Neuronové sítě 2 - Úvod do neuronových sítí

18NES2 -5. hodina, ZS 2024/25

Zuzana Petříčková

6. listopadu 2024

Neuronové sítě 2 - Úvod do neuronových sítí

1. Co jsme dělali minule
2. Ukázky úloh
   - Klasifikace do více tříd
   - Binární klasifikace
   - Regrese
3. Schopnost neuronové sítě zobecňovat
   - Early stopping
   - Regularizační techniky

Co bylo minule

- Učení perceptronové sítě - ukázka
  https://playground.tensorflow.org/
- Předzpracování dat a nastavení hyperparametrů modelu

Předzpracování dat pro MLP

Klíčové kroky předzpracování trénovacích dat:

- **Serializace**
  - Převedení vstupních i výstupních dat na 2D tenzory tvaru (vzory, číselné příznaky)
- **Je třeba se vypořádat s kategorickými proměnnými:**
  - Pokud kategorie lze uspořádat: každou kategorii převedeme na číslo a normalizujeme
  - **One-hot encoding:** Převedení kategorických proměnných na binární reprezentaci (např. kategorie "A", "B", "C" se převedou na [1, 0, 0], [0, 1, 0], [0, 0, 1]).
- **Zabezpečení konzistence dat:**
  - Zkontrolujeme, že všechny vstupní vektory jsou stejně dlouhé, žádné hodnoty nechybí.
  - Chybějící data je třeba nahradit pomocí průměru, mediánu nebo jiných metod.

Předzpracování dat pro MLP

Klíčové kroky předzpracování trénovacích dat:

- **Normalizace/Standardizace vstupů:**
  - **Normalizace**: Škálování hodnot na interval [0, 1] nebo [-1, 1] (v závislosti na zvolené přenosové funkci).
  - **Standardizace**: typicky aby data měla střední hodnotu 0 a směrodatnou odchylku 1.
  - Normalizace je velmi důležitá pro to, aby se model dobře učil
- **Trénovací množina by měla být dostatečně velká a vyvážená.**
  - Někdy je třeba přistoupit k augmentaci trénovací množiny (zvětšení počtu trénovacích vzorů)
- **Rozdělení dat na trénovací, validační a testovací sady:**
  - Obvyklé rozdělení je například 70% trénovací, 15% validační, 15% testovací sada.

Klíčové hyperparametry modelu MLP

Klíčové hyperparametry

- **Velikost modelu**: Počet skrytých vrstev a neuronů v nich
- **Přenosové (aktivační) funkce**: relu, sigmoid, tanh, softmax, linear...
- **Chybová funkce (loss function)**: MSE, binary crossentropy,...
- **Metriky**: accuracy, MSE, precission,..
- **Optimalizátor**: učící algoritmus: SGD, Adam, RMSProp,...
- **Rychlost učení (learning rate)**, popř. další parametry optimalizátoru
- **Batch size (velikost dávky)**
- **Počet epoch**
- **Inicializace vah** Typicky malé náhodné hodnoty
- **Regularizace**: L2, Dropout,...

Ukázky úloh

- klasifikace do více tříd: MNIST
- binární klasifikace: IMDB
- regresní úloha: Boston Housing data

Ukázka úlohy klasifikace do více tříd: MNIST

- 60000 obrázků ručně psaných číslic (odstíny šedé, 28x28)
- obrázky jsou vycentrované a všechny číslice mají cca. stejnou velikost
- 10000 testovacích obrázků od zcela jiných lidí
- výstupní příznak: číslo (číslice) 0,...,9 (10 tříd)
- vlastnosti dat:
  - všechny obrázky mají stejnou velikost $\rightarrow$ nemusíme jejich velikost standardizovat
  - vstupní data jsou 3D $\rightarrow$ musíme je převést na 2D
  - vstupní příznaky nabývají hodnot 0...255 $\rightarrow$ musíme je normalizovat do intervalu $[0, 1]$ nebo $[-1, 1]$

Ukázka úlohy klasifikace do více tříd: MNIST

Nastavení modelu

- softmax přenosová funkce ve výstupní vrstvě
- ReLU (nebo tanh) přenosová funkce ve skrytých vrstvách
- chybová funkce SparseCategoricalCrossentropy, metrika SparseCategoricalAccuracy (přesnost) (pokud labely jsou čísla)
- chybová funkce CategoricalCrossentropy, metrika CategoricalAccuracy (přesnost) (pokud labely jsou one-hot vektory)

Pozorování

- přesnost na testovacích datech je okolo 85%
- chyby na trénovací a validační množině jsou podobné $\rightarrow$ model dobře zobecňuje
- možná by přesnost šlo ještě zlepšit, pokud zvětšíme model, zvýšíme počet epoch, změníme učící algoritmus apod.

