

# Backpropagation

Ing. Viktor Kocur  
viktor.kocur@fmph.uniba.sk

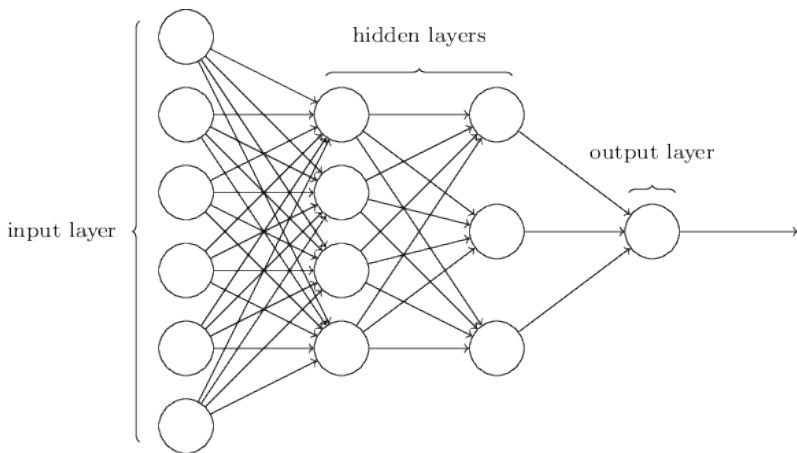
DAI FMFI UK

12.12.2018

# Obsah

- 1 Dopredný výpočet
  - Formalizácia
  - Aktivačné funkcie
  - Výstup
- 2 Trénovanie
  - Cenová funkcia
  - SGD
  - Backpropagation
  - Validačná množina
- 3 Regularizácia
  - L2 Regularizácia
  - Iné metódy

# Fully-connected NN



# Dopredný výpočet

Obecne sa aktivácia (hodnota) nevstupného vrcholu  $p$  počíta pomocou vzorca:

$$a_p = f \left( \sum_{q \in p_{in}} w_{p,q} a_q + b_p \right) = f(z_p) \quad (1)$$

- $a_p$  je aktivácia daného vrcholu
- $w_{p,q}$  je váha vrcholu  $q$  pre vrchol  $p$
- $b_p$  je prah vrcholu  $p$
- $f$  je aktivačná funkcia
- $z_p$  je zjednodušený zápis vstupu aktivačnej funkcie

# Dopredný výpočet

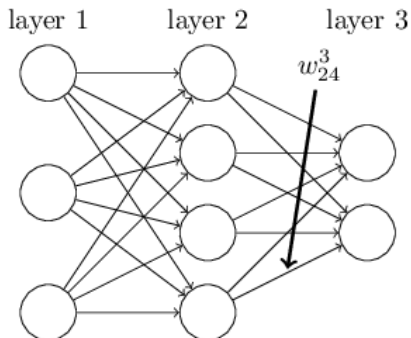
Pre fully-connected siete môžeme tento vzťah formalizovať krajšie:

$$a_j^l = f \left( \sum_k w_{jk}^l a_k^{l-1} + b_j^l \right) = f \left( z_j^l \right) \quad (2)$$

$$a^l = f \left( w^l a^{l-1} + b^l \right) = f \left( z^l \right) \quad (3)$$

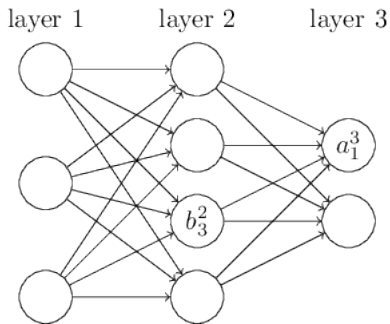
- $a^l$  je vektor aktivácií v  $l$ -tej vrstve
- $w^l$  je váhová matica  $l$ -tej vrstvy ( $\text{size}(l) \times \text{size}(l-1)$ )
- $b^l$  je vektor prahov v  $l$ -tej vrstve
- $f$  je aktivačná funkcia

## Váhy



$$a_j^l = f \left( \sum_k w_{jk}^l a_k^{l-1} + b_j^l \right)$$

## FCN - aktivácia a bias



$$a_j^l = f \left( \sum_k w_{jk}^l a_k^{l-1} + b_j^l \right)$$

# Aktivačné funkcie

- Sigmoid:  $f(z) = \frac{1}{1+e^{-z}}$
- Tanh:  $f(z) = \tanh(z)$
- ReLU:  $f(z) = \max(x, 0)$
- SoftPlus:  $f(z) = \ln(1 + e^z)$
- LeakyReLU:  $f(z) = \max(x, ax), a \leq 1$



# Výsup

Ako interpretovať aktiváciu poslednej vrstvy ( $L$ )? Pri klasifikačnej výbere kategórie  $k$  z množiny  $K$  možných napr:

- Najlepší odhad:  $k = \operatorname{argmax}_{i \in K} (a_i^L(x))$
- Softmax:  $P(k|x) = \frac{e^{z_k^L(x)}}{\sum_{i \in K} e^{z_i^L(x)}}$

# Cenová funkcia

Trénovanie siete je vlastne optimalizačná úloha. Chceme nájsť parametre  $P$ , tak aby sme na trénovacích dátach dosiahli čo najmenšiu cenu (chybu)  $C$ . Na to však musíme definovať cenu na vstupoch  $\{x_1, x_2, \dots, x_n\}$  pri požadovaných výstupoch  $\{y_1, y_2, \dots, y_n\}$ . Možností je opäť viac.

- MSE:  $C = \frac{1}{n} \sum_x \|a^L(x) - y\|_2^2$
- CE:  $C = -\frac{1}{n} \sum_{x,i} y_i \ln(a_i^L(x)) + (1 - y_i) \ln(1 - a_i^L(x))$
- Softmax:  $C = -\frac{1}{n} \sum_x \ln(z_y^L(x))$

# Ako dospejeme k parametrom s optimálnou cenou?

Použijeme gradientný zostup pre všetky parametre!

$$p := p - \eta \frac{\partial C}{\partial p}, \quad (4)$$

kde  $p$  je ľubovoľný parameter nášho modelu,  $C$  je cenová funkcia a  $\eta$  je hyperparameter rýchlosti zostupu.

# Budeme rátať gradient na všetkých trenovacích dátach?

Nie!

Vždy použijeme cenu len pre podmnožinu trénovacej množiny, tzv. minibatch.

# Budeme rátať gradient na všetkých trénovacích dátach?

Nie!

Vždy použijeme cenu len pre podmnožinu trénovacej množiny, tzv. minibatch.

## Stochastic gradient descent

Po jednom kroku zostupu načítame nový minibatch z trénovacích dát. Postupne takto pokryjeme celú trénovacu množinu. To sa volá jedna epocha. Pred začatím ďalšej epochy trénovacu množinu premiešame. Premiešanie trénovacej množiny je náhodné, preto názov stochastic gradient descent.

# Ako zistíme parciálne derivácie pre všetky parametre?

Pre FCN je možné určiť parciálne derivácie podľa parametrov poslednej vrstvy postupne. Určíme si pomocný výraz  $\delta_j^l$ .

$$\delta_j^l = \frac{\partial C}{\partial z_j^l} \quad (5)$$

Pre MSE cenovú funkciu platí:

$$\delta^L = (a^L - y) \odot \sigma' (z^L) \quad (6)$$

Pre CE cenovú funkciu:

$$\delta^L = (a^L - y) \quad (7)$$

# Propagujeme chybu smerom späť!

Chyba vrstvy  $l$  závisí od vrstvy  $l + 1$ !

$$\delta_j^l = \left( (w^{l+1})^T \delta^{l+1} \right) \odot \sigma' (z^l) \quad (8)$$

# Parciálne derivácie

Pre parametre  $w$  a  $b$  platí:

$$\frac{\partial C}{\partial b_j^l} = \delta_j^l \quad (9)$$

$$\frac{\partial C}{\partial w_{j,k}^l} = a_k^{l-1} \delta_j^l \quad (10)$$



# Backprop algoritmus

Stačí to dať dohromady a máme algoritmus jedného kroku SGD.

- Realizujeme dopredný výpočet - pamätáme si  $z^l$  a  $a^l$ .
- Vypočítame  $\delta^L$  podľa (6), alebo (7).
- Spätne propagujeme chybu  $\delta^l$  pomocou (8).
- Vypočítame parciálne derivácie podľa (9) a (10).
- Opakujeme pre celý minibatch, derivácie spriemerujeme a updatneme parametre podľa gradientného zostupu.

# Backprop algoritmus

Stačí to dať dohromady a máme algoritmus jedného kroku SGD.

- Realizujeme dopredný výpočet - pamätáme si  $z^l$  a  $a^l$ .
- Vypočítame  $\delta^L$  podľa (6), alebo (7).
- Spätne propagujeme chybu  $\delta^l$  pomocou (8).
- Vypočítame parciálne derivácie podľa (9) a (10).
- Opakujeme pre celý minibatch, derivácie spriemerujeme a updatneme parametre podľa gradientného zostupu.

## Pozor!

Pred trenovaním je nutné inicializovať premenné! Váhy s nejakým rozdelením, prahy stačia na jednu hodnotu.

# Validačná množina

## Stačí nám trenovacia množina?

Náš model sa snád' úspešne učí na trénovacích dátach. Môžeme pozorovať že cena klesá a presnosť na trénovacej množine stúpa. Máme zaručené, že náš model bude dobre generalizovať svoje znalosti?

