Strojové učenie a neurónové siete - **zadanie 2** - trénovanie neurónových sietí

15. október 2020
I-SUNS cv. 4

Zadanie 2

- Zadanie a bodovanie je k dispozícii na dokumentovom serveri v AIS
- Máte sa naučiť:
  1. Používať validačnú množinu
  2. Vedieť identifikovať podtrénovanie/ pretrénovanie
  3. Sledovať a regulovať proces trénovania siete
  4. Trénovať Stroj s podpornými vektormi
- Dataset:
  - dostupný v AIS
  - Testovacia množina je oddelená, aby sa dali systémy navzájom porovnávať
  - popisy stĺpcov podľa Spotify API

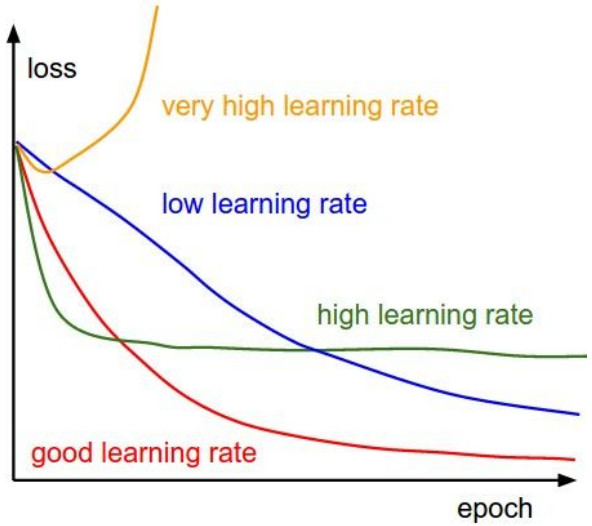
Obsah tejto prezentácie

- Solvery (optimalizátory) - SGD, ADAM
- Delenie trénovacia/testovacia/validačná množina
- Podtrénovanie a pretrénovanie neurónových sietí
- Regularizačné techniky

- O týždeň - SVM

Proces trénovania - opakovanie

- V sieti trénujeme váhy a prahy:
  - Menia sa tak, aby sme znížili* kriteriálnu funkciu (označuje veľkosť chyby pre celý trénovací dataset)
  - Smer (aj veľkosť) zmeny je určený podľa gradientov vypočítaných v rámci spätnej propagácie
  - Presnú veľkosť zmeny pre prah/váhu určuje solver (parametrizovaný min. rýchlosťou učenia)
- Proces trénovania sledujeme na vývoji kriteriálnej funkcie
- Trénovanie ukončujeme:
  - Ak dosiahneme dostatočne malú chybu (dostatočne nízka krit. f-cia)
  - Po predom určenom počte epoch/iterácií
  - Ak sa sieť prestala zlepšovať (resp. začala sa zhoršovať) - krit. f-cia stagnuje, v za *x* epoch sa chyba nezlepšila o min. *y*



