Neuronové sítě

Doc. RNDr. Iveta Mrázová, CSc.

Katedra teoretické informatiky
Matematicko-fyzikální fakulta
Univerzity Karlovy v Praze

Neuronové sítě

– Vrstevnaté neuronové sítě –

Doc. RNDr. Iveta Mrázová, CSc.

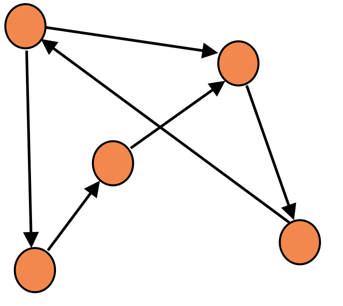
Katedra teoretické informatiky
Matematicko-fyzikální fakulta
Univerzity Karlovy v Praze

Vrstevnaté neuronové sítě (1)

D: Neuronová síť je uspořádaná 6-tice $M=(N,C,I,O,w,t)$, kde:

- N je konečná neprázdná množina neuronů,
- $C \subseteq N \times N$ je neprázdná množina orientovaných spojů mezi neurony
- $I \subseteq N$ je neprázdná množina vstupních neuronů
- $O \subseteq N$ je neprázdná množina výstupních neuronů
- $w: C \rightarrow R$ je váhová funkce
- $t: N \rightarrow R$ je prahová funkce

(R označuje množinu reálných čísel)



Vrstevnaté neuronové sítě (2)

D: **Vrstevnatá síť (BP-síť)** $\boldsymbol{B}$ je neuronová síť s orientovaným acyklickým grafem spojů. Její množina neuronů je tvořena posloupností $\boldsymbol{l+2}$ vzájemně disjunktních podmnožin zvaných vrstvy.

- První vrstva zvaná **vstupní vrstva** je množinou všech vstupních neuronů $\boldsymbol{B}$, tyto neurony nemají v grafu spojů žádné předchůdce; jejich vstupní hodnota $\boldsymbol{x}$ je rovna jejich výstupní hodnotě.
- Poslední vrstva zvaná **výstupní vrstva** je množinou všech výstupních neuronů $\boldsymbol{B}$; tyto neurony nemají v grafu spojů žádné následníky.
- Všechny ostatní neurony zvané skryté neurony jsou obsaženy ve zbylých $\boldsymbol{l}$ vrstvách zvaných **skryté vrstvy**.

Algoritmus zpětného šíření (1)

Cíl: najít takovou matici vah, která by zaručovala pro všechny vstupní vzory z trénovací množiny to, že skutečný výstup neuronové sítě bude stejný jako její požadovaný výstup

Přitom není specifikována ani skutečná, ani požadovaná aktivita skrytých neuronů.

- Pro konečnou množinu trénovacích vzorů lze celkovou chybu vyjádřit pomocí rozdílu mezi skutečným a požadovaným výstupem sítě u každého předloženého vzoru.

Algoritmus zpětného šíření (2)
Chybová funkce

- vyjadřuje odchylku mezi skutečnou a požadovanou odezvou sítě:

požadovaná odezva

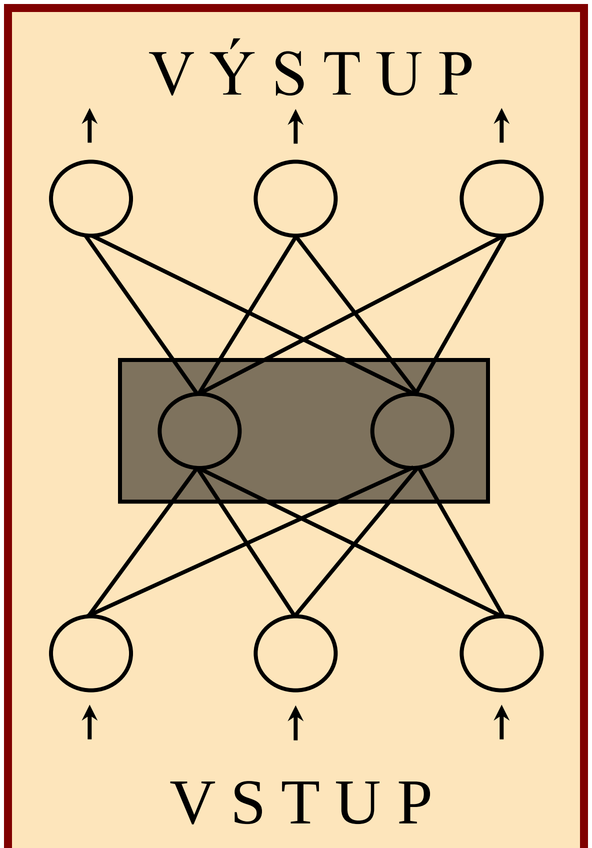
$$E = \frac{1}{2} \sum_{p} \sum_{j} \left(y_{j,p} - d_{j,p} \right)^2$$

vzory výstupní neurony skutečná odezva

- cílem procesu učení je minimalizovat tuto odchylku na dané trénovací množině

⟹ **algoritmus zpětného šíření** (Back-Propagation)

Vrstevnaté neuronové sítě (BP-sítě)



- výpočet skutečné odezvy pro daný vzor
- porovnání skutečné a požadované odezvy
- adaptace vah a prahů
  - proti gradientu chybové funkce
  - od výstupní vrstvy směrem ke vstupní

BP-sítě: adaptační pravidla (1)