Ukázka úlohy binární klasifikace: IMDB

- 50000 filmových recenzí reprozentovaných pomocí slovníku slov (word index, čísla přiřazena dle četnosti v korpusu)
- každá recenze je reprezentovaná pomocí vektoru nejčastějších slov (celkem přes 88000 slov)
- 1 výstupní příznak (sentinent 0/1)
- vlastnosti dat:
  - každá recenze obsahuje jiný počet slov $\rightarrow$ data bude třeba vektorizovat tak, aby všechny vektory byly stejně dlouhé
  - trénovací data dále rozdělíme na trénovací a validační množinu (tu použijeme k sledování správnosti modelu během učení)

Ukázka úlohy binární klasifikace: IMDB

Nastavení modelu

- začneme s poměrně malým modelem, zvolíme spíše větší batch size
- sigmoidální přenosová funkce ve výstupní vrstvě
- ReLU (nebo tanh) přenosová funkce ve skrytých vrstvách
- chybová funkce BinaryCrossentropy, metrika BinaryAccuracy (přesnost)

Pozorování

- přesnost na testovacích datech je okolo 85%
- model se poměrně rychle začne přeučovat (chyba na validační množině roste)
  $\rightarrow$ zkusíme počet epoch snížit, popř. použijeme early stopping nebo nějakou z technik pro zlepšení zobecňování

Ukázka úlohy binární klasifikace: IMDB

Shrnutí

1. Pro binární klasifikaci se používá chybová funkce BinaryCrossentropy (ale lze i MSE) a metrika BinaryAccuracy. Na výstupní vrstvě volíme sigmoidální přenosovou funkci.
2. Pokud mají vstupní vektory různou délku, je třeba je upravit tak, aby měly délku stejnou.
3. Word index je nejjednodušší způsob kódování textů (ale existují i lepší přístupy TF-IDF, Word Embeddings,...).
4. K vyhodnocení toho, jak dobře se model učí a zobecňuje, použijeme validační množinu dat.

Ukázka regresní úlohy: Boston Housing data

- 13 číselných vstupních příznaků, 1 výstupní příznak (cena nemovitosti)

Vlastnosti dat:

- příznaky mají hodně odlišné rozsahy hodnot → každý příznak normalizujeme podle směrodatné odchylky
- trénovacích vzorů je málo (404) →
  1. model musí být malý, jinak hrozí přeučení
  2. pro vyladění parametrů nebude stačit testovací množina → k-násobná křížová validace

K-násobná křížová validace

- umožní nám odhadnout, jak dobře model zobecňuje, i když je trénovací množina poměrně malá
- zobecnění principu rozdělení dat na trénovací a testovací množinu

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 $K = 10$)

Monte-Carlo křížová validace

- umožní nám odhadnout, jak dobře model zobecňuje, i když je trénovací množina poměrně malá
- zobecnění principu rozdělení dat na trénovací a testovací množinu

Základní princip

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 $K = 100$)

Ukázka regresní úlohy: Boston Housing data

Nastavení modelu

- pro úlohu s málo daty budou stačit 1-2 skryté vrstvy
- lineární přenosová funkce ve výstupní vrstvě
- ReLU (nebo tanh) přenosová funkce ve skrytých vrstvách
- chybová funkce MSE, metrika MAE nebo také MSE https://keras.io/api/metrics/regression_metrics/

Pozorování

- pokud model učíme příliš dlouho nebo pokud zvolíme moc velký model, bude se přeučovat (poznáme to podle velké chyby na validačních a testovacích datech)

