Neuronové sítě 1 - Neurony se spojitou přenosovou funkcí

18NES1 - 7. - 8. hodina, LS 2024/25

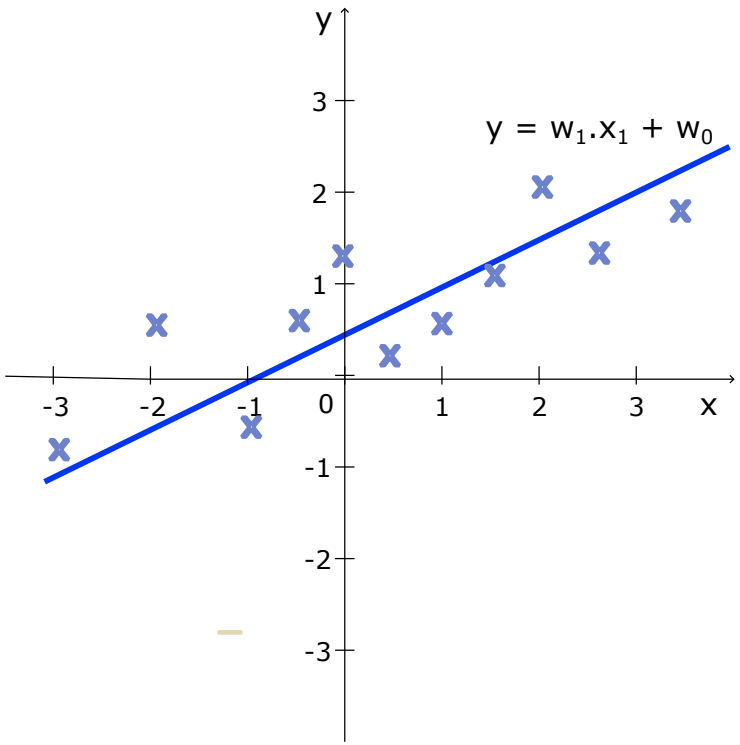
Zuzana Petříčková

16. dubna 2025

Co jsme probírali minule

Lineární neuron a úloha lineární regrese

- vnitřní potenciál: $\xi = \sum_{i=1}^{n} w_i.x_i + w_0$
- výstup: $y = \xi$



- body ve vstupním prostoru prokládáme nadrovinou (přímkou, rovinou,...)
- předpokládáme, že mezi vstupními veličinami $x_1, ..., x_n$ a výstupní y je lineární závislost

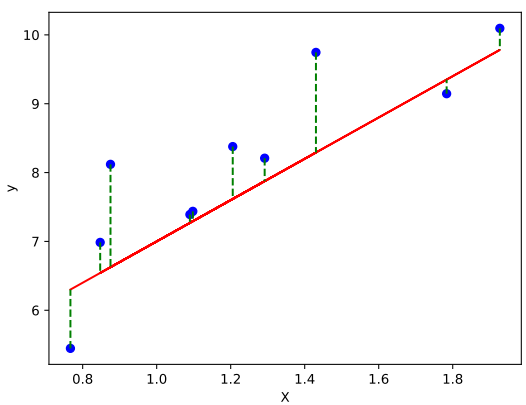
Co jsme probírali minule

Lineární neuron a úloha lineární regrese

- během učení pro každý vzor minimalizujeme rozdíl mezi skutečným a požadovaným výstupem
- místo absolutní hodnoty se používá kvadratická chybová funkce (součet čtverců chyb, SSE)

$$E = \frac{1}{2}\sum_p e_p^2 = \frac{1}{2}\sum_p (d_p - y_p)^2$$

$\rightarrow$ metody nejmenších čtverců



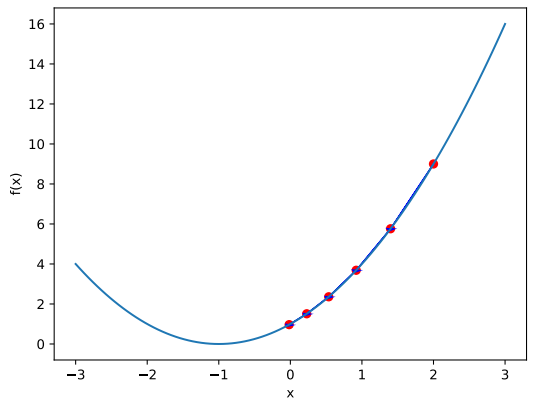
- Algoritmy učení pro lineární neuron (lineární regresi):
  - **Metoda LSQ** - založena na explicitním výpočtu
  - **Gradientní metoda** - lokální metoda (s obecnějším využitím)

Co jsme probírali minule

Gradientní metoda (metoda největšího spádu, gradient descent):

→ **úloha:**

- máme funkci $f(\vec{x}) : R^n \rightarrow R$
- hledáme $\vec{x}$, pro které je $f(\vec{x})$ minimální



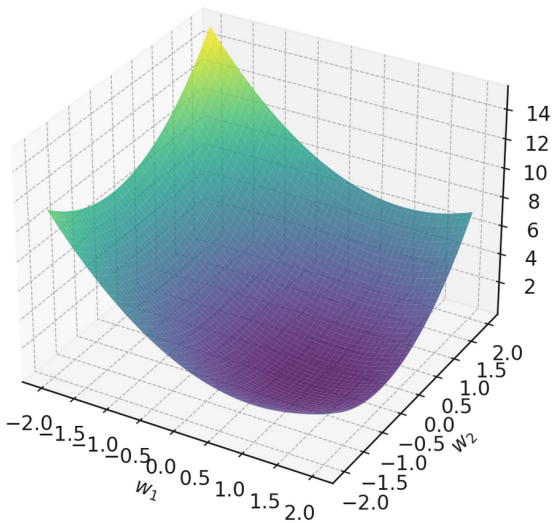
→ **řešení:**

1. začneme v nějakém počátečním bodě $\vec{x}(0)$
2. spočteme gradient $\nabla f(\vec{x}) = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \ldots, \frac{\partial f}{\partial x_n}\right)$
   gradient vyjadřuje směr a velikost největšího růstu funkce v daném bodě
3. v cyklu se posunujeme ,,o kousek" proti směru gradientu:
   $\vec{x}(t+1) = \vec{x}(t) - \alpha \nabla f(\vec{x})$
   α je malé kladné číslo (délka kroku, parametr učení)
   pro jeden vstupní příznak: $x_i(t+1) = x_i(t) - \alpha \frac{\partial f}{\partial x_i}$

Učení lineárního neuronu gradientní metodou

- minimalizujeme chybovou funkci SSE v prostoru vah:

$$E(\vec{w}) = \frac{1}{2}\sum_{p=1}^{N}(d_p - y_p)^2 = \frac{1}{2}\sum_{p=1}^{N}\left(d_p - \sum_{i=0}^{n} w_i.x_{pi}\right)^2 = \sum_{p=1}^{N} E_p(\vec{w})$$



- jedná se o kvadratickou funkci, proto by gradientní metoda při správném nastavení parametrů neměla mít problém najít její globální minimum

Učení lineárního neuronu gradientní metodou

Obecné schéma algoritmu (GD, gradient descent)

