Neuronové sítě 2 - Úvod do neuronových sítí

18NES2 - 4. hodina, ZS 2024/25

Zuzana Petříčková

16. října 2024

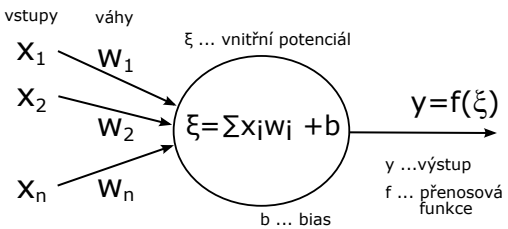
Neuronové sítě 2 - Úvod do neuronových sítí

1. Co jsme dělali minule
2. Vrstevnatá neuronová síť učená gradientní metodou - analýza modelu
   - Předzpracování dat
   - Nastavení hyperparametrů

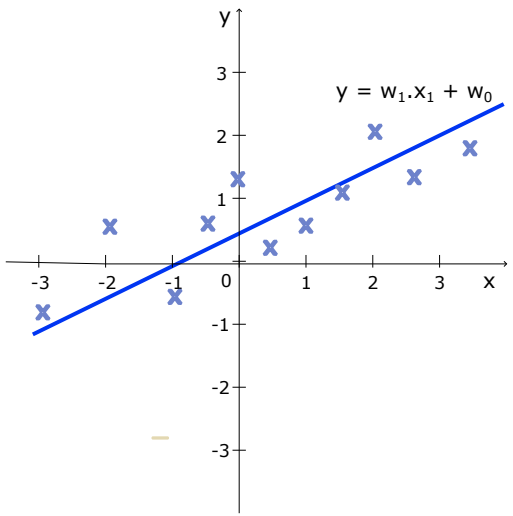
Co bylo minule

- Úvod do umělých neuronových sítí: umělý neuron a jeho interpretace, přenosové funkce
- ukázka: https://playground.tensorflow.org/

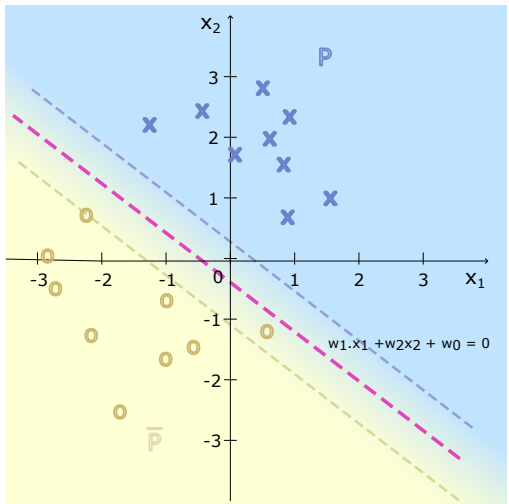
Umělý neuron



Lineární regrese



Lineární klasifikace



- vnitřní potenciál: $\xi = \sum_{i=1}^{n} w_i.x_i + b = \vec{w}\vec{x}^T + b$
- výstup: $y = f(\xi)$ (f ... přenosová / aktivační funkce)

Co bylo minule

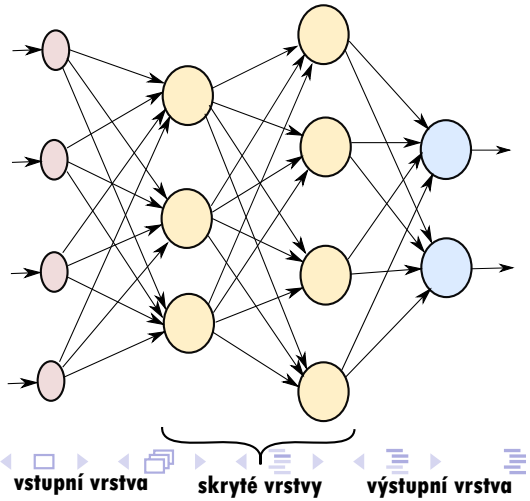
- Umělá neuronová síť (především vstevnatá = MLP) a její učení (gradientní metoda + backpropagation)