Zdroje obrázkov:
http://cs231n.stanford.edu/

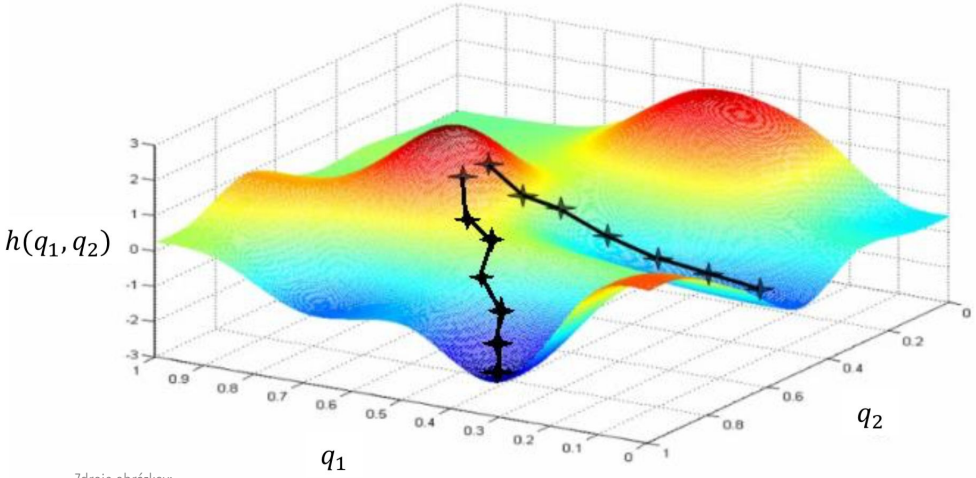
Solvery

Optimalizátory,

Gradient descent (GD)

$$\theta_{i+1} = \theta_i + \alpha \nabla_\theta L(\theta_i)$$

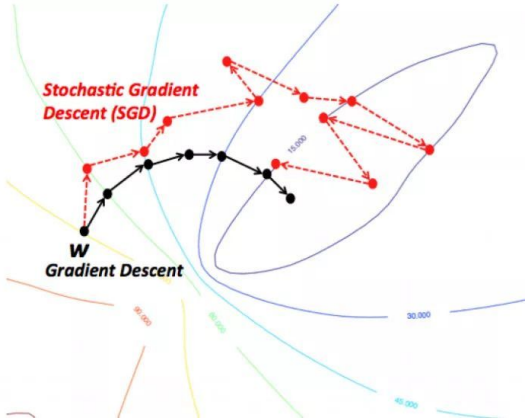
- α - pamater rýchlosti učenia
- θ - váhy
- i - čas (iterácia)



Zdroje obrázkov:
https://shashank-ojha.github.io/ParallelGradientDescent/
https://golden.com/wiki/Stochastic_gradient_descent_(SGD)

Stochastic gradient descent (SGD)

- Posielam dáta do trénovania po častiach - dávkach (batchoch)
- Jeden výpočet úpravy gradientu - 1 iterácia
- Úprava pre celú trénovaciu množinu - 1 epocha
- počet iterácií v jednej epoche?

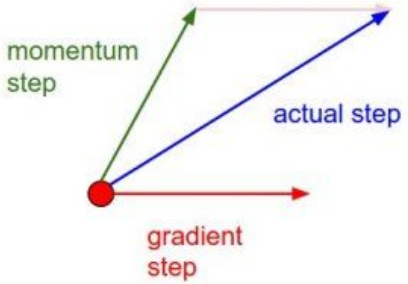


Solvery

GD with momentum

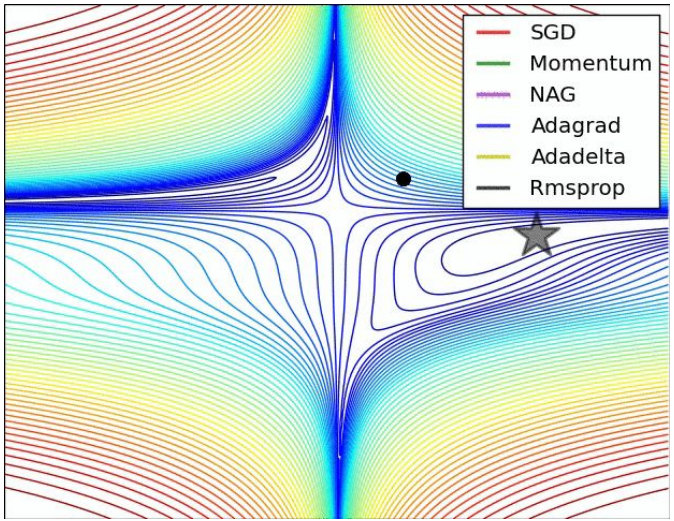
$$v_{i+1} = \gamma v_i + \alpha \nabla_\theta L(\theta_i)$$
$$\theta_{i+1} = \theta_i - v_{i+1}$$

- Gamma γ je hyperparameter (obvykle 0.9)
- Úprava váh - lineárna kombinácia aktuálneho stochastického gradientu a zmeny v predchádzajúcom kroku