1. Inicializuj váhy malými náhodnými reálnými hodnotami ....
   $\vec{w}(0) = (w_0, w_1, ..., w_n)^T$
   Inicializuj parametr učení α_0.... $1 \gg \alpha_0 > 0$
2. Předlož další trénovací vzor $(\vec{x}_t, d_t)$ a spočti skutečný výstup neuronu: $y_t = \vec{x}_t \vec{w}$
3. Adaptuj váhy (proti směru gradientu chybové funkce):

$$\vec{w}(t+1) = \vec{w}(t) - \alpha \nabla E_t(\vec{w}) = \vec{w}(t) + \alpha_t (d_t - y_t) \vec{x}_t^T$$

4. Případně aktualizuj parametr učení : $\alpha_t \to \alpha_{t+1}$
5. Pokud není konec, přejdi ke kroku 2.

Gradientní metoda

- minule jsme se seznámili s různými variantami gradientní metody, s jejími hyperparametry a na příkladech jsme si ukázali důležitost jejich správného nastavení

Závěr

- gradientní metoda je schopná řešit úlohu lineární regrese stejně dobře jako klasická LSQ metoda, **ALE** je náročnější na aplikaci:
  - je to lokální metoda: průběh i výsledek bude pokaždé trochu jiný
  - metoda je velmi citlivá na správné nastavení různých hyperparametrů, inicializaci a předzpracování dat
- výhodou oproti LSQ je, že je méně citlivá na numerické chyby a můžeme ji aplikovat na výrazně širší okruh úloh, kde si klasická LSQ metoda neporadí (nebo ne tak dobře):
  - ,,velká data" (hodně příznaků nebo vzorů)
  - data s odlehlými vzory nebo s velkým šumem,...

Dnešní hodina

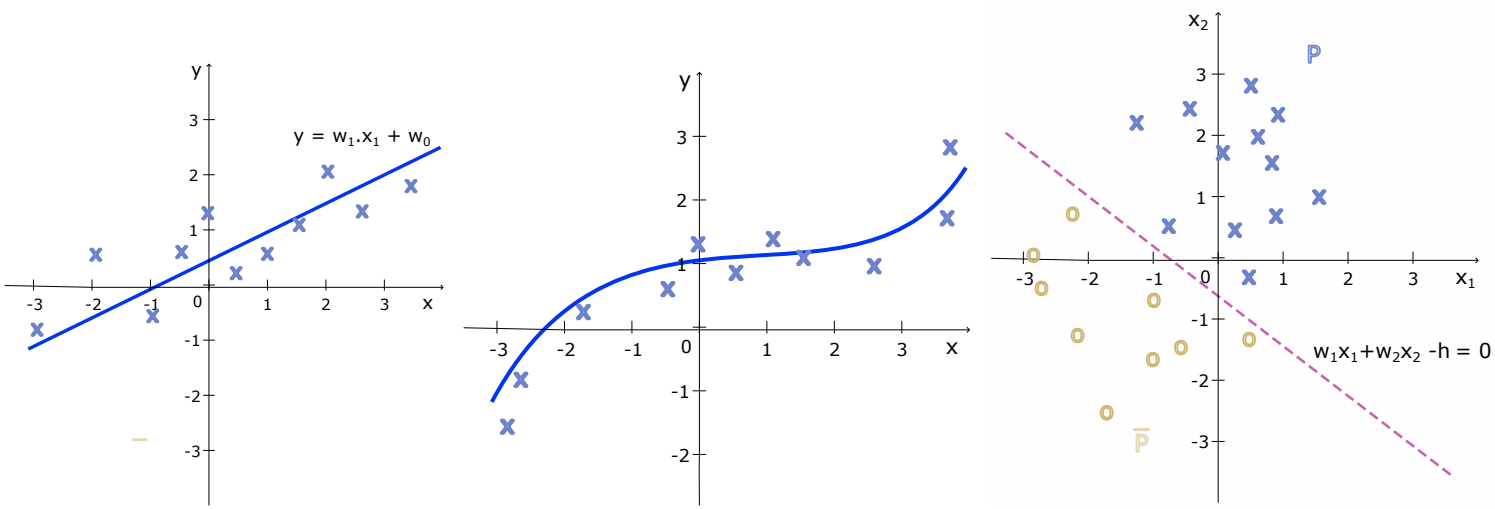
1. Perceptron se spojitou přenosovou funkcí
   - Nejznámější přenosové funkce pro perceptron
   - Učení obecného perceptronu gradientní metodou
2. Neuronová síť tvořená jednou vrstvou neuronů

Od lineární regrese k obecnějšímu modelu perceptronu

- pomocí gradientní metody umíme učit lineární neuron a řešit úlohu lineární regrese

Otázka: Nešla by gradientní metoda použít i pro jiné přenosové funkce?

- pak bychom uměli řešit i úlohu nelineární regrese nebo klasifikace:

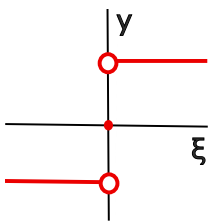


Od lineární regrese k obecnějšímu modelu perceptronu

Otázka: Nešla by gradientní metoda použít i pro jiné přenosové funkce?

- ano, ale jen pro některé: podmínkou je **spojitost** a **diferencovatelnost** (hladkost) přenosové funkce

$\rightarrow$ pro skokovou přenosovou funkci nejde použít:
(otázka: jaká je její derivace?)

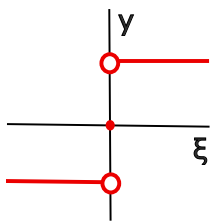


$\rightarrow$ nešla by skoková funkce nahradit nějakou jinou přenosovou funkcí, co by byla spojitá a derivovatelná?

Hledáme spojité funkce, co by se co nejvíce podobaly skokové přenosové funkci

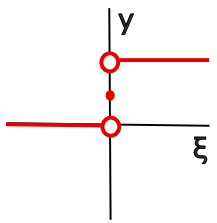
Pro bipolární model neuronu

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



Pro binární model neuronu

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



Hledáme spojité funkce, co by se co nejvíce podobaly skokové přenosové funkci

Požadavky:

- $f(y)$ definovaná a spojitá na $(-\infty, \infty)$
- $lim_{y \to -\infty} f(y) = m < M = lim_{y \to \infty} f(y)$
- $f(y) \approx m$ ... neuron je pasivní
- $f(y) \approx M$ ... neuron je aktivní

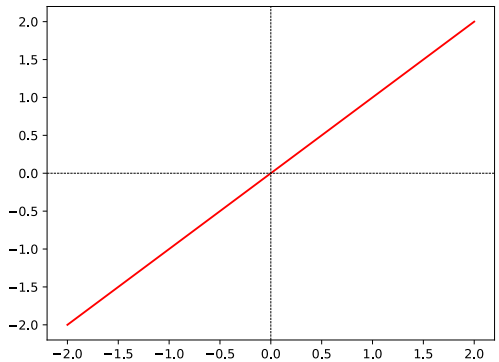
Typické příklady:

- $(m, M) = (-\infty, \infty)$ ... lineární funkce
- $(m, M) = (0, 1)$ - binární model ... sigmoidální funkce
- $(m, M) = (-1, 1)$ - bipolární model ... hyperbolický tangens

