Neuronové sítě 1 - Umělý neuron

18NES1 - 3. a 4. hodina, LS 2024/25

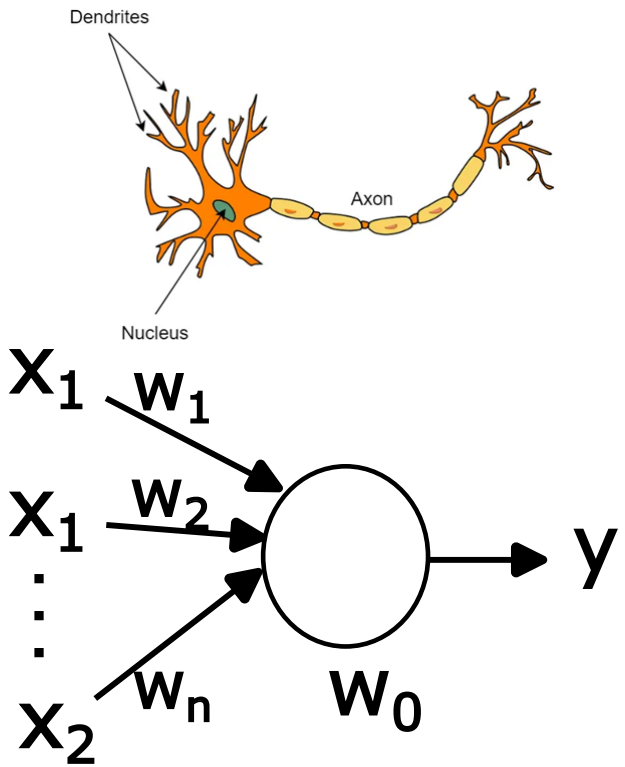
Zuzana Petříčková

February 23, 2025

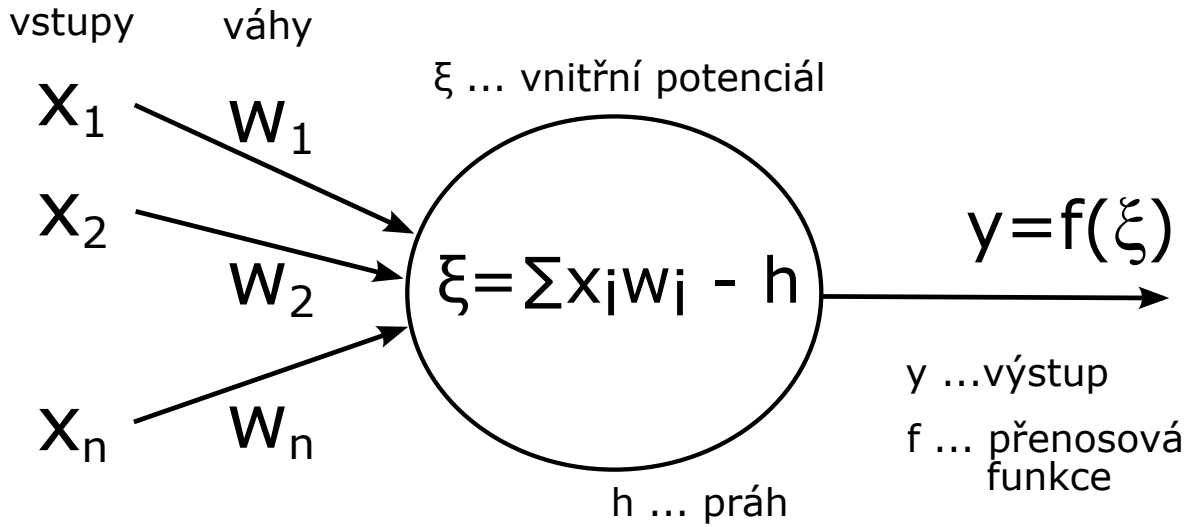
Co jsme probírali minule

Od biologického neuronu k umělému

- Nejstarší modely umělých neuronů: Culloch and Pitts neurons (1943), Perceptrons (Rosenblatt, 1955)
- Perceptron a reprezentace logických funkcí



Matematický model neuronu - původní definice



Klasická definice: práh h

- vnitřní potenciál: $\xi = \sum_{i=1}^{n} w_i x_i - h = \vec{w}\vec{x}^T - h$
- výstup: $y = f(\xi)$ ... **skoková** přenosová gunkce:

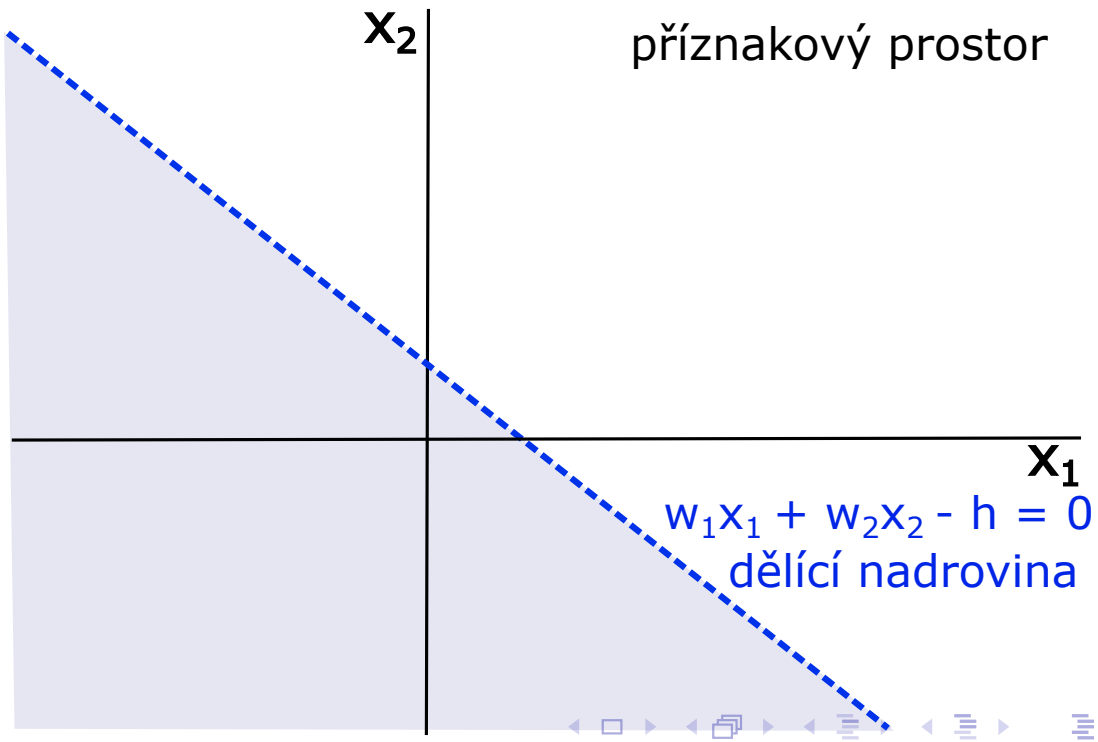
$$f(\xi) = \begin{cases} 1 & \text{pro } \xi > 0 \quad \text{... neuron je aktivní} \\ 0 \text{ (nebo -1)} & \text{pro } \xi < 0 \quad \text{... neuron je pasivní} \\ 0.5 \text{ (nebo 0)} & \text{pro } \xi = 0 \quad \text{... neuron je tichý} \end{cases}$$

Geometrická interpretace umělého neuronu

- vstupy neuronu si představme jako body v n-rozměrném Euklidovském prostoru (vstupní, příznakový prostor)
- položme vnitřní potenciál neuronu $\xi = 0$ a získáme rovnici dělící nadroviny

$$\xi = w_1 x_1 + w_2 x_2 - h = 0$$

$$x_2 = -\frac{w_1}{w_2} x_1 + \frac{h}{w_2}$$



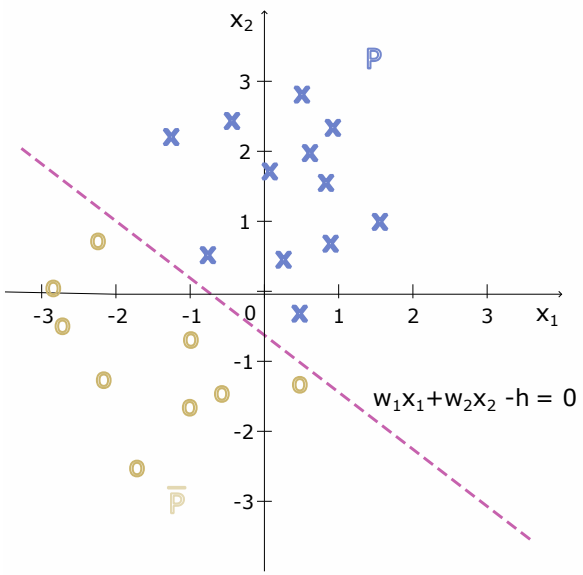
Geometrická interpretace umělého neuronu

$\rightarrow$ perceptron může sloužit jako **lineární klasifikátor:** klasifikuje vzory do dvou tříd (P a $\overline{P}$).

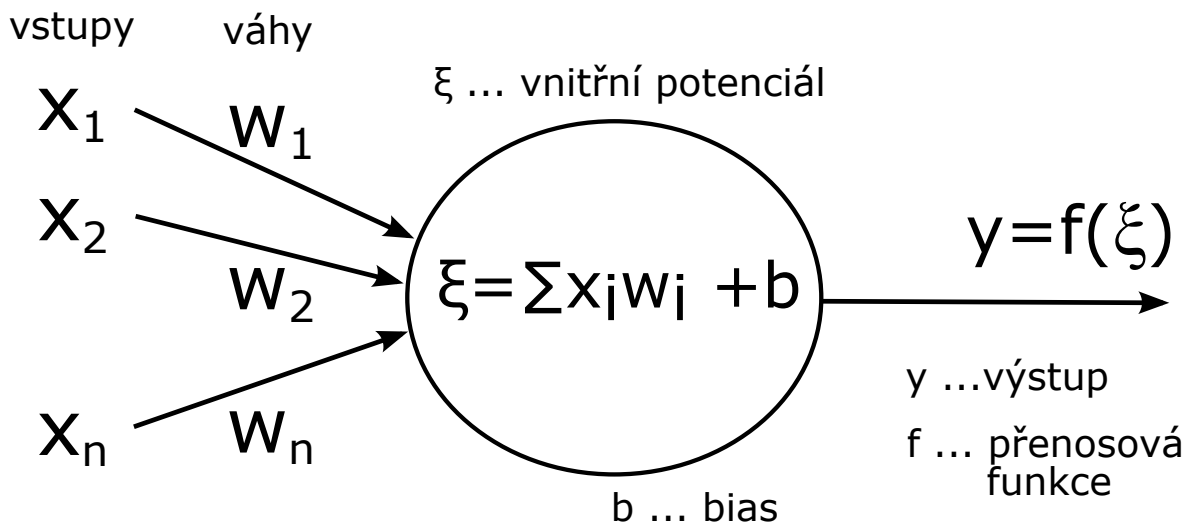
- **lineární:** hranicí mezi třídami je nadrovina
  $\xi = w_1x_1 + w_2x_2 + \cdots + w_nx_n - h = 0$: bod, přímka, rovina,...