Aktualizace synaptických vah proti směru gradientu:

$$w_{ij}(t+1) = w_{ij}(t) + \Delta_E w_{ij}(t)$$

$\Delta_E w_{ij}(t)$ .... přírůstek váhy $\boldsymbol{w_{ij}}$ přispívající k minimalizaci $\boldsymbol{E}$

chyba na výstupu sítě

$$\Delta_E w_{ij} = -\frac{\partial E}{\partial w_{ij}} = -\frac{\partial E}{\partial y_j}\frac{\partial y_j}{\partial \xi_j}\frac{\partial \xi_j}{\partial w_{ij}}$$

potenciál neuronu *j*

skutečný výstup

váha spoje

BP-sítě: adaptační pravidla (2)

Aktualizace synaptických vah pro výstupní vrstvu:

$$\Delta_E w_{ij} \cong -\frac{\partial E}{\partial w_{ij}} = -\frac{\partial E}{\partial y_j}\frac{\partial y_j}{\partial \xi_j}\frac{\partial \xi_j}{\partial w_{ij}} =$$

$$= -\frac{\partial E}{\partial y_j}\frac{\partial y_j}{\partial \xi_j}\frac{\partial}{\partial w_{ij}}\sum_{i'} w_{i'j}\, y_{i'} =$$

$$= -\frac{\partial E}{\partial y_j}\frac{\partial y_j}{\partial \xi_j}\, y_i = -\frac{\partial E}{\partial y_j} f'(\xi_j)\, y_i =$$

$$= -(y_j - d_j)\, f'(\xi_j)\, y_i = \delta_j\, y_i$$

BP-sítě: adaptační pravidla (3)

Aktualizace synaptických vah pro skryté vrstvy:

$$\Delta_E w_{ij} \cong -\frac{\partial E}{\partial w_{ij}} = -\left(\sum_k \frac{\partial E}{\partial \xi_k}\frac{\partial \xi_k}{\partial y_j}\right)\frac{\partial y_j}{\partial \xi_j}\, y_i =$$

$$= -\left(\sum_k \frac{\partial E}{\partial \xi_k}\frac{\partial}{\partial y_j}\sum_{j'} w_{j'k}\, y_{j'}\right)\frac{\partial y_j}{\partial \xi_j}\, y_i =$$

$$= -\left(\sum_k \frac{\partial E}{\partial \xi_k} w_{jk}\right)\frac{\partial y_j}{\partial \xi_j}\, y_i =$$

$$= \left(\sum_k \delta_k w_{jk}\right) f'(\xi_j)\, y_i = \delta_j\, y_i$$

BP-sítě: adaptační pravidla (4)

Výpočet derivace sigmoidální přenosové funkce podle:

$$f'(\xi_j) = \lambda y_j (1 - y_j)$$

Aktualizace vah podle:

$$w_{ij}(t+1) = w_{ij}(t) + \alpha \delta_j y_i + \alpha_m \left(w_{ij}(t) - w_{ij}(t-1) \right)$$

- kde

$$\delta_j = \begin{cases} (d_j - y_j) \lambda y_j (1 - y_j) & \text{pro výstupní neuron} \\ \left(\sum_k \delta_k w_{jk} \right) \lambda y_j (1 - y_j) & \text{pro skrytý neuron} \end{cases}$$

Algoritmus zpětného šíření (1)

Krok 1: Zvolte náhodné hodnoty synaptických vah

Krok 2: Předložte nový trénovací vzor ve tvaru:

[vstup $\vec{x}$, požadovaný výstup $\vec{d}$]

Krok 3: Vypočtěte skutečný výstup

aktivita neuronů v každé vrstvě je dána pomocí:

$$y_j = f\left(\xi_j\right) = \frac{1}{1+e^{-\lambda \xi_j}} \quad , \text{ kde } \; \xi_j = \sum_i y_i w_{ij}$$

Takto vyjádřené aktivity pak tvoří vstup následující vrstvy.

Algoritmus zpětného šíření (2)

Krok 4: Aktualizace vah

Při úpravě synaptických vah postupujte směrem od výstupní vrstvy ke vstupní. Váhy se adaptují podle:

$$w_{ij}(t+1) = w_{ij}(t) + \alpha\delta_j y_i + \alpha_m\left(w_{ij}(t) - w_{ij}(t-1)\right)$$

$$\delta_j = \begin{cases} (d_j - y_j)\,\lambda\, y_j (1 - y_j) & \text{pro výstupní neuron} \\ \left(\sum_k \delta_k w_{jk}\right)\lambda\, y_j (1 - y_j) & \text{pro skrytý neuron} \end{cases}$$

$w_{ij}(t)$ ………. váha z neuronu i do neuronu j v čase t

α , α_m ……..... parametr, resp. moment učení ($0 \leq \alpha , \alpha_m \leq 1$)