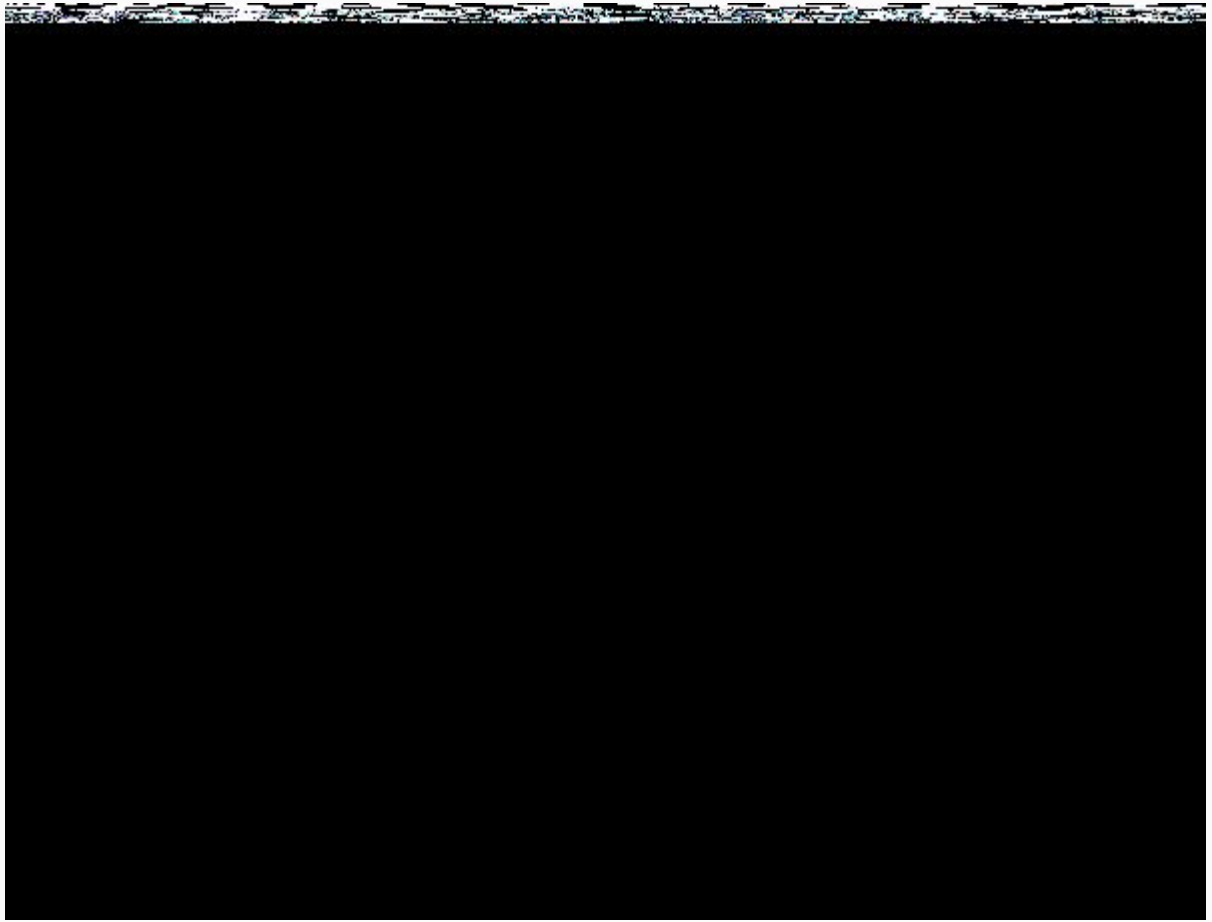


Adaptive moment estimation (ADAM)

- Takisto momentová metóda
- využíva pohyblivé priemery z viacerých minulých iterácií
- Tri hyperparametre: $ß_1$,$ß_2$,ε
- Postupne zmenšuje α

