Neuronové sítě 1 - Vrstevnaté neuronové sítě

18NES1 - 15. hodina, LS 2024/25

Zuzana Petříčková

7. dubna 2025

Co jsme probírali minule

Vrstevnatá neuronová síť (MLP) a algoritmus zpětného šíření (backpropagation)

- Učení vrstevnatých neuronových sítí a nastavení hyperparametrů
- Podrobněji o různých algoritmech učení
- Ukázky různých typů úloh
  - Binární klasifikace, klasifikace do více tříd, regrese, predikce časové řady (poslední úlohu jsme nedokončili)
  - Specifika jednotlivých úloh a příprava dat

Tento týden

1. Dokončení příkladu (predikce časové řady)
2. Neuronové sítě a zobecňování
   - Techniky pro porovnání modelů (obecně i v případě, že máme málo trénovacích dat)
   - Techniky pro prevenci přeučení
   - Ukázky a příklady
3. Úvod do konvolučních neuronových sítí

Praktické ukázky různých typů úloh

- binární klasifikace: Breast Cancer
- klasifikace do více tříd: MNIST
- regresní úloha: Wine quality
- **Predikce časové řady: Denní minimální teploty v Melbourne** - zbývá dokončit

Ukázka úlohy s časovou řadou: Min. denní teploty

- Dataset: Denní minimální teploty v Melbourne, roky 1981–1990
- Cíl: předpovědět teplotu následujícího dne na základě předchozích hodnot

Vlastnosti dat:

- 1 příznak (časová řada – teplota)
- Více než 3600 záznamů – dostatečné pro učení i testování
- Nejde o nezávislé vzory – potřebujeme zachytit časovou závislost

Použití posuvného okénka (sliding window):

- Pro každý vzor použijeme např. posledních 10 dní k predikci následujícího
- Tím vznikne tabulková reprezentace vhodná i pro MLP

Min. denní teploty – model a nastavení

Způsob přípravy vstupu:

- Výroba trénovací, validační a testovací množiny pomocí posuvného okénka (pro každou sadu jiné časové okno!)
- Normalizace vstupních hodnot (např. MinMaxScaler)

Nastavení modelu:

- Vstupní vrstva: počet neuronů odpovídá délce okénka (např. 10)
- Skryté vrstvy s ReLU / tanh
- Výstupní vrstva s lineární přenosovou funkcí (1 hodnota – předpověď teploty)
- Chybová funkce: MSE, metriky: MSE / MAE / RMSE

Pozorování:

- Abychom překonali referenční model, je třeba MLP vyladit
- Volba délky okénka ovlivňuje kvalitu predikce
- Model se může přeučit, pokud použijeme příliš komplexní architekturu

Shrnutí: Časová řada a metoda posuvného okénka

1. Časové řady neobsahují nezávislé vzory – při vytváření trénovací, validační a testovací množiny musíme respektovat časový kontext.
2. Metoda posuvného okénka umožní převést časovou řadu na trénovací množinu pro MLP.
3. Délka okénka (počet vstupních hodnot) je důležitý hyperparametr.
4. Na výstupní vrstvě používáme lineární funkci, jako u regrese.
5. Predikce časových řad lépe zvládají specializované architektury (RNN, LSTM), ale MLP s metodou posuvného okénka je jednoduchý a srozumitelný start.

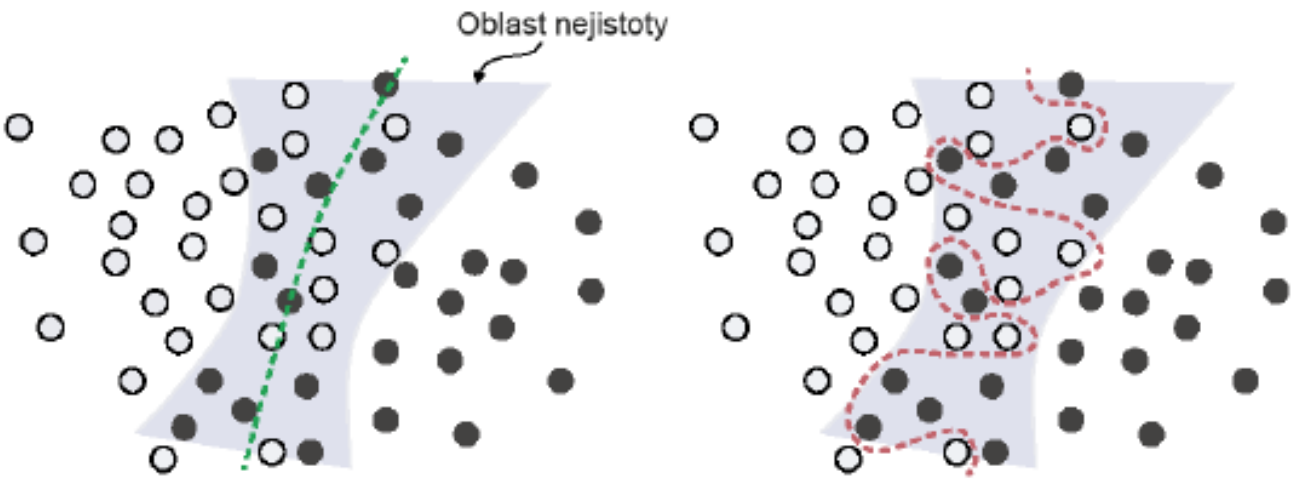
Shrnutí: Časová řada a metoda posuvného okénka