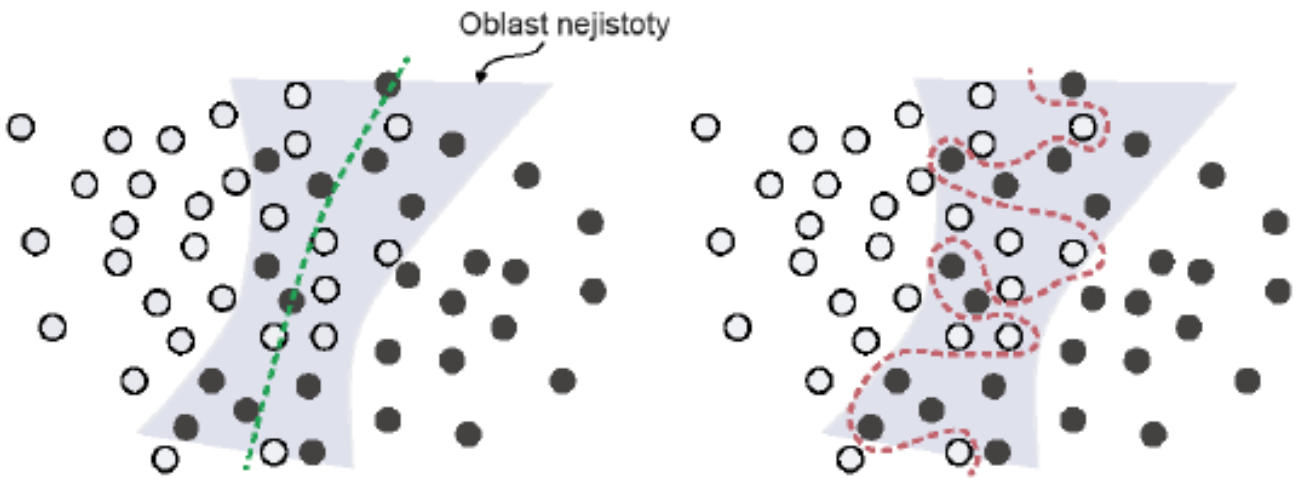
Ukázka regresní úlohy: Boston Housing data

Shrnutí

1. Pro regresi se používá chybová funkce MSE a speciální metriky (MSE, MAE,...). Na výstupní vrstvě volíme lineární přenosovou funkci.
2. Pokud mají vstupní příznaky různé rozsahy hodnot, je dobré každý z nich normalizovat podle směrodatné odchylky.
3. Čím menší trénovací množinu máme, tím menší model musíme učit, aby se nepřeučil.
4. Pokud model učíme příliš dlouho, začne se přeučovat (chyba na validační množině dat se zvyšuje).
5. Pokud máme malé množství dat, je lepší k vyhodnocení toho, jak dobře se model naučil, použít k-násobnou křížovou validaci.

Schopnost neuronové sítě zobecňovat

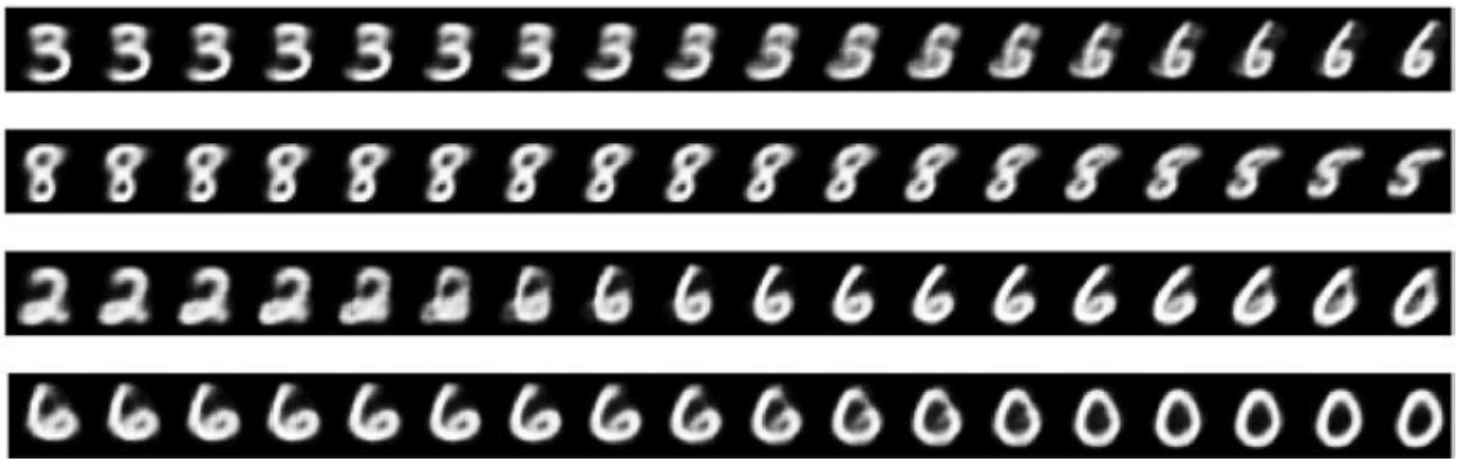
- schopnost dát správný výstup i pro data, co nebyla v trénovací množině
- Ukázka: správně naučený model vs. přeučený model:



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

Schopnost neuronové sítě zobecňovat

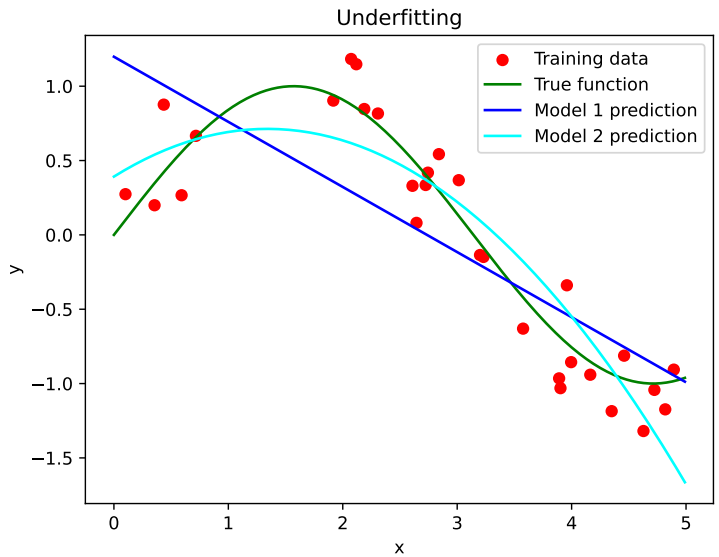
- hranice mezi jednotlivými třídami nemusí být snadno k nalezení:

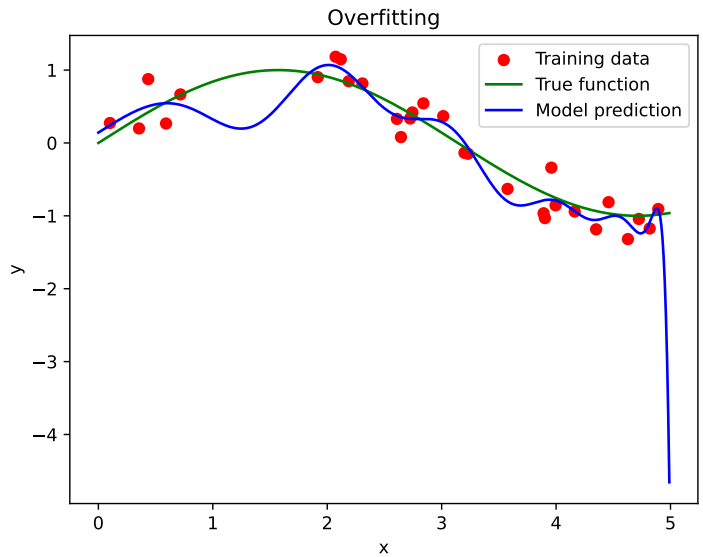


F. Chollet: Deep learning v jazyku Python, obr. 5.7

Schopnost neuronové sítě zobecňovat

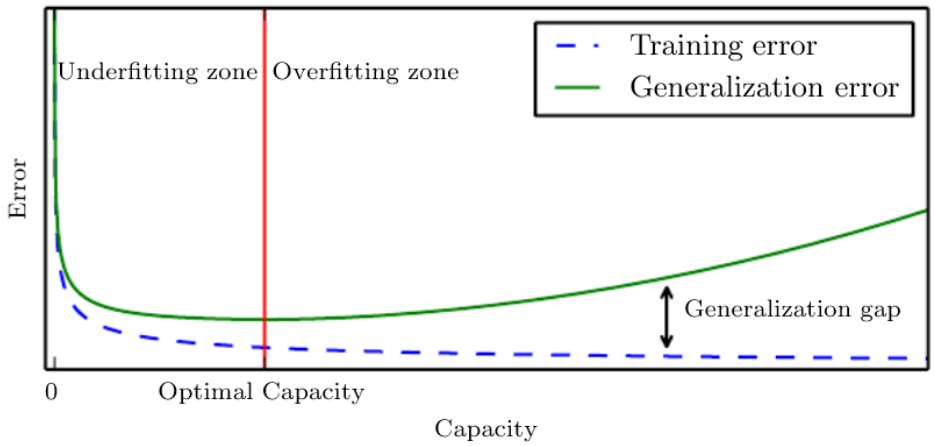
- problematika nedostatečného naučení (underfitting) nebo naopak přeučení (overfitting) v případě regresní úlohy:





Schopnost vrstevnaté neuronové sítě zobecňovat

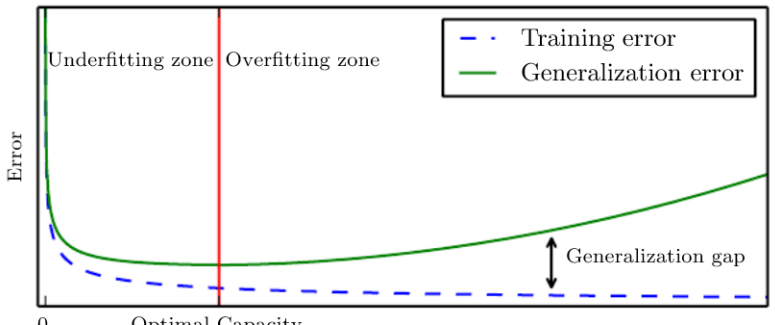
- záleží na architektuře sítě, zjednodušeně na počtu parametrů modelu, tj. **kapacitě**:
- Malá síť
  - potenciálně chybné, ale stabilní predikce (pro různé trénovací množiny a počáteční hodnoty vah)
  - hrozí **underfitting** (síť se nenaučila správně)
- Velká síť
  - větší variabilita
  - hrozí **overfitting** - síť se přeučila, špatně zobecňuje



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

Schopnost vrstevnaté neuronové sítě zobecňovat

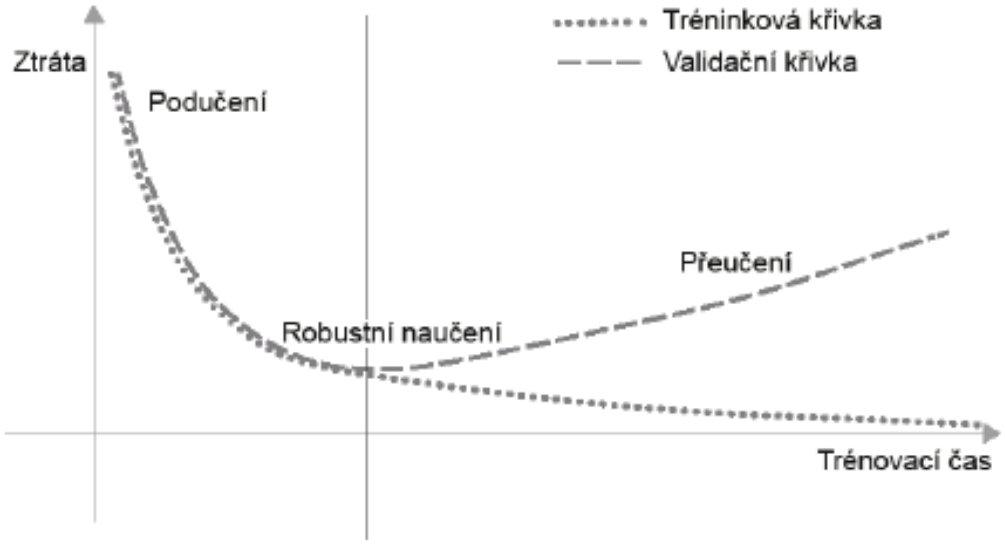
- s **kapacitou** souvisí i **potřebná velikost trénovací množiny:**
- Malá síť
  - potenciálně chybné, ale stabilní predikce (pro různé trénovací množiny a počáteční hodnoty vah)
  - hrozí **underfitting** (síť se nenaučila správně)
  - **k naučení a správnému zobecňování potřebuje méně trénovacích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