Spojité přenosové funkce

Lineární (identita)

- $f(\xi) = \xi$



Využití

- nehodí se příliš pro klasifikační úlohy
- zato velmi vhodná pro regresní úlohy

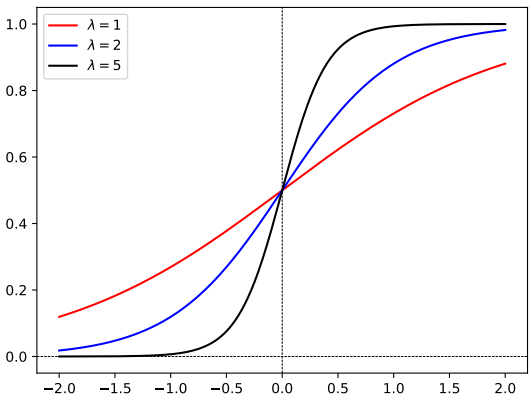
Kde se s ní setkáme:

- jednovrstvá lineární neuronová síť
- ve výstupní vrstvě vícevrstvých/hlubokých sítí u regresních úloh

Spojité přenosové funkce

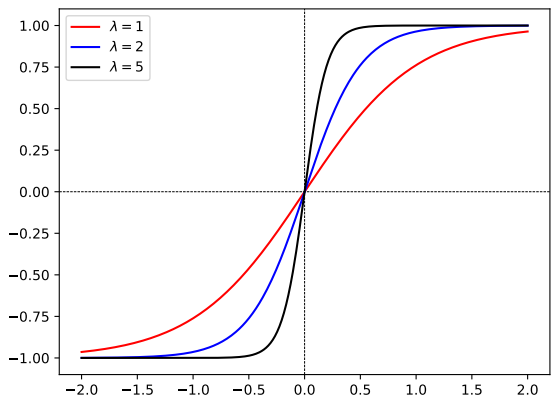
Sigmoidální

- $f(\xi) = \frac{1}{1+e^{-\lambda\xi}}$ ... logsig
- pro binární model



Hyperbolický tangens

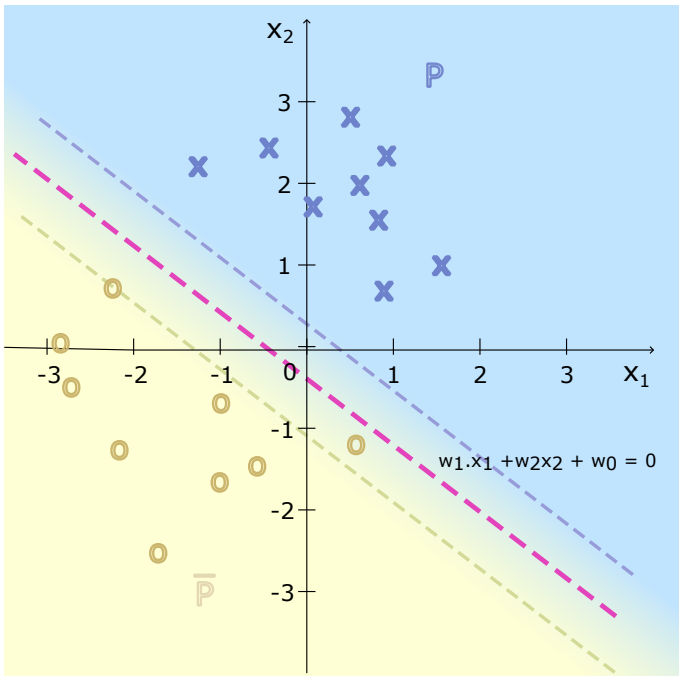
- $f(\xi) = \frac{1-e^{-2\lambda\xi}}{1+e^{-2\lambda\xi}}$ ... tanh
- pro bipolární model



$\rightarrow$,,rozvolněná" skoková přenosová funkce

Spojité přenosové funkce

Sigmoidální funkce a hyperbolický tangens - geometrická interpretace:



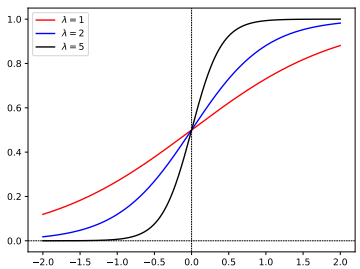
... úloha lineární klasifikace

- pro sigmoidální přenosovou funkci můžeme výstup neuronu přímo interpretovat jako **pravděpodobnost**, že vzor patří do dané třídy

Spojité přenosové funkce

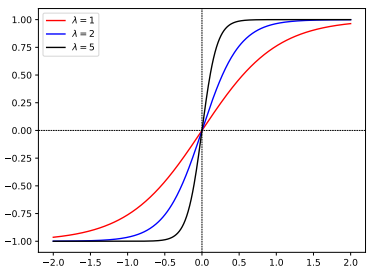
Sigmoidální

- $f(\xi) = \frac{1}{1+e^{-\lambda\xi}}$ ... logsig



Hyperbolický tangens

- $f(\xi) = \frac{1-e^{-2\lambda\xi}}{1+e^{-2\lambda\xi}}$ ... tanh



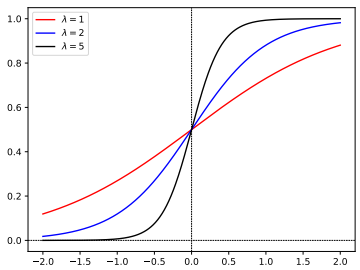
parametr λ ... strmost

- určuje míru nepřesnosti výsledku klasifikace na hranicích
  - $\lambda \to \infty$ ... skoková přenosová funkce
  - čím je λ menší ... tím je širší hranice mezi třídami
  - $\lambda \to 0$ ... neuron nerozlišuje (výstup vždy 0.5 resp. 0)
  - Obvyklá volba $\lambda = 1$ nebo $\lambda = 2$ pro logsig, $\lambda = 1$ pro tanh

Spojité přenosové funkce

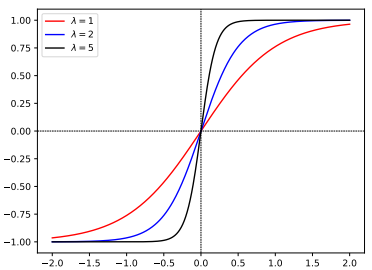
Sigmoidální

- $f(\xi) = \frac{1}{1+e^{-\lambda\xi}}$ ... logsig



Hyperbolický tangens

- $f(\xi) = \frac{1-e^{-2\lambda\xi}}{1+e^{-2\lambda\xi}}$ ... tanh

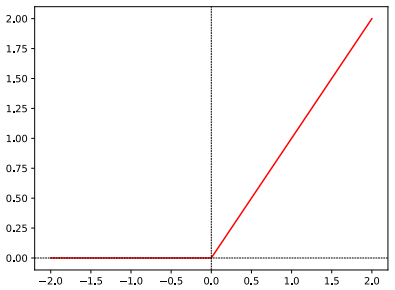


Využití

- pro klasifikační úlohy (rozvolněná prahová funkce)
- ve skrytých vrstvách vrstevnatých nebo hlubokých neuronových sítí, rekurentní sítě