ξ_j , resp. δ_j …. potenciál, resp. chyba na neuronu j

k ……………. index pro neurony z vrstvy nad neuronem j

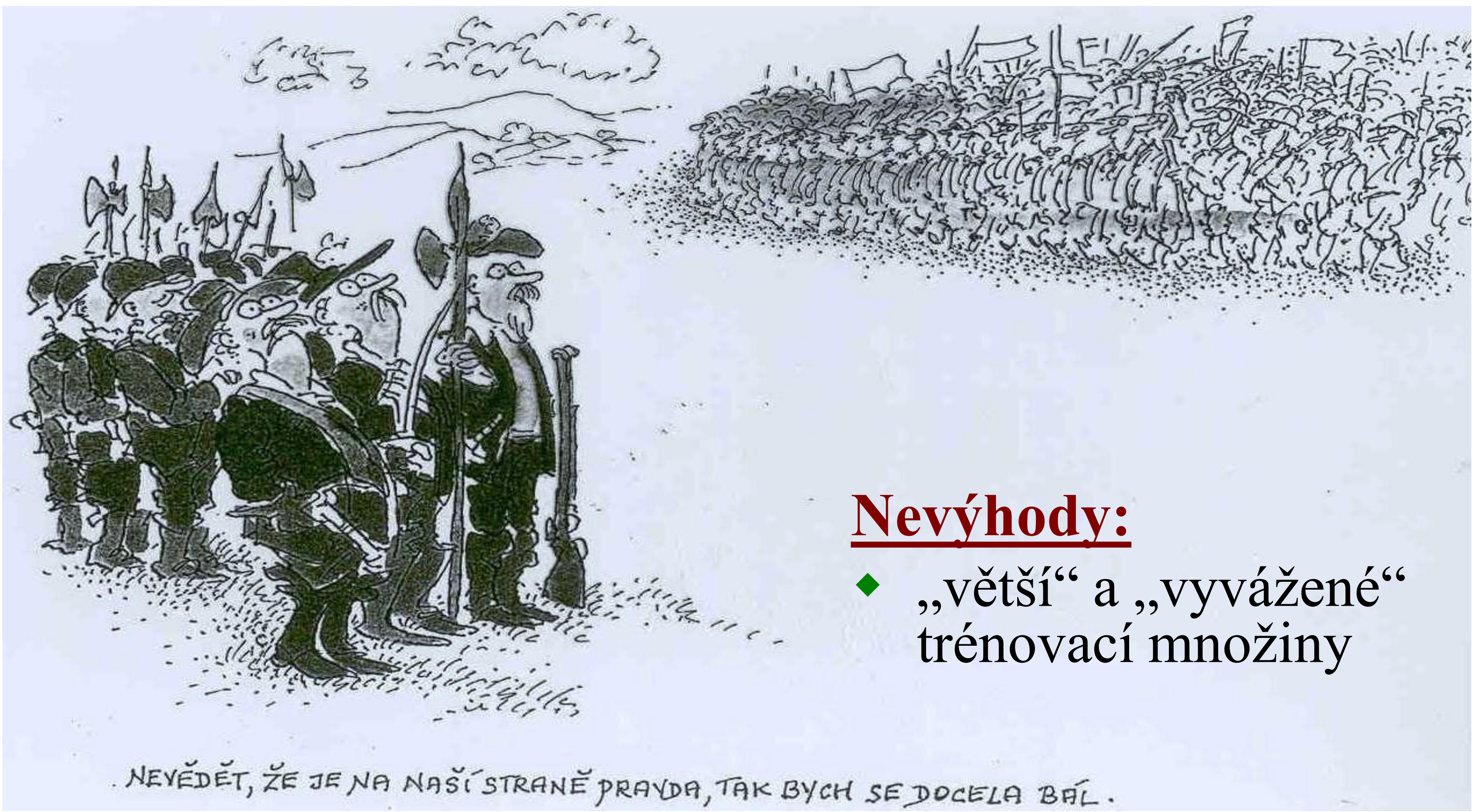
λ ……………. strmost přenosové funkce

Krok 5: Přejdi ke Kroku 2

BP-sítě: analýza modelu

- Jeden z nejpoužívanějších modelů
- Jednoduchý algoritmus učení
- Poměrně dobré výsledky
- **Nevýhody:**
  - interní reprezentace znalostí - „černá skříňka“
  - chybová funkce (znalost požadovaných výstupů)
    - „větší“ a „vyvážené“ trénovací množiny
    - kontrola výstupů sítě při rozpoznávání
  - počet neuronů a generalizační schopnosti sítě
    - prořezávání a doučování

BP-sítě: analýza modelu



Nevýhody:

- „větší“ a „vyvážené“ trénovací množiny

Algoritmus zpětného šíření a urychlení učení (1)

- **Standardní algoritmus zpětného šíření je poměrně pomalý**
  → špatná volba počátečních parametrů ho může ještě zpomalit
- **Problém učení umělých neuronových sítí je obecně NP-úplný**
  → výpočetní náročnost roste exponenciálně s počtem proměnných
  → přesto dosahuje standardní algoritmus zpětného šíření často lepších výsledků než mnohé „rychlé algoritmy“
    - hlavně v případě, že má úloha realistickou úroveň složitosti a velikost trénovací množiny přesáhne kritickou mez

Algoritmus zpětného šíření a urychlení učení (2)

Algoritmy s cílem urychlit proces učení:

- **Zachovávající pevnou topologii sítě**
- **Modulární sítě**
  - Výrazně zlepšují aproximační schopnosti neuronových sítí
- **Adaptace parametrů** (vah, prahů apod.) **i topologie sítě**

Algoritmus zpětného šíření: volba počátečních vah (1)

- **Váhy by měly být rovnoměrně rozdělené na intervalu** $< -\alpha_m , \alpha_m >$
- **Nulová střední hodnota**
  - vede na očekávanou nulovou hodnotu celkového vstupu každého neuronu sítě (potenciálu)
- **Maximální hodnota derivace sigmoidální přenosové funkce pro nulu** ***(~ 0.25)***
  - Větší hodnoty šířené chyby
  - Výraznější změny vah na začátku učení

