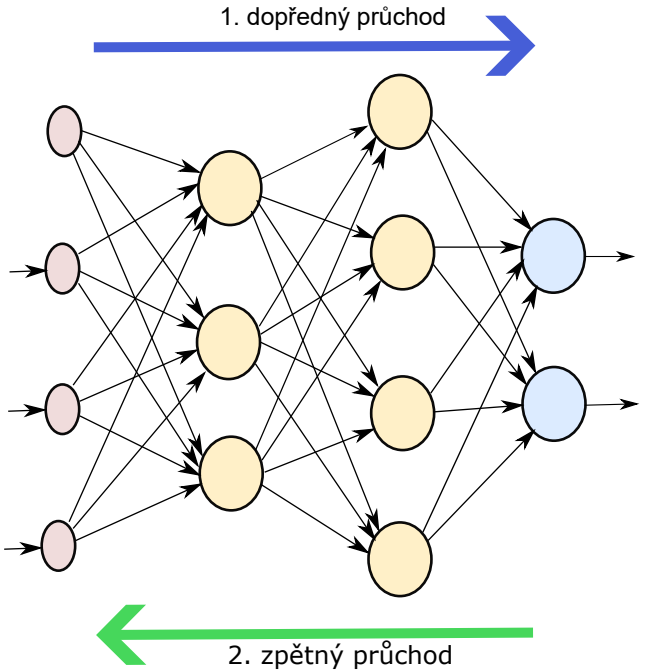
Co jsme probírali minule

Vrstevnatá neuronová síť a algoritmus zpětného šíření (Backpropagation)

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 - 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 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-batch) a ty se iterativně předloží (v rámci mini-batche dávkově)

Algoritmus zpětného šíření - diskuse učící strategie

- Online GD
  - rychlé učení, ale poměrně nestabilní (algoritmus snižuje chybu pro aktuální vzor $\rightarrow$ chyba se může z zvýšit pro ostatní vzory)
  - vyšší citlivost na odlehlé vzory a na volbu hyperparametrů, náhodnost (ale možnost úniku z lokálních minim)
- Batch GD
  - stabilnější, efektivní pro malá data
  - výpočetně a paměťově náročný pro velká data
  - vyšší citlivost na šum v datech
- Mini-batch SGD
  - spojuje výhody obou předchozích strategií
  - používá se pro velké datové sady a hluboké sítě

Architektura vrstevnaté neuronové sítě

- **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 - volba přenosové funkce

Jakou přenosovou funkci použít ve výstupní vrstvě?

- regresní úloha: lineární
- klasifikace do dvou tříd: tanh (případně logsig)
- klasifikace do k tříd: k neuronů s tanh, případně softmax

Jakou přenosovou funkci použít ve skrytých vrstvách?

- tanh - stabilní, ale možné problémy se saturací neuronů
- v případě hlubokých sítí i jiné přenosové funkce, např. ReLU (pozitivně lineární) - rychlejší, efektivnější pro hluboké sítě, ale asymetrie a omezená schopnost reprezentace dat

Vrstevnatá neuronová síť - volba cílové (chybové) funkce

MSE

- velmi vhodná pro regresní úlohy
- pro klasifikační úlohy lze použít (pro sítě bez výstupní softmax vrstvy)
- citlivá k odlehlým hodnotám

Cross-Entropy

- vhodná pro klasifikaci do více tříd ve spojení se softmax přenosovou funkcí ve výstupní vrstvě
- robustnější k odlehlým hodnotám

Binární Cross-Entropy

- varianta Cross-Entropy vhodná pro klasifikaci do dvou tříd společně s logsig přenosovou funkcí ve výstupní vrstvě

Dnešní hodina

- **Analýza modelu vrstevnaté neuronové sítě**
  - Nejprve obecně
  - Rychlost učení a aproximační schopnost
  - Schopnost zobecňovat

Vrstevnaté neuronové sítě s algoritmem zpětného šíření - analýza

- Oblíbený model neuronové sítě
- Jednoduchý algoritmus učení s vysokou variabilitou
- Poměrně dobré aproximační a generalizační schopnosti

Nevýhody

- Lokální metoda učení $\rightarrow$ nemusí najít globální minimum chybové funkce (lokální minima, přeučení)
- Pomalá konvergence
- Interní reprezentace znalostí (černá skříňka).
- Nemá zabudované mechanismy pro zohlednění prostorové struktury dat
- Citlivost na inicializaci, trénovací data a hyperparametry.

Vrstevnaté neuronové sítě s algoritmem zpětného šíření - analýza

Jak zjistit, zda a jak bylo učení úspěšné?

- rychlost učení
- schopnost aproximace (tj. schopnost naučit se danou úlohu)
- schopnost zobecňovat

Co je zásadní pro úspěch algoritmu zpětného šíření?

- normalizace vstupních příznaků. Trénovací množina by měla být dostatečně velká a vyvážená.
- vhodná inicializace vah (např. $\sim N(0, 1)$)
- nastavit správně hyperparametry pro danou úlohu: architektura (počet vrstev a počet neuronů v jednotlivých vrstvách), přenosové funkce, chybová funkce, parametr učení, ...

Vrstevnaté neuronové sítě - analýza modelu
Rychlost učení a aproximační schopnosti

- Algoritmus zpětného šíření je poměrně pomalý.
- Špatná volba počátečních parametrů ho ještě zpomalí.
- Přesto dosahuje často lepších výsledků než mnohé „rychlé algoritmy“
  (hlavně v případě, že má úloha realistickou úroveň složitosti a velikost trénovací množiny přesáhne kritickou mez)

Vrstevnaté neuronové sítě - analýza modelu
Rychlost učení a aproximační schopnosti

Jak urychlit učení a zajistit dobrou aproximaci?

1. Algoritmy, které zachovávají pevnou topologii (architekturu) sítě.
2. Současná adaptace parametrů (vah, prahů) i architektury sítě.
3. Modulární sítě - výrazně zlepšují aproximační schopnosti neuronových sítí.