Spojité přenosové funkce

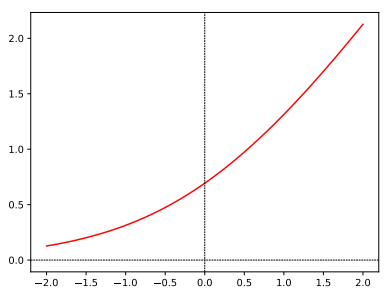
Pozitivně lineární (ReLU, rectified linear unit)



- $f(\xi) = max(0, \xi) = \begin{cases} x, & \text{pro } \xi > 0 \\ 0, & \text{pro } \xi \leq 0 \end{cases}$
  - není diferencovatelná, pouze po částech, ale jednoduše se s ní počítá
  - velmi často se používá ve skrytých vrstvách hlubokých neuronových sítí

Softplus (hladká alternativa k ReLU)

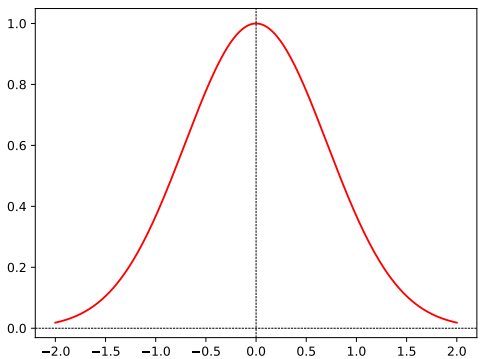
- $f(\xi) = \ln(1 + e^{\xi})$
- Pro velká ξ se chová jako ReLU
- Pro malá ξ se chová jako sigmoida
- Méně populární než ReLU



Lokální přenosové funkce

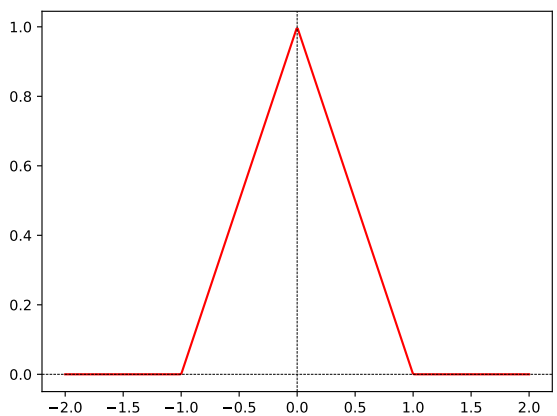
Radiální funkce (gausovská)

- $f(\xi) = e^{-\frac{\xi^2}{\alpha}}$ ... radbas
- $\xi = \frac{|\vec{x}-\vec{w}|}{\beta}$



Trojúhelníková funkce

- $f(\xi) = \begin{cases} 1 - |\xi| & \text{pro } |\xi| \leq 1 \\ 0 & \text{jinak} \end{cases}$
  ... tribas
- $\xi = \frac{|\vec{x}-\vec{w}|}{\beta}$



$\rightarrow$ sítě s lokálními jednotkami, RBF-sítě

Další přenosové funkce

Stochastický model neuronu: stochastická aktivační funkce

- $f(\xi) = 1$ s pravděpodobností $P(\xi)$
- $f(\xi) = 0$ s pravděpodobností $1 - P(\xi)$

$P(\xi)$ je nejčastěji sigmoidální funkce:

- $P(\xi) = \frac{1}{1+e^{-\frac{\xi}{T}}}$
- T ... pseudoteplota

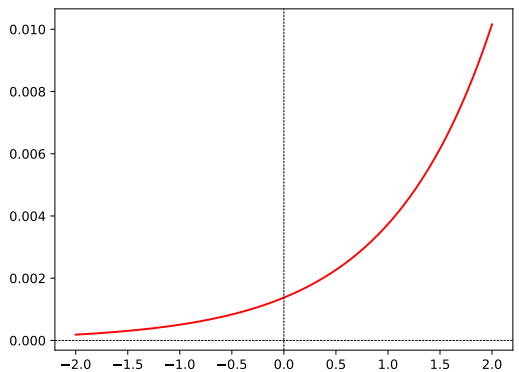
$\rightarrow$ Boltzmanovy stroje, Deep Belief Networks

- zjednodušeně: spočítáme hodnotu sigmoidální funkce, tu bereme jako pravděpodobnost a podle ní si hodíme si korunou na výstup

Další přenosové funkce

Softmax

- speciální přenosová funkce pro klasifikaci do více tříd
- zobecnění funkce argmax, převádí číselné hodnoty na pravděpodobnosti
- $f : R^n -> R^n$, $f(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{N} e^{x_j}}$



→ hodí se pouze do výstupní vrstvy pro klasifikační úlohy

Gradientní metoda pro učení neuronů se spojitou neuronovou funkcí

Gradientní metodu můžeme použít pro učení neuronu s libovolnou **spojitou diferencovatelnou** přenosovou funkcí f (např. sigmoidální funkce, hyperbolický tangens):

- výstup neuronu (pro jeden vzor): $y = f(\xi) = f(\sum_{i=0}^{n} w_i.x_i) = f(\vec{x}\vec{w})$
- maticově (pro celou trénovací množinu): $\vec{y} = f(X\vec{w})$ ($\vec{w}$ je sloupcový vektor)

1. budeme minimalizovat chybovou funkci SSE v prostoru vah:

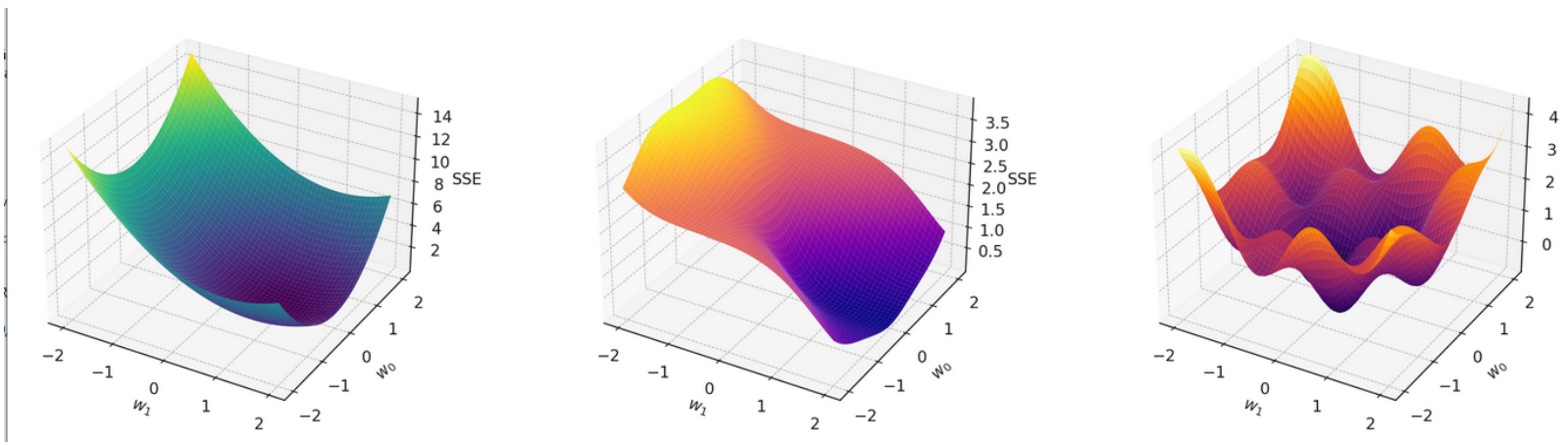
$$E(\vec{w}) = \frac{1}{2}\sum_{p=1}^{N}(d_p - y_p)^2 = \frac{1}{2}\sum_{p=1}^{N}\left(d_p - f(\sum_{i=0}^{n} w_i.x_{pi})\right)^2$$

$$= \sum_{p=1}^{N} E_p(\vec{w}) \ldots E_p(\vec{w}) \text{ je chybová funkce pro jeden vzor}$$

