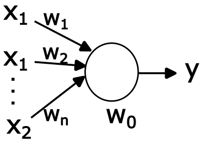
Co jsme probírali minule

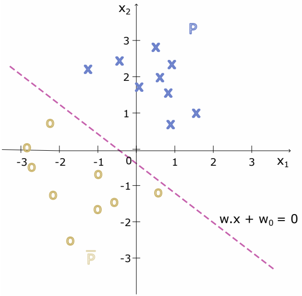
Perceptron

- vnitřní potenciál: $\xi = \sum_{i=0}^{n} w_i . x_i = \vec{w}\vec{x}^T$
- výstup: $y = f(\xi)$



- Perceptron se skokovou přenosovou funkcí a jeho algoritmy učení
  - Rosenblattovo učení a jeho různé varianty
  - Hebbovo učení

Perceptron se skokovou přenosovou funkcí jako lineární klasifikátor



Motivace

Perceptron se skokovou přenosovou funkcí

- **Aplikace:**
  - Lineární klasifikátor do dvou tříd
  - Realizace logických funkcí
- **Problém:** pokud data nejsou lineárně separabilní (např. XOR)

Co s tím?

1. kvadratické nebo kubické rozšíření příznakového prostoru
   např. $x_1, x_2, x_1^2, x_2^2, x_1x_2$
2. neuronová síť s větším počtem perceptronů a vrstev ...
   neumíme ji naučit :(

→ Co takhle použít místo skokové funkce nějakou **spojitou** přenosovou funkci?

→ Umožní nám to řešit i jiné typy úloh (např. regresní)

Dnešní hodina

1. **Přehled základních přenosových funkcí**
2. Lineární neuron a úloha lineární regrese
    - Učení lineárního neuronu metodou LSQ
    - Učení lineárního neuronu gradientní metodou
    - Učení neuronu se spojitou přenosovou funkcí gradientní metodou
3. Neuronová síť s jednou vrstvou neuronů

Přenosové (aktivační) funkce

- vnitřní potenciál neuronu: $\xi = \sum_{i=0}^{n} w_i.x_i = \vec{x}\vec{w}$
- výstup (aktivita) neuronu: $y = f(\xi)$
- $f : R \to R$ ... **přenosová funkce**

Neurony mohou mít různé přenosové funkce s různými vlastnostmi:

- Diskrétní
  - Skoková a její různé varianty
  - Kvantovaná
- Spojitá
  - Saturovaná lineární
  - Pozitivně lineární (ReLU)
  - ...
- Spojitě derivovatelná
  - Lineární
  - Sigmoidální
  - Radiální
  - ...

Podle dosahu:

- Globální
- Lokální
- Pologlobální

Přenosové (aktivační) funkce

Klasické požadavky na globální přenosovou funkci:

- $f(y)$ definovaná 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
- $(m, M) = (-1, 1)$ - bipolární model ... hyperbolický tangens

Již známe: skoková přenosová funkce

Pro bipolární model neuronu

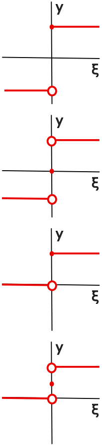
$$f(\xi) = \begin{cases} 1 & \text{pro } \xi \geq 0 \quad \text{... neuron je aktivní} \\ -1 & \text{pro } \xi < 0 \quad \text{... neuron je pasivní} \end{cases}$$

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

Pro binární model neuronu

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

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



Spojité přenosové funkce

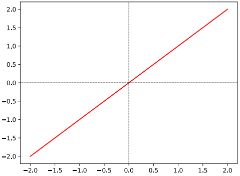
Lineární (identita)

- $f(\xi) = \xi$ ... *purelin*

Využití

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

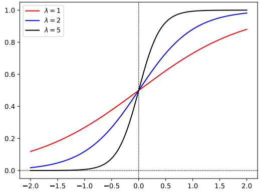
$\rightarrow$ jednovrstvá lineární neuronová síť
$\rightarrow$ ve výstupní vrstvě vícevrstvých/hlubokých sítí