Algoritmus zpětného šíření: volba počátečních vah (2)

Problém:

- **Příliš malé váhy „paralyzují učení“**
  - Chyba šířená z výstupní vrstvy směrem do skrytých vrstev je příliš malá
- **Příliš velké váhy vedou k „saturaci“ neuronů a pomalému učení** (v plochých zónách chybové funkce

→ **Proces učení je pak ukončen v suboptimálním lokálním minimu**

× **Správná volba počátečních vah může riziko uvíznutí v lokálním minimu výrazně snížit**

Algoritmus zpětného šíření: volba počátečních vah (3)

Omezení lokálních minim:

~ inicializace malými náhodnými hodnotami

Motivace:

- **Malé hodnoty vah**
  - Velké absolutní hodnoty vah způsobují saturaci skrytých neuronů (hodně aktivní nebo naopak hodně pasivní pro všechny trénovací vzory) → nelze je dále učit (derivace přenosové funkce – sigmoidy – je téměř nulová)
- **Náhodné hodnoty vah**
  - Cílem je „omezit symetrii“ → skryté neurony by neměly mít navzájem podobnou funkci

Algoritmus zpětného šíření: volba počátečních vah (4)

IDEA:

- **Potenciál skrytého neuronu je dán pomocí:**

$$\xi = w_0 + w_1 x_1 + \dots + w_n x_n$$

x_i … aktivita *i*-tého neuronu z předchozí vrstvy

w_i …váha od *i*-tého neuronu z předchozí vrstvy

- **Střední hodnota potenciálu skrytého neuronu:**

$$E\{\xi_j\} = E\left\{\sum_{i=0}^{n} w_{ij} x_i\right\} = \sum_{i=0}^{n} E\{w_{ij}\} E\{x_i\} = 0$$

  - váhy jsou nezávislé na vzorech
  - váhy jsou náhodné proměnné se střední hodnotou 0

Algoritmus zpětného šíření: volba počátečních vah (5)

IDEA - pokračování:

- **Rozptyl potenciálu ξ je dán pomocí:**

$$\sigma_\xi^2 = E\left\{\left(\xi_j\right)^2\right\} - \underbrace{E^2\left\{\left(\xi_j\right)\right\}}_{=0} = E\left\{\left(\sum_{i=0}^{n} w_{ij}\; x_i\right)^2\right\} - 0 =$$

$$= \sum_{i,k=0}^{n} E\left\{\left(w_{ij} w_{kj}\; x_i\; x_k\right)\right\} =$$

vzájemná nezávislost pro všechna $\boldsymbol{j}$

$$= \sum_{i=0}^{n} E\left\{\left(w_{ij}\right)^2\right\} E\left\{\left(x_i\right)^2\right\}$$

Algoritmus zpětného šíření: volba počátečních vah (5)

IDEA - pokračování:

- **Další předpoklad:** trénovací vzory jsou normalizované a leží v intervalu ***< 0 , 1 >*** . Potom:

$$E\left\{(x_i)^2\right\} = \int_0^1 x_i^2 \, d\, x = \left.\frac{x^3}{3}\right|_0^1 = \frac{1}{3}$$

- jsou-li váhy skrytých neuronů také náhodné proměnné se střední hodnotou 0 rovnoměrně rozložené v intervalu *[-a,a]*. Potom:

$$E\left\{(w_{ij})^2\right\} = \int_{-a}^{a} w_{ij}^2 \cdot \frac{1}{2a} \, d\, w_{ij} = \left.\frac{w_{ij}^3}{3 \cdot 2a}\right|_{-a}^{a} = \frac{a^2}{3}$$

- ***N*** … počet vah vedoucích k danému neuronu

Algoritmus zpětného šíření: volba počátečních vah (6)

IDEA - pokračování:

- **Směrodatná odchylka tedy bude odpovídat:**

$$A = \sigma_\xi = \frac{a}{3}\sqrt{N} \qquad \left(\rightarrow \; a = A\frac{3}{\sqrt{N}} \right)$$