Skoková přenosová funkce:

- $f(\xi) = 1$ pro $\xi > 0$ ... neuron je **aktivní** (třída P)
- $f(\xi) = 0$ (nebo -1) pro $\xi < 0$ ... neuron je **pasivní** (třída $\overline{P}$)
- $f(\xi) = 0.5$ (nebo 0) pro $\xi == 0$ ... neuron je **tichý**, tj. neumí rozhodnout



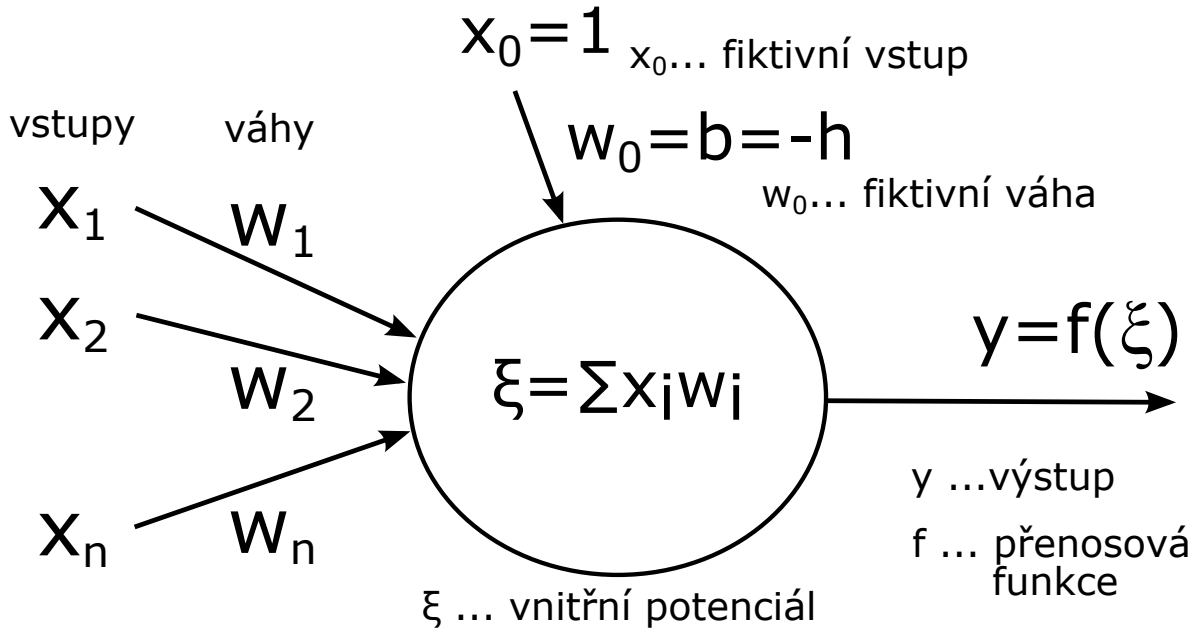
Matematický model neuronu - moderní definice



Alternativní definice: práh $\rightarrow$ bias

- vnitřní potenciál: $\xi = \sum_{i=1}^{n} w_i x_i + b = \vec{w}\vec{x}^T + b$
- výstup: $y = f(\xi)$ (skoková přenosová gunkce ... sign)

Matematický model neuronu - ,,maticová" definice



Alternativní definice: zavedení fiktivního vstupu

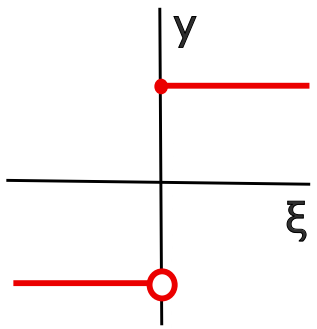
- rozšířený příznakový prostor ... $\vec{x} = (x_0 = 1, x_1, ..., x_n)$
- rozšířený prostor vah ... $\vec{w} = (w_0 = b = -h, w_1, ..., w_n)$
- vnitřní potenciál: $\xi = \sum_{i=0}^{n} w_i x_i = \vec{w}\vec{x}^T$
- výstup: $y = f(\xi)$ (skoková přenosová gunkce ... sign)

Perceptrony (Rosenblatt, 1955)

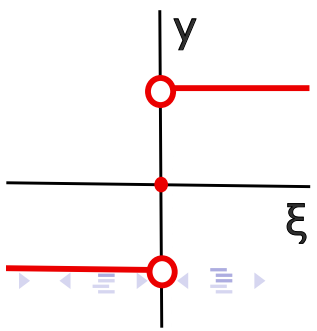
- reálné vstupy ... $x_i \in R$
- reálné váhy ... $w_i \in R$
- výstupy:
  - **binární** ... $y \in \{0, 1\}$
  - **bipolární** ... $y \in \{-1, 1\}$

Varianty skokové přenosové funkce pro bipolární perceptron:

$$f(\xi) = \begin{cases} 1 & \text{pro } \xi \geq 0 \quad \text{... neuron je aktivní} \\ -1 & \text{pro } \xi < 0 \quad \text{... neuron je pasivní} \end{cases}$$



$$f(\xi) = \begin{cases} 1 & \text{pro } \xi > 0 \quad \text{... neuron je aktivní} \\ 0 & \text{pro } \xi = 0 \quad \text{... neuron je tichý} \\ -1 & \text{pro } \xi < 0 \quad \text{... neuron je pasivní} \end{cases}$$



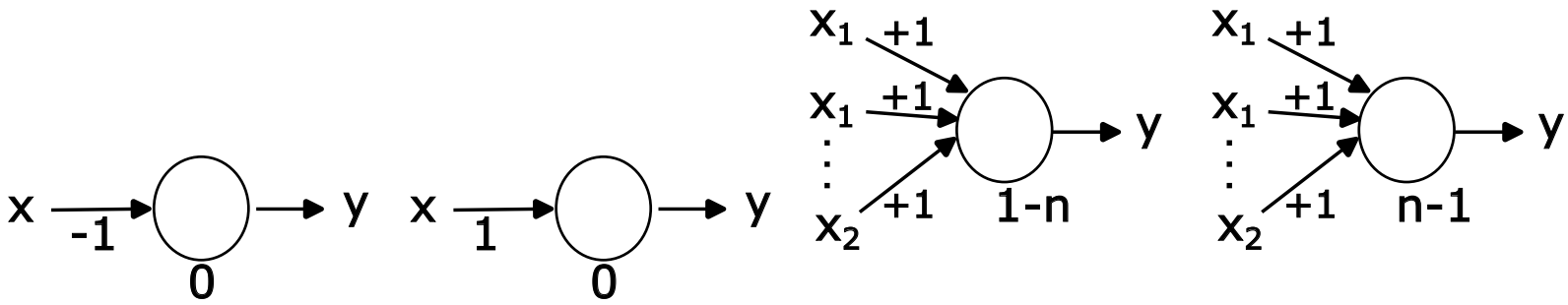
$\rightarrow$ funkce **signum** (sign)

Opakování – Perceptron a reprezentace logických funkcí

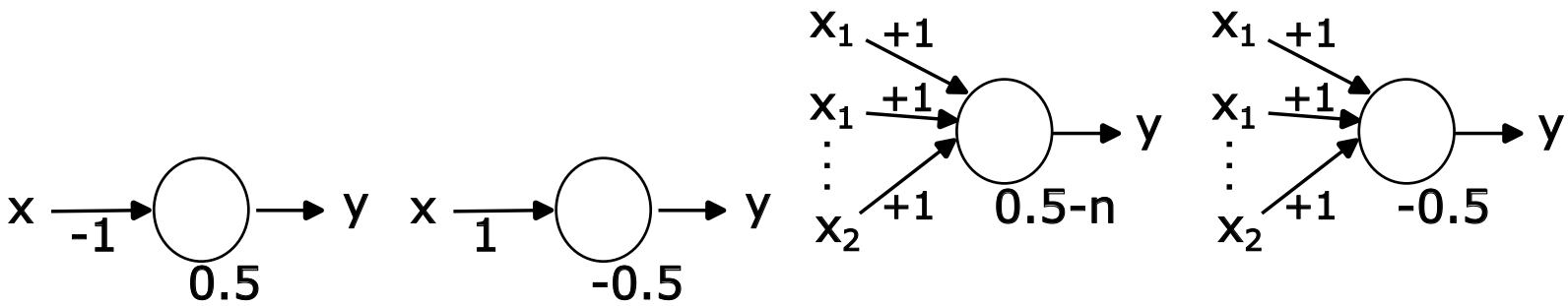
Pomocí perceptronu lze realizovat základní logické funkce

- NOT (negace), ID (identita)
- AND (konjunkce), OR (disjunkce)

Bipolární model (vstupy i výstupy neuroonu -1, 1)



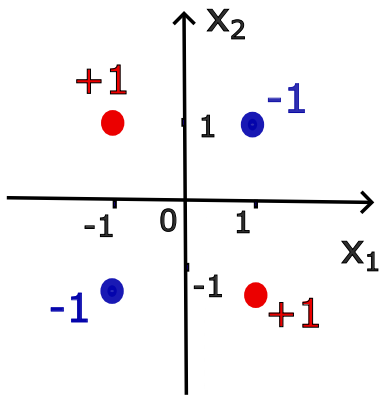
Binární model (vstupy i výstupy neuroonu 0, 1)



Logický prahový obvod

Pomocí perceptronu ale nelze realizovat všechny logické funkce

- např. XOR (exkluzivní disjunkce)



Řešení:

- perceptrony pro NOT, ID, AND a OR můžeme poskládat vhodně za sebe (do neuronové sítě, konkrétně do logického prahového obvodu) a sestavit tak složitější logické funkce

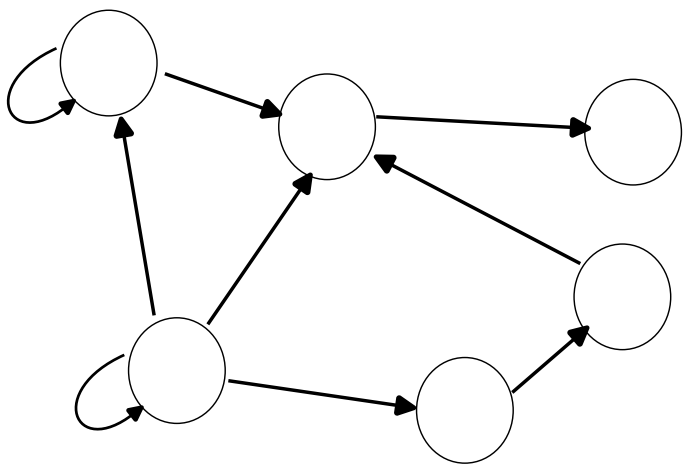
Neuronové sítě 1 - hodina 3: Umělý neuron