Vrstevnaté neuronové sítě - analýza modelu
Rychlost učení a aproximační schopnosti

Jak urychlit učení a zajistit dobrou aproximaci při zachování pevné topologie?

- Vhodná inicializace vah a prahů
- Předzpracování a normalizace vstupních dat
- Vhodná volba parametru učení
- Použití rychlého algoritmu učení (metody druhého řádu,...)

Vhodná inicializace vah a prahů

Pravidlo

- Váhy a prahy by měly být malé, náhodné, rovnoměrně rozdělené, se střední hodnotou 0.

Proč střední hodnota 0?

- Očekávaná hodnota potenciálu každého neuronu bude 0.
- Derivace logsig / tanh je maximální pro 0 (~ 0.25) $\rightarrow$ výraznější změny vah na začátku

Proč náhodné?

- Snaha omezit symetrii. Skryté neurony by neměly mít navzájem podobnou funkci

Vhodná inicializace vah a prahů

Problém vah v průběhu učení

- 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**

Vhodná inicializace vah a prahů

Jak snížit riziko saturace neuronů?

I. Vhodná inicializace vah a prahů - např. podle nějaké heuristiky:

- Základ: $w_{ij}(0) \sim N(0,1)$
- Dobrá heuristika, např. Glorot-Bengio (2010) : $w_{ij}(0) \sim N(0, \sqrt{\frac{6}{n_i+n_j}})$ pro matici vah tvaru $n_i \times n_j$ mezi dvěma vrstvami
- Metoda Nguyen-Widrow: snaží se neurony z dané vrstvy rozhodit ve vstupním prostoru zhruba rovnoměrně

II. Normalizace vstupních dat

- Trénovací vzory by měly být v každém příznaku normalizované v intervalu $[-1,1]$ resp. $[0,1]$ (v závislosti na zvolené přenosové funkci).

Volba vhodného parametru učení

- pro učení je zásadní správné nastavení parametru učení α (learning rate)

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
- adaptivní parametr učení

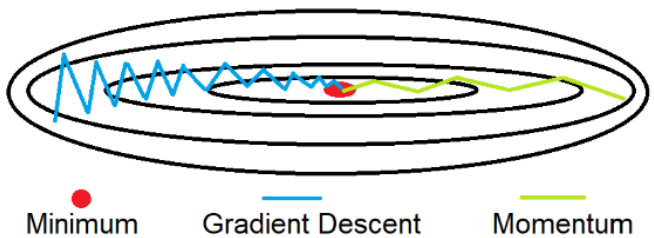
Algoritmus zpětného šíření s momentem

Problém

- V úzkých údolích chybové funkce může vést sledování gradientu k náhlým, velkým a častým oscilacím během učení

Řešení

- Zavedeme člen odpovídající **momentu** (kromě aktuální hodnoty gradientu chybové funkce bereme v úvahu i předchozí změny vah)
- Větší setrvačnost, umožňuje udržovat směr, oslabuje oscilace během učení



https://www.andreaperlato.com/aipost/gradient-descent-with-momentum/

Algoritmus zpětného šíření s momentem

Nové adaptační pravidlo

- Změna váhy hrany z neuronu i do neuronu j v čase (t+1):

$$\begin{aligned} \triangle w_{ij}(t+1) &= -\alpha \frac{\partial E_t}{\partial w_{ij}} + \alpha_m \triangle w_{ij}(t) \\ &= -\alpha \frac{\partial E}{\partial w_{ij}} + \alpha_m (w_{ij}(t) - w_{ij}(t-1)) \end{aligned}$$

- α ... parametr učení
- α_m ... moment učení

Algoritmus zpětného šíření s momentem

Moment učení

$$\triangle w_{ij}(t+1) = -\alpha \frac{\partial E_t}{\partial w_{ij}} + \alpha_m \triangle w_{ij}(t)$$

- Udržuje směr v úzkých údolích chybové funkce
- Snižuje nebezpečí ustálení v nestabilních stavech (lokální minima, sedla)
- Zvyšuje rychlost konvergence (delší úseky beze změny směru přírůstku vah)
- Moc velký α_m - moc velká setrvačnost, mohu přeběhnout minimum chybové funkce

Algoritmus zpětného šíření s momentem

Jak volit parametr učení a moment učení?

- Optimální hodnoty parametrů α a α_m a jejich optimální poměr záleží na charakteru dané úlohy
- V plochých oblastech chybové funkce (sedla):
  - Zde je gradient chybové funkce téměř nulový a to může vést k oscilacím $\rightarrow \alpha$ by měl být velký (chceme se dostat pryč)
- Ve strmých oblastech chybové funkce:
  - Zde by měl být α naopak malý, α_m pak brání oscilacím.

Řešení:

- adaptivní a lokální parametr učení

Adaptivní parametr učení

Lokální parametr učení pro každou váhu

- Změna váhy z neuronu i do neuronu j v čase (t+1):

$$\triangle w_{ij}(t+1) = -\alpha_{ij,t+1}\frac{\partial E_t}{\partial w_{ij}} + \alpha_m \triangle w_{ij}(t)$$

Adaptivní parametr učení

Heuristika:

- Parametr učení by měl klesat s rostoucím počtem epoch
- **Počáteční parametr učení** $0 << \alpha < 1$
  - Umožňuje přeskočit mělká lokální minima
  - Rychlý vývoj vah sítě
- **Závěrečný parametr učení** $\alpha \sim 0$
  - Zabraňuje oscilacím
  - Neměl by klesat moc rychle, stačí:
    $\sum_{t=0}^{\infty} \alpha_t = \infty$,
    $\sum_{t=0}^{\infty} \alpha_t^2 < \infty$

Resilent propagation (Rprop) - Silva & Almeida

Algoritmus založený na změně znaménka parciální derivace:

- Inicializuj $\alpha_{ij,0}$ malými náhodnými hodnotami
- Zrychluj učení, pokud se za poslední dvě po sobě jdoucí iterace nezměnilo znaménko parciální derivace $\frac{\partial E}{\partial w_{ij}}$
- Zpomaluj, pokud se znaménko změnilo

Více variant:

- základní (Silva & Almeida)
- Super SAB
- Rprop+
- ...