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) – pokud máme málo dat

Na co dát pozor při sestavování validační množiny dat

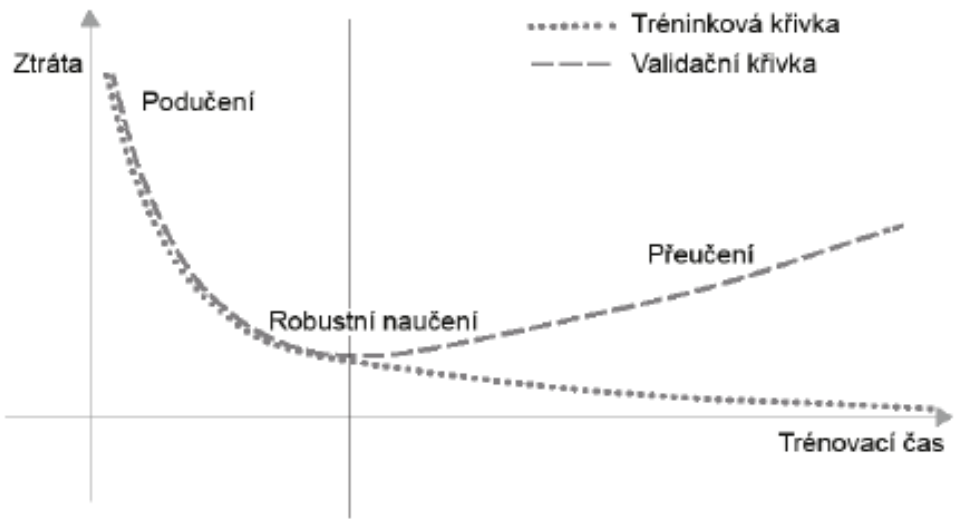
- Všechny tři množiny dat (trénovací, validační i testovací) by měly být stejně reprezentativní (např. obsahovat podobné procento vzorů z jednotlivých tříd)
- Pro časové řady: validační (a testovací) data by měla v čase následovat až za těmi trénovacími
- Pozor na redundance v datech (stejné nebo hodně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

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
- **Feature engineering** (najít lepší, informativnější, vstupní příznaky)
- **Early stopping** - včas zastavit učení za použití validační množiny
- **Zvětšit trénovací množinu** (data augmentation)
- **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