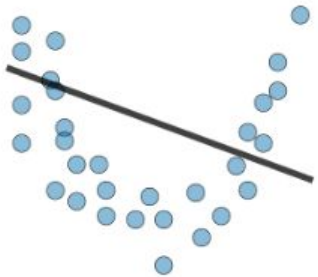
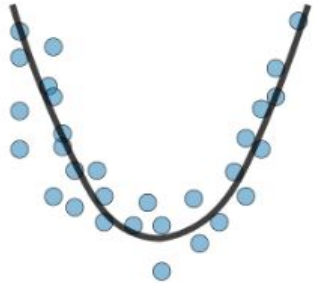
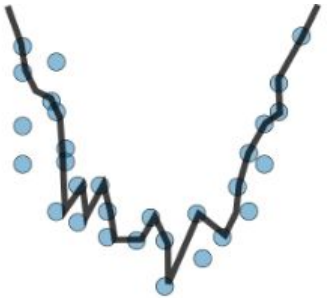
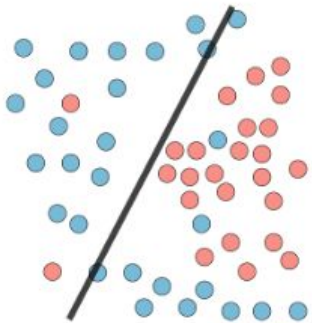
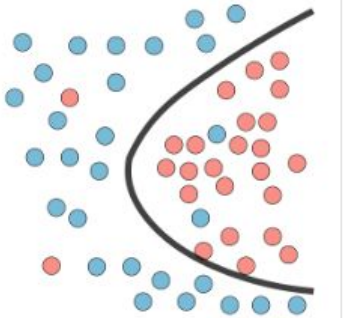
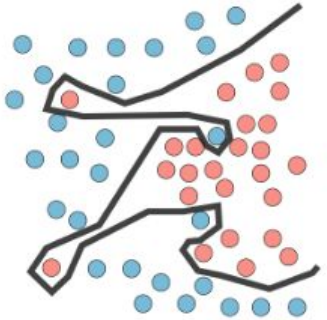


Zdroje obrázkov:
https://twitter.com/alecrad
http://cs231n.stanford.edu/



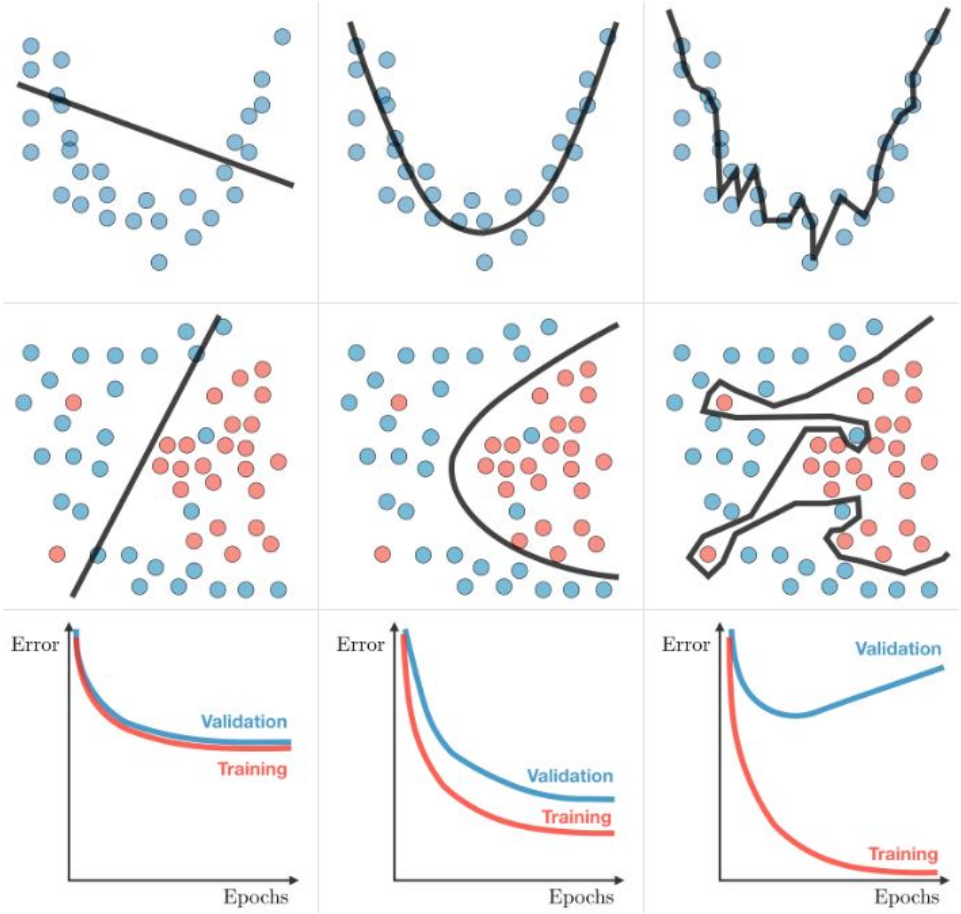
Zdroje obrázkov:
https://towardsdatascience.com/complete-guide-to-adam-optimization-1e5f29532c3d