Gradientní metoda pro učení neuronů se spojitou neuronovou funkcí

- budeme minimalizovat chybovou funkci SSE v prostoru vah:

$$E(\vec{w}) = \frac{1}{2}\sum_{p=1}^{N}(d_p - y_p)^2 = \frac{1}{2}\sum_{p=1}^{N}\left(d_p - f(\sum_{i=0}^{n} w_i.x_{pi})\right)^2$$



- protože se již nemusí jednat o kvadratickou funkci, gradientní metoda ani při sebelepším nastavení hyperparametrů nemusí najít její globální minimum

Gradientní metoda pro neurony se spojitou přenosovou funkcí

1. Nejprve pro chybovou funkci

$$E_p(\vec{w}) = \frac{1}{2}(d_p - y_p)^2 = \frac{1}{2}(d_p - f(\xi_p))^2$$
$$= \frac{1}{2}\left(d_p - f(\sum_{i=0}^{n} w_i.x_{pi})\right)^2,$$

spočteme její parciální derivace:

$$\frac{\partial E_p}{\partial w_i} = \frac{\partial E_p}{\partial y_p}\frac{\partial y_p}{\partial \xi_p}\frac{\partial \xi_p}{\partial w_i} = -(d_p - y_p)f'(\xi_p)x_{pi}$$

Gradientní metoda pro neurony se spojitou přenosovou funkcí

1. Nejprve pro chybovou funkci spočteme její parciální derivace:

$$\frac{\partial E_p}{\partial w_i} = -(d_p - y_p) f'(\xi_p) x_{pi}$$

2. Potom sestavíme adaptační pravidlo:

$$w_i(t+1) = w_i(t) - \alpha \frac{\partial E_p}{\partial w_i} = w_i(t) + \alpha f'(\xi_p)(d_p - y_p) x_{pi}$$

pro vektor vah:

$$\vec{w}(t+1) = \vec{w}(t) - \alpha \nabla E_p(\vec{w}) = \vec{w}(t) + \alpha (d_p - y_p) f'(\xi_p) \vec{x}_p^T$$

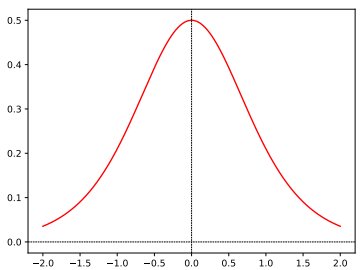
Gradientní metoda pro neurony se spojitou přenosovou funkcí

Výpočet derivace přenosové funkce

Sigmoidální ... logsig

$$f(\xi_p) = \frac{1}{1 + e^{-\lambda \xi_p}}$$

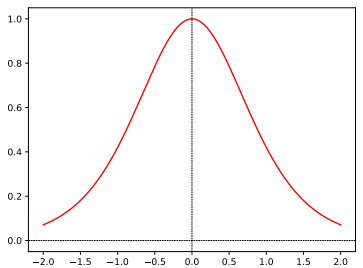
$$f'(\xi_p) = \lambda y_p (1 - y_p)$$



Hyperbolický tangens ... tanh

$$f(\xi) = \frac{1 - e^{-2\lambda \xi}}{1 + e^{-2\lambda \xi}}$$

$$f'(\xi_p) = \lambda^2 (1 + y_p)(1 - y_p)$$



Gradientní metoda pro neurony se spojitou přenosovou funkcí

Obecné schéma algoritmu (GD, gradient descent)

1. Inicializuj váhy malými náhodnými reálnými hodnotami ....
   $\vec{w}(0) = (w_0, w_1, ..., w_n)^T$
   Inicializuj parametr učení α_0.... $1 \gg \alpha_0 > 0$
2. Předlož další trénovací vzor $(\vec{x}_t, d_t)$ a spočti potenciál a skutečný výstup neuronu:
   $\xi_t = \vec{x}_t \vec{w}$
   $y_t = f(\xi_t)$
3. Adaptuj váhy:

$$\vec{w}(t+1) = \vec{w}(t) + \alpha_t f'(\xi_t)(d_t - y_t)\vec{x}_t^T$$

4. Případně aktualizuj parametr učení : $\alpha_t \to \alpha_{t+1}$
5. Pokud není konec, přejdi ke kroku 2.

Diskuse gradientní metody

Opět různé učící strategie

- **pro předkládání vzorů:**
  1. **iterativní předkládání vzorů (Online GD)**
     - rychlé učení (v počtu epoch), ale poměrně nestabilní (algoritmus snižuje chybu pro aktuální vzor $\rightarrow$ chyba se může zvýšit pro ostatní vzory)
  2. **dávkové předkládání vzorů (Batch GD)**
     - stabilnější učení, ale nachází ,,horší" řešení
     - efektivní pro malá data, větší prostorové nároky pro velká data
  3. **dávkové předkládání vzorů po mini-dávkách (SGD, Stochastic GD)**
     - kompromisní řešení, používá se u hlubokých sítí
- **pro nastavení, případně aktualizaci parametru učení**
  - **konstantní** - je třeba ho správně nastavit
  - **adaptivní** - např. klesající v čase - je třeba ho správně nastavit rychlost klesání

Diskuse gradientní metody

Opět různé učící strategie

- **pro inicializaci vah:**
  - váhy inicializujeme na malé náhodné hodnoty (ideálně se střední hodnotou 0)
  - příliš velké nebo vychýlené váhy povedou k tomu, že se perceptron bude učit obtížně
- **pro ukončení učení**
  1. předem daný počet epoch
  2. jakmile je průměrná chyba dostatečně malá ... $E < E_{min}$
  3. jakmile přestane klesat chyba na validační množině dat ... **early stopping**
  4. jakmile je přírůstek vah Δw moc malý ... $|\Delta w| < \delta_{min}$
- lze používat i **jiné chybové funkce** (např. cross-entropy pro klasifikaci) nebo regularizační chybové členy