Resilent propagation (Rprop) - Silva & Almeida

Adaptace parametru učení v čase (t+1)

- $\alpha_{ij,t+1} = u \cdot \alpha_{ij,t}$, jestliže $\frac{\partial E_t}{\partial w_{ij}} \cdot \frac{\partial E_{t-1}}{\partial w_{ij}} > 0$
- $\alpha_{ij,t+1} = d \cdot \alpha_{ij,t}$, jestliže $\frac{\partial E_t}{\partial w_{ij}} \cdot \frac{\partial E_{t-1}}{\partial w_{ij}} < 0$
- Konstanty u, d jsou pevně zvolené tak, že $u > 1$, $d < 1$

Problémy

- Parametr učení roste i klesá exponenciálně vzhledem k u a d
  $\rightarrow$ Problémy mohou nastat, jestliže po sobě následuje mnoho urychlovacích kroků.

Resilent propagation (Rprop) - Super SAB

Algoritmus

- Nastav všechny α_{ij}^0 na počáteční hodnotu α_{start}.
- Proveď krok (t) algoritmu zpětného šíření s momentem.
- Pokud $\frac{\partial E_t}{\partial w_{ij}} \cdot \frac{\partial E_{t-1}}{\partial w_{ij}} > 0$: $\alpha_{ij,t+1} = u \cdot \alpha_{ij,t}$.
- Pokud $\frac{\partial E_t}{\partial w_{ij}} \cdot \frac{\partial E_{t-1}}{\partial w_{ij}} < 0$:
  - Anuluj předchozí změnu vah: $w_{ij}(t+1) = w_{it}(t) - \triangle w_{ij}(t)$
  - Zmenši parametr učení: $\alpha_{ij,t+1} = d \cdot \alpha_{ij,t}$

Vlastnosti

- Řádově rychlejší než standardní algoritmus zpětného šíření
- Poměrně stabilní
- Robustní k volbě počátečních parametrů

Resilent propagation - Rprop+

- Parametr učení není řízený znaménkem parciální derivace chybové funkce, ale změnou velikosti chyby
- Rychlejší než Super SAB

Algoritmus

- Nastav všechny α_{ij}^0 na počáteční hodnotu α_{start}.
- Proveď krok (t) algoritmu zpětného šíření s momentem.
- Pokud $E_t < E_{t-1}$: $\alpha_{ij,t+1} = u \cdot \alpha_{ij,t}$.
- Pokud $E_t > c \cdot E_{t-1}$:
  - Anuluj předchozí změnu vah: $w_{ij}(t+1) = w_{it}(t) - \triangle w_{ij}(t)$
  - Zmenši parametr učení: $\alpha_{ij,t+1} = d \cdot \alpha_{ij,t}$
- Konstanty c, u, d jsou pevně zvolené tak, že $c > 1$, $u > 1$, $d < 1$

Další finty pro zlepšení učení

Výpočet lze pustit opakovaně

- pro různé počáteční váhy, vyberu síť s nejmenší výslednou chybou

Pokud síť nekonverguje nebo konverguje velmi pomalu

- zkusit více neuronů ve vrstvách, popř. více vrstev

Častější předkládání důležitých vzorů

- snížení chyby pro důležité vzory

Další finty pro zlepšení učení

'Žíhání sítě' = přidání náhodného šumu

- při ustálení vah a prahů s vysokou chybou
- Adaptace váhy hrany z neuronu i do neuronu j podle:

$$w_{ij}(t+1) = w_{ij}(t) + N(0, \epsilon)$$

- Parametr ϵ je třeba volit opatrně:
  - moc malé ϵ ... vrátí se zpět
  - moc velké ϵ ... musí se znova dlouho adaptovat

Matlab a vrstevnaté neuronové sítě

Vytvoření modelu:

- regresní úloha ... *fitnet*
- klasifikační úloha ... *patternnet*
- predikce časové řady ... *narxnet*
- obecný model ... ... *feedforwardnet*

net = fitnet([5,5],'traingd');

- první parametr: počty neuronů ve skrutých vrstvách
- druhý parametr: algoritmus učení (zde základní algoritmus zpětného šíření)

Naučení modelu:

[net,tr] = train(net,x,t)

Aplikace modelu:

y = net(x);

Matlab a různé metody učení vrstevnaté neuronové sítě

Algoritmus zpětného šíření a jeho modifikace I.

- *traingd* ... algoritmus zpětného šíření (BP)
- *traingdm* ... BP s momentem
- *traingda* ... BP s adaptivním parametrem učení
- *traingdx* ... BP s adaptivním parametrem učení a momentem