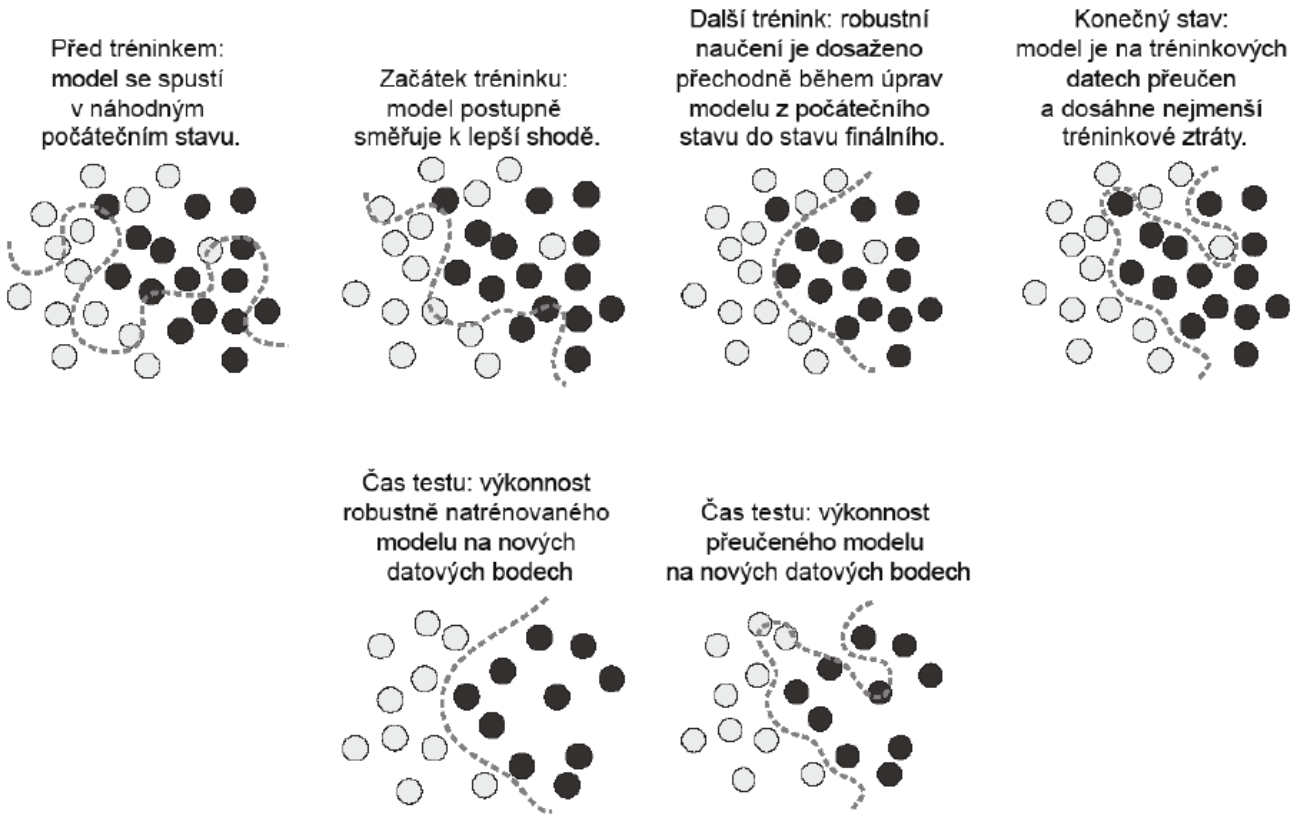
Early stopping

- 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

Jak zvětšit trénovací množinu? (data augmentation)

- pokud data nejsou vyvážená → posílíme nedostatečně zastoupené třídy
- pokud je dat málo → vygenerujeme další data

Dle typu dat

- Obrazová data: různé transformace (rotace, zrcadlení), změna jasu, kombinace obrázků, výřez z obrázku, šum (keras.ImageDataGenerator - v reálném čase)
- Textová data: synonymní náhrady, vynechávání slov,...
- Zvuková data: časový posun, změna rychlosti, přidání šumu
- Sekvenční data: časová oříznutí, náhodné vynechávání dat

Syntetické generování dat

- Přidáním šumu
- Náhodně (Markovovy procesy,...), pomocí simulací
- S využitím generativních modelů

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 prahy 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:

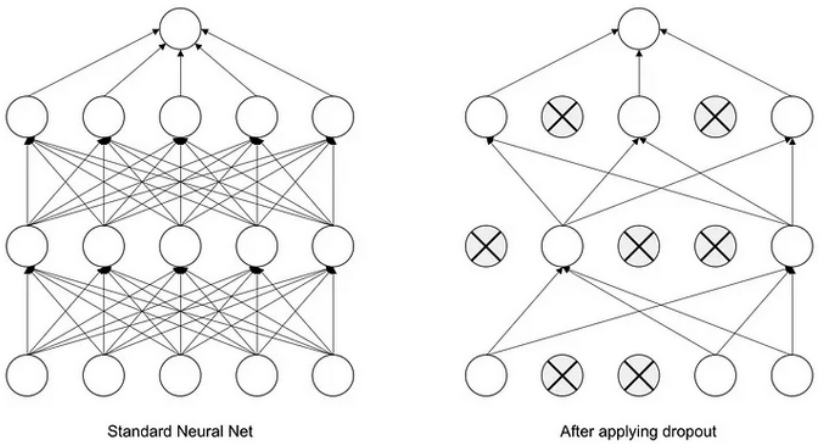
$$w_i(t+1) \quad = \quad w_i(t) - \alpha\frac{\partial E_{loss}(t)}{\partial w_i} - \alpha_r sign(w_i(t))$$

Přidání gaussovského šumu do trénovací množiny

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

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

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

Další techniky pro zlepšení zobecňování u hlubokých neuronových sítí

- Label smoothing (zašumění labelů)
- **Transfer learning (přenesené učení)**
- **Ensembling**
  -
  - **Funkcionální API v Kerasu** (ukázky)
- **Prořezávání** (pruning) ...