Možnosti rozšíření

- Vstupem modelu nemusí být pouze minulé hodnoty výstupu, ale i další hodnoty (např. jiné naměřené hodnoty - tlak vzduchu, vlhkost,...)
- Predikovat nemusíme jen jedno číslo (následující hodnota), ale hodnotu za delší časový interval / více hodnot

Schopnost neuronové sítě zobecňovat

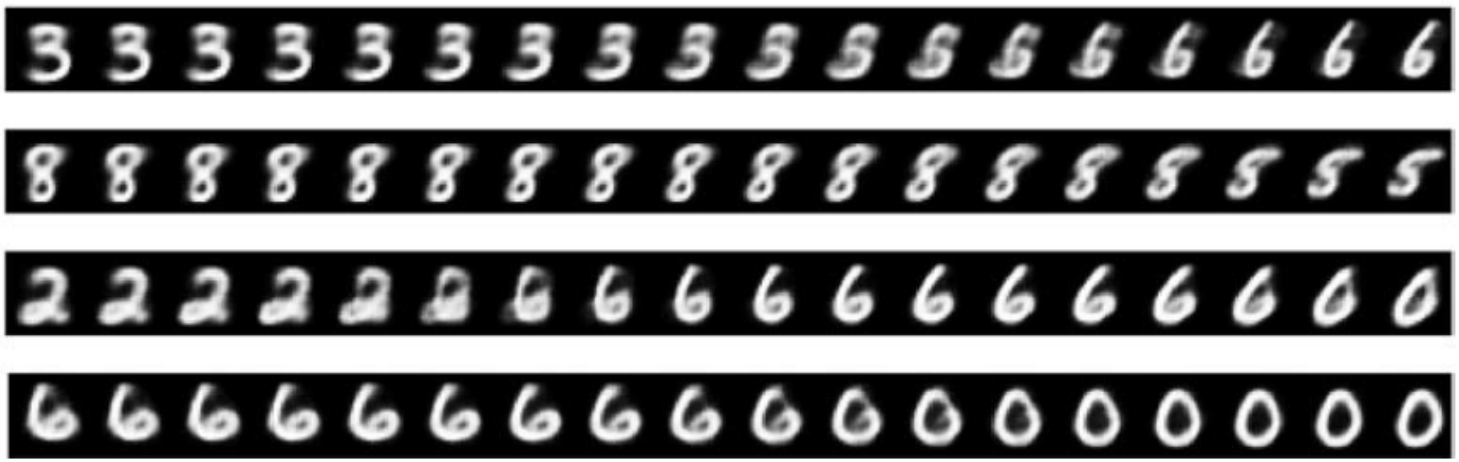
- schopnost dát správný výstup i pro data, co nebyla v trénovací množině
- Ukázka: správně naučený model vs. přeučený model:



F. Chollet: Deep learning v jazyku Python, obr. 5.5

Schopnost neuronové sítě zobecňovat

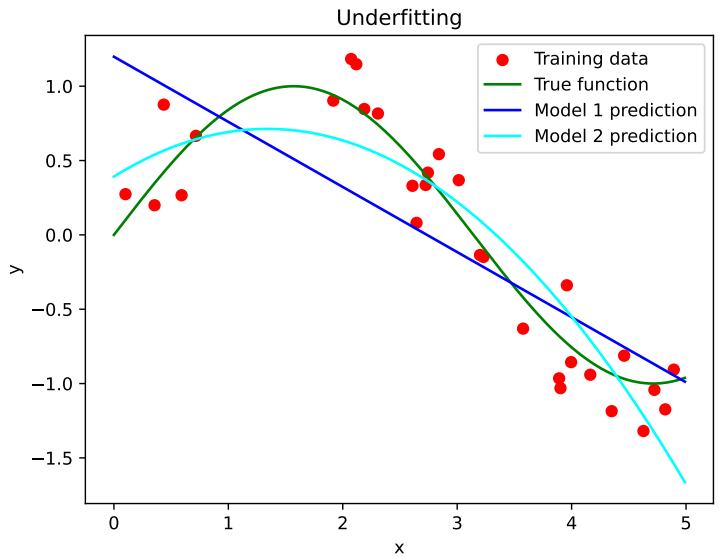
- hranice mezi jednotlivými třídami nemusí být snadno k nalezení:

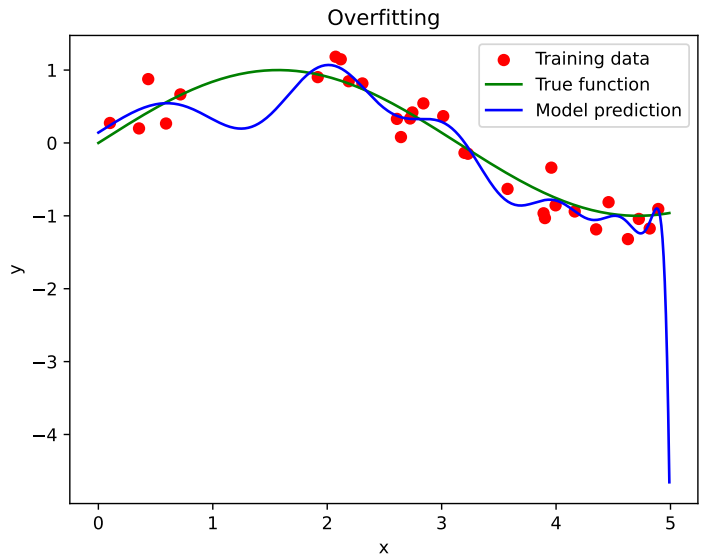


F. Chollet: Deep learning v jazyku Python, obr. 5.7

Schopnost neuronové sítě zobecňovat

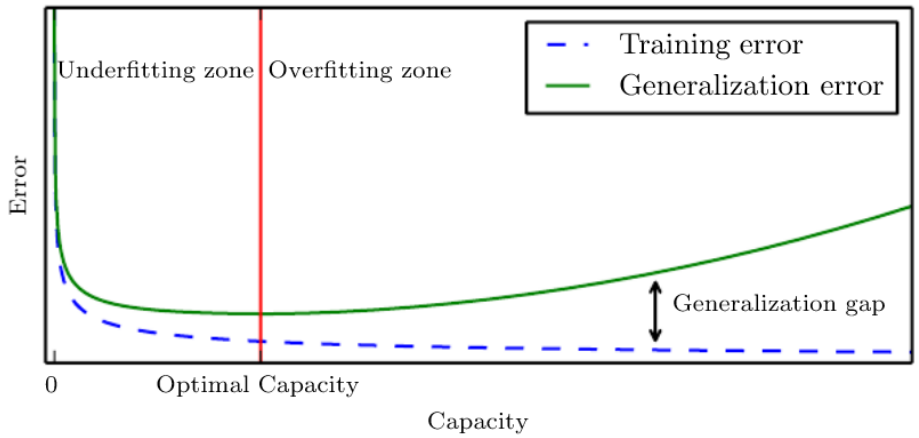
- problematika nedostatečného naučení (underfitting) nebo naopak přeučení (overfitting) v případě regresní úlohy:





Schopnost vrstevnaté neuronové sítě zobecňovat

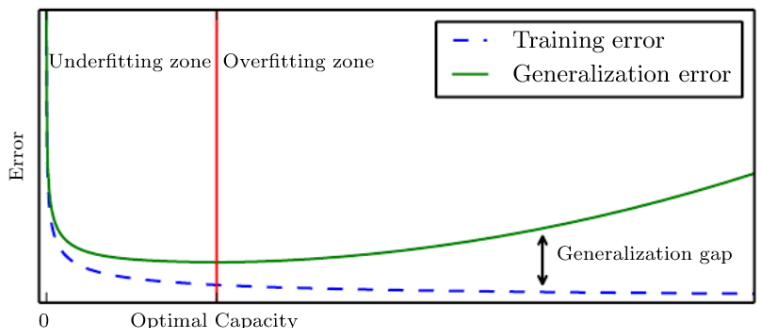
- záleží na architektuře sítě, zjednodušeně na počtu parametrů modelu, tj. **kapacitě**:
- Malý model
  - potenciálně chybné, ale stabilní predikce (pro různé trénovací množiny a počáteční hodnoty vah)
  - hrozí **underfitting** (model se nenaučil správně)
- Velký model
  - větší variabilita
  - hrozí **overfitting** - model se přeučil, špatně zobecňuje



https://www.deeplearningbook.org/, Figure 5.3

Schopnost vrstevnaté neuronové sítě zobecňovat

- s **kapacitou** souvisí i **potřebná velikost trénovací množiny:**
- Malý model
  - potenciálně chybné, ale stabilní predikce (pro různé trénovací množiny a počáteční hodnoty vah)
  - hrozí **underfitting** (model se nenaučil správně)
  - **k naučení a správnému zobecňování potřebuje méně trénovacích dat**
- Velký model
  - větší variabilita
  - hrozí **overfitting** - model se přeučil, špatně zobecňuje
  - **k naučení a správnému zobecňování potřebuje více trénovacích dat**