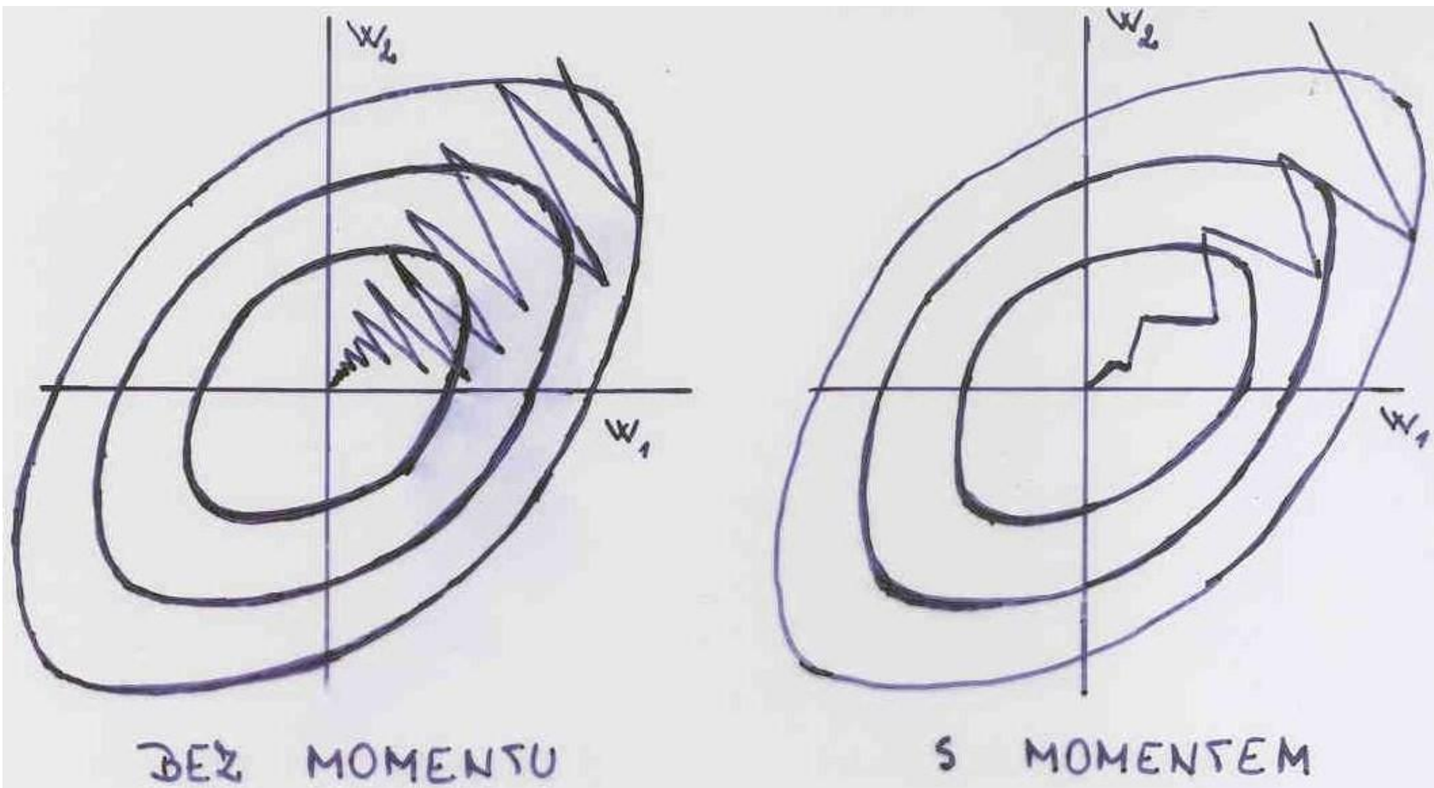
- **Potenciál neuronu by měla být náhodná proměnná se směrodatnou odchylkou** A (navíc nezávislou na počtu vah vedoucích do tohoto neuronu);
- **Počáteční váhy by měly být voleny (zhruba) z intervalu:**

$$\left[-\frac{3}{\sqrt{N}} \cdot A \, , \; \frac{3}{\sqrt{N}} \cdot A \right]$$

- speciálně pro $A = 1$ velký gradient (tj. rychlé učení)

Algoritmus zpětného šíření s momentem (1)

Minimalizace chybové funkce pomocí gradientní metody



Algoritmus zpětného šíření s momentem (2)

- Pokud je pro danou úlohu minimum chybové funkce v „úzkém údolí," může vést sledování gradientu k náhlým (častým a velkým) oscilacím během učení
- **Řešení:** **zavést člen odpovídající momentu**
  - Kromě aktuální hodnoty gradientu chybové funkce je třeba vzít v úvahu i předchozí změny parametrů (vah)

    → **Setrvačnost ~ měla by pomoci zamezit extrémním oscilacím v „úzkých údolích chybové funkce"**

Algoritmus zpětného šíření s momentem (3)

- Pro síť s n různými vahami $w_1, \ldots, w_n$ je změna váhy w_k v kroku $i+1$ dána vztahem

$$\Delta w_k(i+1) = -\alpha \frac{\partial E}{\partial w_k(i)} + \alpha_m \Delta w_k(i) =$$

$$= -\alpha \frac{\partial E}{\partial w_k(i)} + \alpha_m \left(w_k(i) - w_k(i-1) \right)$$

kde: α ….. parametr učení

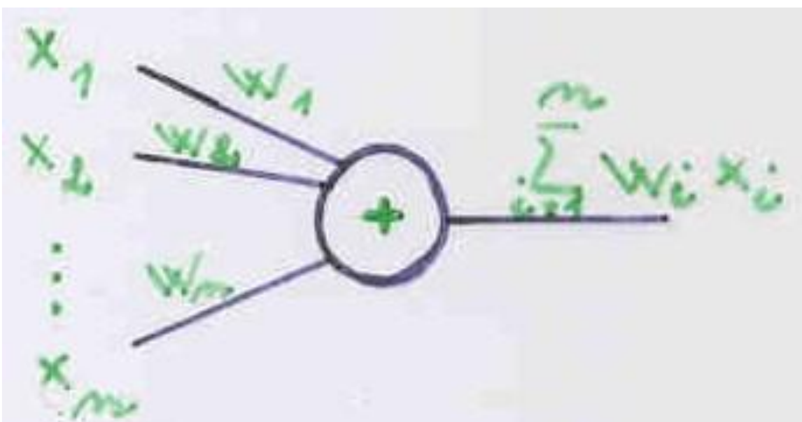
α_m … moment učení

Algoritmus zpětného šíření s momentem (4)

- Chceme-li urychlit konvergenci k minimu chybové funkce:
  - Zvětšit parametr učení až k jisté optimální hodnotě α , která by však stále ještě vedla ke konvergenci v procesu učení
  - Zavedení momentu učení umožňuje oslabit oscilace v procesu učení
- Optimální hodnoty α a α_m závisí na charakteru dané úlohy

