Pokročilé metódy analýzy dát 3

úvod do hlbokého učenia

Peter Bednár

Regularizácia učenia

Regularizácia učenia

- Základné metódy regularizácie:
  - Ohraničujú zložitosť modelu – penalizačné metódy
  - Znižujú varianciu učenia – dropout

- Ohraničenie zložitosti modelu
  - Zmenšíme počet neurónov na skrytej vrstve
  - Odstránime niektoré vstupné atribúty (*feature selection*)
  - Zmenu môžeme vyhodnotiť na nezávislej validačnej množine, alebo krížovou validáciou

Výber atribútov a penalizačné metódy (1)

- Atribút môže byť užitočný pre predikciu iba v kombinácii s ďalším atribútom(mi)
- Testovanie všetkých podmnožín? – príliš veľa kombinácií a učenia
  - Dopredný výber (*forward selection*) – postupne pridávame jeden atribút, ktorý najviac zlepší presnosť modelu
  - Spätné odstraňovanie (*backward elimination*) – postupne odoberáme jeden atribút, tak aby sa nezhoršila presnosť modelu
  - Môžeme striedať pridávanie/odoberanie

Výber atribútov a penalizačné metódy (2)

- Diskrétne ohraničenie modelu – atribút sa odstráni, alebo zachová
- Môžeme mať "spojitý" výber atribútov?
- Pre lineárny model:

$$f(\boldsymbol{x}) = w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_M x_M$$

  - Ak sa absolútna hodnota parametra blíži $|w_j| \to 0$, atribút ma menší vplyv na výslednú predikciu
  - Odstránenie atribútu X_j : $w_j = 0$

Výber atribútov a penalizačné metódy (3)

- Na jednej strane potrebujeme minimalizovať chybu na trénovacích dátach $J(\boldsymbol{w})$
- Na druhej strane minimalizovať zložitosť modelu
  - Penalizácia pre zložitejší model $\sum_{j=1}^{M}|w_j|$
- Namiesto $J(\boldsymbol{w})$ budeme pri učení minimalizovať penalizovanú chybu:

$$\tilde{J}(\boldsymbol{w}) = J(\boldsymbol{w}) + \lambda \sum_{j=1}^{M} |w_j|$$

- $\lambda \geq 0$ je hyper-parameter, ktorý určuje mieru regularizácie
  - Ak zväčšíme λ, váhy budú viac "stláčané" smerom k 0 – viac ohraničený model

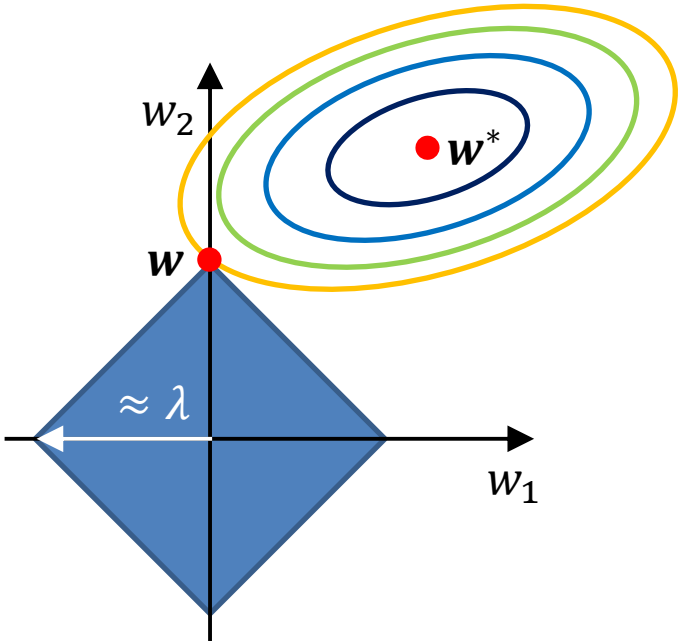
Výber atribútov a penalizačné metódy (4)