# Validačná množina

## Stačí nám trenovacia množina?

Náš model sa snád' úspešne učí na trénovacích dátach. Môžeme pozorovať že cena klesá a presnosť na trénovacej množine stúpa. Máme zaručené, že náš model bude dobre generalizovať svoje znalosti?

## Validačná množina

Časť trénovacích dát si odložíme. Na nich budeme pozorovať ako sa model zlepšuje vždy po niekoľkých krokoch backprop algoritmu. Ak cena na trénovacej množine klesá, ale presnosť na validačnej množine nestúpa došlo k overfittingu, tzv. high variance.

# Ako určiť hyperparametre

## Správne hyperparametre

Ak sa náš model učí príliš pomaly, alebo nedostatočne, prípadne došlo k overfittingu, tak musíme upraviť hyperparametre. Momentálne to sú  $\eta$ , veľkosť minibatch a architektúra siete.

# Ako určiť hyperparametre

## Správne hyperparametre

Ak sa náš model učí príliš pomaly, alebo nedostatočne, prípadne došlo k overfittingu, tak musíme upraviť hyperparametre. Momentálne to sú  $\eta$ , veľkosť minibatch a architektúra siete.

## Train/Val

Na validačnej množine (zväčša ručne) určujeme hyperparametre a na tréovacej množine určujeme parametre modelu.

# Ako určiť hyperparametre

## Správne hyperparametre

Ak sa náš model učí príliš pomaly, alebo nedostatočne, prípadne došlo k overfittingu, tak musíme upraviť hyperparametre. Momentálne to sú  $\eta$ , veľkosť minibatch a architektúra siete.

## Train/Val

Na validačnej množine (zväčša ručne) určujeme hyperparametre a na tréovacej množine určujeme parametre modelu.

## A čo testovacia množina?

Testovacej množiny sa chytáme až úplne na konci!

# Čo ak náš model negeneralizuje dostatočne?

## Viac dát

Pri nedostatočnej schopnosti modelu generalizovať svoje vedomosti je možné zlepšiť situáciu zväčšením trénovacej množiny.



# Čo ak náš model negeneralizuje dostatočne?

## Viac dát

Pri nedostatočnej schopnosti modelu generalizovať svoje vedomosti je možné zlepšiť situáciu zväčšením trénovacej množiny.

## Máme iné možnosti?

Môžeme si vytvoriť syntetické dáta, alebo môžeme obmedziť kapacitu modelu pomocou regularizácie.

# Cenová funkcia s regularizáciou

Jeden spôsob ako zabrániť overfittingu je upraviť cenovú funkciu, tak aby sme preferovali modely s menšiou variance. Vytvoríme tak novú cenovú funkciu.

$$C = C_0 + \lambda R(P), \quad (11)$$

v ktorej vystupuje naša pôvodná cenová funkcia  $C_0$ , tzv. regularizačný člen  $R$ , ktorý závisí len od hodnôt parametrov a hyperparameter  $\lambda$ , ktorý určuje silu regularizačného člena.

# L2 Regularizácia

Najčastejšie sa používa L2 regularizácia. Pre ktorú platí

$$R_{L2} = \frac{1}{2n} \sum_{w,j,k} w_{j,k}^2 \quad (12)$$

Parciálna derivácia ceny podľa  $w$  sa tak zmení na:

$$\frac{\partial C}{\partial w_{j,k}^l} = \frac{\partial C_0}{\partial w_{j,k}^l} + \frac{\partial R}{\partial w_{j,k}^l} = a_k^{l-1} \delta_j^l + \frac{\lambda}{n} w_{j,k}^l \quad (13)$$

# L2 Regularizácia

Ak si teda rozpíšeme update pravidlá dostaneme:

$$w^l := w^l - \eta \frac{\partial C_0}{\partial w^l} - \frac{\eta \lambda}{n} w^l \quad (14)$$

$$w^l := \left(1 - \frac{\eta \lambda}{n}\right) w^l - \eta \frac{\partial C_0}{\partial w^l} \quad (15)$$

A pre prahy jednoducho:

$$\frac{\partial C}{\partial b^l} = b^l - \eta \frac{\partial C_0}{\partial b^l} \quad (16)$$

# L1 a Dropout

## L1

L1 regularizácia je podobná L2. Má iný tvar:  $R = \frac{1}{n} \sum_{w,j,k} |w_{j,k}|$

## Dropout

Pri dropoute sa niektoré vrcholy siete počas učenia deaktivujú s pravdepodobnosťou  $p$ . To zabráni tzv. koadaptácií príznakov. Je nutné si dať pozor a pri normálnom behu siete prenásobiť príslušné váhy parametrom  $p$ .