}^{(4)} = \boldsymbol{w}^{(3)} - \boldsymbol{x} = \begin{bmatrix} 0.5 \\ 0.5 \\ -0.3 \end{bmatrix} - \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0.5 \\ -0.5 \\ -1.3 \end{bmatrix}$$

Iterace 5 – vstup $\boldsymbol{x} = [1, 0, 1]^T \in N$

$$\boldsymbol{w}^T \boldsymbol{x} = 0.5 - 0 - 1.3 = -0.8$$

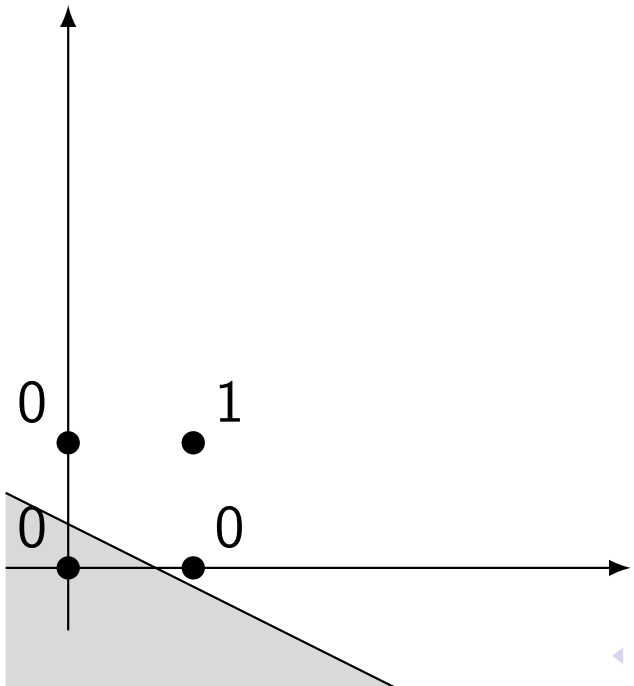
- Výstup $< 0 \rightarrow$ správně klasifikováno.
- Váhy se nemění.

Iterace 6 – vstup $\boldsymbol{x} = [1, 1, 1]^T \in P$

$$\boldsymbol{w}^T \boldsymbol{x} = 0.5 - 0.5 - 1.3 = -1.3$$

- Výstup ¡ 0 $\rightarrow$ špatně klasifikováno.
- Aktualizace:

$$\boldsymbol{w} \leftarrow \boldsymbol{w} + \boldsymbol{x} = \begin{bmatrix} 0.5 \\ -0.5 \\ -1.3 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1.5 \\ 0.5 \\ -0.3 \end{bmatrix}$$