- Penalizačná funkcia $\sum_{j=1}^{M}|w_j|$ je špeciálny prípad normy vektora váh $\boldsymbol{w}$ - ℓ_1 norma
- Môžeme použiť aj iný typ normy, napr. Euklidovskú normu ℓ_2:

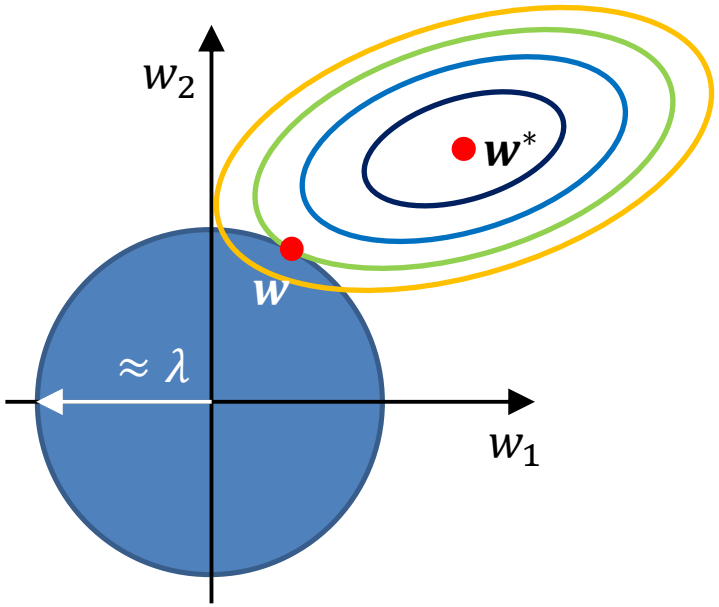
$$\tilde{J}(\boldsymbol{w}) = J(\boldsymbol{w}) + \lambda \sum_{j=1}^{M} w_j^2$$

- ℓ_1 norma (LASSO regresia) – riedke riešenie (niektoré váhy sú nastavené presne na 0)
- ℓ_2 norma (RIDGE regresia, *weight decay*) – tie váhy, ktoré majú na hodnotu chybovej funkcie menší vplyv, sú "stláčané" viac

Výber atribútov a penalizačné metódy (5)



ℓ_1 norma - LASSO regresia



ℓ_2 norma - (RIDGE regresia)

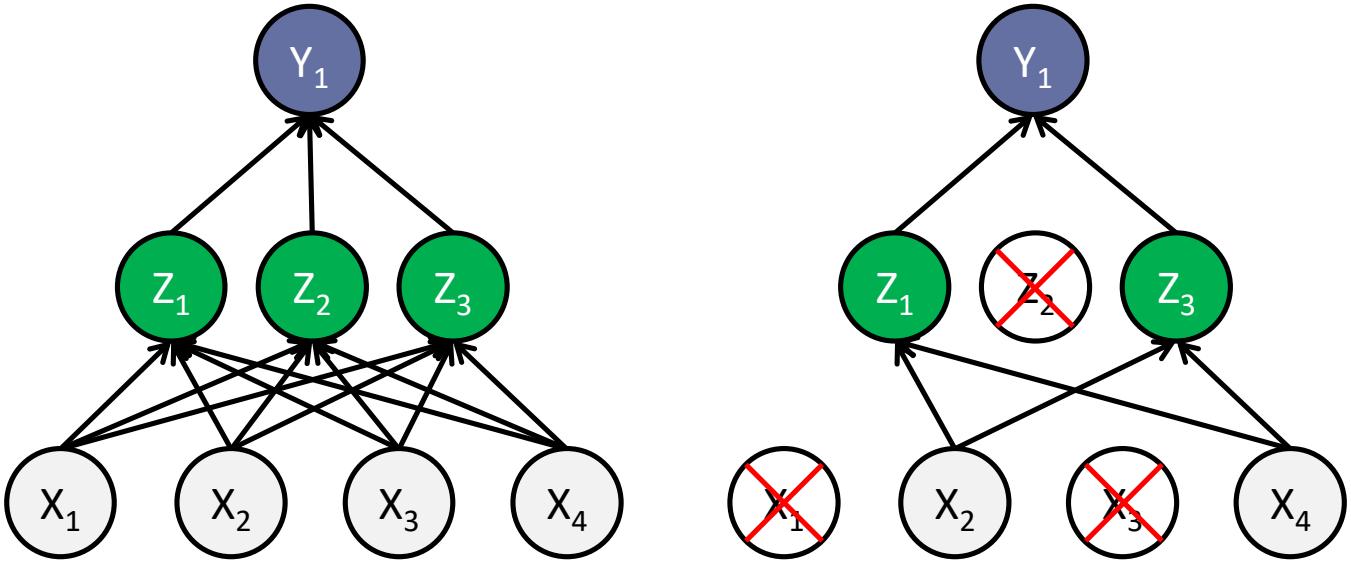
Penalizačná funkcia zodpovedá ohraničeniu prípustných riešení pri učení. Veľkosť oblasti je daná parametrom λ.