Pretrénovanie a podtrénovanie - ilustrácia

	Podtrénovanie	Dobré natrénovanie	Pretrénovanie
Regresný príklad			
Klasifikačný príklad			

Zdroje obrázkov:
https://stanford.edu/~shervine/teaching/cs-229/cheatsheet-machine-learning-tips-and-tricks

Pretrénovanie a podtrénovanie - pozorovanie



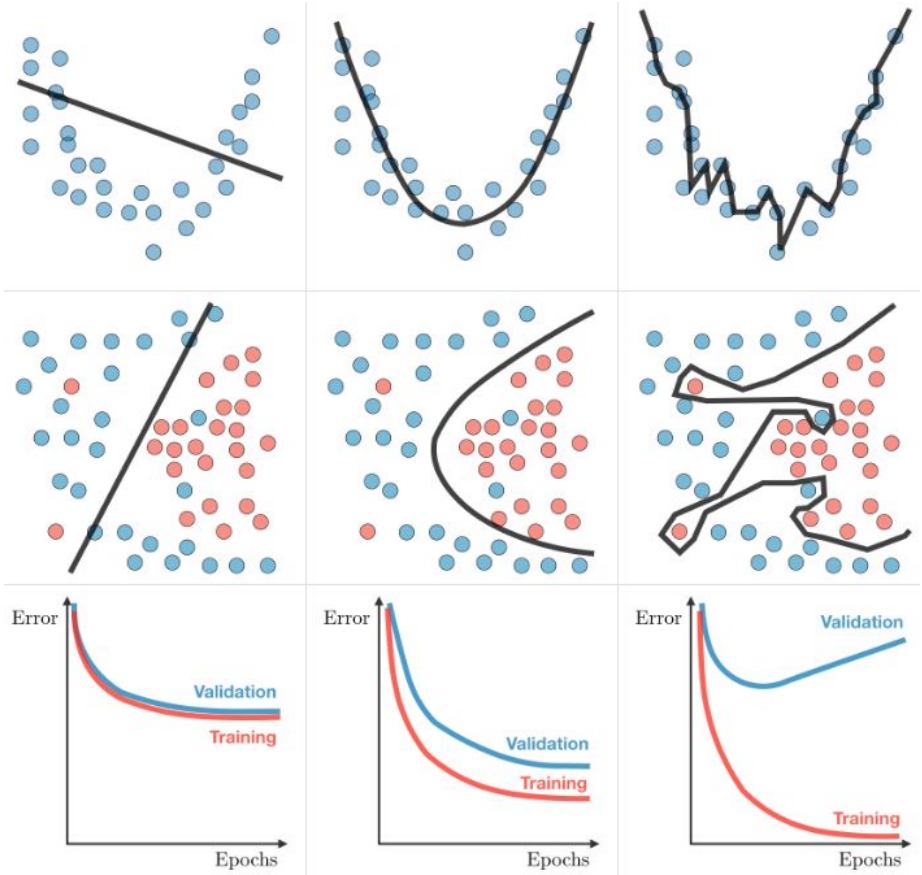
- Pridávam do trénovania **validačnú množinu**
- Samostatná množina dát, pomocou ktorej nepočítam chybu (neovplyvňuje gradienty)
- Overujem na nej, ako dobre vie sieť generalizovať -> rozpoznávať nevidené dáta