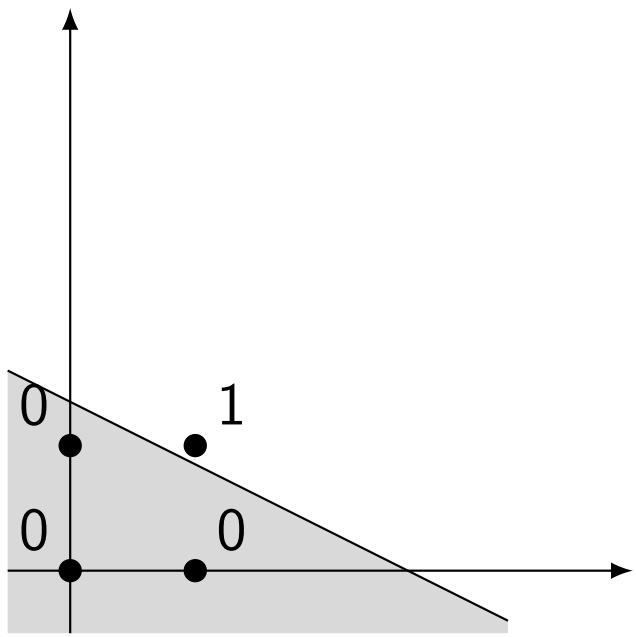


Iterace 7 – vstup $\boldsymbol{x} = [0, 1, 1]^T \in N$

$$\boldsymbol{w}^T \boldsymbol{x} = 0 + 0.5 - 0.3 = 0.2$$

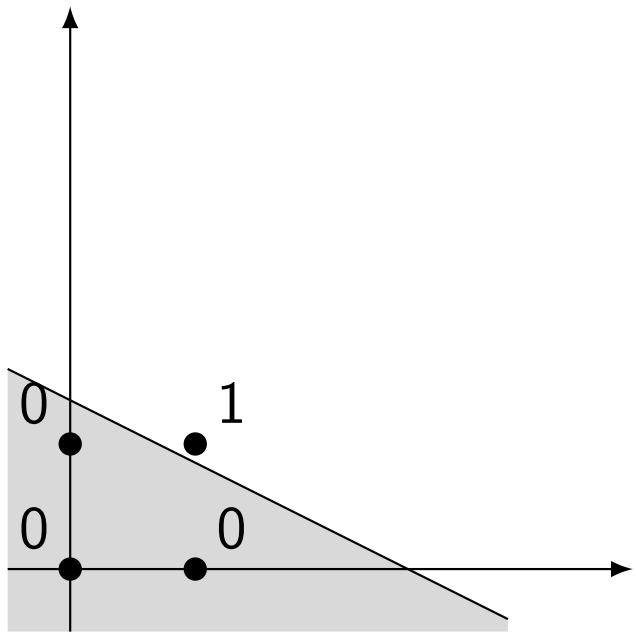
- Výstup $\geq 0 \rightarrow$ špatně klasifikováno.
- Aktualizace:

$$\boldsymbol{w} \leftarrow \boldsymbol{w} - \boldsymbol{x} = \begin{bmatrix} 1.5 \\ 0.5 \\ -0.3 \end{bmatrix} - \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1.5 \\ -0.5 \\ -1.3 \end{bmatrix}$$



Konec – správná klasifikace všech vzorů

- Perceptron nyní správně klasifikuje:
  - všechny negativní vzory: výstup < 0
  - pozitivní vzor $[1, 1, 1]^T$: výstup ≥ 0
- Učení skončilo po 7 váhových aktualizacích.



Co se může a nemůže perceptron naučit?

Pokud jsou P a N **lineárně separabilní**, perceptron se Hebbovským učením naučí dokonale po konečném množství kroků.

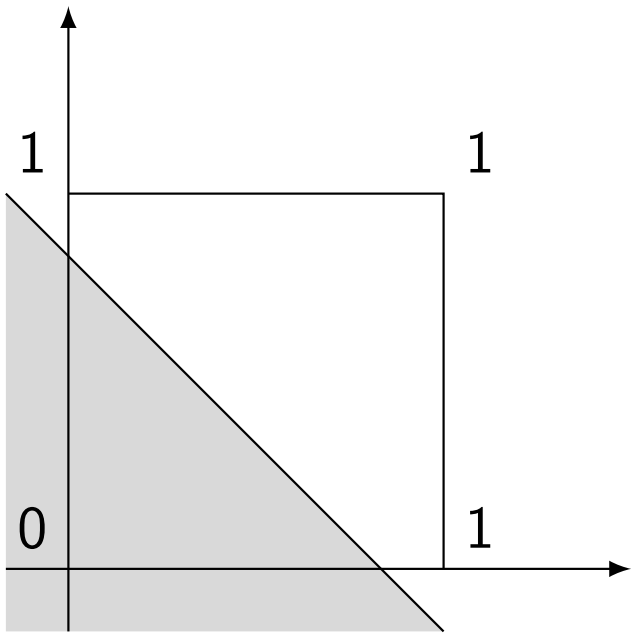
- lineárně separabilní – vektory v P a N lze oddělit přímkou (ve 2d), rovinou (ve 3d), prostorem (ve 4d), ...

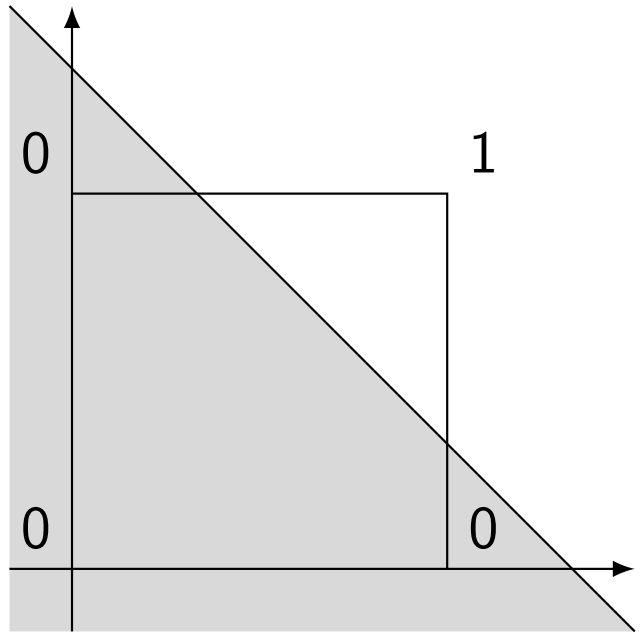
M. L. Minsky, S. Papert
Perceptrons: An Introduction to Computational Geometry.
MIT Press (1969)