Variancia učenia a dropout (1)

- Ako môžeme znížiť varianciu učenia pre daný algoritmus?
- Bagging
  1. Z pôvodných trénovacích dát si náhodným výberom s nahradením vyberieme S rovnako veľkých trénovacích množín
  2. Na vytvorených množinách naučíme S prediktívnych funkcií $f^{(1)}, f^{(2)}, \ldots, f^{(S)}$
  3. Výslednú predikciu vypočítame spriemernením predikcií $f^{(1)}, f^{(2)}, \ldots, f^{(S)}$
- Nevýhody: je potrebné naučiť S modelov

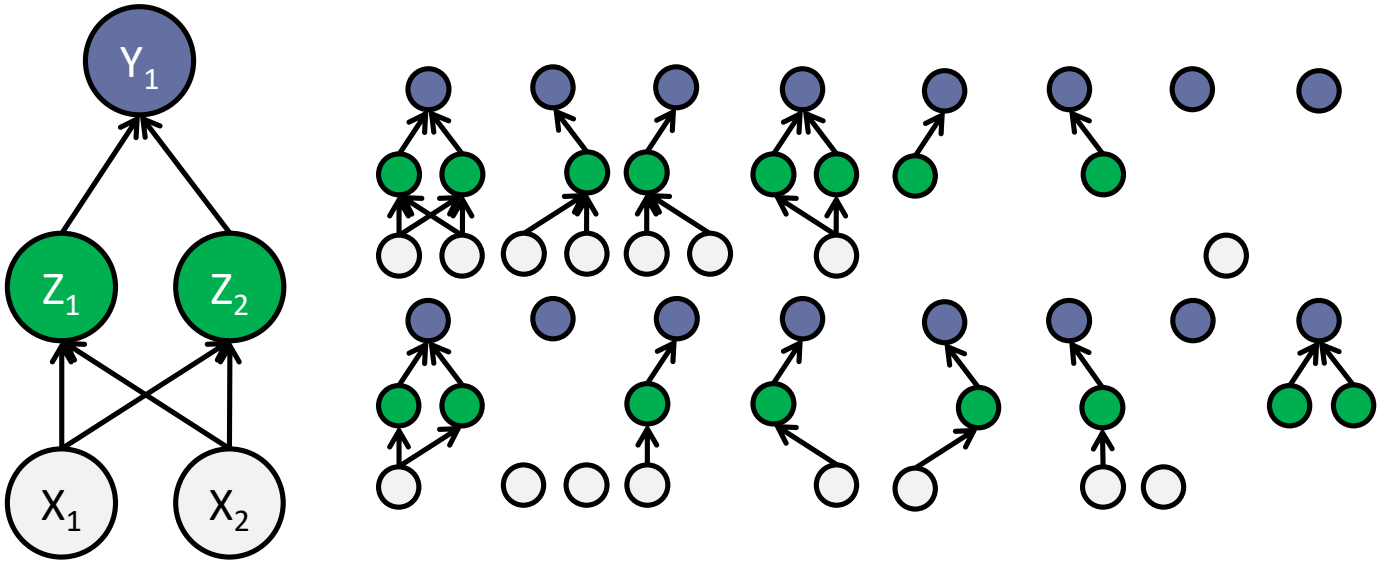
Variancia učenia a dropout (2)