Algoritmus zpětného šíření s momentem (5)

PŘÍKLAD: Lineární přenosová funkce, ***p*** vzorů



X....... matice $(p \times n)$; $\vec{d}$....vektor

$$X = \begin{pmatrix} x_1^{(1)} & \dots & x_n^{(1)} \\ \vdots & \ddots & \vdots \\ x_1^{(p)} & \dots & x_n^{(p)} \end{pmatrix} \quad ; \quad \vec{d} = \begin{pmatrix} d^{(1)} \\ \vdots \\ d^{(p)} \end{pmatrix}$$

$\rightarrow$ MINIMALIZACE ***E*** :

$$E = \left\| X\vec{w} - \vec{d} \right\|^2 = \left(X\vec{w} - \vec{d} \right)^T \left(X\vec{w} - \vec{d} \right) = \vec{w}^T \left(X^T X \right) \vec{w} - 2\vec{d}^T X\vec{w} + \vec{d}^T \vec{d}$$

Algoritmus zpětného šíření s momentem (6)

- ***E*** je kvadratická funkce → pomocí gradientní metody lze nalézt minimum

  Interpretace: ***E*** představuje paraboloid v ***n*** – rozměrném prostoru; jeho tvar je určen vlastními čísly korelační matice $\boldsymbol{X^T X}$

→ Gradientní metoda dává nejlepší výsledky pro „nejdůležitější“ osy „stejné délky“ (principal axes)

→ Pokud se „délky“ os hodně liší, vede gradientní metoda k oscilacím

Algoritmus zpětného šíření s momentem (7)

- Oscilace pak lze omezit malým α a větším momentem učení α_m
  - × **příliš malé hodnoty** α

    → **nebezpečí lokálních minim**
  - × **příliš velké hodnoty** α

    → **nebezpečí oscilací**

Algoritmus zpětného šíření s momentem (8)

V nelineárním případě je pro oblasti vzdálené od lokálního minima gradient chybové funkce téměř nulový – možnost výskytu oscilací

→ v takovém případě může pomoci větší parametr učení
  → návrat ke „konvexním“ oblastem chybové funkce

Řešení:

- **Adaptivní parametr učení**
- **Předzpracování trénovací množiny**
  - dekorelace vstupních vzorů (PCA, …)

Algoritmus zpětného šíření: strategie pro urychlení procesu učení (1)

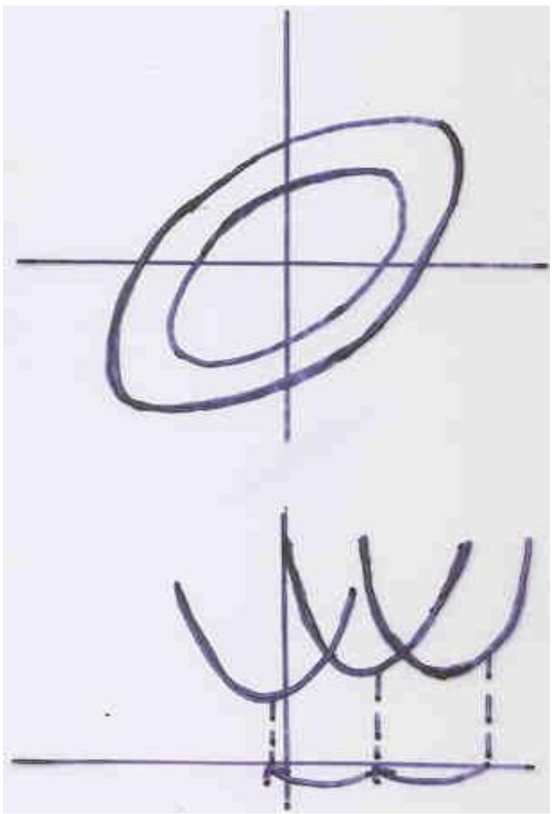
1. **Adaptivní parametr učení:**

   lokální parametr učení α_i pro každou váhu w_i

   adaptace vah podle: $$\Delta w_i = -\alpha_i \frac{\partial E}{\partial w_i}$$

- **Varianty algoritmů:**
  - Silva & Almeida
  - Delta-bar-delta
  - Super SAB

Algoritmus Silvy & Almeidy (1)



Předpoklad: síť má ***n*** vah

- Kvadratická chybová funkce:

$$c_1^2 w_1^2 + c_1^2 w_1^2 + \ldots + c_1^2 w_1^2 + \sum_{i \neq j} d_{ij} w_i w_j + C$$

- Krok v ***i*** –tém směru minimalizuje

$$c_i^2 w_i^2 + k_1 w_i + k_2$$

$\boldsymbol{k_1}$, $\boldsymbol{k_2}$ jsou konstanty, které závisí na hodnotě „zmrazených“ proměnných v daném iteračním bodě ($\boldsymbol{c_i}$ udává zakřivení paraboly)

Algoritmus Silvy & Almeidy (2)

<u>Heuristika:</u>

- URYCHLUJ, pokud se za poslední dvě po sobě jdoucí iterace nezměnilo znaménko parciální derivace
- ZPOMALUJ, pokud se znaménko změnilo

$\Delta_i E^{(k)}$ … parciální derivace chybové funkce podle váhy w_i v k –té iteraci

$\alpha_i^{(0)}$ … počáteční hodnota parametru učení $(i = 1, \ldots, n)$
inicializace malými náhodnými hodnotami