Malá odbočka: algoritmy učení hlubokých neuronových sítí

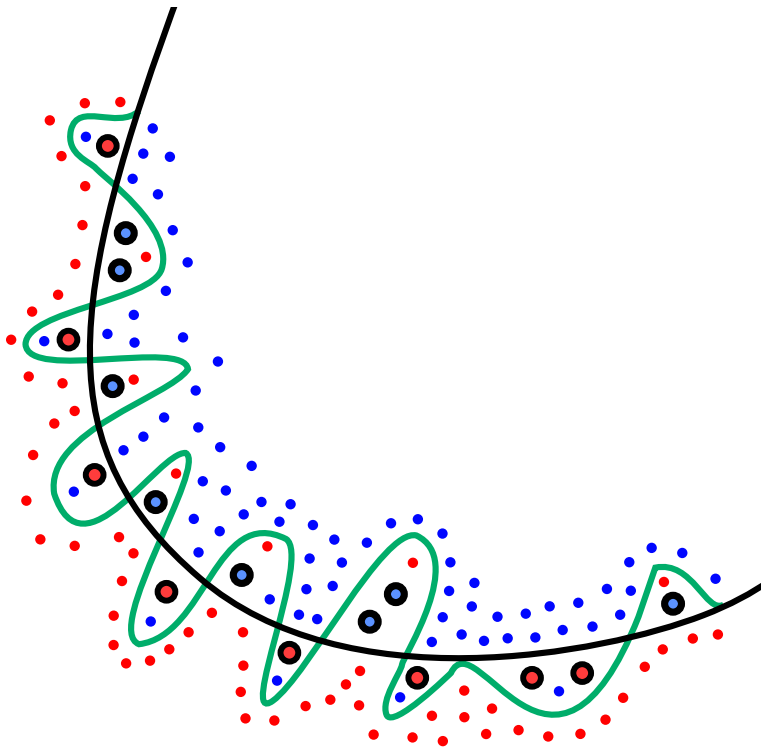
- vycházejí z GD a často používají adaptivní a lokální parametr učení
  - Gradient Descent (základní algoritmus zpětného šíření)
  - SGD (stochastický algoritmus zpětného šíření využívající mini-batche)
  - AdaGrad
  - RMSprop
  - Adadelta
  - Adam (aktuálně zřejmě nejpopulárnější)
  - AdaMax
  - NAdam
  - FTRL ...

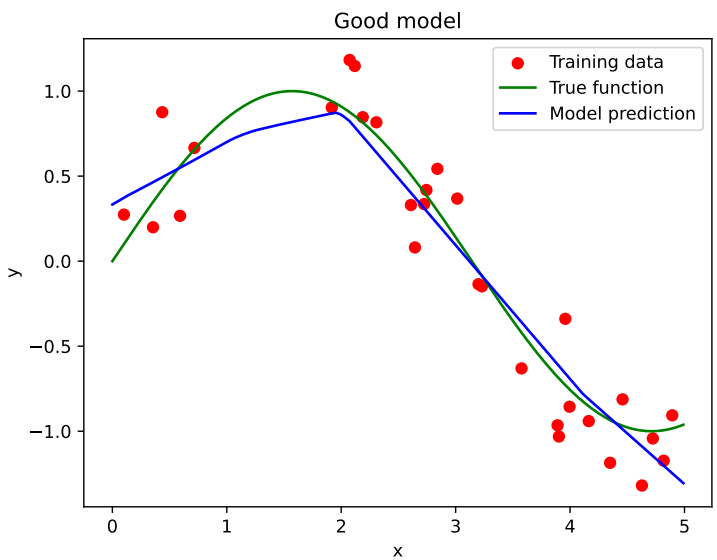
Dnešní hodina

- **Analýza modelu vrstevnaté neuronové sítě**
  - Nejprve obecně
  - Rychlost učení a aproximační schopnost
  - **Schopnost zobecňovat**