Schopnost vrstevnaté neuronové sítě zobecňovat

Věta: Vztah mezi kapacitou a potřebným počtem trénovacích vzorů

- Pro síť s 1 skrytou vrstvou, počtem parametrů (vah a biasů) w a počtem neuronů h a s omezením pro generalizační chybu ϵ, je počet trénovacích vzorů N potřebných pro správné zobecňování:

$$N \geq \frac{w}{\epsilon} log_2(\frac{h}{\epsilon})$$

$\rightarrow$ model nemůže dobře zobecňovat, jestliže

$$N < \frac{w}{\epsilon}$$

$\rightarrow$ Pro požadovanou přesnost (Accuracy) alespoň 90 % je třeba vybrat alespoň $10 \cdot w$ vzorů

Generalizace u hlubších neuronových sítí

Vztah mezi kapacitou a potřebným počtem trénovacích vzorů

- **Rámcový odhad potřebného počtu trénovacích vzorů pro sítě s více vrstvami:**
  Pro hlubokou síť s celkovým počtem parametrů w a požadovanou generalizační chybou ϵ platí:

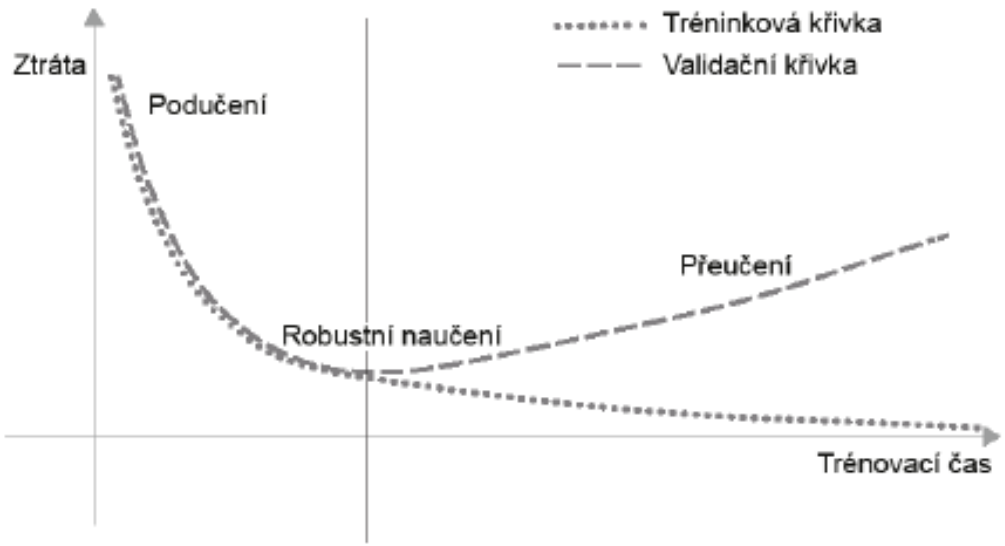
$$N \geq O\left(\frac{w \cdot \log w}{\epsilon}\right)$$

- S rostoucím počtem vrstev (a parametrů) roste i potřeba většího počtu trénovacích dat.
- Empiricky: často potřebujeme **řádově více trénovacích vzorů než parametrů**.
- Pro dobrou generalizaci je třeba buď:
  - použít dostatečně velkou trénovací množinu,
  - nebo použít vhodnou techniku pro zlepšení modelu zobecňovat

Jak měřit, zda model dobře zobecňuje?

Techniky založené na samplování

- **Využití validační množiny dat**
  - trénovací množinu rozdělíme na dvě části: trénovací (např. 70 %) a validační (30%)
  - model učíme pouze na trénovací podmnožině
  - pomocí validační množiny odhadneme generalizační chybu



F. Chollet: Deep learning v jazyku Python, obr. 5.1

- **Využití krosvalidace** (např. k-násobná křížová validace)

Jak měřit, zda model dobře zobecňuje? Využití validační množiny dat

Na co dát pozor při sestavování validační množiny dat

- Všechny tři množiny dat (trénovací, validační i testovací) by měly být stejně reprezentativní (např. obsahovat podobné procento vzorů z jednotlivých tříd)
- Pro časové řady: validační (a testovací) data by měla v čase následovat až za těmi trénovacími
- Pozor na redundance v datech (stejné nebo hodně podobné vzory v trénovací a validační či testovací množině) - mohou zkreslit výsledek

Křížová validace (krosvalidace)

- Umožní nám odhadnout, jak dobře model zobecňuje, obzvlášť když je trénovací množina malá
- Zobecnění principu rozdělení dat na trénovací a testovací množinu
- Pomáhá detekovat overfitting / underfitting
- Umožňuje srovnání modelů nebo konfigurací hyperparametrů