Příklad: Logické funkce AND a OR

Pro $a, b \in \{0, 1\}$,

$$a \texttt{ AND } b = a \cdot b$$
$$a \texttt{ OR } b = \min(a + b, 1)$$



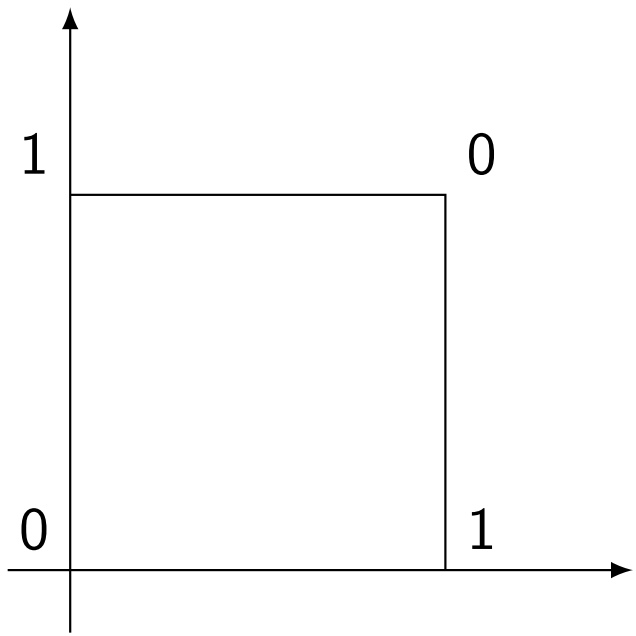


Příklad: Logická funkce XOR

- XOR neboli logická non-ekvivalence

Pro $a, b \in \{0, 1\}$,

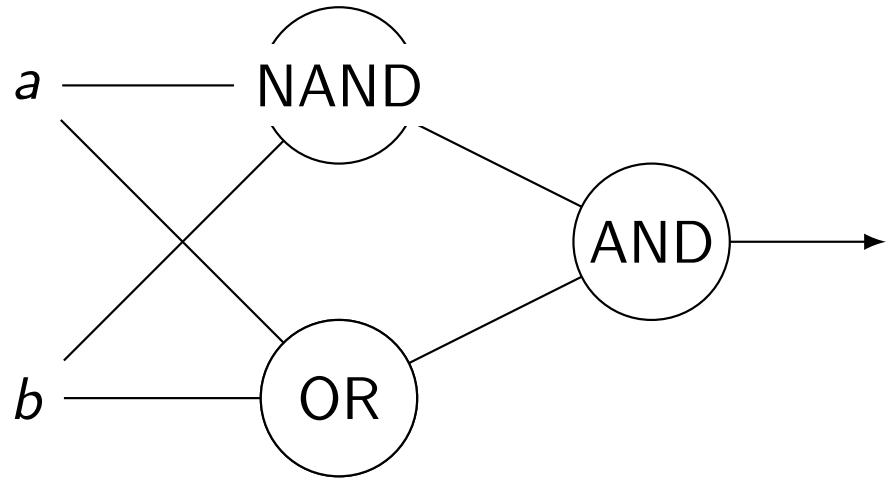
$$a \texttt{ XOR } b = \begin{cases} 0 & \text{pokud } a = b \\ 1 & \text{jinak.} \end{cases}$$