Zdroje obrázkov:
https://stanford.edu/~shervine/teaching/cs-229/cheatsheet-machine-learning-tips-and-tricks

Pretrénovanie a podtrénovanie - opravy

- Podtrénovanie:
  - Zastavovacia podmienka
  - Iný model
  - Viac príznakov
  - Väčšia sieť

- Pretrénovanie:
  - Zastavovacia podmienka
  - Viac trénovacích dát
  - Menej príznakov
  - Menšia sieť
  - Regularizácia



Zdroje obrázkov:
https://stanford.edu/~shervine/teaching/cs-229/cheatsheet-machine-learning-tips-and-tricks

Regularizačné techniky - L1/2 regularizácia

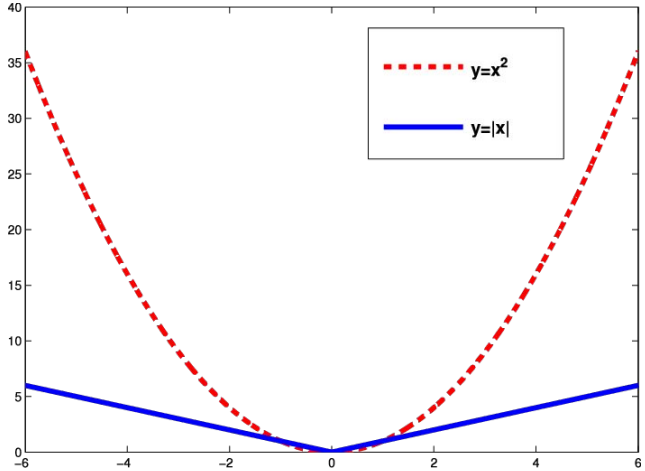
- Penalizácia váh je pridávaná do kriteriálnej funkcie
- L1 regularizácia:

$$Loss = Error(y, \hat{y}) + \lambda \sum_{i=1}^{N} |w_i|$$

- L2 regularizácia:

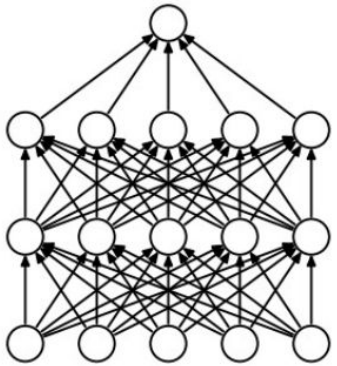
$$Loss = Error(y, \hat{y}) + \lambda \sum_{i=1}^{N} w_i^2$$

- λ je hyperparameter - sila regularizácie - (musí byť väčšia ako 0)
- Penalizujú príliš vysoké a nízke váhy
- L1 - sparse vektor váh, L2 - veľkosti váh zhruba rovnaké

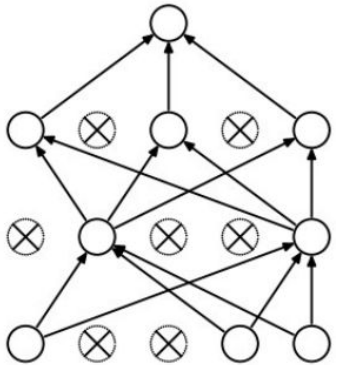


Regularizačné techniky - dropout/dropconnect

- Náhodné vypínanie váh (neurónov) počas trénovania
- Jeden hyperparameter - pomer vypnutých/zapnutých váh



(a) Standard Neural Net



(b) After applying dropout.