Typy křížové validace:

- **Monte Carlo křížová validace** - vhodná pro středně velké datové sady - flexibilnější, náhodný přístup
- **K-násobná křížová validace** - vhodnější pro velmi malé datové sady - nezapomene na žádný vzor

Monte-Carlo křížová validace

Základní princip: opakované náhodné dělení

1. Pro $i = 1, ..., k$:
   - Náhodně rozděl trénovací množinu T na trénovací množinu T_1 a testovací množinu T_2 (např. v poměru 70:30)
   - nauč model na trénovací množině T_1 a použij T_2 jako testovací množinu
   - zaznamenej chybu modelu na testovací množině T_2
2. Spočti průměr a směrodatnou odchylku chyby přes k pokusů (obvykle $K = 100$)

K-násobná křížová validace

- oproti Monte-Carlo systematicky (a deterministicky) pokrývá celá data (žádný vzor nevynechá)

Základní princip

1. Rozděl trénovací množinu T na k stejně velkých disjunktních podmnožin T_1,..., T_k
2. Pro $i = 1, ..., k$:
   - nauč model na trénovací množině $T \setminus T_i$ a použij T_i jako testovací množinu
   - zaznamenej chybu modelu na testovací množině T_i
3. Spočti průměr a směrodatnou odchylku chyby přes k pokusů (obvykle $K = 10$)

Příklad

regularization_mnist.ipynb

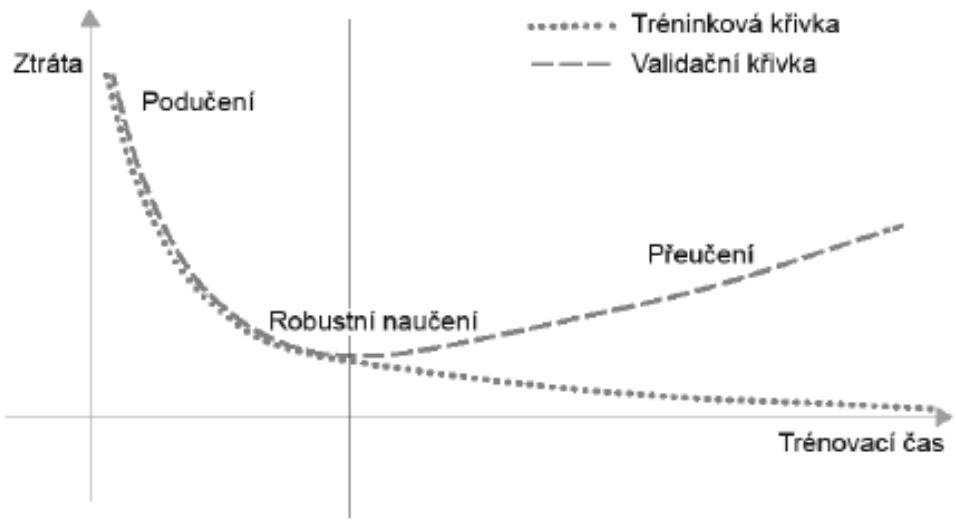
- Praktický příklad: datová sada MNIST (číslice) a omezená velikost trénovací množiny
- Úkol: experimentujte s velikostí modelu a počtem vzorů v trénovací množině
- Sledujte, jak s klesající velikostí trénovací množiny (a s rostoucí velikostí modelu) roste chyba na validační a testovací množině dat (a tedy klesá schopnost modelu správně zobecňovat (nebo vůbec se naučit danou úlohu)

Jak zajistit, aby (hluboká) neuronová síť dobře zobecňovala?

- **Najít "optimální" architekturu** pro danou trénovací množinu (počet vrstev, neuronů, přenos. fce)
  - **Neural Architecture Search** (NAS), př. knihovna AutoKeras
- **Zvětšit trénovací množinu** (data augmentation)
- **Feature engineering** (najít lepší, informativnější, vstupní příznaky)
- **Early stopping** - včas zastavit učení za použití validační množiny
- **Regularizační techniky**
  - **L1/L2 regularizace, Dropout**, DropConnect
  - Label smoothing (zašumění labelů)
- **Normalizace** dat, vah, výstupů vrstev,....
- **Transfer learning (přenesené učení)** a **Ensembling**
- **Hyperparameter tuning** (Grid Search, Random Search, Bayesian Optimization), př. knihovna Keras Tuner