Řešení XOR-problému

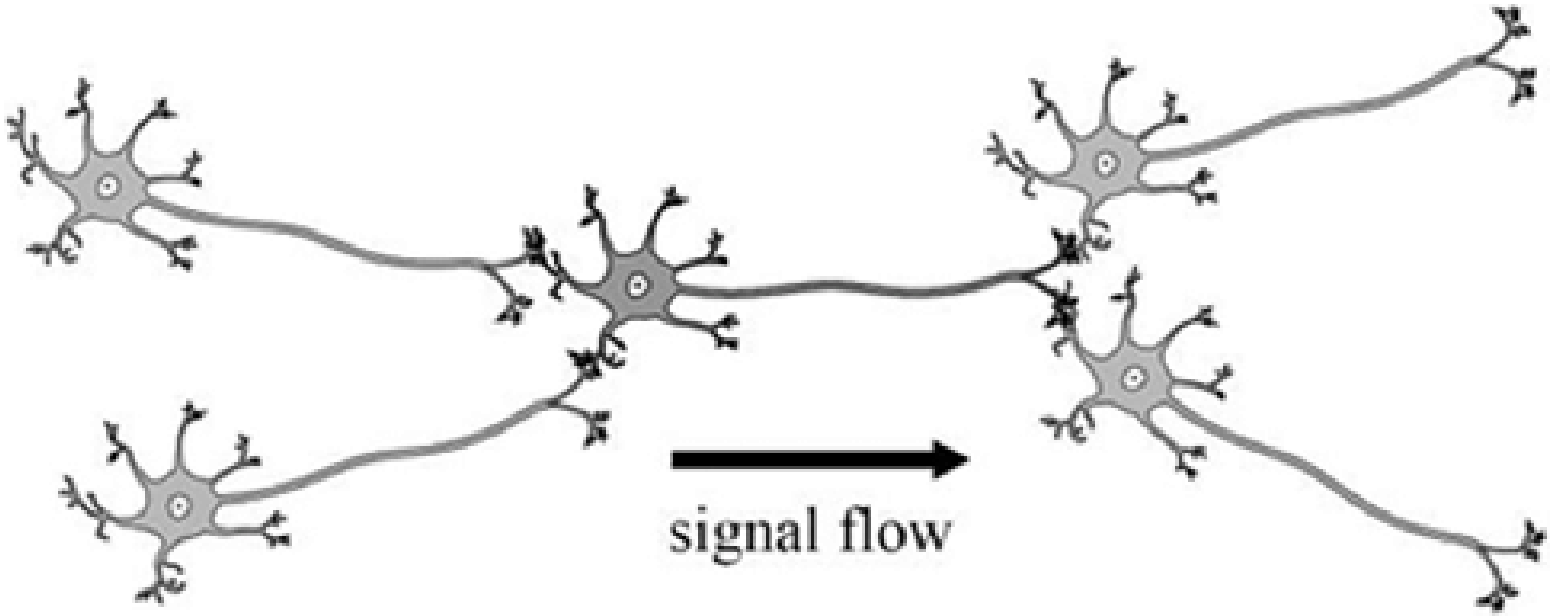
Pro každé $a, b \in \{0, 1\}$ platí.

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



- Víme, že AND a OR jsou lineárně separabilní, ... a pro NAND to platí také.
- Takto zapojené perceptrony by počítaly XOR

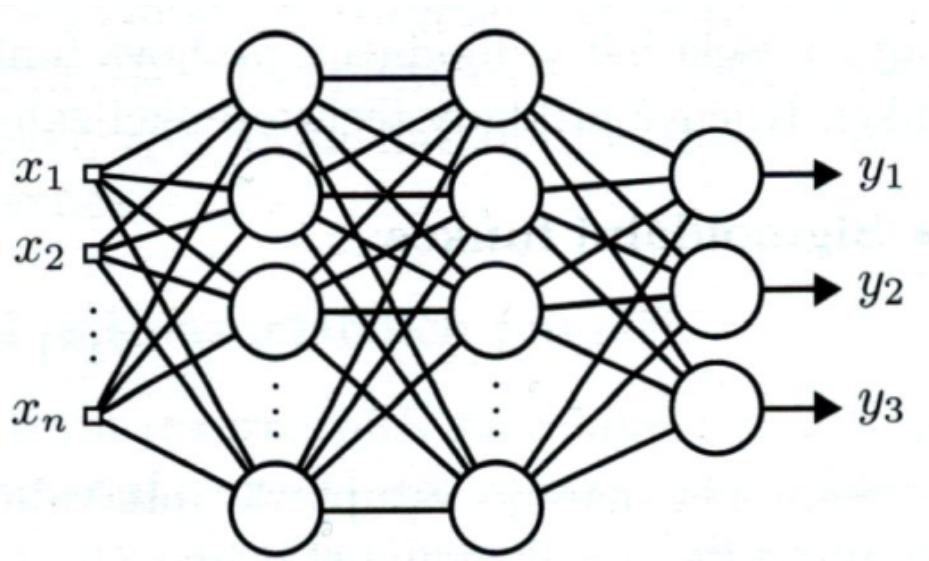
Ostatně biologické neurony taky neexistují izolovaně