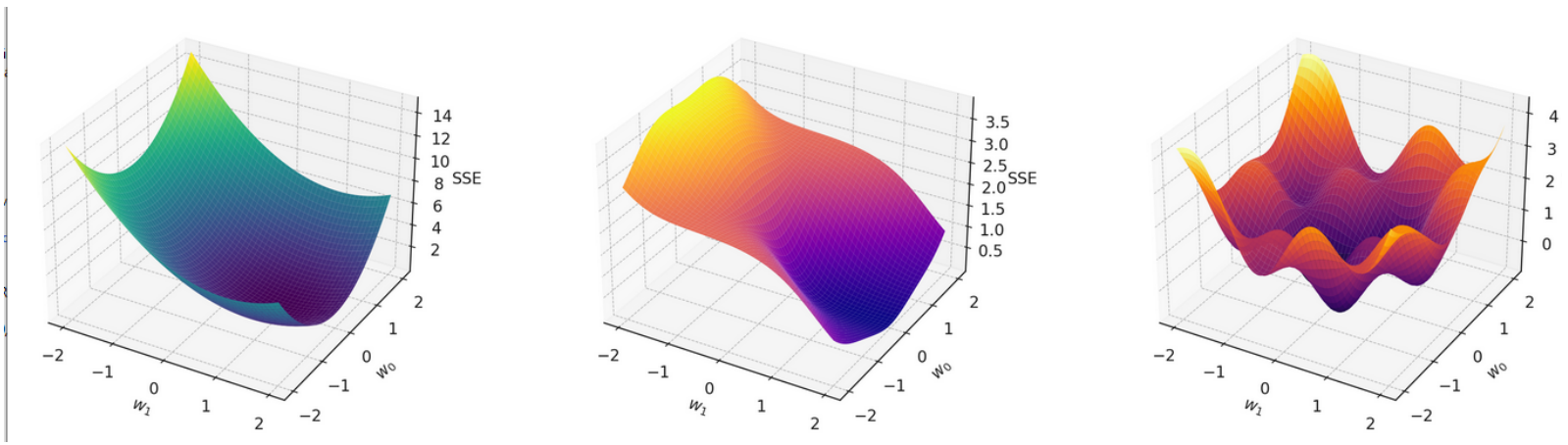
Diskuse gradientní metody

Důležitost předzpracování trénovacích dat:

- Aby se model gradientní metodou efektivně a rychle učil, je vhodné, aby nejen počáteční váhy, ale i vstupní vzory měly hodnoty v malém rozsahu, např. z intervalu $[-1, 1]$ nebo $[0, 1]$.
- Normalizace vstupních dat pomáhá zabránit problémům s odlišnými měřítky vstupních veličin a vede k rychlejší konvergenci.
- Metody normalizace:
  - min-max normalizace do intervalu $[-1, 1]$ - hodí se pro rovnoměrně rozložená data bez odlehlých vzorů
  - normalizace podle směrodatné odchylky - hodí se, obsahují-li data extrémní hodnoty a outliery

Diskuse gradientní metody

- oproti lineárnímu neuronu je situace složitější, protože chybová funkce nemusí být kvadratická
- gradientní metoda tak může snadněji skončit v lokálním minimu chybové funkce
- pro obecný neuron tak bude gradientní metoda ještě citlivější k volbě parametrů a k předzpracování dat (normalizace) než v případě lineárního neuronu



Diskuse gradientní metody

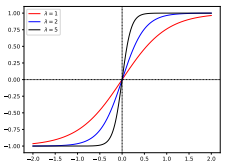
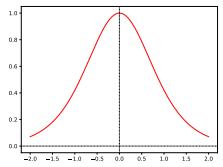
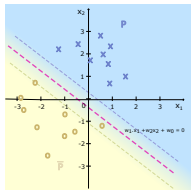
Riziko saturace neuronu v průběhu učení:

- neuron má ,,velké" váhy a už se ,,nechce" dál učit, i když by mohl ... přírůstek vah je příliš malý

Co ovlivňuje, jak velký bude přírůstek vah?

$$\triangle\vec{w}(t) + \alpha_t f'(\xi_t)(d_t - y_t)\vec{x}_t^T$$

- $(d_t - y_t)$ ... rozdíl mezi skutečným a požadovaným výstupem
- α ... parametr učení (learning rate)
- $\vec{x}_t^T$ ... vstupní vzor $\rightarrow$ důležitost normalizace
- $f'(\xi_t)$ ... pro sigmoidu / hyperbolický tangens klesá s rostoucí vzdáleností vzoru od dělící nadroviny

$\rightarrow$ riziko **saturace** neuronu

Diskuse gradientní metody

Jak se vyhnout saturaci neuronu?

- normalizace vstupních dat ,,kolem nuly" a inicializace vah ,,kolem nuly" ... oboje vede na očekávanou nulovou hodnotu potenciálu a rychlé učení z počátku
- vypořádat se s odlehlými vzory
- popřípadě pro sigmoidu / hyperbolický tangens použít jinou chybovou funkci než SSE, např. **cross-entropy**:

$$E = -\sum_p \left(d_p \log y_p + (1 - d_p) \log(1 - y_p)\right) \quad \text{(pro sigmoidu)}$$

$$E = -\sum_p \left(\frac{1 + d_p}{2} \log \frac{1 + y_p}{2} + \frac{1 - d_p}{2} \log \frac{1 - y_p}{2}\right) \quad \text{(pro tanh)}$$

interpretace: Jak moc se predikovaná pravděpodobnost třídy (např. 0.9 ku 0.1) liší od ideální hodnoty (1 ku 0)?

Diskuse gradientní metody

Chybová funkce cross-entropy

- velmi výhodná pro lineární klasifikaci v kombinaci s přenosovými funkcemi sigmoid nebo tanh.
- Např. pro sigmoidu:

$$E = -\sum_p (d_p \log y_p + (1 - d_p) \log(1 - y_p))$$

- gradient bude roven: $\frac{\partial E_p}{\partial y_p} = -\frac{d_p}{y_p} + \frac{(1-d_p)}{(1-y_p)}$
- po dosazení do vzorečku se některé členy navzájem výhodně vykrátí a platí:

$$\frac{\partial E_p}{\partial w_i} = \frac{\partial E_p}{\partial y_p} \frac{\partial y_p}{\partial \xi_p} \frac{\partial \xi_p}{\partial w_i} = \left(\frac{1 - d_p}{1 - y_p} - \frac{d_p}{y_p} \right) y_p(1-y_p)x_{pi} = (y_p - d_p)x_{pi}$$

$\rightarrow$ zbavili jsme se problematických členů a snížili tak riziko saturace

Příklady - 1. Dokončení příkladů na lineární regresi

linear_neuron.ipynb
Příklad 3 a Příklad 4 - úloha lineární regrese v jednorozměrném a dvourozměrném vstupním prostoru

- Ukážeme si využití testovací a validační množiny dat během učení a při zhodnocení toho, jak dobře se lineární neuron úlohu naučil
- Opět budeme expeimentovat s natavením hyperparametrů a budeme se je snažit vyladit pro danou úlohu
- Porovnáme mezi sebou iterativní a dávkový algoritmus
- Vyzkoušíme si techniku early stopping

Příklady - 1. Dokončení příkladů na lineární regresi

Dobrovolný domácí úkol z minule za bonusové body

- Modifikujte data 4 (definujte si vlastní, unikátní lineární funkci s unikátním šumem).
- Experimentujte pro tuto úlohu s nastavením hyperparametrů (parametr učení, počet epoch, učící strategie,...) a snažte se je co nejlépe vyladit.
- Stručně zhodnoťte výsledky experimentu (jaké nastavení hyperparametrů byste pro danou úlohu doporučili a proč)
- Srovnejte nejlepší dosažené výsledky s metodou LSQ
- Výsledný notebook (včetně textového zhodnocení) mi můžete poslat na email