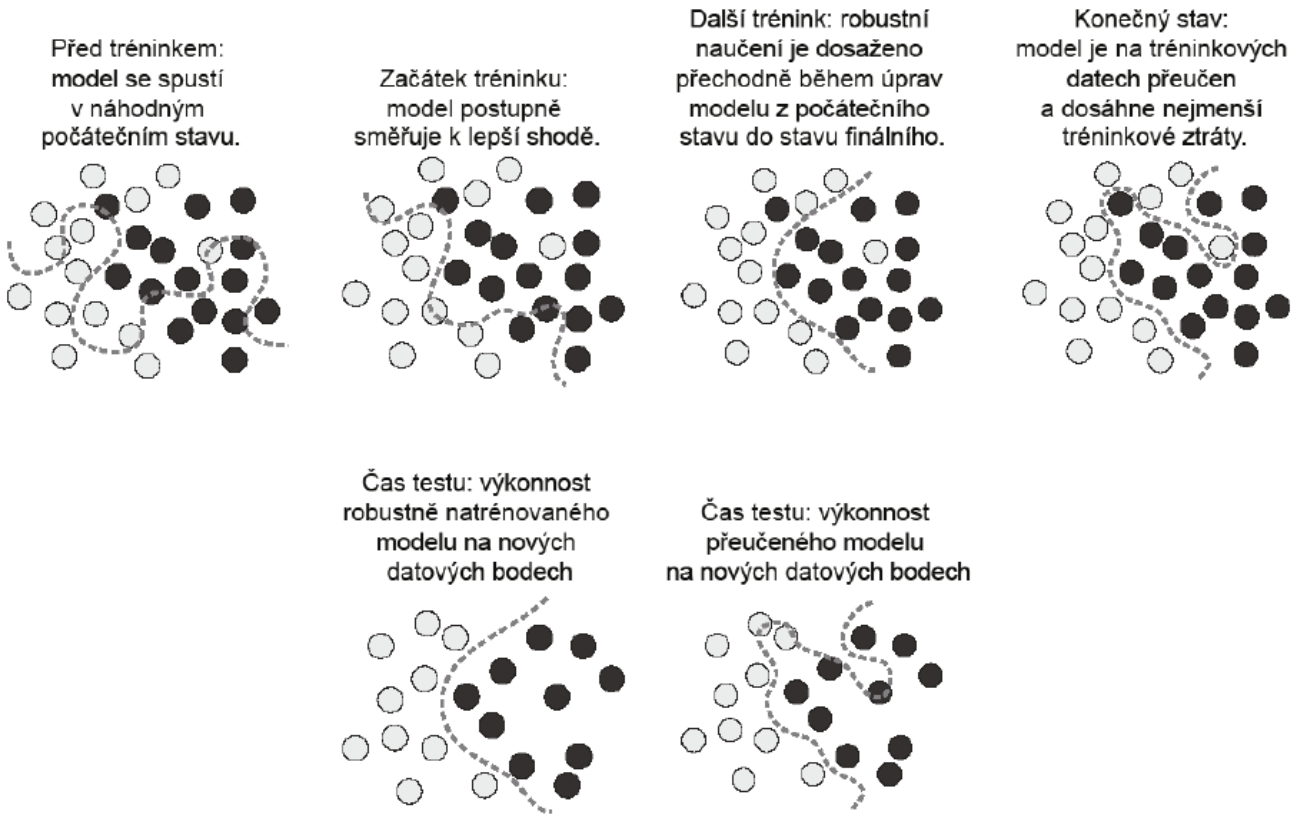
Early stopping

- trénovací množinu rozdělíme na dvě části: trénovací (např. 70-90 procent) a validační
- model učíme pouze na trénovací podmnožině
- učení ukončíme ve chvíli, kdy začne růst chyba na validační množině dat
- pozor: validační a testovací množina musí být na sobě zcela nezávislé!



F. Chollet: Deep learning v jazyku Python, obr. 5.1

Early stopping



Obrázek 5.10:
Přechod od náhodného modelu k přeučenému modelu a dosažení robustního výsledku jako mezistavu

F. Chollet: Deep learning v jazyku Python, obr. 5.10

Regularizační techniky

Základní princip

- Přidávají k základní chybové funkci (např. E_{loss}) další penalizační členy:
  $E = c_{loss}E_{loss} + c_A E_A + c_B E_B + ...$
- **Occamova břitva:** Menší sítě s jednodušší, hladší funkcí lépe zobecňují.
- Existuje celá řada jednoduchých i sofistikovaných penalizačních členů.