Spojité přenosové funkce

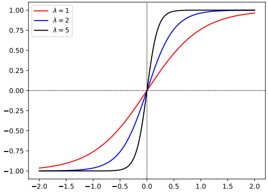
Sigmoidální

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



Hyperbolický tangens

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

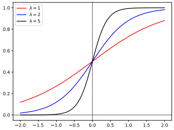


$\rightarrow$ „rozvolněná" skoková přenosová funkce

Spojité přenosové funkce

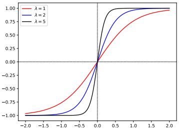
Sigmoidální

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



Hyperbolický tangens

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



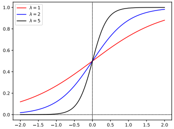
parametr λ ... strmost

- 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 tansig

Spojité přenosové funkce

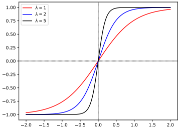
Sigmoidální

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



Hyperbolický tangens

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



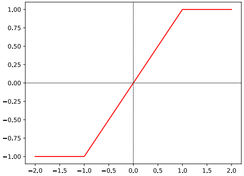
Využití

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

Spojité přenosové funkce

Saturovaná lineární

- $f(\xi) = \begin{cases} \xi, & \text{pro } |\xi| \leq 1 \\ 1, & \text{pro } \xi > 1 \\ -1, & \text{pro } \xi < -1 \end{cases}$
- bipolární i binární varianta ... *satlins*, *satlin*

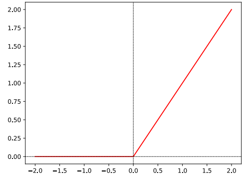


Pozitivně lineární (ReLU, rectified linear unit)

- $f(\xi) = max(0, \xi) = \begin{cases} x, & \text{pro } \xi > 0 \\ 0, & \text{pro } \xi \leq 0 \end{cases}$

  ... *poslin*

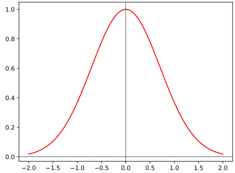
$\rightarrow$ hluboké neuronové sítě



Lokální přenosové funkce

Radiální funkce (gausovská)

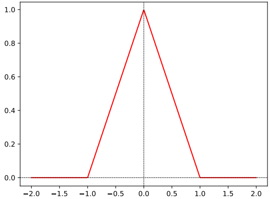
- $f(\xi) = e^{-\frac{\xi^2}{\alpha}}$ ... radbas
- $\xi = \frac{|\vec{x}-\vec{w}|}{\beta}$



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:

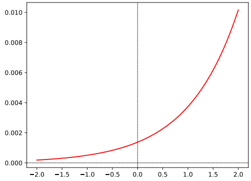
- $P(\xi) = \frac{1}{1+e^{-\frac{\xi}{T}}}$
- T ... pseudoteplota

$\rightarrow$ Boltzmanovy stroje, Deep Believe Networks

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

Dnešní hodina

1. Přehled základních přenosových funkcí
2. **Lineární neuron a úloha lineární regrese**
    - Učení lineárního neuronu metodou LSQ
    - Učení lineárního neuronu gradientní metodou
    - Učení neuronu se spojitou přenosovou funkcí gradientní metodou
3. Neuronová síť s jednou vrstvou neuronů

Lineární neuron

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

Cíl učení:

- máme trénovací množinu ve tvaru $T = (X, \vec{d})$

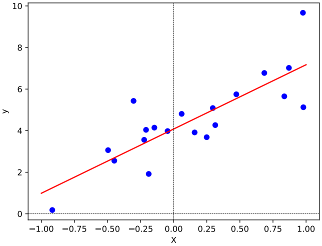
$x_{10} = 1$	x_{11}	...	x_{1n}	d_1
...	...		...	...
$x_{N0} = 1$	x_{N1}	...	x_{Nn}	d_N

$\rightarrow$ hledáme $\vec{w}$, aby platilo: $\vec{d} = \vec{y}$, tj. $\vec{d} = X\vec{w}$

- jedná se o úlohu **lineární regrese**

Lineární neuron - geometrická interpretace

- výstup neuronu: $y = w_1 x + w_0$
- (x_k, d_k) jsou body v rovině
- prokládáme body přímkou:

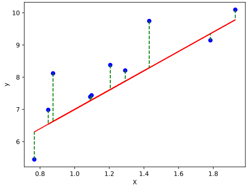


$\rightarrow$ obecně: prokládáme body nadrovinou
(předpokládáme, že mezi vstupními veličinami a výstupní je lineární závislost)

Lineární neuron

Jak učit lineární neuron (tj. model lineární regrese)?