- Dropout – jednoduchá schéma pre regularizáciu učenia
- Pri učení v každom kroku náhodne vynecháme niektoré neuróny a neaktualizujeme ich váhy – pri učení stačí nastaviť výstup vynechaného neurónu na 0



Variancia učenia a dropout (3)

- Aproximovane učíme model zložený zo všetkých pod-sietí, ktoré je možné vytvoriť vynechaním neurónov z danej architektúry (na rozdiel od bagging-u, modely nie sú učené nezávisle, ale počas učenia zdieľajú rovnaké váhy).



Variancia učenia a dropout (4)

- Neuróny zachováme s pravdepodobnosťou $0 < p \leq 1$ (resp. vynecháme s pravdepodobnosťou $1 - p$)
- Hyper-parameter p určuje mieru regularizácie (1 bez regularizácie), pre každú vrstvu môžeme zvoliť inú hodnotu
- Keďže počas testovania/predikcie bude neurón vždy prítomný, je potrebné upraviť očakávaný vstup na nasledujúcej vrstve:
  - Výstupné váhy z neurónu vynásobíme po skončení učenia pravdepodobnosťou p

Hlboké neurónové siete

Hlboké siete (1)

- Základná úloha skrytej vrstvy je nelineárne transformovať vstupné dáta a vyextrahovať latentné príznaky. Tieto príznaky sa potom skombinujú lineárnym modelom do výslednej predikcie
- Zložité závislosti = viac skrytých vrstiev: snažíme sa naučiť hierarchiu zložitejších príznakov postupne odvodených z predchádzajúcich vrstiev

Hlboké siete (2)

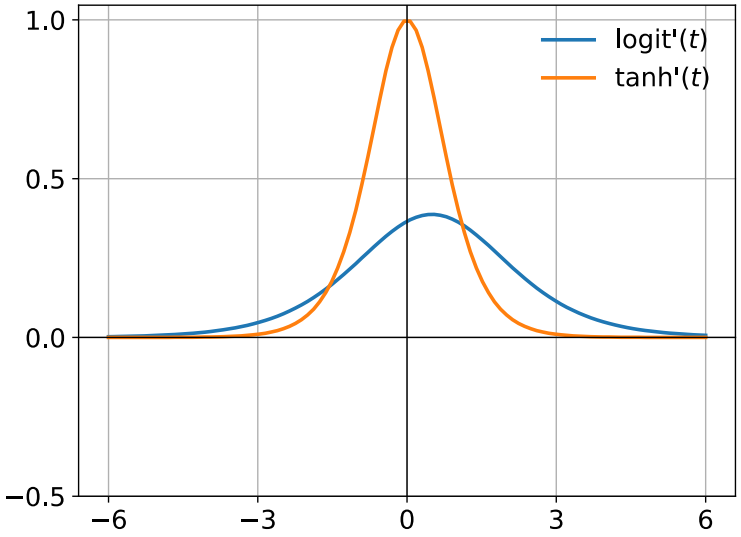
- Exponenciálne nám rastie počet parametrov (váh).
- Problémy s veľkou kapacitou modelu
  - Potrebujeme dostatočný počet trénovacích príkladov aby sme dobre aproximovali zložité závislosti a zároveň znížili varianciu učenia (a teda zabránili preučeniu)
- Numerické problémy pri učení gradientovou metódou
  - "Vymiznutie gradientu", "explodovanie" gradientu

Ako ohraničiť model?