Regularizační techniky

Weight decay, L2-regularizace (Werbos, 1988)

- asi nejznámější a nejpoužívanější penalizační člen:

$$E = \beta E_{loss} + (1-\beta)\frac{1}{2}\|\vec{w}\|_2^2 = \beta E_{loss} + (1-\beta)\sum_i w_i^2$$

  i je index přes všechny váhy a biasy v síti
  $\beta \in [0,1]$ udává váhu jednotlivých chybových členů

- Adaptace vah podle:

$$w_i(t+1) \quad = \quad w_i(t) - \alpha\frac{\partial E_{loss}(t)}{\partial w_i} - \alpha_r w_i(t)$$

- V průběhu učení se snižují absolutní hodnoty vah
- Prevence paralýzy sítě
- Je možné ze sítě odstranit hrany s příliš malými vahami

Regularizační techniky

Lasso, L1-regularizace

- Umožňuje efektivněji vynulovat některé váhy:

$$E = \beta E_{loss} + (1 - \beta)\frac{1}{N_w}\sum_{i=1}^{N_w}|w_i|$$

- Adaptace vah podle:

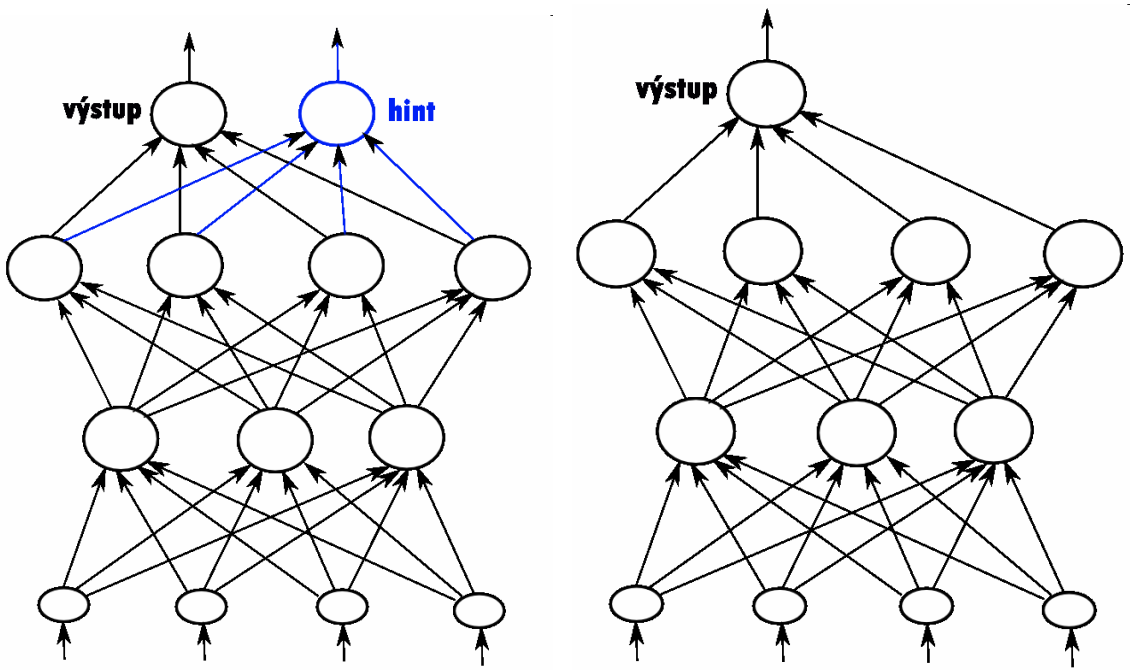
$$w_i(t+1) \quad = \quad w_i(t) - \alpha\frac{\partial E_{loss}(t)}{\partial w_i} - \alpha_r sign(w_i(t))$$

Přidání gaussovského šumu do trénovací množiny

- trénovací množinu rozšíříme o ,,zašuměné" trénovací vzory
- má podobný efekt jako L2-regularizace

Učení s nápovědou

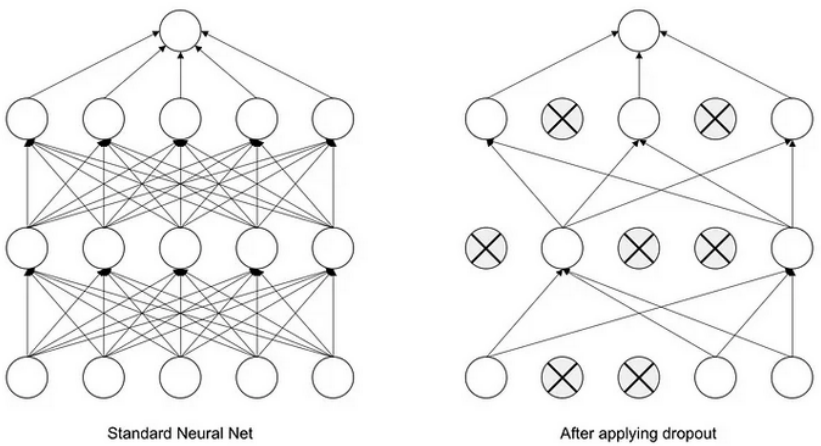
(Mostafa,1993; Suddarth,1990)



- Zvyšuje schopnost sítě zobecňovat a zrychluje učení.
- Vede k hladší funkci sítě, podporuje prořezávání.