Schopnost vrstevnaté neuronové sítě zobecňovat

- schopnost dát správný výstup i pro data, co nebyla v trénovací množině

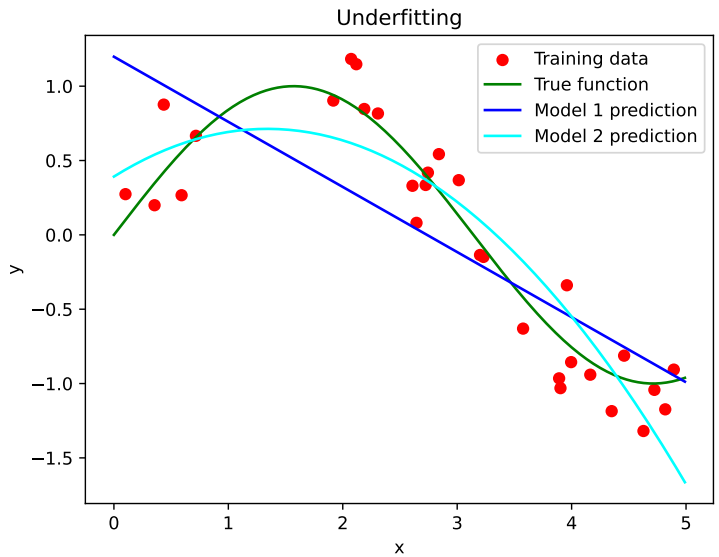


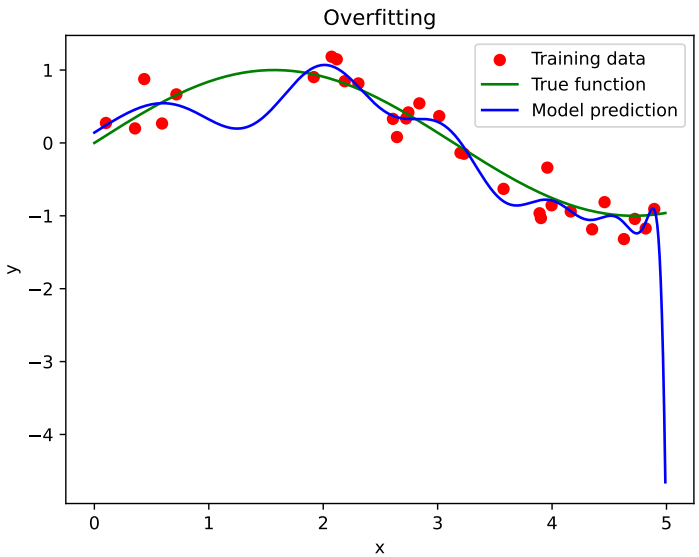


https://upload.wikimedia.org/wikipedia/commons/1/19/Overfitting.svg

Schopnost vrstevnaté neuronové sítě zobecňovat

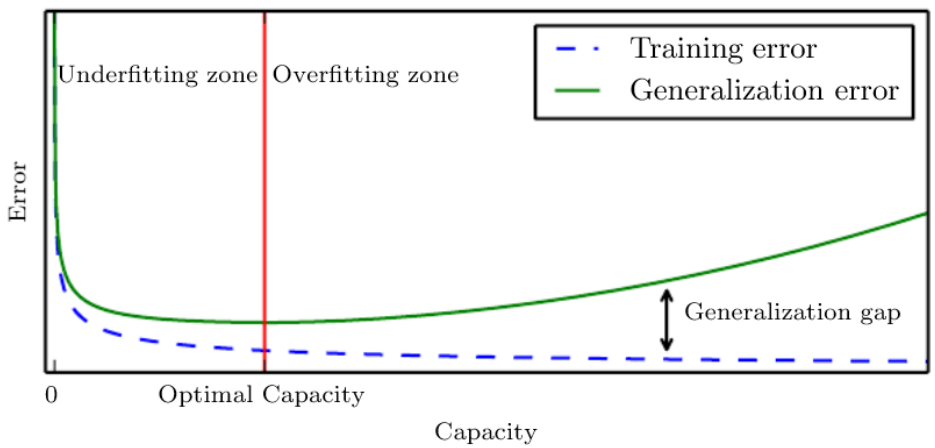
- problematika nedostatečného naučení (underfitting) nebo naopak přeučení (overfitting):





Schopnost vrstevnaté neuronové sítě zobecňovat

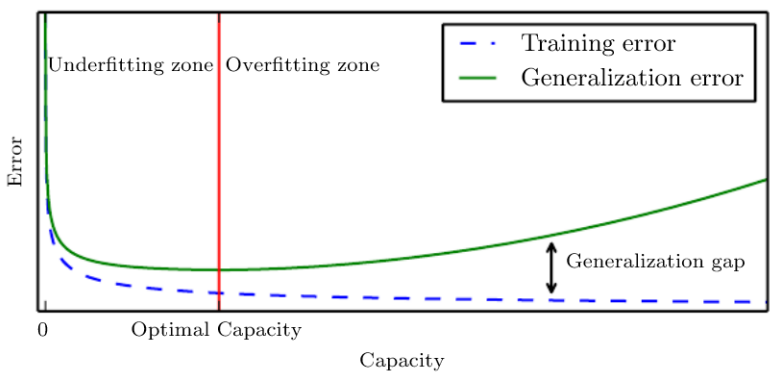
- záleží na architektuře sítě, zjednodušeně na počtu parametrů modelu, tj. **kapacitě**:
- Malá síť
  - potenciálně chybné, ale stabilní predikce (pro různé trénovací množiny a počáteční hodnoty vah)
  - hrozí **underfitting** (síť se nenaučila správně)
- Velká síť
  - větší variabilita
  - hrozí **overfitting**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



Schopnost vrstevnaté neuronové sítě zobecňovat

Věta - vztah mezi kapacitou a potřebným počtem trénovacích vzorů

- Pro síť s 1 skrytou vrstvou, počtem vah 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 alespoň 90 % je třeba vybrat alespoň 10 $\cdot$W vzorů

Jak zajistit, aby vrstevnatá neuronová síť dobře zobecňovala?

- Zvětšit trénovací množinu (data augmentation)
- Najít "optimální" architekturu pro danou trénovací množinu
- Samplovací techniky
  - Early stopping - včas zastavit učení za použití validační množiny (už jsme probírali).
  - Využití krosvalidace při ladění hyperparametrů
    - K-násobná křížová validace
    - Monte-Carlo křížová validace
    - ...
- Regularizační techniky
- Dropout
- Učení s nápovědou
- Prořezávání
- ...

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 $K = 100$)

Regularizační techniky

Základní princip

- Přidávají k základní chybové funkci (např. MSE) další penalizační členy:
  $E = c_{mse} E_{mse} + c_A E_A + c_B E_B + ...$
- **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:

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

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

- Adaptace vah podle:

$$w_{ij}(t+1) \quad = \quad w_{ij}(t) - \alpha\frac{\partial E_t}{\partial w_{ij}} - \alpha_r w_{ij}(t)$$

- 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:

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

- Adaptace vah podle:

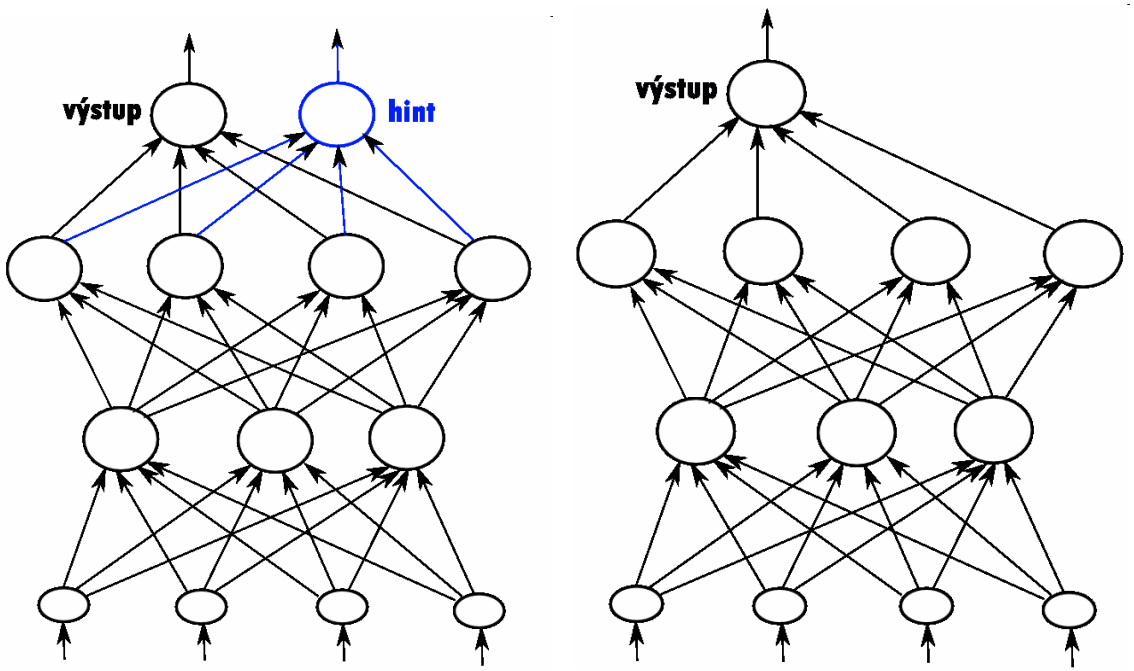
$$w_i(t+1) \quad = \quad w_i(t) - \alpha\frac{\partial E_t}{\partial w_i} - \alpha_r sign(w_i(t))$$