- Regularizácia učenia – nemôžeme však model preregulovať, pretože potrebujeme mať dostatočnú kapacitu na aproximovanie zložitých závislostí
- Ak vieme, že sú vstupné dáta štruktúrovane usporiadané, napr. priestorovo (obrázok = 2D mriežka), alebo sekvenčne (text, audio):
  - Ten istý príznak vyextrahovaný pre časť vstupu môže byť užitočný aj pre inú časť – tie isté váhy môžeme zdieľať pre rôzne časti vstupu
  - Pre usporiadané dáta vieme teda navrhnúť efektívnejšiu architektúru s menším počtom parametrov

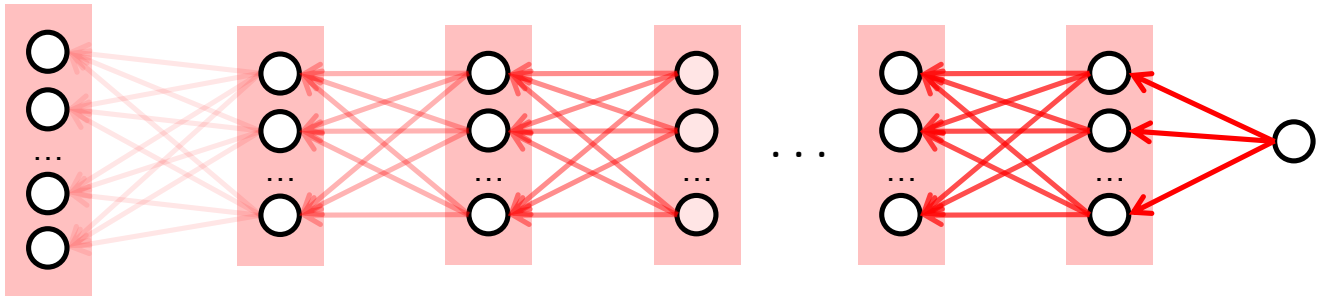
Numerické problémy pri učení (1)

- Gradient je určený na každej vrstve pravidlom spätného šírenia chyby: vstup × g'(vstup) × vážená suma chyby na nasledujúcej vrstve
  - tzn. rekurzívne násobíme derivácie aktivačnej funkcie
  - Derivácie sigmoidálnych funkcií: logit' < 0.45, tanh' < 1



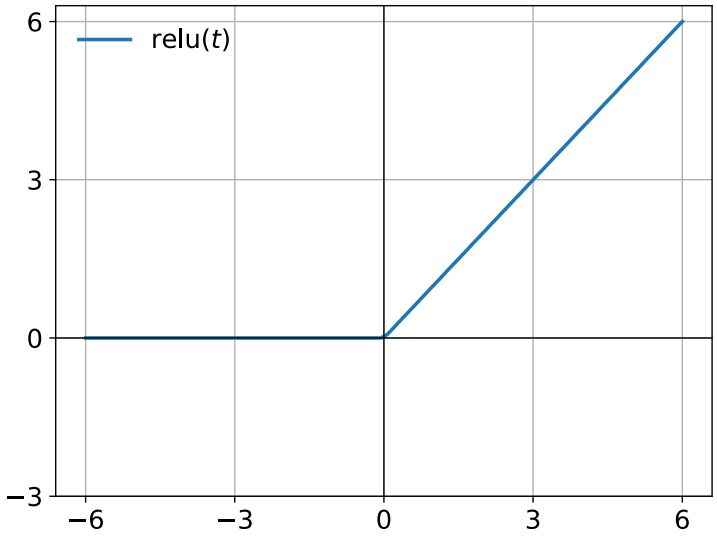
Vymiznutie gradientu

- Pri hlbokých sieťach sa chyba môže postupne znižovať až gradient úplne "vymizne" – učenie sa utlmí



Aktivačná funkcia ReLU

- Preto sa pri hlbokých sieťach používa častejšie rektifikačná aktivačná funkcia (ReLU): $g(x) = \max(0, x)$, derivácia ReLU pre $x < 0, g'(x) = 0$ a pre $x > 0, g'(x) = 1$



Numerické problémy pri učení (2)