- **metoda nejmenších čtverců**:
  - chceme aby se skutečný výstup neuronu y_p se co nejméně lišil od požadovaného d_p
  - minimalizujeme $\frac{1}{2}\sum_{p=1}^{N}(d_p - y_p)^2$ … součet čtverců (SSQ)



Jak na to?

1. metoda LSQ - založena na explicitním výpočtu
2. gradientní metoda (metoda největšího spádu)

Lineární neuron - učení metodou LSQ

- máme trénovací množinu ve tvaru $T = (X, \vec{d})$

$$
\begin{array}{|cccc|c|}
\hline
x_{10} = 1 & x_{11} & \dots & x_{1n} & d_1 \\
\dots & \dots & & \dots & \dots \\
x_{N0} = 1 & x_{N1} & \dots & x_{Nn} & d_N \\
\hline
\end{array}
$$

$\rightarrow$ hledáme $\vec{w}$, aby platilo: $\vec{d} = \vec{y}$, tj. $\vec{d} = X\vec{w}$

Řešíme tedy soustavu rovnic:

$$
\begin{array}{ccccccccc}
w_0 x_{10} & + & w_1 x_{11} & + & \dots & + & w_n x_{1n} & = & d_1 \\
\dots & \dots & & \dots & \dots & & \dots & & \dots \\
w_0 x_{N0} & + & w_1 x_{N1} & + & \dots & + & w_n x_{Nn} & = & d_N
\end{array}
$$

Lineární neuron - učení metodou LSQ

- podmínka na lin. nezávislost sloupců X
- podmínka na hodnost: $h(X|\vec{d}) = h(X)$
- pokud $h(X|\vec{d}) = h(X) = n + 1 \rightarrow$ soustava má právě jedno řešení
- obecně: $\frac{1}{2}\sum_{p=1}^{N}(d_p - y_p)^2 = min$

Odvození (Gauss):

$$X\vec{w} = \vec{d}$$
$$X^T(X\vec{w}) = X^T\vec{d}$$
$$(X^TX)\vec{w} = X^T\vec{d}$$

...

Lineární neuron - učení metodou LSQ

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

1. pokud existuje inverzní matice, tj. $|X^T X| \neq 0$:

$$\vec{w} = (X^T X)^{-1} X^T \vec{d}$$

2. pokud $|X^T X| = 0 \rightarrow$ soustava má více řešení $\rightarrow$ provedeme regularizaci:

$$\vec{w} = (X^T X + \lambda I)^{-1} X^T \vec{d}, \lambda > 0$$

$$\vec{w} = \lim_{\lambda \to 0+} (X^T X + \lambda I)^{-1} X^T \vec{d}$$

Lineární neuron - učení metodou LSQ

Příklad 1 ... $\vec{w} = (X^T X)^{-1} X^T \vec{d}$

x_0	x_1	x_2	d
+1	-1	-1	+1
+1	-1	+1	+1
+1	+1	-1	+1
+1	+1	+1	-1

$$X^T X = \begin{pmatrix} 1 & 1 & 1 & 1 \\ -1 & -1 & 1 & 1 \\ -1 & 1 & -1 & 1 \end{pmatrix} \begin{pmatrix} 1 & -1 & -1 \\ 1 & -1 & 1 \\ 1 & 1 & -1 \\ 1 & 1 & 1 \end{pmatrix} = \begin{pmatrix} 4 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 4 \end{pmatrix}$$

$$(X^T X)^{-1} = \begin{pmatrix} \frac{1}{4} & 0 & 0 \\ 0 & \frac{1}{4} & 0 \\ 0 & 0 & \frac{1}{4} \end{pmatrix}$$