- Ale jsou zapojeny do sítě – nervové soustavy

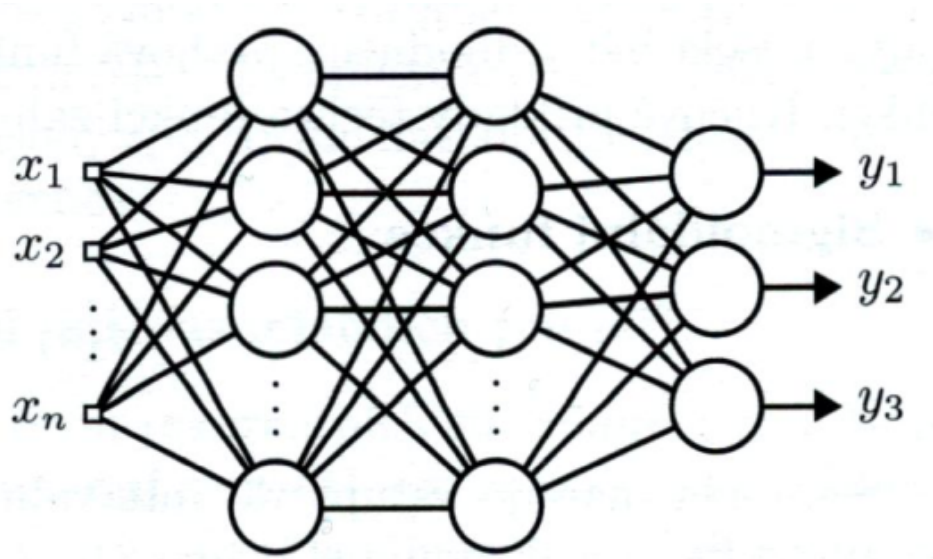
Přechod k vícevrstvým neuronovým sítím

Přechod k vícevrstvým neuronovým sítím (MLP, Multi-Layer Perceptrons) představoval průlom, který umožnil řešení lineárně **ne**separabilních úloh.



- Z praktických důvodů se perceptrony organizují do vrstev.

Funkce sítě



Máme n vstupů, k perceptronů v první vrstvě.
Kolik tam je parametrů?

a) $n + k$

a) $n + k + 1$

c) $n \cdot k$

d) $(n + 1) \cdot k$

Oddělení lineární a nelineární části

Rozdělme funkci perceptronu

$$f_{\boldsymbol{w}}(\boldsymbol{x}) = \begin{cases} 1 & \text{pokud } \sum_{i=1}^{n} w_i x_i + b \geq 0 \\ 0 & \text{jinak} \end{cases}$$

na dvě části:

- lineární:

$$\ell_{\boldsymbol{w}}(\boldsymbol{x}) = \sum_{i=1}^{n} w_i x_i + b$$

- nelineární:

$$s(y) = \begin{cases} 1 & \text{pokud } y \geq 0 \\ 0 & \text{jinak} \end{cases}$$

A když to máme v síti...

Řekněme, že máme

- tři vstupy: $\boldsymbol{x} = (x_1, x_2, x_3)$
- dva perceptrony v první vrstvě: $\boldsymbol{w}_1 = (w_{11}, w_{21}, w_{31}, b_1)$ a $\boldsymbol{w}_2 = (w_{12}, w_{22}, w_{32}, b_2)$

Lineární část:

$$y_1 = \ell_{\boldsymbol{w}_1}(\boldsymbol{x}) = \sum_{i=1}^{n} w_{i1} x_i + b_1$$

$$y_2 = \ell_{\boldsymbol{w}_2}(\boldsymbol{x}) = \sum_{i=1}^{n} w_{i2} x_i + b_2$$

Lineární část:

$$y_1 = \ell_{\boldsymbol{w_1}}(\boldsymbol{x}) = \sum_{i=1}^{n} w_{i1} x_i + b_1$$

$$y_2 = \ell_{\boldsymbol{w_2}}(\boldsymbol{x}) = \sum_{i=1}^{n} w_{i2} x_i + b_2$$

Můžeme tedy psát

$$y_j = \ell_{\boldsymbol{w_j}}(\boldsymbol{x}) = \sum_{i=1}^{n} w_{ij} x_i + b_j \qquad \text{pro } j \in \{1, 2\}$$

Násobení vektoru a matice, a sčítání vektorů

Vstupní vektor:

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

Váhová matice:

$$W = \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \\ w_{31} & w_{32} \end{bmatrix}$$

Vektor biasů

$$\boldsymbol{b} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

Výstup:

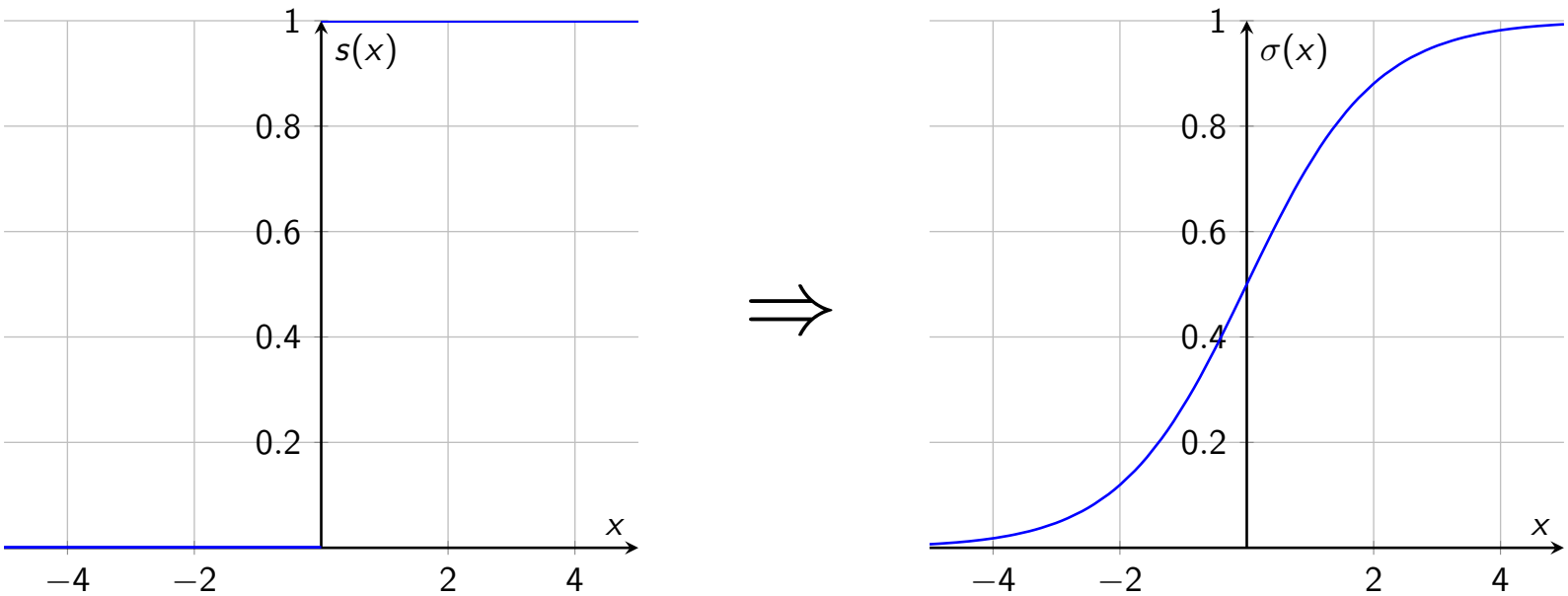
$$\boldsymbol{y} = W^T \boldsymbol{x} + \boldsymbol{b}$$

Na nelineární vrstvě pak nic zajímavého...

- Jen místo „schodkové funkce“

$$s(y) = \begin{cases} 1 & \text{pokud } y \geq 0 \\ 0 & \text{jinak} \end{cases}$$

použijeme její „vyhlazení“:



Proč je výhodné vyjadřovat FFNN pomocí matic?

- Výpočet FFNN (např. $\boldsymbol{y} = \sigma(W^T \boldsymbol{x} + \boldsymbol{b})$) je vlastně jen:
  - násobení matice a vektoru
  - součet vektorů
  - aplikace funkce po složkách

Proč je výhodné vyjadřovat FFNN pomocí matic?

- Výpočet FFNN (např. $\boldsymbol{y} = \sigma(W^T \boldsymbol{x} + \boldsymbol{b})$) je vlastně jen:
  - násobení matice a vektoru
  - součet vektorů
  - aplikace funkce po složkách
- Tyto operace se dají skvěle počítat paralelně
- **Grafické karty (GPU)** jsou navrženy právě pro hromadné výpočty s maticemi
- **Proto může neuronová síť běžet rychle** – i když má miliony vah

Proč je výhodné vyjadřovat FFNN pomocí matic?

- Výpočet FFNN (např. $\boldsymbol{y} = \sigma(W^T \boldsymbol{x} + \boldsymbol{b})$) je vlastně jen:
  - násobení matice a vektoru
  - součet vektorů
  - aplikace funkce po složkách
- Tyto operace se dají skvěle počítat paralelně
- **Grafické karty (GPU)** jsou navrženy právě pro hromadné výpočty s maticemi
- **Proto může neuronová síť běžet rychle** – i když má miliony vah
- Moderní AI funguje jen díky tomu, že:
  - lze výpočet zapsat maticově
  - máme hardware, který to umí počítat extrémně rychle

Maticový tvar = most mezi matematikou a výkonným hardwarem

Problém: jak ale učit perceptronovou síť

- (Ne)můžeme učit každý perceptron zvlášť.