Algoritmus Silvy & Almeidy (3)

- V ***k*** –té iteraci se hodnota parametru učení aktualizuje pro další krok (pro každou váhu) podle:

$$\alpha_i^{(k+1)} = \begin{cases} \alpha_i^{(k)} u \;, & \text{jestliže} \quad \nabla_i E^{(k)} \cdot \nabla_i E^{(k-1)} > 0 \\ \alpha_i^{(k)} d \;, & \text{jestliže} \quad \nabla_i E^{(k)} \cdot \nabla_i E^{(k-1)} < 0 \end{cases}$$

- Konstanty ***u*** a ***d*** jsou pevně zvolené tak, že ***u* > *1*** a ***d* < *1***
- Adaptace vah podle: $\Delta^{(k)} w_i \;=\; -\alpha_i^{(k)} \nabla_i E^{(k)}$

Algoritmus Silvy & Almeidy (4)

Problémy:

- Parametr učení roste i klesá exponenciálně vzhledem k ***u*** a ***d***

→ **Problémy mohou nastat, jestliže po sobě následuje mnoho urychlovacích kroků**

Algoritmus Delta-bar-delta

- Větší důraz na urychlování (především z malých počátečních vah)
- ***k*** –tá iterace: $\alpha_i^{(k+1)} = \begin{cases} \alpha_i^{(k)} + u \;, & \text{jestliže} \quad \nabla_i E^{(k)} \cdot \delta_i^{(k-1)} > 0 \\ \alpha_i^{(k)} \cdot d \;, & \text{jestliže} \quad \nabla_i E^{(k)} \cdot \delta_i^{(k-1)} < 0 \\ \alpha_i^{(k)} & \text{jinak} \end{cases}$

  u, d … předem zvolené pevné konstanty

  $\delta_i^{(k)} = (1 - \Phi) \nabla_i E^{(k)} + \Phi \, \delta_i^{(k-1)}$, kde Φ je konstanta
- Aktualizace vah bez momentu: $\Delta^{(k)} w_i = -\alpha_i^{(k)} \nabla_i E^{(k)}$

Algoritmus Super SAB

- **Adaptivní akcelerační strategie** pro algoritmus zpětného šíření
  - Řádově rychlejší než původní algoritmus zpětného šíření
  - Poměrně stabilní
  - Robustní vzhledem k volbě počátečních parametrů
- **Využívá momentu:**
  - Urychlení konvergence v plochých oblastech váhového prostoru
  - V příkrých oblastech váhového prostoru moment tlumí oscilace způsobené změnou znaménka gradientu

Super SAB – algoritmus učení (1)

α^+ ……... multiplikativní konstanta pro zvětšování parametru učení ($\alpha^+ = 1.05$)

α^- ……... multiplikativní konstanta pro zmenšování parametru učení ($\alpha^- = 2$)

α_{START} … počáteční hodnota parametru α_{ij}, $\forall i, j$ ($\alpha_{START} = 1.2$)

α_m ……... moment ($\alpha_m = 0.3$)

Krok 1: nastav všechna α_{ij} na počáteční hodnotu α_{START}

Krok 2: proveď Krok (t) algoritmu zpětného šíření s momentem

Krok 3: pokud se nezměnilo znaménko derivace (podle w_{ij}), zvětši parametr učení (proveď $\forall w_{ij}$) : $\alpha_{ij}(t+1) = \alpha^+ \cdot \alpha_{ij}(t+1)$

Super SAB – algoritmus učení (1)

Krok 4: pokud se změnilo znaménko derivace (podle w_{ij}):

- anuluj předchozí změnu vah (která způsobila změnu znaménka gradientu): $\Delta w_{ij}(t+1) = -\Delta w_{ij}(t)$
- zmenši parametr učení: $\alpha_{ij}(t+1) = \alpha_{ij}(t) \,/\, \alpha^{-}$
- a polož: $\Delta w_{ij}(t+1) = 0$

(při dalším kroku učení se pak nebude brát v úvahu změna z předchozího kroku)

Krok 5: přejdi ke Kroku 2

Algoritmus zpětného šíření: strategie pro urychlení procesu učení (2)

2. **Algoritmy druhého řádu:**
   - berou v úvahu více informací o tvaru chybové funkce než jen gradient $\rightarrow$ zakřivení chybové funkce
   - metody 2. řádu používají kvadratickou aproximaci chybové funkce

$\vec{w} = (w_1, \ldots, w_n)$ … vektor všech vah sítě

$E(\vec{w})$ ………… chybová funkce

$\rightarrow$ Taylorova řada pro aproximaci chybové funkce E:

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

Algoritmy druhého řádu (2)

$\nabla^2 E(\vec{w})$ ……. Hessovská matice ***(n × n)*** parciálních derivací druhého řádu:

$$\nabla^2 E(\vec{w}) = \begin{pmatrix} \frac{\partial^2 E(\vec{w})}{\partial w_1^2} & \frac{\partial^2 E(\vec{w})}{\partial w_1 \partial w_2} & \cdots & \frac{\partial^2 E(\vec{w})}{\partial w_1 \partial w_n} \\ \frac{\partial^2 E(\vec{w})}{\partial w_2 \partial w_1} & \frac{\partial^2 E(\vec{w})}{\partial w_2^2} & \cdots & \frac{\partial^2 E(\vec{w})}{\partial w_2 \partial w_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 E(\vec{w})}{\partial w_n \partial w_1} & \frac{\partial^2 E(\vec{w})}{\partial w_n \partial w_2} & \cdots & \frac{\partial^2 E(\vec{w})}{\partial w_n^2} \end{pmatrix}$$