- Pre $x < 0$ (polovicu rozsahu vstupných hodnôt) je hodnota ReLU aktivácie aj derivácia 0 (tzn. neurón nemá vplyv na predikciu a neučí sa): *leaky* ReLU
  - $g(x) = \begin{cases} x & ak\ x \geq 0 \\ 0.01x & inak \end{cases}$
- Keďže aktivácia nie je ohraničená, gradient môže mať naopak veľmi veľkú hodnotu ("explodovanie" gradientu)
  - Je potrebné na každej vrstve kontrolovať veľkosť gradientu a ohraničiť ho zvolenou maximálnou hodnotou, alebo stabilizovať učenie normalizáciou

Normalizácia (1)

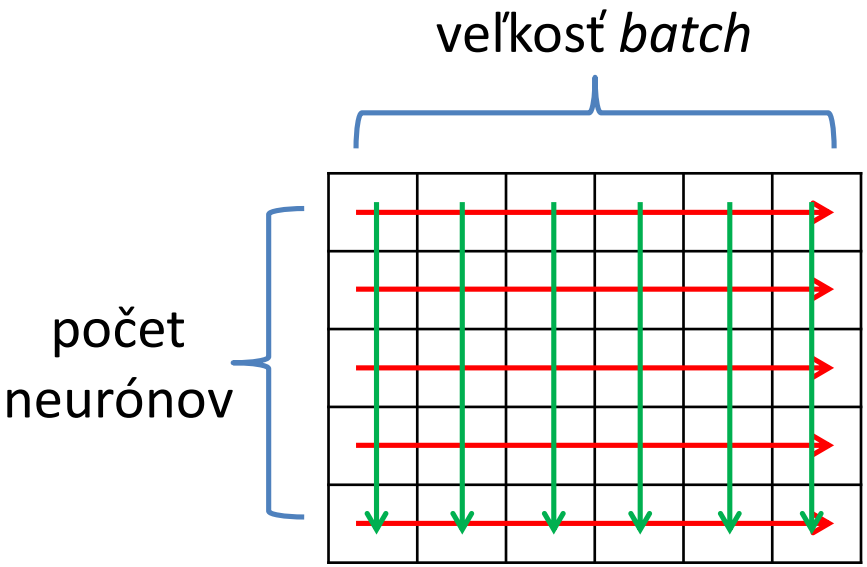
- Gradient: vstup × g'(vstup) × vážená suma chyby na nasledujúcej vrstve
- Ak zmenšíme vstup, zmenšíme aj gradient
- Vstup = vážená suma výstupov z predchádzajúcej vrstvy - znormujeme výstup z každej vrstvy

$$s_{i,j} = \frac{s_{i,j} - \mu}{\sqrt{\sigma^2 + \epsilon}}$$

- μ – stredná hodnota aktivácií na danej vrstve, σ – variancia aktivácií na danej vrstve, ϵ – malá hodnota aby sa zabránilo deleniu 0

Normalizácia (2)

- Strednú hodnotu a varianciu vypočítame pre podmnožinu príkladov pri učení (*batch*) – pre *batch* môžeme výstup z každej vrstvy zapísať ako maticu S



- *batch normalization* – po riadkoch
- *layer normalization* – po stĺpcoch

Učenie hlbokých sietí - zhrnutie

- Regularizácia učenia
  - Snažíme sa zabrániť preučeniu
  - Penalizácia – ohraničíme zložitosť modelu (LASSO, RIDGE)
  - Dropout – znížime varianciu učenia
- Numerické problémy pri učení
  - (leaky) ReLU aktivačná funkcia pre skryté vrstvy (potlačí vymiznutie gradientu)
  - Normalizácia – stabilizuje učenie ohraničením veľkosti aktivácie na skrytých vrstvách (potlačí „explodovanie“ gradientu) – *batch/layer*