Vrstevnatá neuronová síť (multi-layer neural network, MLP):

- **sekvenční** architektura, neurony jsou uspořádány do vrstev
- **dense layers (plně propojené vrstvy)**: všechny neurony v jedné vrstvě jsou propojeny právě se všemi neurony z následující vrstvy
- speciální **vstupní vrstva** - odpovídá vstupům sítě

Výstup (odezva) modelu

- odpovídá výstupům (aktivitám) neuronů ve výstupní vrstvě
- spočte se dopředným průchodem (forward pass)



Učení vrstevnaté neuronové sítě (MLP model)

Data, na základě kterých se model učí

- trénovací množina $T = (X, D)$
  - X ... vstupní vzory: 2D tenzor tvaru (N, n), N je počet trénovacích vzorů, n je počet vstupních příznaků
  - D ... požadovaný výstup: 2D tenzor tvaru (N, m), m je počet výstupních příznaků
- trénovací vzor (training pattern) ... $(\vec{x_i}, \vec{d_i})$,
  - $\vec{x_i}$ ... vektor délky n ... vstupní vzor (input pattern)
  - $\vec{d_i}$ ... vektor délky m ... požadovaný (očekávaný) výstup (target)

Cíl učení:

- Nastavit váhy (a biasy) všech neuronů v síti tak, aby byl skutečný výstup sítě Y stejný jako požadovaný (D).

Učení vrstevnaté neuronové sítě (MLP model)

Základní princip (zjednodušeně)

1. náhodně inicializuj parametry modelu (váhy a biasy, tj. W a b)
2. opakuj trénovací cyklus:
   - připrav dávku (batch) trénovacích vzorů X a odpovídajících požadovaných výstupů D
   - spočti skutečný výstup (predikci) modelu ... pro jednu vrstvu by to bylo $Y = f(XW + \vec{b})$
   - spočti chybu modelu (jak moc se liší Y a D)
   - aktualizuj W a b tak, aby se chyba modelu o něco zmenšila

$\rightarrow$ **gradientní metoda (gradient descent)**

Hezké vizualizace hladin chybové funkce:

https://jithinjk.github.io/blog/nn_loss_visualized.md.html

https://izmailovpavel.github.io/curves_blogpost/

Gradientní metoda (metoda největšího spádu, gradient descent)

Obecné schéma algoritmu

1. Inicializuj váhy a biasy malými náhodnými reálnými hodnotami
   Inicializuj parametr učení α_0 ... $1 > \alpha_0 > 0$
2. Předlož další dávku trénovacích vzorů (X_t, D_t) a spočti pro ně skutečnou odezvu modelu Y_t a chybe E_t
3. Adaptuj všechny váhy a biasy (indexované pomocí i):