Lineární neuron - učení metodou LSQ

Příklad 1 ... $\vec{w} = (X^T X)^{-1} X^T \vec{d}$

x_0	x_1	x_2	d
+1	-1	-1	+1
+1	-1	+1	+1
+1	+1	-1	+1
+ 1	+1	+1	-1

$$\vec{w} = \begin{pmatrix} \frac{1}{4} & 0 & 0 \\ 0 & \frac{1}{4} & 0 \\ 0 & 0 & \frac{1}{4} \end{pmatrix} \begin{pmatrix} 1 & 1 & 1 & 1 \\ -1 & -1 & 1 & 1 \\ -1 & 1 & -1 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ 1 \\ -1 \end{pmatrix}$$

$$= \begin{pmatrix} \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ -\frac{1}{4} & -\frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{4} & -\frac{1}{4} & \frac{1}{4} \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ 1 \\ -1 \end{pmatrix} = \begin{pmatrix} \frac{1}{2} \\ -\frac{1}{2} \\ -\frac{1}{2} \end{pmatrix}$$

Lineární neuron - učení metodou LSQ

Příklad 2 ... $\vec{w} = (X^T X)^{-1} X^T \vec{d}$

x_0	x_1	x_2	d
+1	+1	-1	+1
+1	+1	+1	-1

$$X^T X = \begin{pmatrix} 1 & 1 \\ 1 & 1 \\ -1 & 1 \end{pmatrix} \begin{pmatrix} 1 & 1 & -1 \\ 1 & 1 & 1 \end{pmatrix} = \begin{pmatrix} 2 & 2 & 0 \\ 2 & 2 & 0 \\ 0 & 0 & 2 \end{pmatrix}$$

$$h(X^T X) = 2 \rightarrow |X^T X| = 0 \rightarrow (X^T X)^{-1} \mathit{neexistuje}$$

Provedeme regularizaci:

$$\vec{w} = (X^T X + \lambda I)^{-1} X^T \vec{d}, \lambda > 0$$

$$\vec{w} = \lim_{\lambda \to 0+} (X^T X + \lambda I)^{-1} X^T \vec{d}$$

Lineární neuron - učení metodou LSQ

Příklad 2 ... $\vec{w} = (X^T X + \lambda I)^{-1} X^T \vec{d}, \lambda > 0$

x_0	x_1	x_2	d
+1	+1	-1	+1
+1	+1	+1	-1

$$X^T X + \lambda I = \begin{pmatrix} 2+\lambda & 2 & 0 \\ 2 & 2+\lambda & 0 \\ 0 & 0 & 2+\lambda \end{pmatrix}$$

Po větší troše počítání:

$$(X^T X + \lambda I)^{-1} = \begin{pmatrix} \frac{2+\lambda}{\lambda^2+4\lambda} & -\frac{2}{\lambda^2+4\lambda} & 0 \\ -\frac{2}{\lambda^2+4\lambda} & \frac{2+\lambda}{\lambda^2+4\lambda} & 0 \\ 0 & 0 & \frac{1}{2+\lambda} \end{pmatrix}$$

Lineární neuron - učení metodou LSQ

Příklad 2 ... $\vec{w} = (X^T X + \lambda I)^{-1} X^T \vec{d} = X^+ \vec{d}, \lambda > 0$

$$X^+ = (X^T X + \lambda I)^{-1} X^T = \begin{pmatrix} \frac{2+\lambda}{\lambda^2+4\lambda} & -\frac{2}{\lambda^2+4\lambda} & 0 \\ -\frac{2}{\lambda^2+4\lambda} & \frac{2+\lambda}{\lambda^2+4\lambda} & 0 \\ 0 & 0 & \frac{1}{2+\lambda} \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 1 & 1 \\ -1 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} \frac{\lambda}{\lambda^2+4\lambda} & \frac{\lambda}{\lambda^2+4\lambda} \\ \frac{\lambda}{\lambda^2+4\lambda} & \frac{\lambda}{\lambda^2+4\lambda} \\ -\frac{1}{2+\lambda} & \frac{1}{2+\lambda} \end{pmatrix} = \begin{pmatrix} \frac{1}{\lambda+4} & \frac{1}{\lambda+4} \\ \frac{1}{\lambda+4} & \frac{1}{\lambda+4} \\ -\frac{1}{2+\lambda} & \frac{1}{2+\lambda} \end{pmatrix}$$