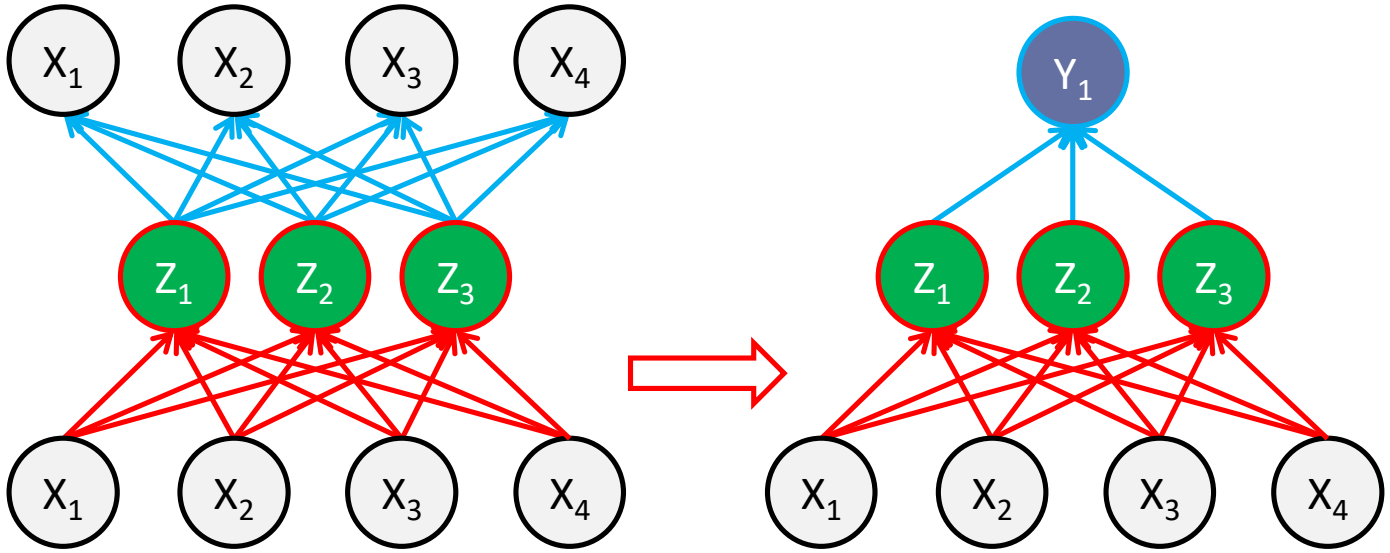
Využitie neoznačených dát

Vieme pri predikcii využiť neoznačené dáta?

- Vytvoriť manuálne veľkú trénovaciu množinu označených príkladov je veľmi prácne. Pre veľa domén je však možné jednoducho získať veľké množstvo neoznačených dát
- Semikontrolované učenie reprezentácie
  - Latentné príznaky naučíme na neoznačených dátach (naučená reprezentácia však nemusí byť najlepšia pre danú klasifikačnú úlohu – transformáciu môže byť potrebné doučiť na označených príkladoch)
  - Autokodéry (*autoencoders*) – základná architektúra pre neurónové siete

Semikontrolované učenie reprezentácie (1)

- Transformáciu vstupných dát na latentné premenné Z_1, Z_2, ... pred-učíme na neoznačených dátach. Výsledné váhy medzi vstupnou a skrytou vrstvou doučíme na označených príkladoch.