Neuronová síť

- Skládá se z neuronů, které jsou navzájem pospojovány tzv. hranami
- Výstup jednoho neuronu může být vstupem jednoho nebo více dalších neuronů

Architektura (topologie) neuronové sítě

- Orientovaný graf, neurony představují uzly, synaptické vazby představují hrany



Neuronová síť

Výstupní neurony

- jejich výstupy tvoří dohromady výstup neuronové sítě
- **typicky:** nevedou z nich žádné hrany do jiných neuronů

Vstupní neurony

- mají na vstupu vstupní vzory
- **typicky:** nevedou do nich žádné hrany z jiných neuronů

Výstup (odezva) neuronové sítě

- Výstupy (aktivity) výstupních neuronů

Neuronová síť

Definice: Neuronová síť je šestice (N,C,I,O,w,t):

- N je konečná neprázdná množina neuronů,
- $C \subseteq N \times N$ je neprázdná množina orientovaných spojů mezi neurony (hran)
- $I \subseteq N$ je neprázdná množina vstupních neuronů
- $O \subseteq N$ je neprázdná množina výstupních neuronů
- $w : C \to R$ je váhová funkce
- $t : N \to R$ je prahová funkce

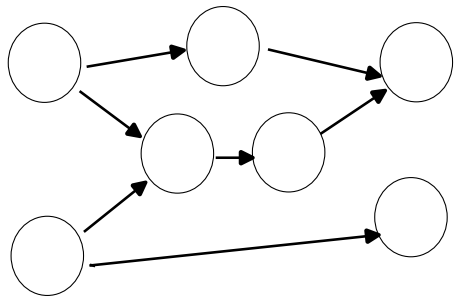
Konfigurace neuronové sítě

- Váhy všech hran a prahy (biasy) všech neuronů

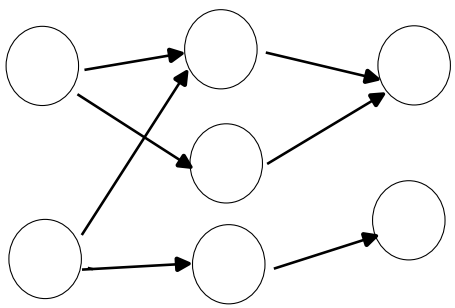
Neuronová síť

Architektura neuronové sítě

- cyklická, rekurentní
- acyklická, dopředná - ,,všechny hrany jdou stejným směrem" (tj. graf lze topologicky uspořádat)

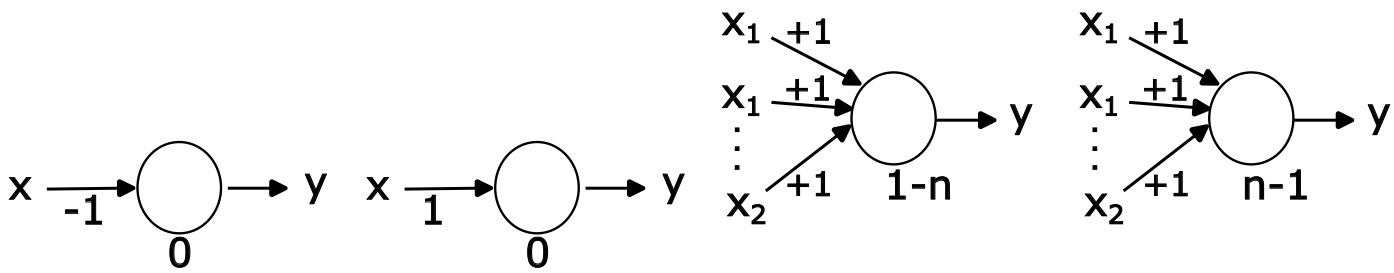


- hierarchická (vrstevnatá) - dělí se na vrstvy, propojeny jsou jen neurony ze dvou po sobě jdoucích vrstev



Logický prahový obvod

- z perceptronů pro NOT, ID, AND, OR a ID můžeme sestavit neuronovou síť reprezentující složitější logické funkce



- AND realizuje průnik konvexních útvarů a OR jejich sjednocení

Otázky: Jak se změní logická funkce realizovaná perceptronem, když:

- vynásobím všechny váhy (včetně biasu) kladným číslem?
- vynásobím všechny váhy (včetně biasu) záporným číslem (např. -1)?
- vynásobím některé váhy -1?

Logický prahový obvod - příklady

Příklad 1: Předpověď úrody

úroda = ((teplo ∧ déšť) ∨ (teplo ∧ zavlažování)) ∧ hnojiva ∧ ¬škůdci

1. Navrhněte perceptronovou síť pro tuto logickou funkci s využitím základních logických operací.
   - Kolik má vstupů a výstupů, kolik neuronů a kolik vrstev?
2. Navrhněte perceptronovou síť, která bude mít hierarchickou (vrstevnatou) architekturu.
3. Minimalizujte tuto logickou funkci a navrhněte neuronovou síť pro minimalizovanou verzi.
4. Lze tuto logickou funkci reprezentovat jedním perceptronem?

Logický prahový obvod - příklady

Příklad 2: Majoritní obvod

$y = (x_1 \wedge x_2) \vee (x_1 \wedge x_3) \vee (x_2 \wedge x_3)$

1. Navrhněte neuronovou síť pro majoritní obvod s využitím základních logických operací.
2. Lze ho reprezentovat jedním perceptronem?
3. Jaké bude řešení v obecném případě (pro obecné n)?

Perceptronová síť jako logický prahový obvod

Věta

Každou logickou formuli lze vyjádřit v **disjunktivním normálním tvaru (DNF)**, tj. jako disjunkci konjunkcí atomů, kde atomy tvoří proměnné nebo jejich negace.

- disjunkce: $F = K_1 \vee K_2 \vee ... \vee K_n$
- konjunkce: $K_i = A_{i1} \wedge A_{i2} \wedge ... \wedge A_{in_i}$
- atomy: $A_{ij} = L$ nebo $A_{ij} = \neg L$

Příklad: $y = (x_2 \wedge x_4) \vee \neg x_1 \vee (x_2 \wedge \neg x_3)$

Důsledek

Každou logickou funkci mohu vyjádřit pomocí perceptronové neuronové sítě.

- **Otázka**: Navrhněte schéma takové perceptronové neuronové sítě. Kolik vrstev bude tato neuronová síť mít?

Perceptronová síť jako logický prahový obvod

Podobně
Každou logickou formuli lze vyjádřit v **konjunktivním normálním tvaru (CNF)**, tj. jako konjunkci disjunkcí atomů, kde atomy tvoří proměnné nebo jejich negace.

- konjunkce: $F = D_1 \wedge D_2 \wedge ... \wedge D_n$
- disjunkce: $D_i = A_{i1} \vee A_{i2} \vee ... \vee A_{in_i}$
- atomy: $A_{ij} = L$ nebo $A_{ij} = \neg L$

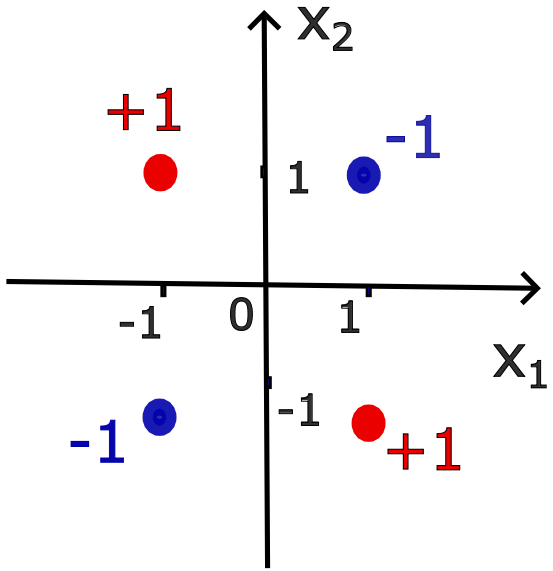
Příklad: $y = (x_2 \vee x_4) \wedge \neg x_1 \wedge (x_2 \vee \neg x_3)$
Realizace pomocí perceptronové sítě : analogicky jako u DNF

- AND realizuje průnik konvexních útvarů a OR jejich sjednocení

Logický prahový obvod - příklady

Příklad 3: Exkluzivní OR (XOR)

- XOR nelze realizovat jedním perceptronem



- XOR lze realizovat pomocí perceptronové sítě

Logický prahový obvod - příklady

Příklad 3: Exkluzivní OR (XOR) - dobrovolný domácí úkol do příště

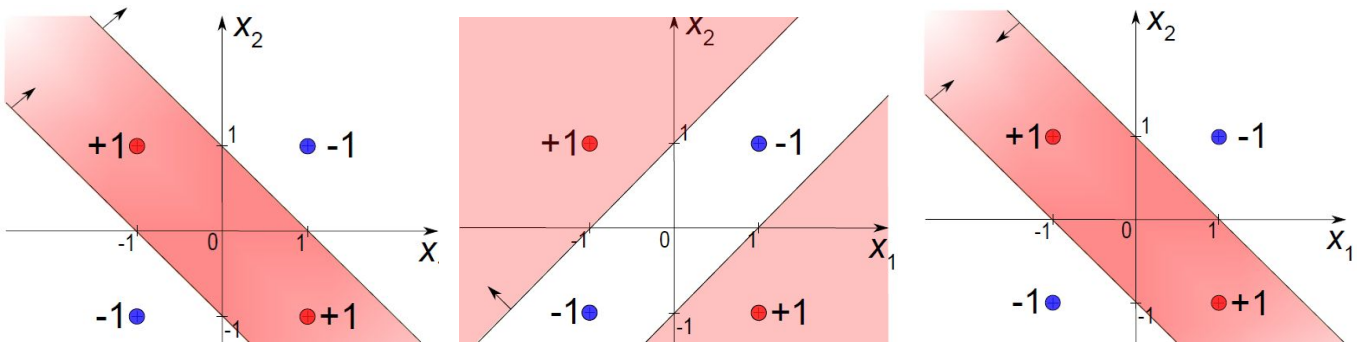
1. XOR můžeme pomocí základních logických operací (AND, OR, NOT) reprezentovat různě, zvládnete navrhnout několik různých způsobů?
2. Navrhněte co nejmenší neuronovou síť, která bude reprezentovat XOR. Kolik bude obsahovat neuronů?