Lineární neuron - učení metodou LSQ

Příklad 2 ... $\vec{w} = (X^T X + \lambda I)^{-1} X^T \vec{d} = X^+ \vec{d}, \lambda > 0$

- $\lambda = 1$:

$$\vec{w} = X^+ \vec{d} = \begin{pmatrix} \frac{1}{5} & \frac{1}{5} \\ \frac{1}{5} & \frac{1}{5} \\ -\frac{1}{3} & \frac{1}{3} \end{pmatrix} \begin{pmatrix} 1 \\ -1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ -\frac{2}{3} \end{pmatrix}$$

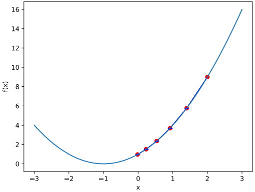
- $\lambda \to 0$:

$$\vec{w} = X^+ \vec{d} = \begin{pmatrix} \frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{4} \\ -\frac{1}{2} & \frac{1}{2} \end{pmatrix} \begin{pmatrix} 1 \\ -1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ -1 \end{pmatrix}$$

Odbočka: gradientní metoda (metoda největšího spádu)

Úloha:

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



$\to$ **řešení gradientní metodou:**

1. začneme v nějakém počátečním bodě $\vec{x}(0)$
2. spočteme gradient $\nabla f(\vec{x}) = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \dots, \frac{\partial f}{\partial x_n}\right)$
   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 parametr: $x_i(t+1) = x_i(t) - \alpha \frac{\partial f}{\partial x_i}$

Odbočka: gradientní metoda (metoda největšího spádu)

Problémy:

- pro malé α je učení pomalé
- pro velkéα kmitá (přeskakuje řešení)
- nemusí najít globální minimum (např. uvízne v lokálním)

Jak tedy nastavit parametr učení? $\rightarrow$ různé heuristiky:

- např. $\alpha_j > 0$ malé, $\sum_{i=0}^{\infty} \alpha_i = \infty$, $\sum_{i=0}^{\infty} \alpha_i^2 < \infty$
  $\alpha_j = \frac{\alpha_0}{1+j}$ (Robins-Moore, 1951)

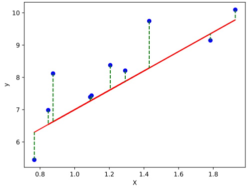
Učení lineárního neuronu gradientní metodou (metodou největšího spádu)

Připomenutí: lineární neuron (model lineární regrese)

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

Metoda nejmenších čtverců

- chceme aby se skutečný výstup neuronu y_p se co nejméně lišil od požadovaného d_p
- minimalizujeme $\frac{1}{2}\sum_{p=1}^{N}(d_p - y_p)^2$ ... součet čtverců



Učení lineárního neuronu gradientní metodou

Metoda nejmenších čtvreců

- minimalizujeme $\frac{1}{2}\sum_{p=1}^{N}(d_p - y_p)^2$ ... součet čtverců

Řešení gradientní metodou

1. budeme minimalizovat chybovou funkci SSE v prostoru vah:

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

Učení lineárního neuronu gradientní metodou

1. Nejprve pro chybovou funkci

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

spočteme její parciální derivace:

$$\frac{\partial E}{\partial w_i} = \left(d_p - \sum_{i=0}^{n} w_i.x_{pi}\right)(-x_{pi}) = -(d_p - y_p)x_{pi}$$

2. Potom sestavíme adaptační pravidlo:

$$w_i(t+1) = w_i(t) - \alpha\frac{\partial E}{\partial w_i} = w_i(t) + \alpha(d_p - y_p)x_{pi}$$

pro vektor vah:

$$\vec{w}(t+1) = \vec{w}(t) - \alpha\nabla E(\vec{w}) = \vec{w}(t) + \alpha(d_p - y_p)\vec{x}_p^T$$

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.... $1 > \alpha_0 > 0$
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:
   $$\vec{w}(t+1) = \vec{w}(t) + \alpha_t (d_t - y_t) \vec{x}_t^T$$
4. Případně aktualizuj parametr učení : $\alpha_t \to \alpha_{t+1}$
5. Pokud není konec, přejdi ke kroku 2.