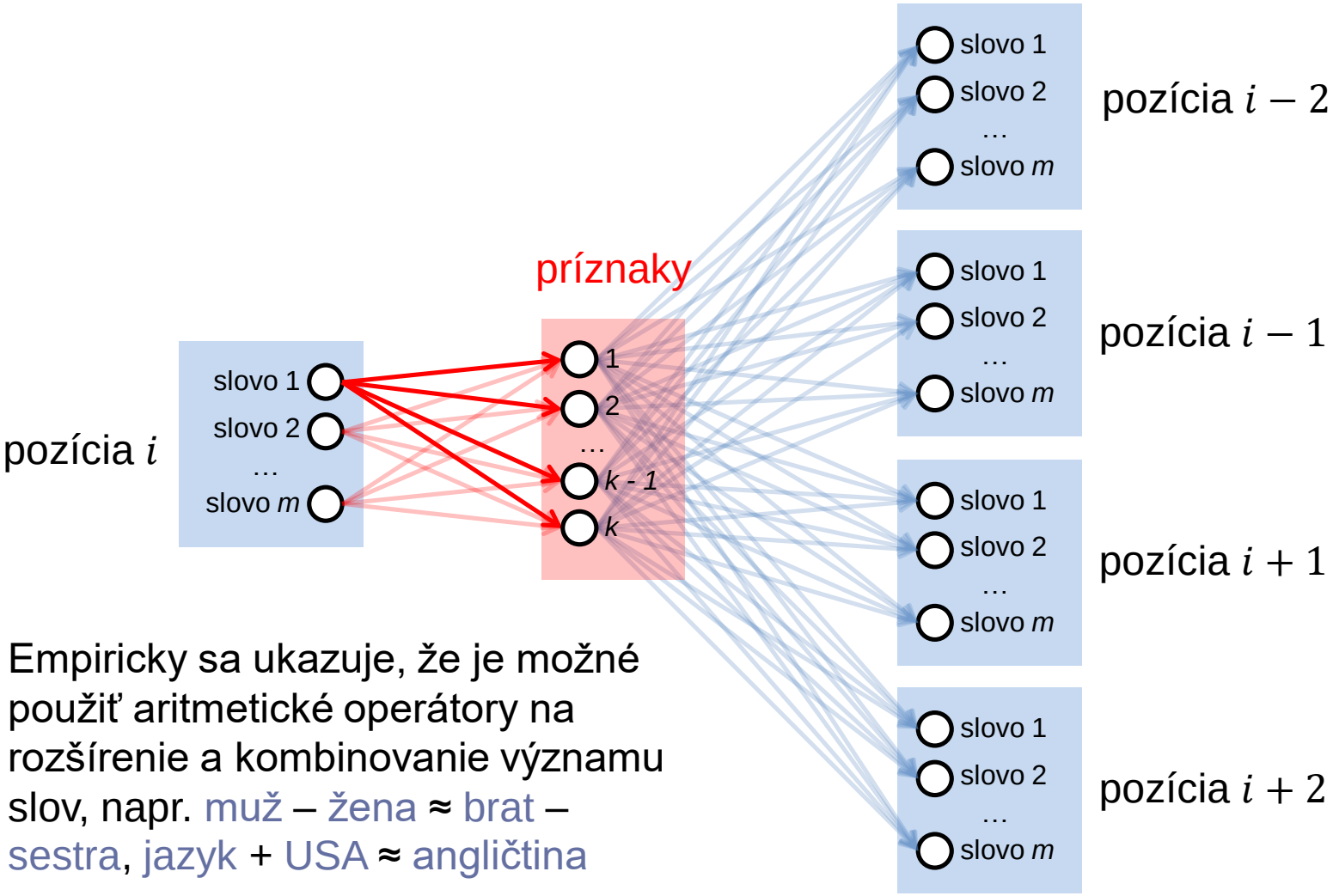


Semikontrolované učenie reprezentácie (2)

- Distribuovaná reprezentácia slov

- Predpoklady: jazykové vlastnosti slova sa dajú odvodiť od okolitých slov, ktoré sa spolu vyskytujú s daným slovom
- Ak máme veľkú množinu textov, naučíme sieť, ktorá bude pre každé slovo predpovedať, aké slová môžu nasledovať, alebo predchádzať danému slovu
  - V parametroch skrytej vrstvy by mali byť zakódované vlastnosti slova
  - Vektor parametrov potom použije ako vstupnú reprezentáciu slova pri riešení textových úloh (klasifikácia, extrahovanie informácií, automatický preklad, ...)

Semikontrolované učenie reprezentácie (3)



Empiricky sa ukazuje, že je možné použiť aritmetické operátory na rozšírenie a kombinovanie významu slov, napr. muž – žena ≈ brat – sestra, jazyk + USA ≈ angličtina