$$w_i(t+1) = w_i(t) - \alpha_t \frac{\partial E_t}{\partial w_i}$$

$$b_i(t+1) = b_i(t) - \alpha_t \frac{\partial E_t}{\partial b_i}$$

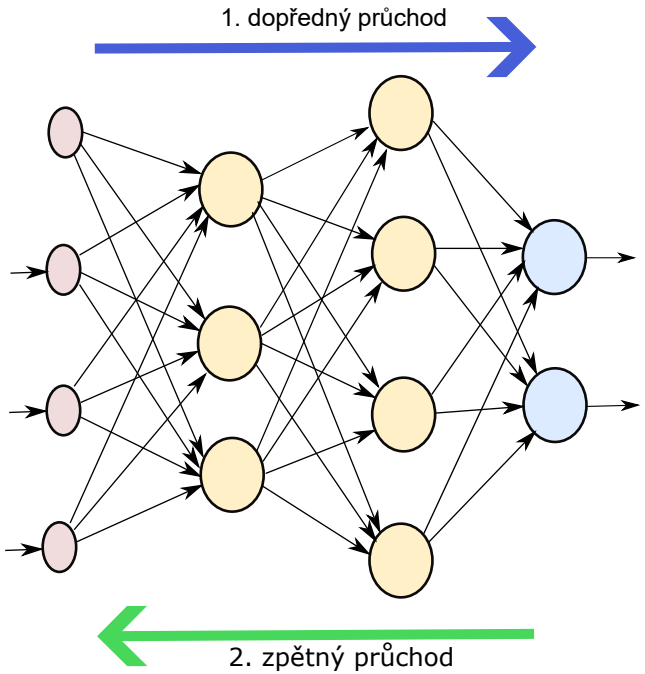
4. Případně aktualizuj parametr učení : $\alpha_t \to \alpha_{t+1}$
5. Pokud není konec, přejdi ke kroku 2.

Algoritmus zpětného šíření (Backpropagation)

(Werbos, Rumelhart, 1974-1986)

Základní princip

1. Spočteme skutečnou odezvu sítě pro daný trénovací vzor.
   - od vstupní vrstvy směrem k výstupní
2. Porovnáme skutečnou a požadovanou odezvu sítě.
3. Adaptujeme váhy a prahy:
   - proti směru gradientu chybové funkce
   - od výstupní vrstvy směrem ke vstupní



Algoritmus zpětného šíření - diskuse učící strategie

Jak předkládat trénovací vzory?

1. **iterativně po epochách (online GD)**: během jedné epochy se každý vzor předloží právě jednou, v rámci každé epochy vzory náhodně uspořádáme
   - maximální počet epoch .... kolikrát se předloží celá trénovací množina
2. **dávkově po epochách (batch GD)**:
   - celá trénovací množina se předloží najednou v jedné dávce a váhy se adaptují najednou pro celou trénovací množinu
3. **dávkově po mini-batchích (SGD, stochastic gradient descent)**
   - v každé epoše se trénovací množina náhodně rozdělí na malé podmnožiny vzorů (mini-dávka, mini-batch) a ty se iterativně předloží (v rámci mini-batche dávkově)
   - aktuálně nejpoužívanější

Algoritmus zpětného šíření - diskuse učící strategie

Kdy ukončit učení?

1. předem daný maximální počet epoch
2. jakmile přestane klesat chyba na validační množině dat: **early stopping**
3. jakmile je přírůstek vah Δw moc malý ... $|\Delta w| < \delta_{min}$
4. jakmile je průměrná chyba dostatečně malá ... $E < E_{min}$
5. časový limit

Vrstevnatá neuronová síť učená gradientní metodou - analýza modelu

Výhody:

- Jednoduchý univerzální model s poměrně dobrými aproximačními a generalizačními schopnostmi
- Univerzální aproximátor – zvládá aproximaci jakékoliv spojité funkce (pro nelineární přenos. fce stačí jedna vrstva). Ale problém učení je NP-úplný.
- Vhodný pro úlohy klasifikace i regrese.
- Schopnost zachytit komplexní nelineární vztahy.
- Využívá backpropagation pro efektivní učení gradientní metodou.
- Dobře zobecňuje

Vrstevnatá neuronová síť učená gradientní metodou - analýza modelu

Nevýhody:

- Je nutné správně nastavit a vyladit hyperparametry
- Omezení na tvar vstupních a výstupních dat
- Pomalá konvergence.
- Lokální metoda učení – může nalézt suboptimální řešení.
- Náchylnost k přeučení – pokud není dobře nastavená regularizace nebo early stopping.
- Nemá zabudované mechanismy pro zohlednění prostorové struktury dat
- Citlivost na inicializaci, trénovací data a hyperparametry.

Vrstevnatá neuronová síť učená gradientní metodou - analýza modelu

Jak zjistit, zda a jak bylo učení úspěšné?