D. E. Rumelhart, J. L. McClelland (Eds.)
Parallel Distributed Processing.
MIT Press (1986)

Algoritmus *backpropagation*

- Hledá minimum chybové funkce
- ... a to pomocí gradientního sestupu.

Chybová funkce a trénování sítě

- Cílem učení je, aby síť dávala správný výstup pro různé vstupy
- Máme množinu trénovacích příkladů:
  - vstup $\rightarrow$ správný (očekávaný) výstup
  - podobně jako dříve: pozitivní (P) a negativní (N) příklady

Chybová funkce a trénování sítě

- Cílem učení je, aby síť dávala správný výstup pro různé vstupy
- Máme množinu trénovacích příkladů:
  - vstup $\rightarrow$ správný (očekávaný) výstup
  - podobně jako dříve: pozitivní (P) a negativní (N) příklady
- Pro každý vstup síť něco spočítá, ale může se mýlit
- **Chybová funkce** říká, *jak moc* se síť mýlí
  - Například: součet čtverců rozdílů mezi skutečným a správným výstupem
  - Čím větší chyba, tím horší síť

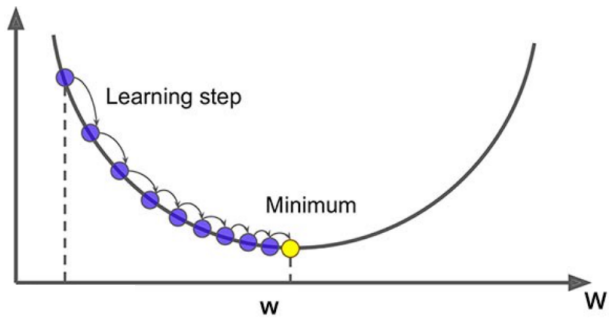
Chybová funkce a trénování sítě

- Cílem učení je, aby síť dávala správný výstup pro různé vstupy
- Máme množinu trénovacích příkladů:
  - vstup $\rightarrow$ správný (očekávaný) výstup
  - podobně jako dříve: pozitivní (P) a negativní (N) příklady
- Pro každý vstup síť něco spočítá, ale může se mýlit
- **Chybová funkce** říká, *jak moc* se síť mýlí
  - Například: součet čtverců rozdílů mezi skutečným a správným výstupem
  - Čím větší chyba, tím horší síť
- **Cíl:** změnit váhy tak, aby se chyba zmenšovala

Chybová funkce je jako známka – říká, jak moc se síť spletuje

Co dělá gradientní sestup (jednoduše)?

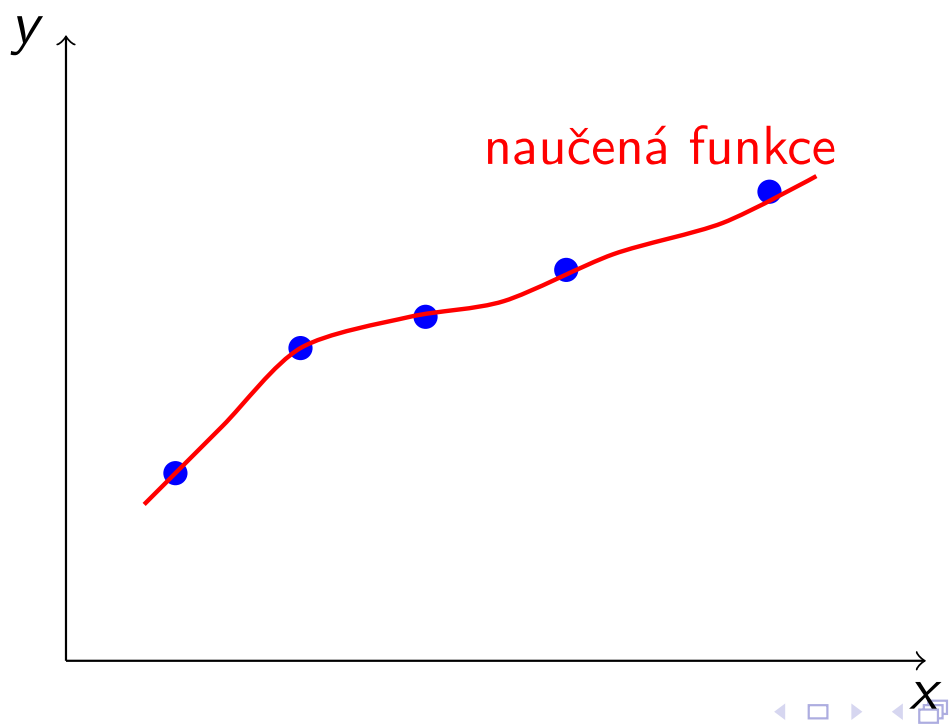
- Představ si obyčejnou funkci jedné proměnné $f(x)$
- Chci najít její **minimum** – třeba tam, kde je chyba neuronové sítě nejmenší
- Zkusím náhodné x a spočítám $f(x)$ a derivaci $f'(x)$
- Pokud je $f'(x) > 0$, funkce roste $\rightarrow$ jdu doleva (snižuju x)
- Pokud je $f'(x) < 0$, funkce klesá $\rightarrow$ jdu doprava (zvyšuji x)
- Každým krokem se posouvám blíž k minimu