Příklady - 1. Gradientní metoda pro perceptron s přenosovou funkcí hyperbolický tangens

tanh_perceptron.ipynb

- Jednoduché příklady: žlučník (příklad 1), hospoda (příklad 2)
  - Pozorování: gradientní hodnota si s úlohami poradí, i když učení trvá déle a je obtížnější nastavit správně hyperparametry
  - Všimněte si, že výstupy neuronu nejsou čisté 1 a -1
- Úloha lineární regrese (příklad 3)
  - Pozorovaní: přenosová funkce tanh je pro úlohu lineární regrese nevhodná
- Náhodně generované překrývající se shluky(příklad 4)
  - Pozorování: gradientní metoda si s úlohou opět dobře poradí, platí tytéž postřehy jako pro příklady 1 a 2
  - + Ukázka, že špatně inicializované váhy představují pro gradientní metodu velký problém
  - + Ukázka, že pokud máme k dispozici málo dat, hrozí přeučení + early stopping

Příklady - Gradientní metoda pro perceptron s přenosovou funkcí hyperbolický tangens

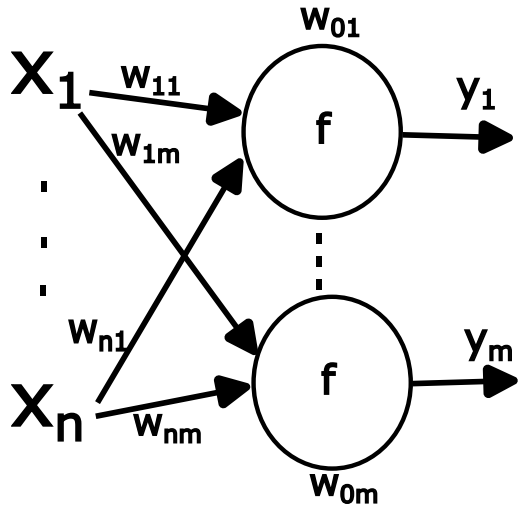
tanh_perceptron.ipynb

- Nenormalizovaná data (příklad 5)
  - Ukázka, že nenormalizovaná, např. vychýlená data představují pro gradientní metodu velký problém
- Odlehlá data (příklad 6)
  - Ukázka, že odlehlá data nemusí pro iterativní gradientní metodu společně s funkcí hyperbolický tangens znamenat problém
  - Naopak uvidíme, že odlehlá data mohou způsobit problémy při normalizaci dat podle minima a maxima

Dnešní hodina

1. Perceptron se spojitou přenosovou funkcí
   - Nejznámější přenosové funkce pro perceptron
   - Učení obecného perceptronu gradientní metodou
2. Neuronová síť tvořená jednou vrstvou neuronů
   - Feature engineering

Neuronová síť tvořená jednou vrstvou neuronů



1. Neurony s lineární přenosovou funkcí
   $\rightarrow$ **mnohorozměrná lineární regrese**
   - lineární neuronová síť
2. Neurony se sigmoidou nebo tanh přenosovou funkcí
   $\rightarrow$ **lineární klasifikace do více tříd (úloha rozpoznávání vzorů)**
   - jednovrstvý perceptron

Odbočka: Feature engineering

Cíl:

- Připravit vstupní data tak, aby se na nich vůbec mohl model neuronové sítě učit a aby se na nich **učil dobře**:

Co všechno spadá do feature engineeringu?

- **vektorizace:** perceptron a perceptronové sítě pracují s číselnými vektory $\rightarrow$ vstupní vzory je třeba převést na vektory čísel
- **normalizace:** numerické hodnoty vstupních příznaků je třeba normalizovat (ideálně na interval $[-1, 1]$)
- vypořádání se s chybějícími nebo chybnými hodnotami, šumem, odlehlými vzory,...
- výběr relevantních příznaků (feature selection), vytváření nových příznaků,...

Odbočka: Feature engineering

- **Cíl:** Připravit vstupní data tak, aby se na nich vůbec mohl model neuronové sítě učit a aby se na nich **učil dobře**:
  - **Lepší konvergence:** rychlejší a stabilnější proces učení.
  - **Zvýšení prediktivní síly modelu:** model lépe rozpozná skryté vzory v datech.
  - **Minimalizace šumu:** odstranění irelevantních nebo redundantních příznaků.

Otázka ohledně vektorizace:

- Jak s **kategoriálními** hodnotami příznaků?

Feature engineering

Motivační příklad: klasifikace do více tříd a kategoriální hodnoty

Velikost	Srst	Mluví?	Pohyb	Třída
Malá	Krátká	Ne	Běhá	Kočka
Velká	Dlouhá	Ne	Běhá	Pes
Malá	Žádná	Ano	Létá	Papoušek
Střední	Krátká	Ne	Běhá	Kočka
Malá	Žádná	Ne	Plave	Kapr
...			...	

Jak na to?

1. nejprve hodnoty příznaků převedeme z kategoriálních hodnot na numerické:
   - ordinal encoding, one-hot encoding, embeddings,...
2. pak případně hodnoty dále normalizujeme

Feature engineering - Převod kategoriálních proměnných na číselné hodnoty

Ordinal (label) encoding

- Pro kategorie s pořadím, např. *nízký, střední, vysoký* a pro právě dvě kategorie, např. *levý, pravý*
  - *Nízký* $= 0$, *Střední* $= 1$, *Vysoký* $= 2$
  - *Levý* $= 0$, *Pravý* $= 1$
- hodnoty můžeme dále normalizovat na interval $[-1, 1]$:
  - *Nízký* $=$ -1, *Střední* $= 0$, *Vysoký* $= 1$
  - *Levý* $=$ -1, *Pravý* $= 1$
- **nevhodné** pro nezávislé kategorie $\rightarrow$ takové kódování je pro model matoucí:
  - *Kočka* $= 0$, *Pes* $= 1$, *Papoušek* $= 2$
  - Je pes něco mezi kočkou a papouškem?

Feature engineering - Převod kategoriálních proměnných na číselné hodnoty

One-hot encoding

- pro nenávislé kategorie, např. barva auta, zvíře
  - *Červená* $\rightarrow [1, 0, 0]$, *Modrá* $\rightarrow [0, 1, 0]$, *Zelená* $\rightarrow [0, 0, 1]$
  - *Pes* $\rightarrow [1, 0, 0, 0]$, *Papoušek* $\rightarrow [0, 1, 0, 0]$, *Kočka* $\rightarrow [0, 0, 1, 0]$ *Kapr* $\rightarrow [0, 0, 0, 1]$
- popř. opět můžeme normalizovat:
  - *Červená* $\rightarrow [1, -1, -1]$, *Modrá* $\rightarrow [-1, 1, -1]$, *Zelená* $\rightarrow [-1, -1, 1]$

Embeddings

- pro složitější vztahy mezi kategoriálními hodnotami, např. reprezentace slov a vět v přirozeném jazyce

Feature engineering

Motivační příklad: klasifikace do více tříd a kategoriální

	Velikost	Srst	Mluví?	Třída
hodnoty	Malá	Krátká	Ne	Kočka
	Velká	Dlouhá	Ne	Pes
	Malá	Žádná	Ano	Papoušek
	Střední	Krátká	Ne	Kočka
	...			...