Zdroje obrázkov:
Srivastava, Nitish, et al. "Dropout: a simple way to prevent neural networks from overfitting", JMLR 2014

Regularizačné techniky - normalizácia dávok

- *Batch normalization*
- Normalizujeme nielen vstupy ale aj váhy počas trénovania
- Obvykle implementovaná ako vlastná vrstva (pred akt. funkciou)
- Sieť je odolnejšia pred zlou inicializáciou, zlepšuje sa úspešnosť aj rýchlosť konvergencie

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

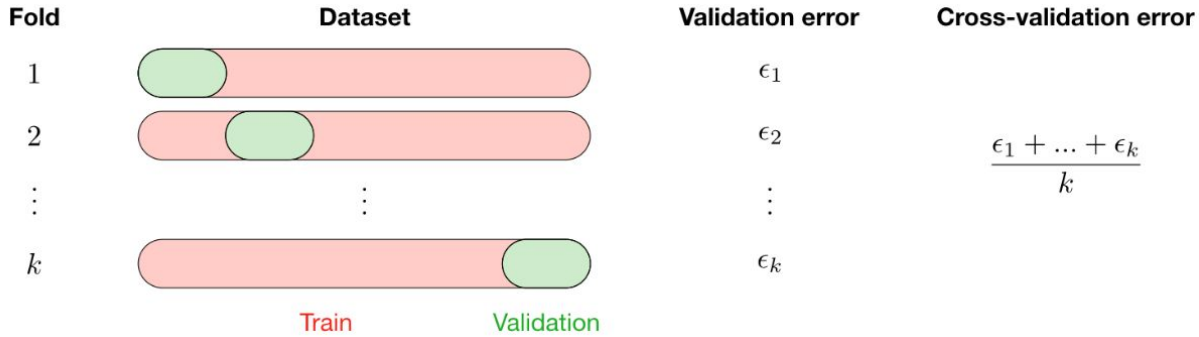
$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma \widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Algorithm 1: Batch Normalizing Transform, applied to activation x over a mini-batch.

Trénovanie v praxi

1. Cross-validácia

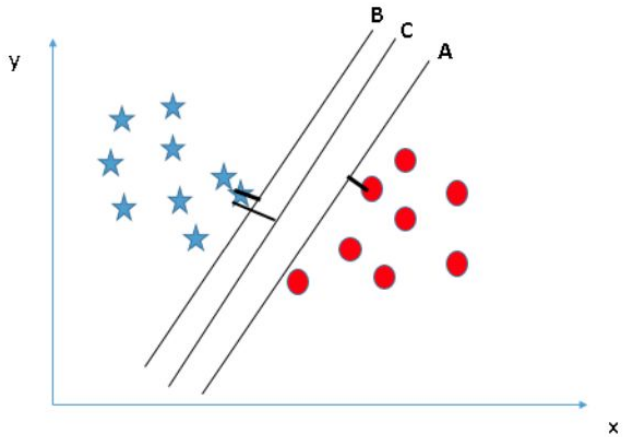


2. Grid search parametrov - pokus/omyl pri hľadaní hyperparametrov za mňa môže spraviť program
   - sklearn.model_selection.GridSearchCV — scikit-learn 0.23.2 documentation

Zdroje obrázkov:
https://stanford.edu/~shervine/teaching/cs-229/cheatsheet-machine-learning-tips-and-tricks

Metóda podporných vektorov - intuícia

- Učenie s učiteľom
- Hľadá hyperroviny oddeľujúce triedy (klasifikácia) / približnú funkciu (regresia) - podobné neurónkam
- Hľadá váhy *w* pre parametre, ktorých kombinácia so vstupmi predikuje výstup - podobné neurónkam
- Predstavuje koncept podporných vektorov, ktoré pomáhajú hľadať dobré hyperroviny - optimalizujem pre "ťažké" vstupy, nie pre všetky
- Vyberáme hyperparametre:
  - Druh kernelu
  - C
  - Gamma
  - Stupeň (pri polynomiálnom)



Zdroje obrázkov:
https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/

Priestor na otázky