Přidání gaussovského šumu do trénovací množiny

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

Učení s nápovědou

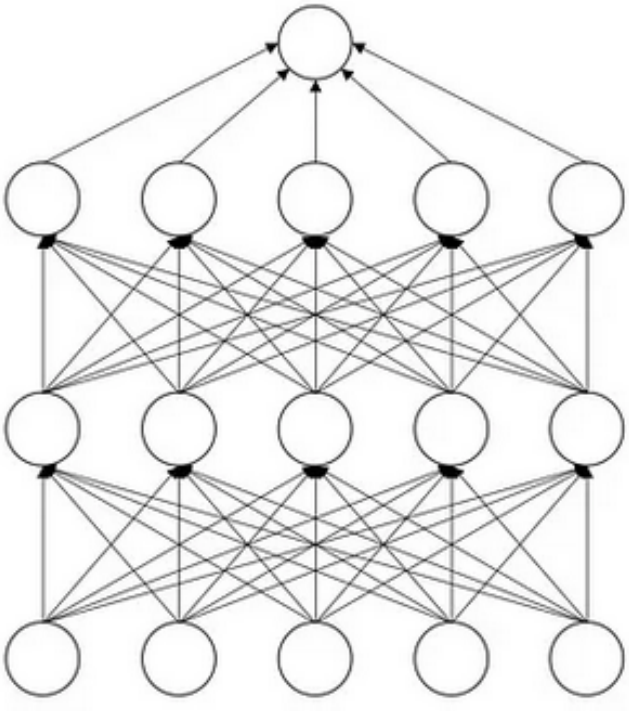
(Mostafa,1993; Suddarth,1990)



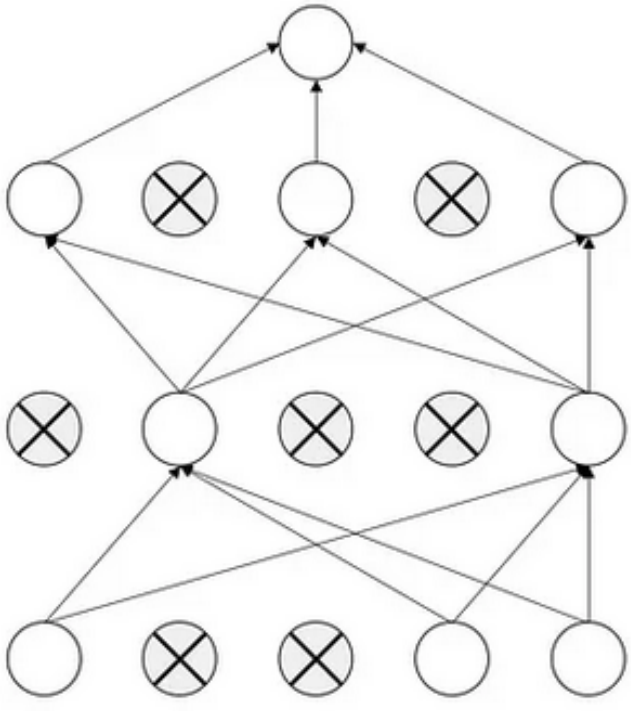
- Zvyšuje schopnost sítě zobecňovat a zrychluje učení.
- Vede k hladší funkci sítě, podporuje prořezávání.

Dropout (Srivastava et al., 2014)

- spočívá v náhodném vypínání (deaktivování) některých skrytých neuronů během učení
- při testování jsou všechny neurony aktivované



Standard Neural Net



After applying dropout

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

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 $S_{ij} = \frac{\partial y_j}{\partial y_i}$ ...

Odbočka: Matlab a mělká neuronová síť - implementované techniky pro zlepšení zobecňování

Early stopping a rozdělení dat mezi trénovací, validační a testovací množinu

- lze nastavit *net.divideFcn* a jí příslušné *net.divideParam*
- *dividerand* ... náhodně (*trainRatio*, *valRatio*, *testRatio*)
- *divideint* ... bez přeuspořádání (*trainRatio*, *valRatio*, *testRatio*)
- *divideind* ... indexy zadá uživatel (*trainInd*, *valInd*, *testInd*)

Regularizační techniky - Weight decay

- *net.performParam.regularization* = *0.5* ... váha chybového členu
- *trainbr* ... automatická regularizace

Malá odbočka: zlepšení zobecňování u hlubokých neuronových sítí

- používají se další techniky pro vylepšení chopnosti modelu zobecňovat:
  - Early stopping
  - L2-regularizace, řídká regularizace
  - Dropout, DropConnect
  - různé normalizace (Batch normalization, Weight standardization, Layer normalization)
  - Label smoothing - vyhlazení požadovaných výstupů
  - Data augmentation (MixUp - kombinace obrázků, Cutout - výřez z obrázku,...) ...