Velikost	Srst	Mluví?	Kočka	Pes	Papoušek	Kapr
-1	0	-1	1	-1	-1	-1
1	1	-1	-1	1	-1	-1
-1	-1	1	-1	-1	1	-1
0	0	-1	1	-1	-1	-1
...	...	...	...		...	

Ukázka: categorical_values.ipynb

Feature engineering

Příklad klasifikace do více tříd a kategoriální hodnoty

Velikost	Srst	Mluví?	Kočka	Pes	Papoušek	Kapr
-1	0	-1	1	-1	-1	-1
1	1	-1	-1	1	-1	-1
-1	-1	1	-1	-1	1	-1
0	0	-1	1	-1	-1	-1
...	...	...	...		...	

Jak úlohu naučíme?

1. pro každou kategorii naučíme jeden neuron (jeden po druhém) a slepíme výsledky ...

Velikost	Srst	Mluví?	Kočka
-1	0	-1	1
0	1	-1	-1
-1	-1	1	-1
0	0	-1	1
...			...

Feature engineering

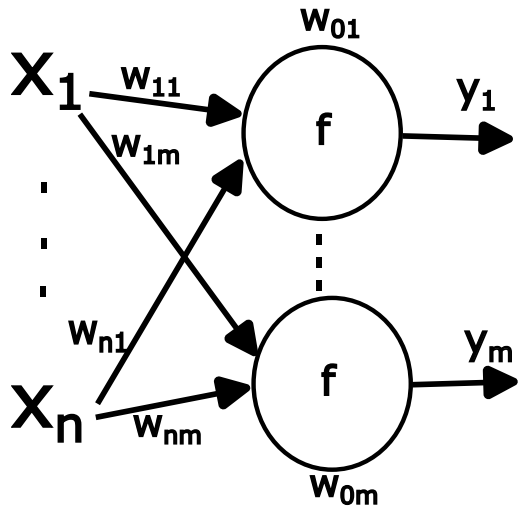
Příklad klasifikace do více tříd a kategoriální hodnoty

Velikost	Srst	Mluví?	Kočka	Pes	Papoušek	Kapr
-1	0	-1	1	-1	-1	-1
1	1	-1	-1	1	-1	-1
-1	-1	1	-1	-1	1	-1
0	0	-1	1	-1	-1	-1
...	...	...	...		...	

Jak úlohu naučíme?

2. **lépe:** sestrojíme neuronovou síť s jednou vrstvou neuronů a naučíme ji najednou

Neuronová síť tvořená jednou vrstvou neuronů

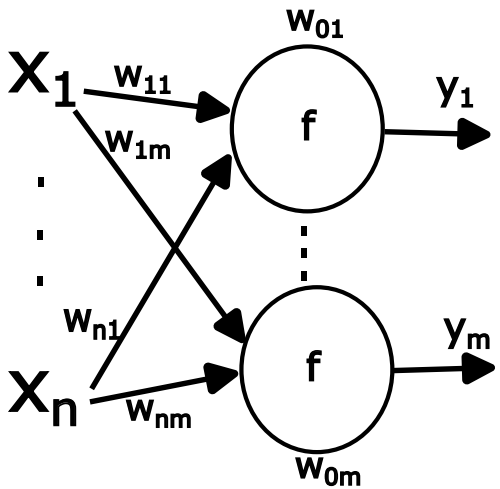


- Model je reprezentován maticí vah

$$W = \begin{pmatrix} w_{01} & w_{02} & \dots & w_{0m} \\ \dots & \dots & & \dots \\ w_{n1} & w_{n2} & \dots & w_{nm} \end{pmatrix}$$

  každému neuronu odpovídá jeden sloupec matice
- mají-li jednotlivé neurony přenosovou funkci $f : R \to R$, definujeme $f(\vec{\xi}) = (f(\xi_1), ..., f(\xi_N))^T$

Neuronová síť tvořená jednou vrstvou neuronů



$$W = \begin{pmatrix} w_{01} & w_{02} & \dots & w_{0m} \\ \dots & \dots & & \dots \\ w_{n1} & w_{n2} & \dots & w_{nm} \end{pmatrix}$$

- Výstup modelu:

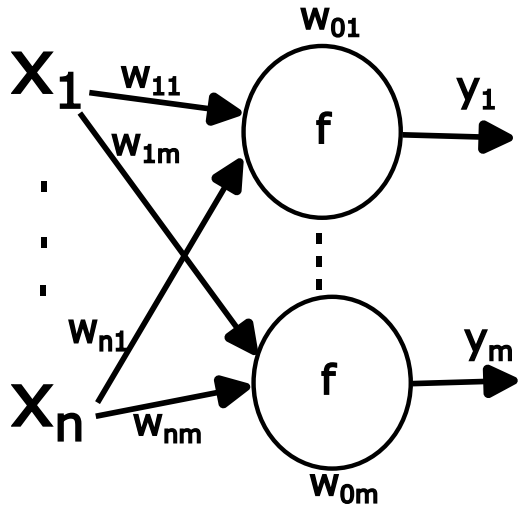
$$\vec{y} = f(\vec{\xi}) = f(\vec{x}W)$$

- Trénovací množina je ve tvaru $T = (X, D)$

$x_{10} = 1$	x_{11}	...	x_{1n}	d_{11}	...	d_{1m}
...	...		...	...		...
$x_{N0} = 1$	x_{N1}	...	x_{Nn}	d_{N1}	...	d_{Nm}

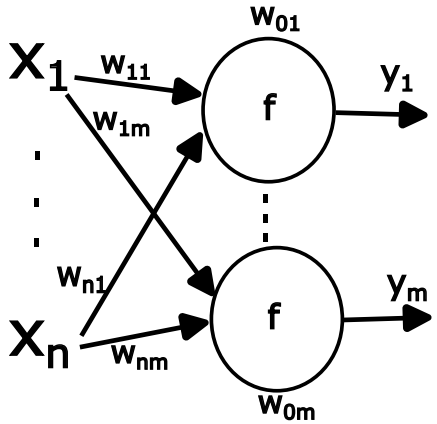
$$Y = f(\Xi) = f(XW)$$

Neuronová síť tvořená jednou vrstvou neuronů



1. Neurony s lineární přenosovou funkcí
   $\rightarrow$ **mnohorozměrná lineární regrese**
   - lineární neuronová síť
2. Neurony se sigmoidou nebo tanh přenosovou funkcí
   $\rightarrow$ **lineární klasifikace do více tříd (úloha rozpoznávání vzorů)**
   - jednovrstvý perceptron

Lineární neuronová síť



- tvořena jednou vrstvou lineárních neuronů (více vrstev by nepřineslo žádný benefit - na rozmyšlenou)
- mnohorozměrná lineární regrese
- výstup modelu:

$$Y = XW$$

Učení metodou LSQ

$$W = (X^T X)^{-1} X^T D$$

Učení gradientní metodou