x_1	x_2	$y = x_1 \otimes x_2$
-1	-1	-1
-1	+1	+1
+1	-1	+1
+1	+1	-1

Logický prahový obvod - příklady

Příklad 3: Exkluzivní OR (XOR) - dobrovolný domácí úkol do příště

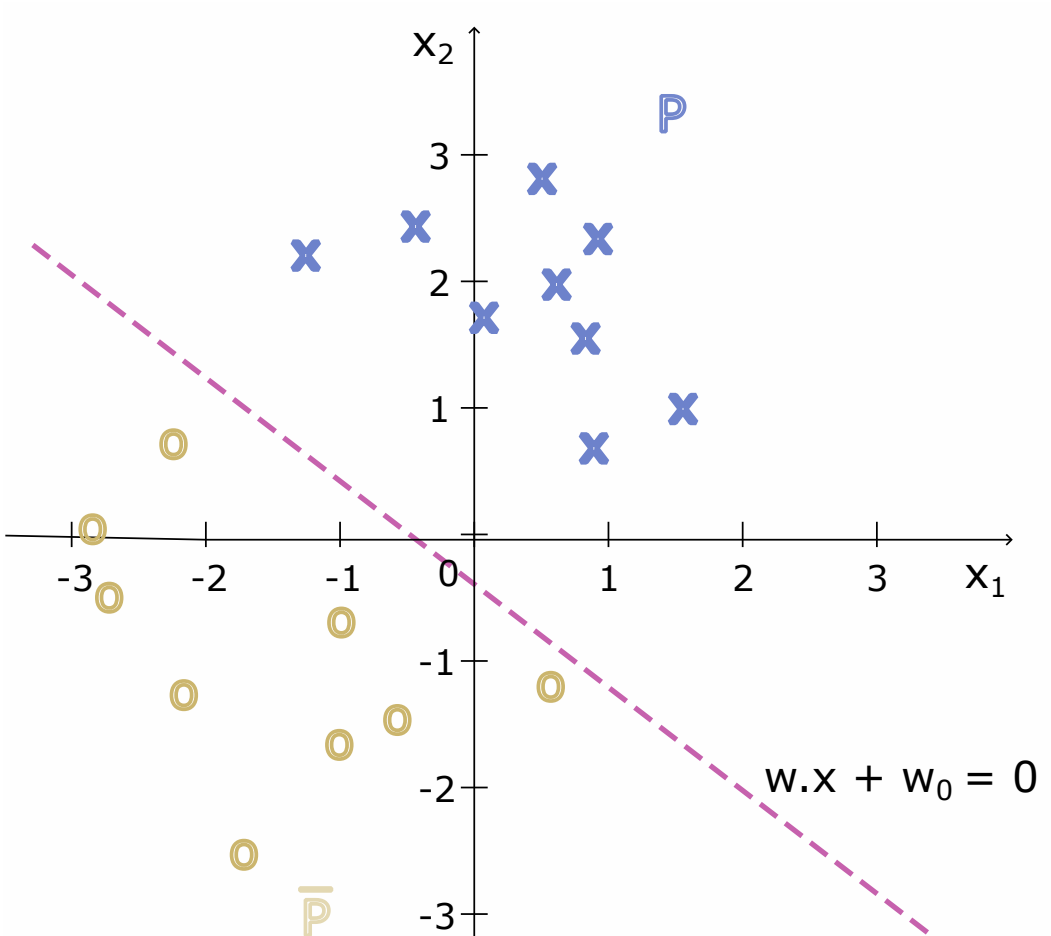
1. XOR můžeme pomocí základních logických operací (AND, OR, NOT) reprezentovat různě, zvládnete navrhnout několik různých způsobů?
2. Navrhněte co nejmenší neuronovou síť, která bude reprezentovat XOR. Kolik bude obsahovat neuronů?



Neuronové sítě 1 - hodina 3: Umělý neuron

Lineární separabilita

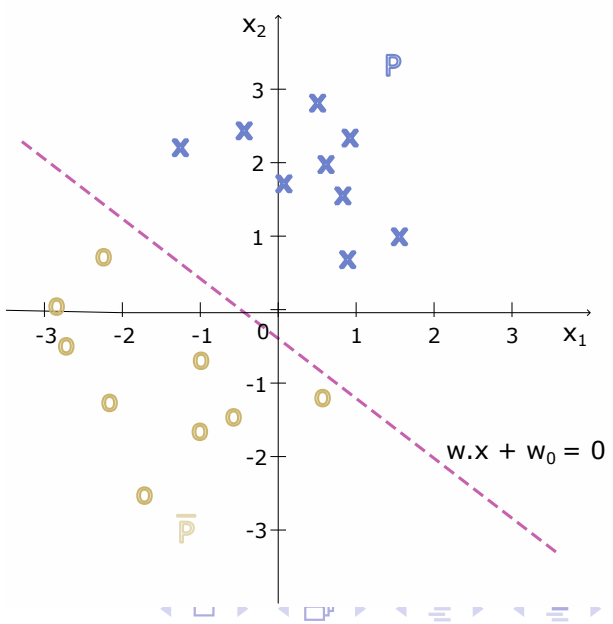
→ perceptron může sloužit jako **lineární klasifikátor:** klasifikuje vzory do dvou tříd (zde P, $\overline{P}$) pomocí **dělící nadroviny**
$\vec{w}\vec{x} + w_0 = \sum_{i=1}^{n} w_i.x_i + w_0 = 0$



Lineární separabilita

Definice:
Dvě množiny P, $\overline{P}$ jsou **lineárně separabilní** v n-rozměrném prostoru, pokud existují čísla $w_0, ..., w_n$ taková, že pro každý bod $\vec{x} \in P$ platí $\sum_{i=1}^{n} w_i.x_i + w_0 > 0$ a pro každý bod $\vec{x} \in B$ platí $\sum_{i=1}^{n} w_i.x_i + w_0 < 0$.

Proč se zavedl tento pojem?
→ vědci zkoumali, které funkce lze realizovat pomocí perceptronu (nebo obecně lin. klasifikátoru) a které ne



Lineární separabilita

Pro Booleovský prostor a $n = 2$

- celkem $2^4 = 16$ logických funkcí, 14 z nich je lineárně separabilních:
  - 6 jednoduchých logických funkcí:
    0, 1, A, B, $\neg A$, $\neg B$
    $\rightarrow$ triviální
  - 8 variant konjunkce a disjunkce:
    $A \wedge B$, $A \wedge \neg B$, $\neg A \wedge B$, $\neg A \wedge \neg B$
    $A \vee B$, $A \vee \neg B$, $\neg A \vee B$, $\neg A \vee \neg B$,
    $\rightarrow$ trochu složitější
- 2 funkce, které nejsou lineárně separabilní a perceptronem je nelze realizovat:
  - $A \otimes B$ (XOR) a $A \Leftrightarrow B$ (ekvivalence)

Lineární separabilita

Pro obecný Booleovský prostor:

- $n = 2 \rightarrow 14$ z $2^4 = 16$ logických funkcí je lineárně separabilních.
- $n = 3 \rightarrow 104$ z $2^8 = 256$
- $n = 4 \rightarrow 1882$ z $2^{16} = 65536$
- n obecné ... ??

$\rightarrow$ logických funkcí, které nelze reprezentovat pomocí perceptronu, je hodně a jejich procento roste s dimenzí příznakového (vstupního) prostoru

Co s tím?

- místo jednoho neuronu použijeme perceptronovou síť
- rozšíříme příznakový prostor o další proměnné

Neuronové sítě 1 - hodina 2: Umělý neuron

Perceptron - algoritmus učení

Už víme:
Perceptron (se skokovou přenosovou funkcí) můžeme použít jako **lineární klasifikátor** vstupních vzorů do dvou množin/tříd (P a $\overline{P}$).

Dělicí nadrovina
určená (n+1)–rozměrným váhovým vektorem $\vec{w}$ je množina všech bodů $\vec{x} \in R^n$, pro které $\vec{w} \cdot \vec{x} + w_0 = 0$

Problém:
Nalézt takové váhy, resp. práh/bias, které by umožnily rozdělit vstupní vzory správně do množin P a $\overline{P}$- pomocí dělící nadroviny

Možné řešení:
Perceptronový (Rosenblattův) algoritmus učení (Rosenblatt, 1959)

Perceptron - algoritmus učení

Data, na základě kterých se bude model učit:

- trénovací množina T
  - množina N trénovacích vzorů $T = \{(\vec{x_1}, d_1), ..., (\vec{x_N}, d_N)\}$
- trénovací vzor (training pattern) ... $(\vec{x}, d)$,
  - $\vec{x} = (x_1, ..., x_n)$ ... vstupní vzor (input pattern), má n příznaků
  - $d \in \{-1, 1\}$ ... požadovaný (očekávaný) výstup
- T můžeme rozdělit do dvou množin P a $\overline{P}$:
  - P ... kladné (pozitivní) vzory ($d = 1$)
  - $\overline{P}$ ... záporné (negativní) vzory ($d = -1$)
  - $T = P \cup \overline{P}$

Perceptron - algoritmus učení

Data, na základě kterých se bude model učit:

- Trénovací množina $T = \{(\vec{x_1}, d_1), ..., (\vec{x_N}, d_N)\}$:
- Maticová reprezentace $T = (X|\vec{d})$:

x_{11}	x_{12}	...	x_{1n}	d_1	... 1. trénovací vzor
x_{21}	x_{22}	...	x_{2n}	d_2	... 2. trénovací vzor
...	...		...	...	
x_{N1}	x_{N2}	...	x_{Nn}	d_N	... Ntý trénovací vzor

- Pro rozšířený příznakový prostor:

$x_{10} = 1$	x_{11}	...	x_{1n}	d_1	... 1. trénovací vzor
$x_{20} = 1$	x_{21}	...	x_{2n}	d_2	... 2. trénovací vzor
...	...		...	...	
$x_{N0} = 1$	x_{N1}	...	x_{Nn}	d_N	... Ntý trénovací vzor

Perceptron - algoritmus učení

Cíl učení:

- Nastavit váhy (a bias) neuronu tak, aby správně klasifikoval všechny trénovací vzory, tj:
  - $y_p = d_p$ ... pro všechny trénovací vzory z T
    y_p ... skutečná odezva (výstup) neuronu pro vstupní vzor $\vec{x}_p$
- Perceptron se skokovou přenosovou funkcí:

$$y_p = sign(\xi_p)$$

$$\xi_p = \sum_{i=1}^{n} w_i x_{pi} + w_0 = \sum_{i=0}^{n} w_i x_{pi} = \vec{w} \cdot \vec{x}_p$$

$\vec{x}_p = (1, x_{p1}, ..., x_{pn})$ ... rozšířený vstupní vzor

$\rightarrow$ Chceme:

- $\vec{w} \cdot \vec{x_p} < 0$ ... pro rozšířené trénovací vzory z $\overline{P}$
- $\vec{w} \cdot \vec{x_p} > 0$ ... pro rozšířené trénovací vzory z P

Perceptron - algoritmus učení

Chceme:

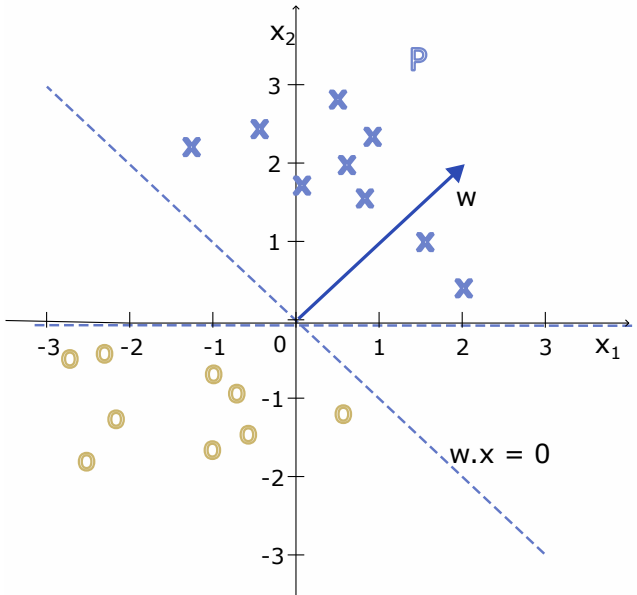
- $\vec{w} \cdot \vec{x_p} < 0$ ... pro rozšířené trénovací vzory z $\overline{P}$
- $\vec{w} \cdot \vec{x_p} > 0$ ... pro rozšířené trénovací vzory z P

Možnosti:

1. Soustava nerovnic nemá řešení $\rightarrow$ hledáme neperfektní řešení
2. Existuje alespoň jedno perfektní řešení $\vec{w}$ v R^{n+1} $\rightarrow$ existuje nekonečně mnoho řešení v R^{n+1} (dokonce v Z^{n+1})

Možné doplňkové podmínky:

- $|w_0| + |w_1| ... + |w_n| = min$
- $\sqrt{{w_0}^2 + \cdots {w_n}^2} = min$
- $max(|w_0|, ..., |w_n|) = min$



Perceptron - algoritmus učení

Značení:

- $y = f(\vec{x})$ ... skutečná odezva (výstup) neuronu pro vstupní vzor $\vec{x}$ (a d je požadovaná odezva)

Cílová (chybová) funkce

- počet chybně klasifikovaných vzorů: $E = \sum_{(\vec{x},d)\in T} [d \neq f(\vec{x})]$
- pro bipolární model:

$$E = \sum_{x\in P} \frac{1}{2}(1 - f(\vec{x})) + \sum_{x\in \overline{P}} \frac{1}{2}(1 + f(\vec{x}))$$

- pro binární model:

$$E = \sum_{x\in P} (1 - f(\vec{x})) + \sum_{x\in \overline{P}} f(\vec{x})$$

Cíl učení:

- Minimalizace E v prostoru vah. Nejlépe $E = 0$.

Perceptron - Rosenblattův algoritmus učení

Myšlenka a odvození:

- budeme pracovat v rozšířeném příznakovém a váhovém prostoru

Chceme:

- $\vec{w} \cdot \vec{x} < 0$ pro $(\vec{x}, d) \in \overline{P}$
- $\vec{w} \cdot \vec{x} > 0$ pro $(\vec{x}, d) \in P$

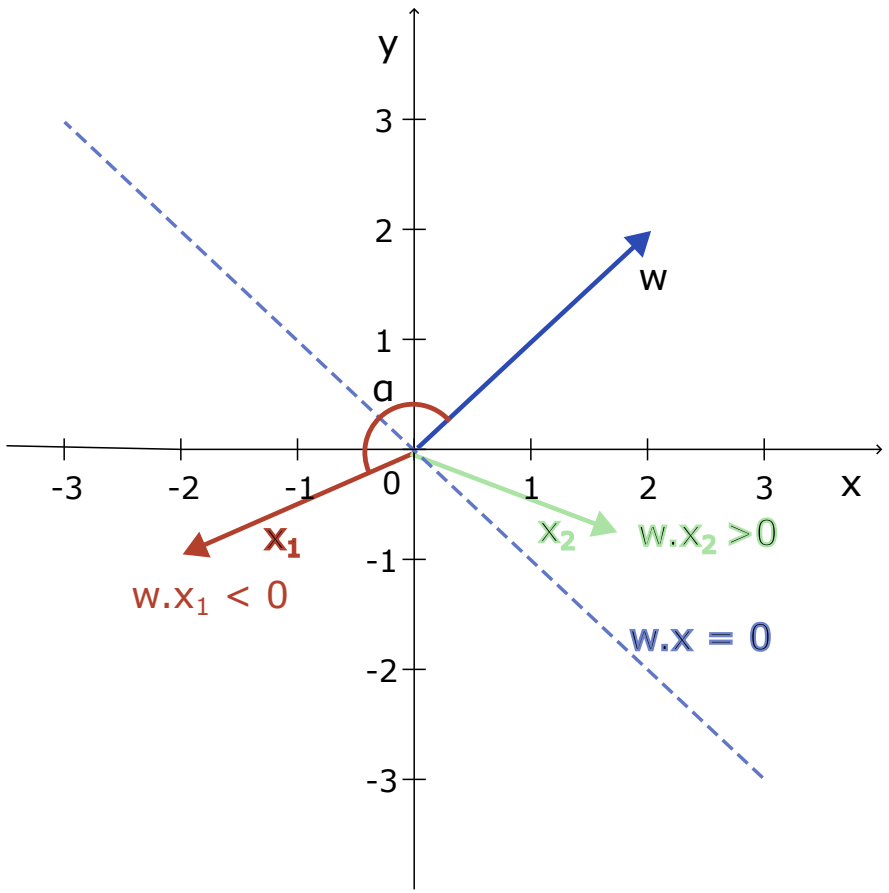
Z definice skalárního součinu:

- $\vec{w} \cdot \vec{x} = |\vec{w}||\vec{x}|cos(\alpha)$

$\rightarrow$

- $\vec{w} \cdot \vec{x} = 0$ ... $|\alpha| = 90°$
- $\vec{w} \cdot \vec{x} > 0$ ... $|\alpha| < 90°$
- $\vec{w} \cdot \vec{x} < 0$ ... $180 \geq |\alpha| > 90°$

Geometrická interpretace:



Perceptron - Rosenblattův algoritmus učení

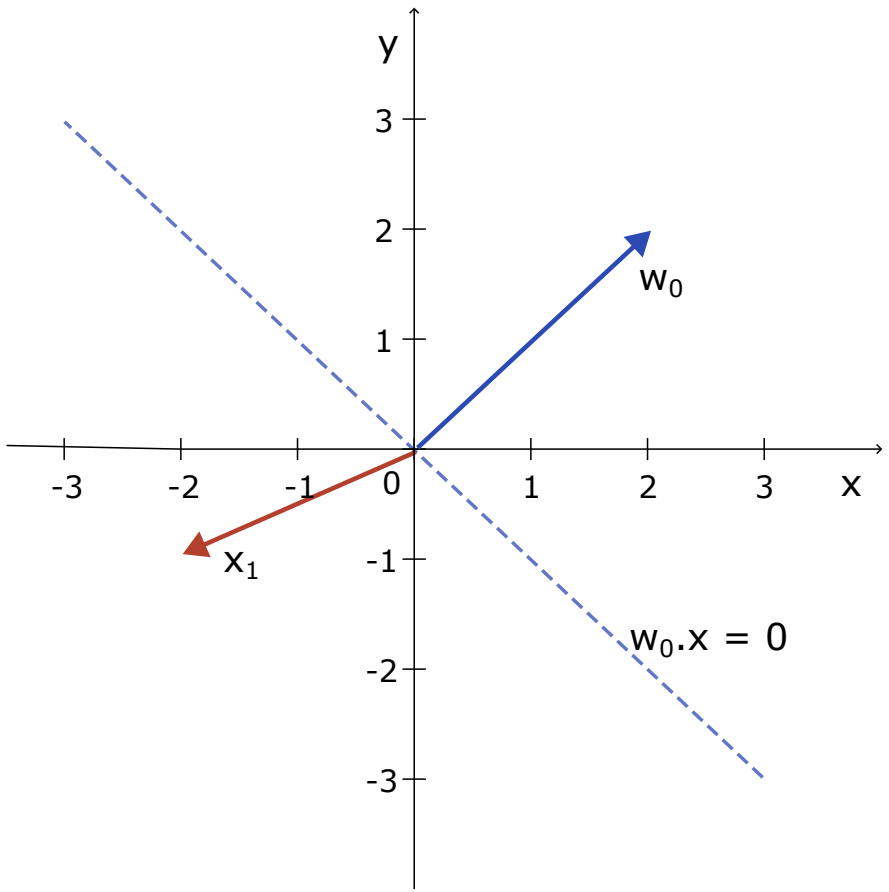
Myšlenka a odvození:

- $\vec{w_0}$ ... počáteční (aktuální) vektor vah
- $(\vec{x_1}, d_1 = 1) \in P$ ... trénovací vzor, pro který model dává nesprávný výstup: $\vec{w_0} \cdot \vec{x_1} \leq 0$

Co uděláme:

- otočíme $\vec{w_0}$, aby se zmenšil úhel mezi $\vec{w_0}$ a $\vec{x_1}$ $\rightarrow$ ideálně $|\alpha| < 90$

Geometrická interpretace:



Perceptron - Rosenblattův algoritmus učení

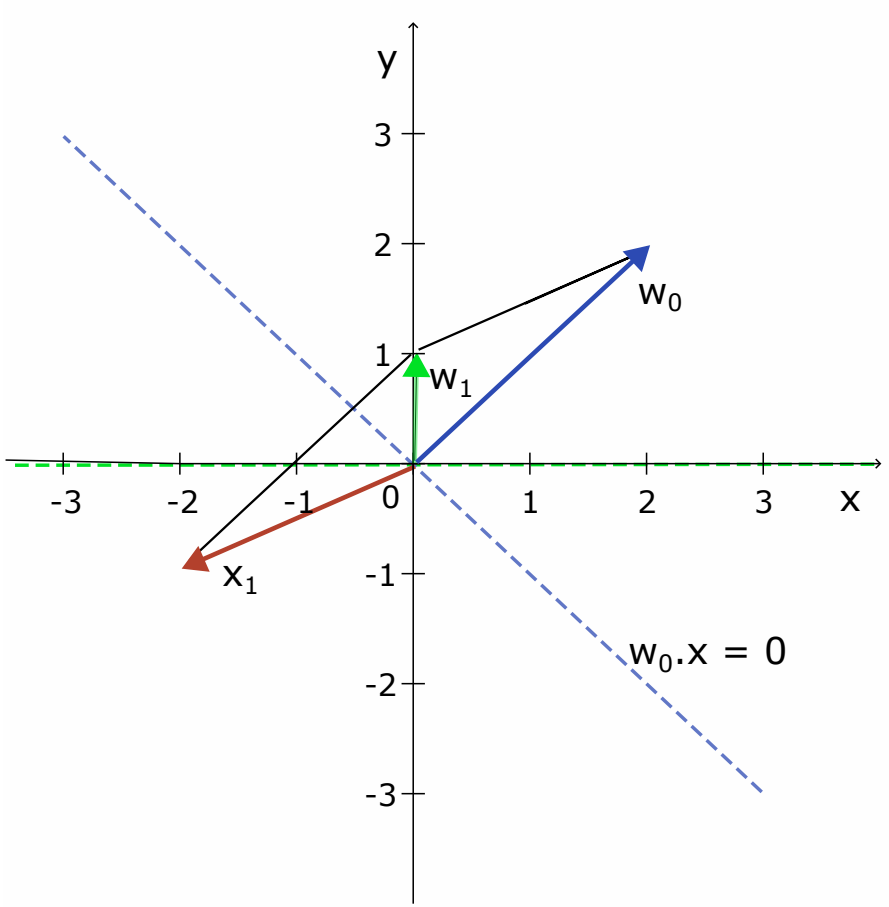
Myšlenka a odvození:

- $\vec{w_0}$ ... počáteční (aktuální) vektor vah
- $(\vec{x_1}, d_1 = 1) \in P$ ... trénovací vzor, pro který model dává nesprávný výstup:
  $\vec{w_0} \cdot \vec{x_1} \leq 0$

Co uděláme:

- otočíme $\vec{w_0}$, aby se zmenšil úhel mezi $\vec{w_0}$ a $\vec{x_1}$
  $\rightarrow$ ideálně $|\alpha| < 90$
  $\rightarrow$ přičteme $\vec{x_1}$ k $\vec{w_0}$
  $\vec{w_1} = \vec{w_0} + \vec{x_1}$

Geometrická interpretace:



Perceptron - Rosenblattův algoritmus učení

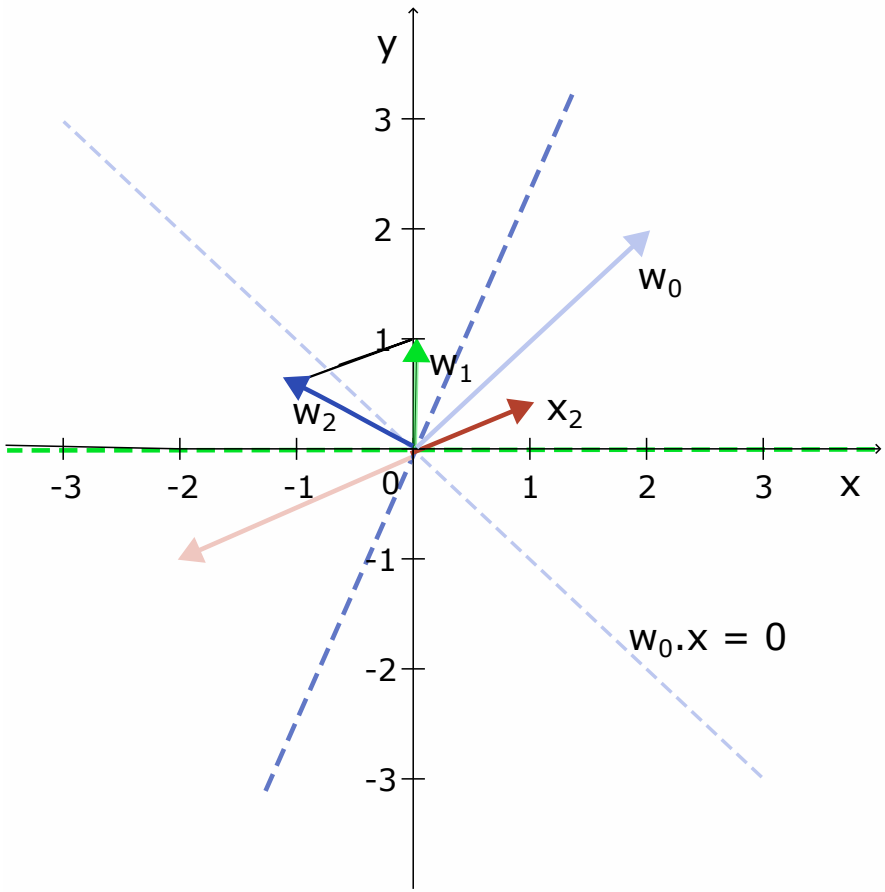
Myšlenka a odvození:

- $\vec{w_1}$ ... aktuální vektor vah
- trénovací vzor $(\vec{x_2}, d_2 = -1) \in \overline{P}$, pro který: $\vec{w_1} \cdot \vec{x_2} \geq 0$

Co uděláme:

- otočíme $\vec{w_1}$, aby se zvětšil úhel mezi $\vec{w_1}$ a $\vec{x_2}$
  $\rightarrow$ ideálně $|\alpha| > 90$
  $\rightarrow$ odečteme $\vec{x_2}$ od $\vec{w_0}$
  $\vec{w_2} = \vec{w_1} - \vec{x_2}$

Geometrická interpretace:



Perceptron - Rosenblattův algoritmus učení (1959)

1. Inicializuj váhy:
   $\vec{w}(0) = (w_0, w_1, ..., w_n)^T$ ... vektor vah v čase 0 (včetně prahu/biasu) $w_0 = b = -h$
2. Předlož další trénovací vzor $(\vec{x}_t, d_t)$:
   $\vec{x}_t = (x_{t0} = 1, x_{t1}, , ..., x_{tn})$ ... vstupní vzor
   d_t ... požadovaný výstup
3. Spočti skutečný výstup (odezvu sítě):
   $y_t = sign(\vec{x}_t \vec{w})$
4. Adaptuj váhy:

$$\vec{w}(t+1) = \begin{cases} \vec{w}(t) & \text{pokud } y(t) = d(t) \\ \vec{w}(t) + \vec{x}_t^T & \text{pokud } y_t \neq 1, d_t = 1 \\ \vec{w}(t) - \vec{x}_t^T & \text{pokud } y_t \neq -1, d_t = -1 \end{cases}$$

   jinak napsáno: $\vec{w}(t+1) = \vec{w}(t) + \vec{x}_t^T sign(d_t - y_t)$
5. Pokud t nedosáhl maximální hodnoty, přejdi ke kroku 2.

Perceptron - Rosenblattův algoritmus učení (1959)

Jak předkládat trénovací vzory? .... různé strategie:

1. **iterativně**: jeden po druhém
   - **po tzv. epochách**: během jedné epochy se každý vzor se předloží právě jednou
   - počet epoch .... kolikrát se předloží celá trénovací množina
2. **náhodně** ... v každé iteraci se zvolí náhodný trénovací vzor
3. vhodná **kombinace předchozích strategií**:
   1. vrámci každé epochy vzory náhodně uspořádáme
   2. nejprve předkládáme vzory náhodně, na konci učení systematicky projdeme všechny vzory

Perceptron - Rosenblattův algoritmus učení (1959)

Jak inicializovat váhy? .... různé strategie:

- $\vec{w}(0) = \vec{0}$
- náhodně: malé náhodné hodnoty
- použít nějakou heuristiku: např. průměr vzorů z P - průměr vzorů z $\overline{P}$
- ...

Kdy ukončit učení?

1. předem daný počet iterací nebo epoch
2. jakmile $E = 0$, popř. jakmile je chyba dostatečně malá ... $E < E_{min}$

Perceptron - Rosenblattův algoritmus učení (1959)

Co když jsou vstupní vzory různě velké?

- nebo co když je jeden nebo několik vzorů tzv. odlehlých (outliers)?
  $\rightarrow$ může to vést k výraznému zpomalení učení
- **Řešení:** normalizace vstupních vektorů na stejnou velikost:

$$\vec{x}_{new} = \frac{\vec{x}}{|\vec{x}|} = \frac{\vec{x}}{\sqrt{x_0^2 + x_1^2 + ... + x_n^2}}$$

  $\rightarrow$ normalizovaný Rosenblattův algoritmus učení

Perceptron - Rosenblattův algoritmus učení (1959)

Výhody

- Triviální algoritmus
- Pro lineárně separabilní množiny algoritmus konverguje, tj. nalezne správné řešení v konečném počtu kroků *(Rosenblatt, 1959)*

Nevýhody

- Velmi pomalý algoritmus
  - skutečný počet kroků roste exponenciálně s počtem vstupů
  - odlehlé vzory (pokud neprovedeme normalizaci)
- Umí klasifikovat jen lineárně separabilní množiny
- Chybí rozšíření pro více vrstev
- Špatné zobecňování ... nemusí najít ,,ideální" dělící nadrovinu

Perceptron - Rosenblattův algoritmus učení (1959)

Příklad 1

	x_0	x_1	x_2	d
a	+1	-1	-1	+1
b	+1	-1	+1	+1
c	+1	+1	-1	+1
d	+1	+1	+1	-1

- $\vec{w}(0) = \vec{0}$
- vzory předkládáme iterativně (v rámci epochy náhodně)
  c, b, d, a; b, a, c, d; ...

Perceptron - Rosenblattův algoritmus učení (1959)

Příklad 1

	x_0	x_1	x_2	d
a	+1	-1	-1	+1
b	+1	-1	+1	+1
c	+1	+1	-1	+1
d	+1	+1	+1	-1

Řešení c, b, d, a; b, a, c, d; ...

	w_0	w_1	w_2	d	ξ	y	operace
$\vec{w}(0)$	0	0	0				
c	+1	+1	-1	+1	0	0	+
$\vec{w}(1)$	+1	+1	-1				
b	+1	-1	+1	+1	-1	-1	+
$\vec{w}(2)$	+2	0	0				

Perceptron - Rosenblattův algoritmus učení (1959)

Příklad 1

	x_0	x_1	x_2	d
a	+1	-1	-1	+1
b	+1	-1	+1	+1
c	+1	+1	-1	+1
d	+1	+1	+1	-1

Řešení c, b, d, a; b, a, c, d; ...

	w_0	w_1	w_2	d	ξ	y	operace
$\vec{w}(2)$	+2	0	0				
d	+1	+1	+1	-1	+2	+1	-
$\vec{w}(3)$	+1	-1	-1				
a	+1	-1	-1	+1	+3	+1	nic

Perceptron - Rosenblattův algoritmus učení (1959)

Příklad 1

	x_0	x_1	x_2	d
a	+1	-1	-1	+1
b	+1	-1	+1	+1
c	+1	+1	-1	+1
d	+1	+1	+1	-1

Řešení c, b, d, a; b, a, c, d; ...