Učení lineárního neuronu gradientní metodou

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

1. **iterativně po epochách**: během jedné epochy se každý vzor předloží právě jednou, vrámci každé epochy vzory náhodně uspořádáme
   - počet epoch .... kolikrát se předloží celá trénovací množina
2. **dávkově po epochách**:
   - celá trénovací množina se předloží najednou a váhy se adaptují také najednou

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

$$\vec{w}(t+1) = \vec{w}(t) + \alpha_t \sum_{p=1}^{N} (d_p - y_p)\vec{x}_p^T = \vec{w}(t) + \alpha_t X^T(\vec{d} - \vec{y})$$

3. **dávkově po minibatchích** SGD (stochastic gradient descend)
   - náhodně se vybere malá podmnožina vzorů (minibatch) a předloží se dávkově

Učení lineárního neuronu gradientní metodou

Jak a jak často aktualizovat parametr učení?

- typicky jednou za epochu e:

$$\alpha_e = \frac{\alpha_0}{e}$$

$$\alpha_e = \frac{\alpha_0}{\sqrt{e}}$$

Kdy ukončit 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
4. jakmile je přírůstek vah Δw moc malý ... $|\Delta w| < \delta_{min}$

Učení lineárního neuronu gradientní metodou

Co když jsou vstupní vzory „velké"?

- to by mohlo způsobovat velké problémy při učení (ovlivní to rychlost a stabilitu učení, zobecňování,..)

Řešení: normalizace hodnot vstupních příznaků

- min-max normalizace na interval $[-1, 1]$:

$$X_{ij}^{new} = 2 * \frac{X_{ij} - m_j}{M_j - m_j} - 1$$

$m_j = \min_k(X_{kj})$, $M_j = \max_k(X_{kj})$

- normalizace podle směrodatné odchylky:

$$X_{ij}^{new} = \frac{X_{ij} - E(X_{kj})}{S(X_{kj})}$$

  - $E(X_{kj}) = \frac{1}{N}\sum_{k=1}^{N} X_{kj}$ je průměr (střední hodnota) sloupce j matice X
  - $S(X_{kj}) = \frac{1}{N-1}\sum_{k=1}^{N}(X_{kj} - E(X_{kj}))^2$ je směrodatná odchylka sloupce j matice X

Zobecnění gradientní metody

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

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

1. budeme minimalizovat chybovou funkci SSE v prostoru vah:

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

Gradientní metoda pro neurony se spojitou přenosovou funkcí

1. Nejprve pro chybovou funkci

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

spočteme její parciální derivace:

$$\frac{\partial E}{\partial w_i} = \left(d_p - f(\sum_{i=0}^{n} w_i.x_{pi})\right) f'(\xi_p)(-x_{pi}) = -(d_p - y_p)f'(\xi_p)x_{pi}$$

2. Potom sestavíme adaptační pravidlo:

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

pro vektor vah:

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

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.... $1 > \alpha_0 > 0$
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.

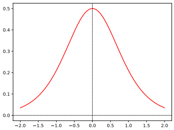
Gradientní metoda pro neurony se spojitou přenosovou funkcí

Výpočet derivace přenosové funkce

Sigmoidální ... logsig

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

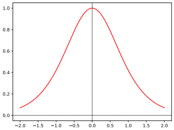
$$f'(\xi_p) = \lambda y_p (1 - y_p)$$



Hyperbolický tangens ... tanh

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

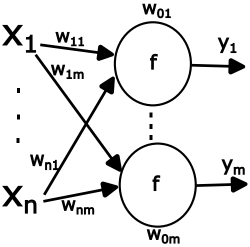
$$f'(\xi_p) = \lambda^2 (1 + y_p)(1 - y_p)$$



Dnešní hodina

1. Přehled základních přenosových funkcí
2. Lineární neuron a úloha lineární regrese
   - Učení lineárního neuronu metodou LSQ
   - Učení lineárního neuronu gradientní metodou
   - Učení neuronu se spojitou přenosovou funkcí gradientní metodou
3. **Neuronová síť s jednou vrstvou neuronů**

Neuronová síť s jednou vrstvou neuronů

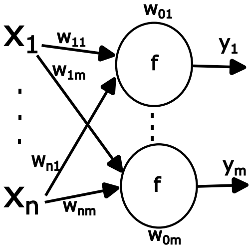


- Model je reprezentován maticí vah

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

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

Neuronová síť s jednou vrstvou neuronů



- Výstup modelu:

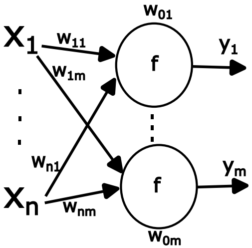
$$\vec{y} = f(\vec{\xi}) = f(\vec{x}W)$$

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

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

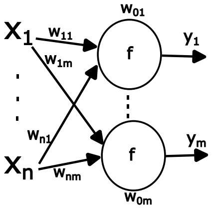
$$Y = f(\Xi) = f(XW)$$

Neuronová síť s jednou vrstvou neuronů



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

Lineární neuronová síť



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

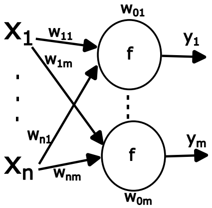
$$Y = XW$$

Učení metodou LSQ

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

Učení gradientní metodou

Jednovrstvá neuronová síť se spojitou přenosovou funkcí



Učení gradientní metodou

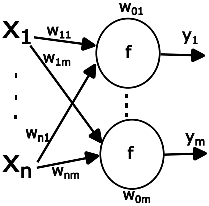
1. váhové vektory všech neuronů adaptuji simultánně
2. SGD (stochastická gradientní metoda) - v každém kroku adaptuji váhy jen jednoho náhodně zvoleného neuronu

Jednovrstvá neuronová síť se spojitou přenosovou funkcí

Obecné schéma algoritmu (GD, gradient descent)

1. Inicializuj váhy malými náhodnými reálnými hodnotami .... $W(0)$ tvaru $(n+1) \times m$
   Inicializuj parametr učení α_0.... $1 > \alpha_0 > 0$
2. Předlož další trénovací vzor $(\vec{x}_t, \vec{d}_t)$ a spočti potenciál a skutečný výstup modelu: $\vec{\xi}_t = (\vec{x}_t W)$
   $\vec{y}_t = f(\vec{\xi}_t)$
3. Adaptuj váhy:
   $$W(t+1) = W(t) + \alpha_t \vec{x}_t^T [f'(\vec{\xi}_t) \circ (\vec{d}_t - \vec{y}_t)]$$
4. Případně aktualizuj parametr učení : $\alpha_t \rightarrow \alpha_{t+1}$
5. Pokud není konec, přejdi ke kroku 2.

Lineární klasifikace do více tříd



- Bipolární neurony se skokovou nebo tansig přenosovou funkcí

Příklad klasifikační úlohy

x_1	x_2	x_n	**Třída**
1.5	-2.6	3.7	Třída 1
2.1	3.2	-0.5	Třída 2
-1.0	1.8	2.9	Třída 1
-2.2	0.5	2.0	Třída 3
...	...	...	...

Lineární klasifikace do více tříd

Příklad klasifikační úlohy

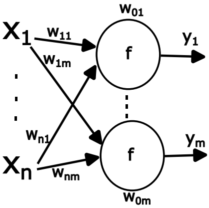
x_1	x_2	x_3	**Třída**
1.5	-2.6	3.7	Třída 1
2.1	3.2	-0.5	Třída 2
-1.0	1.8	2.9	Třída 1
-2.2	0.5	2.0	Třída 3
...	...	...	...

- pokud by vstupní příznaky nebyly normalizované, měly by se normalizovat

$\rightarrow$ **Trénovací množina pro bipolární model:**

x_0	x_1	x_2	x_3	d_1	d_2	d_3
1	1.5	-2.6	3.7	1	-1	-1
1	2.1	3.2	-0.5	-1	1	-1
1	-1.0	1.8	2.9	1	-1	-1
1	-2.2	0.5	2.0	-1	-1	-1

Lineární klasifikace do více tříd



- pro každou třídu naučíme jeden perceptron (simultánně nebo jeden po druhém)

Jak zjistím vítěznou třídu?

- argmax

$$k_{max} = argmax_k y_k$$

- softmax - pouze pro spojitou přenosovou funkci $f : R^n -> R^n$,

$$f(y_k) = \frac{e^{y_k}}{\sum_{j=1}^{m} e^{y_j}}$$