Dnešní hodina

- **Analýza modelu vrstevnaté neuronové sítě**
  - Nejprve obecně
  - Rychlost učení a aproximační schopnost
  - Schopnost zobecňovat
  - **Rychlost učení a aproximační schopnost** - doplnění

Další strategie pro urychlení učení

Algoritmy druhého řádu

- vhodné pro učení ,,mělkých" neuronových sítí
- berou v úvahu více informací o tvaru chybové funkce: nejen gradient, ale i zakřivení
- Kvadratická aproximace chybové funkce
  - $\vec{w}$ ... vektor všech vah a prahů sítě (délky n)
  - $E(\vec{w})$ ... chybová funkce
- Taylorův rozvoj:

$$E(\vec{w}+\vec{h}) \approx E(\vec{w}) + \bigtriangledown E(\vec{w})^T \vec{h} + \frac{1}{2}\vec{h}^T \bigtriangledown^2 E(\vec{w})\vec{h}$$

Algoritmy druhého řádu

Taylorův rozvoj

$$E(\vec{w}+\vec{h}) \approx E(\vec{w}) + \bigtriangledown E(\vec{w})^T \vec{h} + \frac{1}{2}\vec{h}^T \bigtriangledown^2 E(\vec{w})\vec{h}$$

- $\vec{w}$ ... vektor všech vah a prahů sítě (délky n)
- $\vec{h}$ ... změna vektoru vah
- $E(\vec{w})$ ... chybová funkce
- $\bigtriangledown E(\vec{w})$ je vektor parciálních derivací... $(\frac{\partial E}{\partial w_i})_{i=1}^n$
- $\bigtriangledown^2 E(\vec{w})$ je Hessovská matie $(n \times n)$ parciálních derivací druhého řádu ... $(\frac{\partial^2 E}{\partial w_i \partial w_j})_{i,j=1}^n$

Algoritmy druhého řádu

Gradient chybové funkce

- Taylorův rozvoj:
  $E(\vec{w}+\vec{h}) \approx E(\vec{w}) + \bigtriangledown E(\vec{w})^T \vec{h} + \frac{1}{2}\vec{h}^T \bigtriangledown^2 E(\vec{w})\vec{h}$
- Zderivujeme $E(\vec{w}+\vec{h})$ podle $\vec{h}$:
  $\bigtriangledown E(\vec{w}+\vec{h}) \approx \bigtriangledown E(\vec{w})^T + \vec{h}^T \bigtriangledown^2 E(\vec{w}) = 0$
- Z rovnice vyjádříme $\vec{h}$:

$$\vec{h} = -\left(\bigtriangledown^2 E(\vec{w})\right)^{-1} \bigtriangledown E(\vec{w})$$

Newtonovská metoda

- Adaptace vah (a prahů) iterativně podle:

$$\vec{w}(t+1) = \vec{w}(t) - \left(\bigtriangledown^2 E(\vec{w})\right)^{-1} \bigtriangledown E(\vec{w})$$

- Rychlá konvergence, ale problémem může být výpočet inverzní Hessovské matice

Algoritmy druhého řádu

Pseudonewtonovské metody

- Pracují se zjednodušenou aproximací Hessovské matice
- Např. mohu brát v úvahu jen prvky na hlavní diagonále (ostatní prohlásím za nulové): $(\frac{\partial^2 E}{\partial w_i^2})$
- Adaptace vah (a prahů) iterativně podle:

$$w_i(t+1) = w_i(t) - \left(\frac{\partial^2 E}{\partial w_i^2}\right)^{-1} \frac{\partial E}{\partial w_i}$$

- Odpadá výpočet inverzní Hessovské matice
- Nemusí fungovat dobře, pokud se chybová funkce hodně liší od kvadratické

Algoritmy druhého řádu

Pseudonewtonovské metody -Quickprop

- nepočítá přímo ani prvky Hessovské matice na diagonále
- používá diskrétní aproximaci parciální derivace druhého řádu

$$\frac{\partial^2 E_t}{\partial w_i^2} \approx \frac{\frac{\partial E_t}{\partial w_i} - \frac{\partial E_{t-1}}{\partial w_i}}{w_i(t) - w_i(t-1)}$$

- Adaptace vah (a prahů) iterativně podle:

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

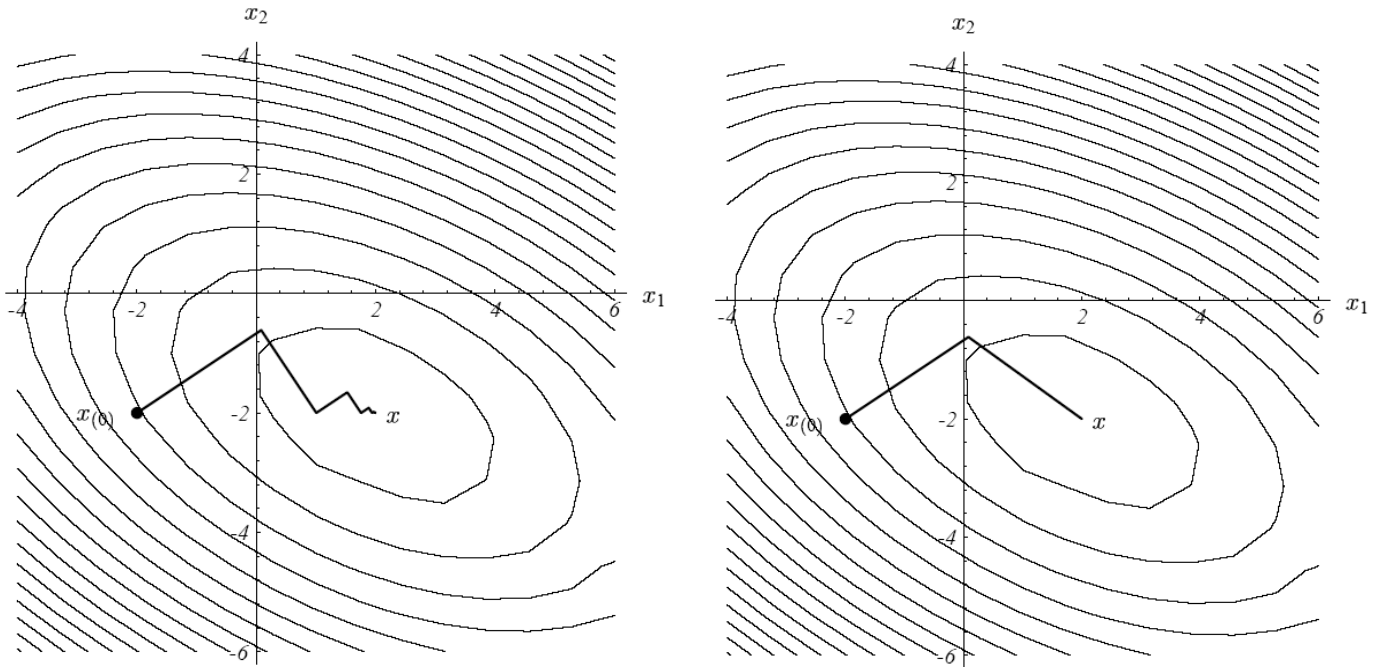
- tzv ,,sekantový krok"