	w_0	w_1	w_2	d	ξ	y	operace
$\vec{w}(4)$	+1	-1	-1				
b	+1	-1	+1	+1	+1	+1	nic
a	+1	-1	-1	+1	+3	+1	nic
c	+1	+1	-1	+1	+1	+1	nic
d	+1	+1	+1	-1	-1	-1	nic, konec

$$\rightarrow \vec{w} = (+1, -1, -1)^T$$

Perceptron - Další algoritmy učení

Další varianty perceptronového učícího algoritmu

1. Dávkový algoritmus učení
   - celou trénovací množinu předložíme najednou:

$$\vec{w}(t+1) = \vec{w}(t) + \sum_{p=1}^{N} \vec{x}_p^T \, sign(d_p - y_p)$$

2. Rosenblattův algoritmus s parametrem učení
   - při přičítání/odčítání vzory vážíme:
     $\vec{w}(t+1) = \vec{w}(t) + \alpha \vec{x}_t^T \, sign(d_t - y_t)$
3. Přihrádkový algoritmus
   - pro nalezení optimálního řešení i pro lineárně neseparabilní množiny

Hebbovo učení

- každý trénovací vzor se předloží právě jednou:

$$\vec{w}(t+1) = \vec{w}(t) + d_t \vec{x}_t^T$$

Perceptron - Dávkový algoritmus učení

- celou trénovací množinu předložíme najednou:

$$\vec{w}(t+1) = \vec{w}(t) + \sum_{p=1}^{N} \vec{x}_p^T \, sign(d_p - y_p)$$

- Pro maticovou reprezentaci:
  $w(t) = (w_0, w_1, ..., w_n)^T$
  $T = (X, \vec{d})$

$x_{10} = 1$	x_{11}	...	x_{1n}	d_1
...	...		...	...
$x_{N0} = 1$	x_{N1}	...	x_{Nn}	d_N

$$\vec{w}(t+1) = \vec{w}(t) + X^T \, sign(\vec{d} - \vec{y})$$

Perceptron - Dávkový algoritmus učení

1. Inicializuj váhy:
   $\vec{w}(0) = (w_0, w_1, ..., w_n)^T$ ... vektor vah v čase (epoše) 0
2. Předlož celou trénovací množinu $(X, \vec{d})$

$x_{10} = 1$	x_{11}	...	x_{1n}	d_1
...	...		...	
$x_{N0} = 1$	x_{N1}	...	x_{Nn}	d_N

3. Spočti skutečný výstup (odezvu sítě) $\vec{y}$:

$$\vec{y} = sign(X\vec{w})$$

4. Adaptuj váhy:

$$\vec{w}(t+1) = \vec{w}(t) + X^T sign(\vec{d} - \vec{y})$$

5. Pokud počet epoch nedosáhl maximální hodnoty, přejdi ke kroku 2.

Perceptron - Dávkový algoritmus učení

Výhody a nevýhody

- maticová reprezentace, možnost paralelizace výpočtu
- často rychlejší a stabilnější konvergence než u Rosenblattova algoritmu (ale ne vždy)
- výsledek učení nezávisí na pořadí vzorů
- větší paměťová náročnost
- důkaz konvergence? ... konečnost algoritmu nelze zaručit :(

Bipolární Perceptron - Hebbovo učení (1949)

- každý trénovací vzor předložíme právě jednou:
  1. Inicializuj váhy:
     $\vec{w}(0) = \vec{0}^T$
  2. Pro každý trénovací vzor $\vec{x}_t$ $(t = 1, \cdots, N)$ uprav váhy:

$$\vec{w}(t+1) = \vec{w}(t) + d_t \vec{x}_t^T$$

maticově:

$$\vec{w} = X^T \vec{d}$$

Výhody a nevýhody

- jednodušší než Rosenblattův algoritmus
- při učení není třeba počítat skutečný výstup
- výsledek učení nezávisí na pořadí vzorů
- váhy lze snadno interpretovat
- nezaručuje nalezení perfektního řešení

Bipolární Perceptron - Hebbovo učení (1949)

Příklad 1

	x_0	x_1	x_2	d
a	+1	-1	-1	+1
b	+1	-1	+1	+1
c	+1	+1	-1	+1
d	+1	+1	+1	-1

Bipolární Perceptron - Hebbovo učení (1949)

Příklad 1

	x_0	x_1	x_2	d
a	+1	-1	-1	+1
b	+1	-1	+1	+1
c	+1	+1	-1	+1
d	+1	+1	+1	-1

Řešení:

	w_0	w_1	w_2
$\vec{w}(0)$	0	0	0
$d_a\vec{x}_a$	+1	-1	-1
$\vec{w}(1)$	+1	-1	-1
$d_b\vec{x}_b$	+1	-1	+1
$\vec{w}(2)$	+2	-2	0

Bipolární Perceptron - Hebbovo učení (1949)

Příklad 1

	x_0	x_1	x_2	d
a	+1	-1	-1	+1
b	+1	-1	+1	+1
c	+1	+1	-1	+1
d	+1	+1	+1	-1

Řešení:

	w_0	w_1	w_2
$\vec{w}(2)$	+2	-2	0
$d_c\vec{x}_c$	+1	+1	-1
$\vec{w}(3)$	+3	-1	-1
$d_d\vec{x}_d$	-1	-1	-1
$\vec{w}(4)$	+2	-2	-2

$\rightarrow \vec{w} = (+2, -2, -2)^T$

Bipolární Perceptron - Hebbovo učení (1949)

Příklad 1

	x_0	x_1	x_2	d
a	+1	-1	-1	+1
b	+1	-1	+1	+1
c	+1	+1	-1	+1
d	+1	+1	+1	-1

Řešení maticově: $\vec{w} = X^T \vec{d}$

$$\begin{pmatrix} +1 & +1 & +1 & +1 \\ -1 & -1 & +1 & +1 \\ -1 & +1 & -1 & +1 \end{pmatrix} \begin{pmatrix} +1 \\ +1 \\ +1 \\ -1 \end{pmatrix} = \begin{pmatrix} +2 \\ -2 \\ -2 \end{pmatrix}$$

$\rightarrow \vec{w} = (+2, -2, -2)^T$ ... naučil se perceptron správně?

Bipolární Perceptron - Hebbovo učení (1949)

Příklad 1

Řešení : $\vec{w} = (+2, -2, -2)^T$ ... naučil se perceptron správně?

	x_0	x_1	x_2	d	ξ	y	
a	+1	-1	-1	+1	+6	+1	ok
b	+1	-1	+1	+1	+2	+1	ok
c	+1	+1	-1	+1	+2	+1	ok
d	+1	+1	+1	-1	-2	-1	ok

$\rightarrow$ ano (ale nemusí to tak zdaleka být vždy)

Poznámka

- stejný by byl první krok dávkového algoritmu učení pro $\vec{w}_0 = \vec{0}$
- Hebbovo učení můžeme použít pro inicializaci vah pro Rosenblattův algoritmus

Perceptron - Rosenblattův algoritmus s parametrem učení

1. Inicializuj váhy a práh malými náhodnými hodnotami:
   $\vec{w}(0) = (w_0, w_1, ..., w_n)$ ... vektor vah v čase 0 (včetně prahu/biasu) $w_0 = b = -h$
2. Předlož další trénovací vzor $(\vec{x}_t, d_t)$:
   $\vec{x}_t = (x_{t0} = 1, x_{t1}, , ..., x_{tn})$ ... vstupní vzor
   d_t ... požadovaný výstup
3. Spočti skutečný výstup (odezvu sítě):
   $y_t = sign(\vec{w} \cdot \vec{x}_t)$
4. Adaptuj váhy:

$$\vec{w}(t+1) = \begin{cases} \vec{w}(t) & \text{pokud } y(t) = d(t) \\ \vec{w}(t) + \alpha \vec{x}_t^T & \text{pokud } y_t \neq 1, d_t = 1 \\ \vec{w}(t) - \alpha \vec{x}_t^T & \text{pokud } y_t \neq -1, d_t = -1 \end{cases}$$

jinak napsáno: $\vec{w}(t+1) = \vec{w}(t) + \alpha \vec{x}_t^T sign(d_t - y_t)$
α ... parametr učení

Perceptron - Rosenblattův algoritmus s parametrem učení

Jak ovlivní volba parametru učení výsledek?

- výrazné zrychlení učení
- Doporučení: $\alpha \in (0, 1]$
- Nejlépe: α zpočátku velké, postupně $\alpha \to 0$

Výhody a nevýhody

- Obvykle rychlejší než Rosenblattův algoritmus
- Pro lineárně separabilní množiny algoritmus konverguje, tj. nalezne správné řešení v konečném počtu kroků *(Rosenblatt, 1959)*
- Skutečný počet kroků roste exponenciálně s počtem vstupů
- Problém je, pokud množiny vzorů nejsou lineárně separabilní $\to$ algoritmus diverguje

Perceptron - Přihrádkový algoritmus (Gallant, 1990)

Idea

- Použiji (iterativní) Rosenblattův algoritmus učení
- Nejlepší doposud nalezený vektor vah mám uložený v přihrádce
- Pokud najdu lepší váhový vektor (tj. s menší chybou), uložím ho do přihrádky

Výhody

- I pro lineárně neseparabilní množiny vzorů nalezne algoritmus nejlepší řešení.
  - Pokud je trénovací množina konečná a složky váhového vektoru a vstupních vektorů jsou racionální, lze ukázat, že přihrádkový algoritmus konverguje k optimálnímu řešení s pravděpodobností 1 (Gallant, 1990).

Neuronové sítě 1 - hodina 3: Umělý neuron

Příklady - 1. Ukázky učících algoritmů v Pythonu

rosenblatt_perceptron.ipynb

- Rosenblattův učící algoritmus - vybrané varianty (iterativní, dávkový, s parametrem učení), Hebbovo učení
- Zobrazení dat a dělící nadroviny
- Přiklady: učení jednoducých logických funkcí:

Negace AND

x_1	x_2	d
-1	-1	+1
-1	+1	+1
+1	-1	+1
+1	+1	-1

Ukázka: Jak se perceptron učí pomocí různých variant algoritmu a jak se postupně posouvá dělící nadrovina.

XOR (Exkluzivní OR)

x_1	x_2	$d = x_1 \otimes x_2$
-1	-1	-1
-1	+1	+1
+1	-1	+1
+1	+1	-1

Ukázka: Jak probíhá učení perceptronu v případě, že data nejsou lineárně separabilní.

Příklady - 2. Různé praktické úlohy

perceptron_examples.ipynb

- Učení jednoducých logických funkcí
- Data s odlehlými vzory.
- Náhodně vygenerovaná data.
- Písmena.
- Ručně psané číslice.

Učení jednoducých logických funkcí

Příklad 1 – Žlučníkový záchvat

vlašák	bůček	léky	Bude mi zle?
+1	-1	-1	+1
+1	-1	+1	-1
+1	+1	-1	+1
-1	+1	+1	-1
+1	+1	-1	+1
+1	+1	+1	+1

- Jak dlouho se bude perceptron učit? Naučí se danou úlohu?
- Jsou nějaké rozdíly mezi učícími algoritmy?