- rychlost učení
- schopnost naučit se danou úlohu
- schopnost zobecňovat

Co je zásadní pro úspěch algoritmu zpětného šíření?

- vhodné předzpracování trénovacích dat
- vhodná inicializace vah (např. $\sim N(0, 1)$)
- nastavit správně hyperparametry pro danou úlohu

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,...
- **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,...

Velikost modelu a její vliv na výkon MLP

- **Vstupní a výstupní vrstva a počty neuronů v nich**: je dané tvarem dat
- **Velikost modelu:** počet skrytých vrstev a počet neuronů v jednotlivých vrstvách
  - Malý model má omezenou kapacitu a není schopen zachytit složitější vztahy v datech
  - Příliš malý model může mít problémy s podučením (underfitting), zatímco příliš velký model se může přeučit (overfitting).
  - Správný počet vrstev a neuronů závisí na složitosti úlohy a velikosti dat.

Velikost modelu a její vliv na výkon MLP

Praktické doporučení:

- Začni s menším modelem a postupně přidávej vrstvy/neurony podle potřeby.
- Použij validační data pro monitorování výkonu modelu a vyhnutí se přeučení.
- Pokud se model přeučuje, zvaž použití technik jako regularizace nebo dropout.

Velikost modelu a její vliv na výkon MLP

Jakou zvolit architekturu?

- **mělká (shalllow)** - model s jednou skrytou vrstvou
  - je vhodnější pro jednodušší úlohy - model se pro ně učí rychleji a lépe zobecňuje
  - vystačí si s menšími trénovacími daty (naopak velká trénovací data mu nesvědčí)
  - je snazší ho pochopit a interpretovat
- **hluboká (deep)** - model s více (nebo i mnoha) skrytými vrstvami
  - je vhodnější pro složitější úlohy s velkými trénovacími daty - učí se pro ně lépe
  - je schopna zachytit i složité vztahy mezi daty
  - vyžaduje jiné učící mechanismy a potýká se v praxi s jinými problémy než mělký model

Vrstevnatá neuronová síť - Jaké zvolit přenosové (aktivační) funkce?

Jakou přenosovou funkci použít ve výstupní vrstvě?

- regresní úloha: lineární (linear)
- klasifikace do dvou tříd: sigmoidální (sigmoid)
- klasifikace do k tříd: softmax

Jakou přenosovou funkci použít ve skrytých vrstvách?

- hyperbolický tangens (tanh) - stabilní, symetrická, ale možné problémy se saturací neuronů, oblíbená u rekurentních modelů
- v případě hlubokých sítí se často používá ReLU (pozitivně lineární) - rychlejší, efektivnější pro hluboké sítě, ale asymetrie a omezená schopnost reprezentace dat

Vrstevnatá neuronová síť - Jakou zvolit cílovou (chybovou) funkci?

Pro regresní úlohu:

- **MSE (loss='mean_squared_error')**
  - velmi vhodná pro regresní úlohy, nejčastěji používaná
  - citlivá k odlehlým hodnotám
- **MAE (loss='mean_absolute_error')** - robustnější k odlehlým hodnotám
- **Huber Loss (loss='huber')** - kombinace předchozích
- ...

Vrstevnatá neuronová síť - Jakou zvolit cílovou (chybovou) funkci?

Pro klasifikaci:

- **Binární Crossentropy (loss = 'binary_crossentropy')** - vhodná pro klasifikaci do dvou tříd společně s sigmoidální přenosovou funkcí ve výstupní vrstvě
- **Kategorická Crossentropy (loss = 'categorical_crossentropy')**
  - vhodná pro klasifikaci do více tříd ve spojení se softmax přenosovou funkcí ve výstupní vrstvě
  - pro one-hot-coded výstupy
- **Řídká Kategorická Crossentropy (loss = 'sparse_categorical_crossentropy')**
  - jako kategorická crossentropy, ale pracuje s číselnými indexy tříd místo one-hot encodingu

Vrstevnatá neuronová síť - Jakou zvolit metriku pro sledování výkonu modelu?