Gradientní sestup hledá dno údolí

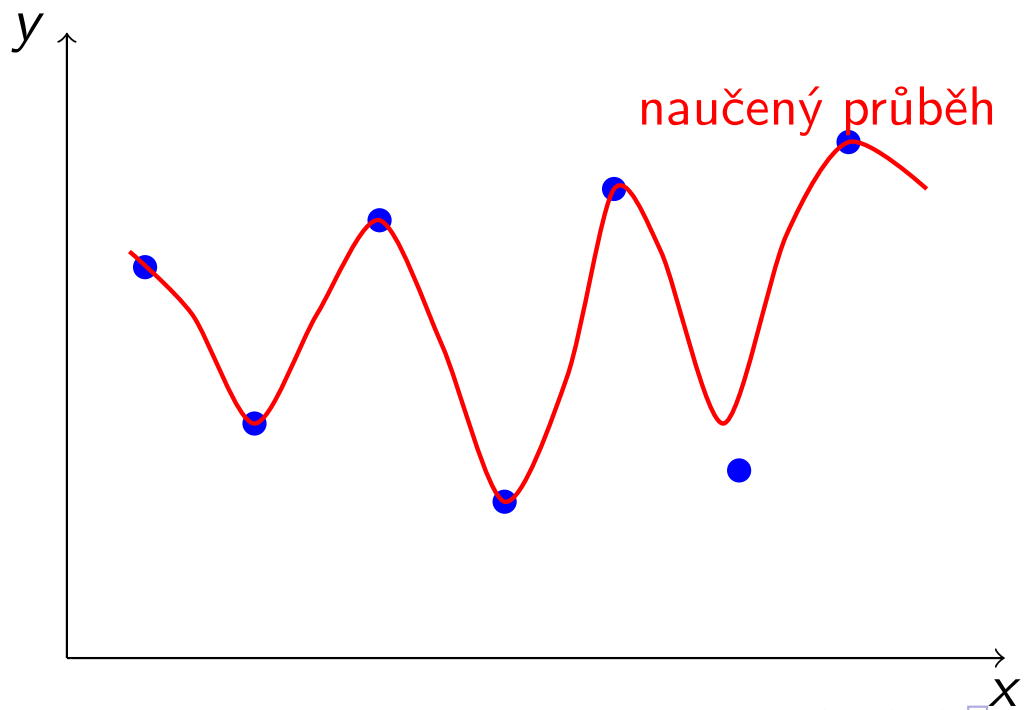
Neuronová síť jako naučená funkce

- Představ si, že chceme, aby síť převedla vstup na správný výstup
- Například:
  - vstup = výška člověka, výstup = odhad váhy
  - vstup = obrázek, výstup = kategorie („kočka“, „pes“...)
- Síť se učí nějakou funkci: vstup $\rightarrow$ výstup
- **Čím více trénovacích příkladů, tím lépe síť tuto funkci odhadne**



Univerzální aproximátor – co to znamená?

- Neuronová síť se může naučit **téměř libovolnou funkci**
- ... pokud má dostatečně mnoho neuronů a správná data
- To znamená:
  - může se naučit rovnou čáru
  - ale i klikatou, vlnitou, nebo složitou hranici mezi třídami
- Nezáleží na tom, jak „zvláštní“ funkce je – síť se ji *dokáže* naučit (aproximovat)



Proč jsou neuronové sítě tak silné

- Neuronové sítě dokážou:
  - naučit se téměř libovolný vztah mezi vstupem a výstupem
  - zpracovat složitá data (obrázky, texty, signály. . .)
- Mají schopnost generalizace – učí se ze vzorků, aplikují na nové případy
- Proto jsou dnes základem umělé inteligence

Neuronové sítě $\approx$ naučitelné univerzální převodníky