Učení jednoducých logických funkcí

Příklad 2 - Hospoda (variace na majoritní obvod)

Pavel	Pepa	Honza	Jde se na pivo?
+1	-1	-1	-1
+1	-1	+1	+1
-1	+1	-1	-1
-1	+1	+1	+1
+1	+1	-1	+1
+1	+1	+1	+1
-1	-1	-1	-1
-1	-1	+1	-1

- Jak dlouho se bude perceptron učit? Naučí se danou úlohu?
- Jsou nějaké rozdíly mezi učícími algoritmy?

Příklady - Různé praktické úlohy

Vzájemné porovnávání modelů strojového učení, zde různých variant Rosenblattova algoritmu učení

- Obykle nás zajímá chyba (jak dobře se model úlohu naučil) a efektivita (zde počet epoch/čas)
- Protože je Rosenblattův algoritmus částečně stochastický, průběh a výsledek učení může být pokaždé jiný
- Proto je lepší zopakovat experiment např. 100 krát a porovnat průměrné hodnoty veličin (případně jejich rozptyl).

Pro různé úlohy může vyjít jako nejlepší jiná varianta modelu/algoritmu.

Příklady - Různé praktické úlohy

Příklad 3 - Data s odlehlými vzory

- Máme data, kde vstupní vektory mají různou délku (zde je odlehlý 3. vzor)

x_1	x_2	y
-0.5	-0.5	1
0.3	-0.5	1
-40	50	-1
-0.5	0.5	-1
-0.1	1.0	1

- Jak dlouho se bude perceptron učit? Naučí se danou úlohu?
- Jak dlouho se bude perceptron učit, pokud normalizujeme všehny vstupní vektory na délku 1?

Příklady - Různé praktické úlohy

Příklad 4 - Náhodně generovaná data

- Pokud porovnáváme různé modely nebo algoritmy, je dobré začít s uměle, např. náhodně, generovanými daty
- V tomto případě data nejsou zcela lineárně separabilní (je přidán náhodný šum)
- Generujte data opakovaně (různě) a pozorujte, jak se perceptrony učí. Která varianta Rosenblattova algoritmu je podle vás pro tuto úlohu nejlepší?

Příklad 5 - Náhodně generované shluky

- Potom vygenerujeme data obsahující dva shluky vzorů
- Generujte data opakovaně (různě) a pozorujte, jak se perceptrony učí. Která varianta Rosenblattova algoritmu je podle vás pro tuto úlohu nejlepší?

Příklady - Různé praktické úlohy

Příklad 5 - Písmena letters_example.ipynb

- Využijeme připravenou datovou sadu **letters.csv**
- Písmena byla segmentována z **letters.png**
- Prohlédněte si datovou sadu a zobrazte si některá písmenka
- Vytvoříme testovací množinu: data s přidaným šumem nebo následně vyhlazená
- Naučte perceptron pomocí různých algoritmů (a variant) rozpoznávat jednotlivá písmena.
- Určete chybu klasifikace na trénovací množině i na testovacích množinách (popř. i počet epoch / čas učení)
- S jak moc velkým šumem v datech si ještě perceptron poradil?
- Podívejte se, se kterými písmenky měl perceptron největší problémy.
- Který učící algoritmus si vedl nejlépe?

Příklady - Různé praktické úlohy

Příklad 6 - Ručně psané číslice

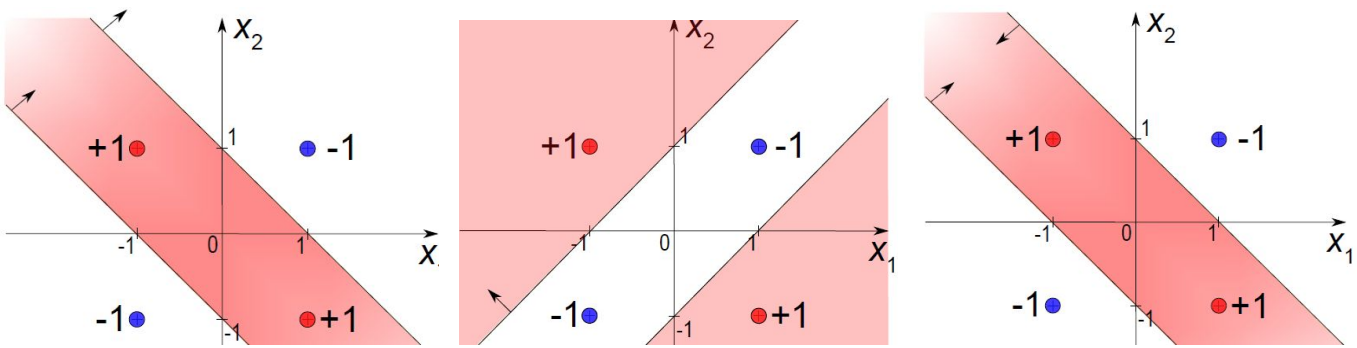
- Využijeme připravenou datovou sadu **OcrData.csv** s ručně psanými číslicemi
- Prohlédněte si datovou sadu a zobrazte si některé číslice (využijte předpřipravený skript)
- Naučte perceptron pomocí různých algoritmů (a variant) rozpoznávat jednotlivé číslice.
- Určete (a porovnejte) chybu klasifikace na trénovací množině (popř. i počet epoch / čas učení)
- Podívejte se, se kterými číslicemi měl perceptron největší problémy
- Který učící algoritmus si vedl nejlépe?

Neuronové sítě 1 - hodina 3: Umělý neuron

XOR - Řešení dobrovolného domácího úkolu

Příklad : Exkluzivní OR (XOR) - dobrovolný domácí úkol

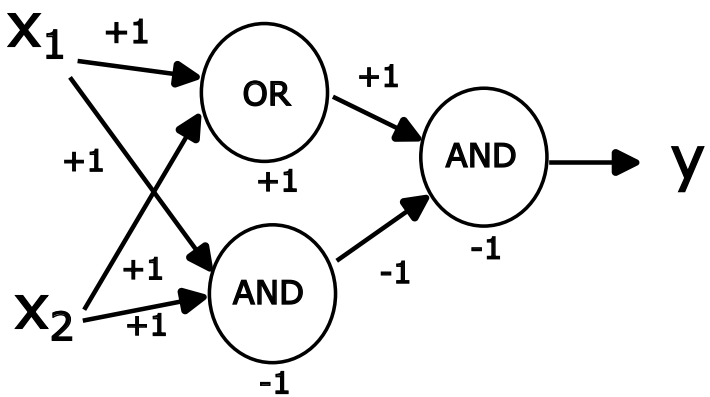
1. XOR můžeme pomocí základních logických operací (AND, OR, NOT) reprezentovat různě, zvládnete navrhnout několik různých způsobů?
2. Navrhněte co nejmenší neuronovou síť, která bude reprezentovat XOR. Kolik bude obsahovat neuronů?

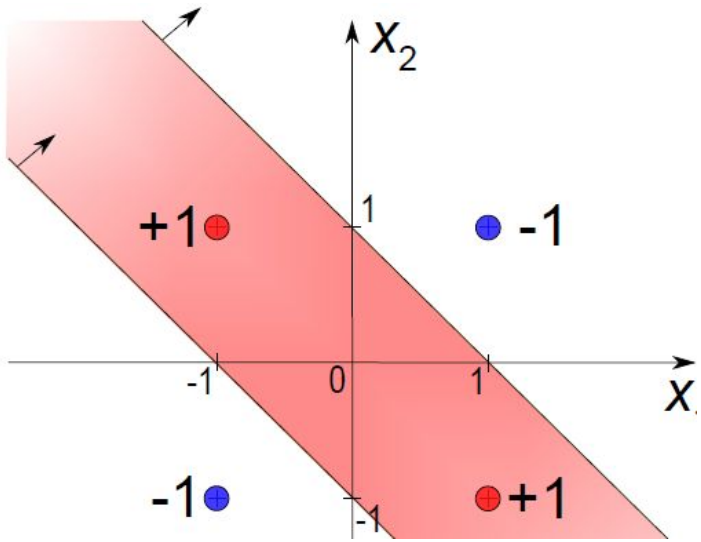


XOR - Řešení dobrovolného domácího úkolu

Příklad : Exkluzivní OR (XOR)

- 1. řešení: $x_1 \otimes x_2 = (x_1 \vee x_2) \wedge \neg(x_1 \wedge x_2)$

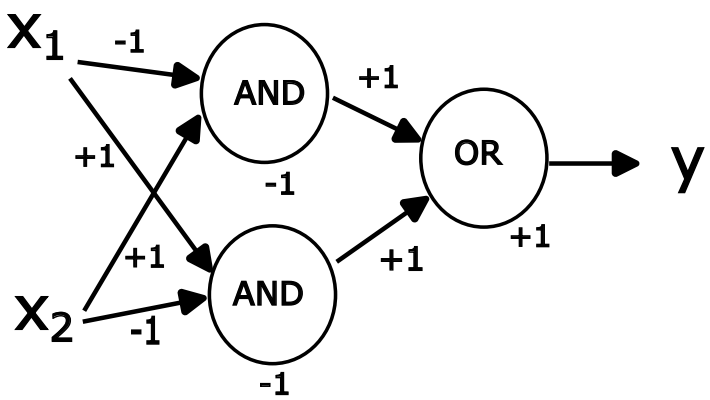


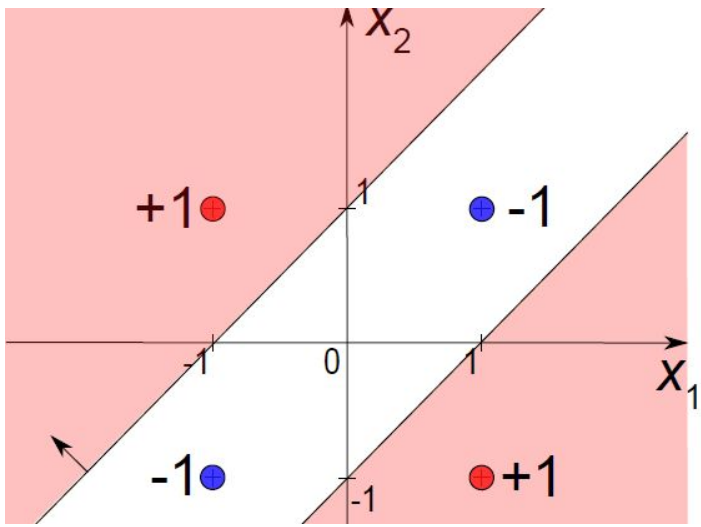


XOR - Řešení dobrovolného domácího úkolu

Příklad : Exkluzivní OR (XOR)

- 2. řešení: $x_1 \otimes x_2 = (x_1 \wedge \neg x_2) \vee (\neg x_1 \wedge x_2)$





XOR - Řešení dobrovolného domácího úkolu

Příklad : Exkluzivní OR (XOR)

- 3. řešení:
  $x_1 \otimes x_2 = (x_1 \vee x_2) \wedge \neg(x_1 \wedge x_2) = (x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2)$