Klasifikace:

- **Accuracy (Přesnost)** - podíl správně klasifikovaných vzorů
  - **binární klasifikace**: **accuracy**, **binary_accuracy**
  - **klasifikace do více tříd: categorical_accuracy**, **sparse_categorical_accuracy**
- **Další metriky pro binární klasifikaci**:
  - **AUC** - sleduje plochu pod ROC křivkou, vhodné pro nevyvážená data.
  - **Precision**, **Recall**, **F1** -vhodné při potřebě minimalizovat falešně pozitivní nebo negativní.

Regrese

- **mean_squared_error** (MSE) - pro běžná data
- **mean_absolute_error** (MAE) - pro data s outliery

Algoritmy učení (optimalizátory) hlubokých neuronových sítí

- vycházejí z gradientní metody a často používají adaptivní a lokální parametr učení
  - **SGD (Stochastic Gradient Descent)** (základní algoritmus, stochastický algoritmus zpětného šíření využívající mini-batche, stabilní)
  - **Adam** (aktuálně zřejmě nejpopulárnější, adaptivní parametr učení, rychlejší učení)
  - **RMSprop** (vhodný pro sekvenční a online data)
  - AdaGrad, Adadelta, AdaMax, NAdam, FTRL,...
- každý optimalizátor má další parametry (např. SGD: **learning_rate, momentum, nesterov**)
  často můžeme zachovat defaultní nastavení

Volba vhodného parametru učení

- pro učení algoritmem SGD je zásadní správné nastavení parametru učení (learning_rate)
- parametr řídí, jak rychle se model učí

Jak volit parametr učení?

- moc malý ... pomalé učení (malé změny vah) - nebezpečí uvíznutí v suboptimálním lokálním minimu
- moc velký ... velké skoky - nebezpečí oscilací, mohu přeskočit lokální minimum chybové funkce

Co pomůže?

- nastavení optimální hodnoty parametru učení pro danou úlohu
- učení s momentem (parametry: momentum, nesterov)
- adaptivní parametr učení (Adam, RMSProp)

Klíčové hyperparametry modelu MLP

Další klíčové hyperparametry modelu MLP

- **Batch size (velikost dávky)** -počet vzorů zpracovaných v jedné dávce.
- **Počet epoch**
- **Inicializace vah** Typicky malé náhodné hodnoty
- **Techniky pro zabránění přeučení** - např. L2-regularizace, Dropout, Early stopping

Problém saturace neuronů

- Nastává v průběhu učení u klasické vrstevnaté neuronové sítě s tradičními přenosovými funkcemi typu tanh
- Jsou-li váhy a prahy moc malé, šíří se sítí příliš malá chyba a učení je moc pomalé.
- Moc velké váhy naopak vedou k tzv. **saturaci** neuronů a pomalému učení (v plochých oblastech chybové funkce)
  - Neurony jsou hodně aktivní nebo hodně pasivní pro všechny trénovací vzory a nelze je dále učit, protože derivace přenosové funkce je téměř nulová
    $\rightarrow$ **paralýza sítě** a nekontrolovaný růst vah

$\rightarrow$ **proces učení je pak zastaven v suboptimálním lokálním minimu chybové funkce**

Problém saturace neuronů

Jak snížit riziko saturace?

- relu přenosová funkce ve skrytých vrstvách namísto tanh
- normalizace trénovacích dat, batch normalization, layer normalization
- vhodná inicializace vah a biasů: He inicializace (pro ReLU) nebo Xavier (Glorot) inicializace (pro sigmoid/tanh) https://keras.io/api/layers/initializers/
- snížit parametr učení nebo použít algoritmus s adaptivním parametrem učení

MLP model - co zbývá

- techniky pro zlepšení schopnosti modelu zobecňovat
- techniky pro porovnání modelů
- ladění hyperparametrů
- vytvoření ,,nesekvenčního" modelu (např. ensemble model)