Dropout (Srivastava et al., 2014)

- vysoce účinná metoda
- spočívá v náhodném vypínání (deaktivování) některých skrytých neuronů během učení
- při testování a používání modelu jsou všechny neurony aktivované
- implementované přidáním speciální **dropout** vrstvy za každou plně propojenou vrstvu



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

Normalizace

- Různé druhy normalizace: dat, vah, výstupů jednotlivých vrstev
- Často implementována pomocí speciálních vrstev
- **Batch normalization** – normalizuje napříč vzorky v batchi pro každý neuron (vhodné pro MLP, CNN)
- **Layer normalization** – normalizuje napříč neurony ve vrstvě pro každý vzorek (vhodné pro RNN, Transformery)
- Normalizace pomáhá zamezit problémům jako je saturace neuronů nebo mizející gradienty
- Normalizace celkově stabilizuje učení hlubokých sítí

Prořezávání vrstevnaté neuronové sítě

Myšlenka:

- Vyhodnocení, které části modelu jsou důležité
  - hrany
  - skryté neurony
  - vstupní příznaky
- Odstranění zbytečných částí modelu

Důvody:

- Zrychlení výpočtu, snížení prostorové náročnosti.
- Zlepšení schopnosti sítě zobecňovat, detekce a řešení problému s přeučením (overfitting)
- Vytvoření sítě s jasnou strukturou.
- Automatická detekce důležitých vstupních příznaků.

Prořezávání vrstevnaté neuronové sítě

Algoritmus:

1. Naučíme model s dostatečně velkou architekturou.
2. Dokud klesá chyba na validační množině (nebo dokud nepřekročí určitou mez):
   1. Spočteme relevanci skrytých neuronů nebo hran.
   2. Odstraníme nejméně relevantní neuron či hranu (neurony či hrany).
   3. Doučíme síť.

Problémy:

- Výpočet relevance skrytých neuronů nebo hran.
- Strategie, jak odstraňovat neurony / hrany.

Prořezávání vrstevnaté neuronové sítě

Používané míry relevance (pro skryté neurony):

- **Součet vah z neuronu**: $W_i = \sum_j (w_{ij})^2$.
  Nejjednodušší, ale účinná strategie.
- **Goodness factor**: $G_i = \sum_p \sum_j (y_i^p w_{ij})^2$
  Zvýhodňuje neurony, ze kterých vedou hrany s velkou vahou a které aktivní jsou pro vetšinu vstupních vzorů.
- **Consuming energy**: $E_i = \sum_p \sum_j y_i^p w_{ij} y_j$
  Zvýhodňuje neurony, které jsou často aktivní, a to společně s neurony v následující vrstvě.
- **Citlivostní koeficienty**
  $S_{ij} = \frac{\partial y_j}{\partial w_i}$ nebo $S_{ij} = \frac{\partial y_j}{\partial y_i}$ ...

Další techniky pro zlepšení zobecňování u hlubokých neuronových sítí

- **Label smoothing (zašumění labelů)** – pomáhá vyhnout se přílišné jistotě modelu tím, že mírně rozptýlí požadované výstupy
- **Transfer learning (přenesené učení)** - využití modelu předučeného na jiné množině dat
- **Ensembling** – kombinace více modelů zlepšuje stabilitu a přesnost predikcí ...

Praktické rady: co dělat, když náš model dobře nezobecňuje?

- Získejte lepší trénovací data nebo lepší příznaky
- Snižte velikost modelu, použijte adaptivní parametr učení, vylaďte hyperparametry
- Použijte dropout
- Místo dropoutu můžete pro větší modely zkusit použít Batch normalizaci a pro menší L2-regularizaci

Příklady

regularization_mnist.ipynb

- Praktický příklad: datová sada MNIST (číslice) a omezená velikost trénovací množiny
- Ukázky základních technik pro zlepšení schopnosti modelu zobecňovat (early stopping, regulraizace, dropout)
- Ukázka křížové validace

images_simple_mlp_autoencoder.ipynb

- Jednoduchý autoencoder (architektura, kdy je vstup a požadovaný výstup stejný)
- Ilustrace toho, jak může učení ovlivnit regularizace (zde - rozšíření trénovací množiny o další vzory)
- Zkuste experimentovat s trénovací množinou i s počtem neuronů a podívejte se na výsledky

Dobrovolný úkol

images_simple_mlp_autoencoder.ipynb

- Použijte vlastní obrázky nebo fotografie a upravte notebook tak, aby výrazně pozměnil jejich barevnost (ale jinak než v původním notebooku)
- Zvolte si vlastní trénovací obrázek a vlastní rozšíření trénovací množiny
- Je možné, že bude třeba nastavit počet neuronů ve skryté vrstvě na míru danému obrázku.
- Pošlete mi pozměněný notebook, původní a pozměněné obrázky.