Algoritmy druhého řádu

Pseudonewtonovské metod - Levenberg-Marquardtův algritmus

- velmi oblíbený algoritmus
- rychlejší a přesnější v oblasti minima chybové funkce
- kombinace gradientní a Newtonovy metody
- pro jeden výstup y: $\frac{\partial^2 E}{\partial w_i \partial w_j} \approx \frac{\partial y}{\partial w_i} \cdot \frac{\partial y}{\partial w_j}$

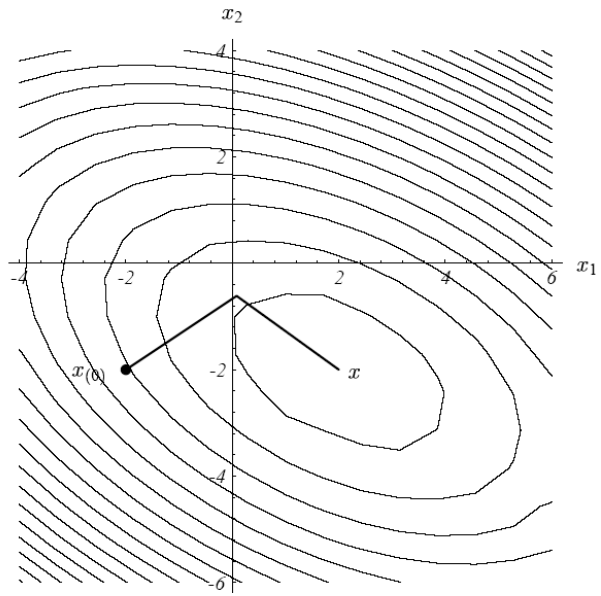
Metody konjugovaných gradientů



Super-rychlé metody, robustní k volbě počátečních parametrů

Metody konjugovaných gradientů

- Adaptace vah (a prahů) iterativně podle: $\vec{w}(t+1) = \vec{w}(t) + \alpha_t \vec{g}_t$
  - α_t ... délka kroku (parametr učení).
  - $\vec{g}_t$ ... směr kroku.
- Vektory $\vec{g}_t$ jsou spočteny tak, aby byly navzájem konjugované dáno Hessovská matice $\bigtriangledown^2 E(\vec{w})$: $\vec{g}_{t1}^T \cdot \bigtriangledown^2 E(\vec{w}) \cdot \vec{g}_{t2} = 0$



Metody druhého řádu - shrnutí

- vhodné pro ,,mělké" neuronové sítě
- typicky navržené pro dávkovou strategii učení
- mohou řádově urychlit učení (ve smyslu počtu epoch), ale roste výpočetní náročnost jedné iterace
- pro hluboké sítě a velké datové sady se zatím příliš nepoužívají (vysoká prostorová a časová náročnost)

Relaxační metody

Základní metoda

- V každém kroku se aktualizuje jedna náhodně zvolená váha w_i
- Počítá diskrétní aproximaci gradientu: $\triangle \vec{w}_i = -\alpha \frac{E(\vec{w}+\vec{\beta}) - E(\vec{w})}{\beta_i}$
- $\vec{\beta}$ ... k váze w_i je přičtena malá náhodná perturbace

Rychlejší metoda

- V každém kroku se perturbuje výstup jednoho (i-tého) neuronu y_i o hodnotu $\triangle y_i$
- Požadovaný nový potenciál neuronu i bude: $\sum_k w_{ki,t+1} y_k = f^{-1}(y_i + \triangle y_i)$
- Adaptace váhy w_{ji} podle: $w_{ji}(t+1) = w_{ji}(t) \frac{f^{-1}(y_i + \triangle y_i)}{\sum_k w_{ki,t} y_k}$

Matlab a různé metody učení vrstevnaté neuronové sítě

Algoritmus zpětného šíření a jeho modifikace I.

- *traingd* ... algoritmus zpětného šíření (BP)
- *traingdm* ... BP s momentem
- *traingda* ... BP s adaptivním parametrem učení
- *traingdx* ... BP s adaptivním parametrem učení a momentem

Pseudonewtonovské metody

- *trainlm* ... Levenberg-Marquardtův algoritmus
- *trainoss* ... Quickprop
- *trainbfg* ... BFG

Matlab a různé metody učení vrstevnaté neuronové sítě

Algoritmus zpětného šíření a jeho modifikace I. Metody konjugovaných gradientů

- *trainscg* ... Moeller ...Metoda škálovaných konjugovaných gradientů
- *traincgf* ... Fletcher-Reeves
- *traincgp* ... Polak-Ribiere
- *traincgb* ... Powell-Beale

Relaxační metoda

- *trainrp* ... Resilent method