Algoritmy druhého řádu (3)

→ Gradient chybové funkce (zderivováním $\nabla E\left(\vec{w}+\vec{h}\right)$):

$$\nabla E\left(\vec{w}+\vec{h}\right)^T \approx \nabla E(\vec{w})^T + \vec{h}^T \nabla^2 E(\vec{w})$$

→ Gradient by měl být nulový (hledá se minimum E):

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

==> **Newtonovské metody:**

- **Pracují iterativně**
- **Aktualizace vah v k – té iteraci podle:**

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

- **Rychlá konvergence**

× **Problémem může být výpočet inverzní Hessovské matice**

Algoritmy druhého řádu (4)

Pseudonewtonovské metody:

- Pracují se „zjednodušenou aproximací“ Hessovské matice
- V úvahu se berou pouze prvky na diagonále: $\left(\dfrac{\partial^2 E(\vec{w})}{\partial w_i^2} \right)$
- Ostatní prvky Hessovské matice jsou vynulovány
- Adaptace vah podle:

$$w_i^{(k+1)} = w_i^{(k)} - \frac{\nabla_i E(\vec{w})}{\dfrac{\partial^2 E(\vec{w})}{\partial w_i^2}}$$

Algoritmy druhého řádu (5)

Pseudonewtonovské metody:

- Odpadá potřebná inverze Hessovské matice
- Metody „dobře fungují,“ pokud má chybová funkce kvadratický tvar, v opačném případě však mohou nastat problémy

- **Varianty algoritmů:**
  - Quickprop
  - Levenberg-Marquardtův algritmus

Algoritmus Quickprop (1)

- Bere v úvahu i informace 2. řádu
  - × Minimalizační kroky provádí pouze v jednom rozměru
  - → Informace o zakřivení chybové funkce ve směru adaptace se získává ze současné a předchozí parciální derivace chybové funkce
- Nezávislá optimalizace pro každou váhu pomocí kvadratické jednorozměrné aproximace chybové funkce

Algoritmus Quickprop (2)

- Aktualizace vah v ***k*** – té iteraci podle:

$$\vec{w}_i^{(k+1)} = \vec{w}_i^{(k)} + \Delta^{(k)} w_i \quad , \text{ kde}$$

$$\Delta^{(k)} w_i = \Delta^{(k-1)} w_i \cdot \frac{\nabla_i E^{(k)}}{\nabla_i E^{(k-1)} - \nabla_i E^{(k)}}$$

Předpoklad: chybová funkce byla spočtena v kroku ***(k – 1)*** a v kroku ***k*** , a to pro váhy s rozdílem $\Delta^{(k-1)} w_i$ - buď standardním algoritmem zpětného šíření nebo pomocí algoritmu Quickprop

Algoritmus Quickprop (3)

- Aktualizaci vah lze psát i jako:

$$\Delta^{(k)} w_i = - \frac{\nabla_i E^{(k)}}{\dfrac{\nabla_i E^{(k)} - \nabla_i E^{(k-1)}}{\Delta^{(k-1)} w_i}}$$

- Jmenovatel odpovídá diskrétní aproximaci parciální derivace 2. řádu $\partial^2 E(\vec{w}) / \partial w_i^2$
- Quickprop ~ pseudonewtonovská diskrétní metoda, která používá tzv. „ **SEKANTOVÝ KROK** “

Levenberg-Marquardtův algoritmus

- rychlejší a přesnější v oblasti minima chybové funkce
- kombinace gradientní a Newtonovy metody

gradient

$$\vec{w}_{\min} = \vec{w}_0 - (H + \lambda I)^{-1} \cdot \vec{g}$$

Hessovská matice

- pro 1 výstup: $g_i = \frac{\partial e}{\partial w_i} = 2(y-d)\frac{\partial y}{\partial w_i}$

$$\frac{\partial^2 e}{\partial w_i \partial w_j} = 2\left[\frac{\partial y}{\partial w_i}\frac{\partial y}{\partial w_j} + \cancel{(y-d)\frac{\partial^2 y}{\partial w_i \partial w_j}}\right]$$

Algoritmus zpětného šíření: strategie pro urychlení procesu učení (3)

3. **Relaxační metody** – perturbace vah:

- V každé iteraci se počítá diskrétní aproximace gradientu porovnáním chybové funkce pro výchozí váhy $E(\vec{w})$ a chybové funkce pro mírně změněné váhy $\vec{w}'$ (k váze $\boldsymbol{w_i}$ byla přičtena malá perturbace $\boldsymbol{\beta}$) – $E(\vec{w}')$
- Aktualizace vah pomocí: $\Delta w_i = -\alpha \dfrac{E(\vec{w}') - E(\vec{w})}{\beta}$
- Aktualizace se iterativně opakuje pro vždy náhodně zvolenou váhu

Relaxační metody (2)

Alternativa s rychlejší konvergencí:

- Perturbace výstupu ***i*** – tého neuronu $\boldsymbol{o_i}$ o $\boldsymbol{\Delta o_i}$
- Vypočítá se rozdíl ***E - E´***
- Pokud je rozdíl kladný ***(> 0),*** lze nové chyby ***E´*** dosáhnout výstupem $\boldsymbol{o_i + \Delta o_i}$ pro ***i*** – tý neuron
- V případě sigmoidální přenosové funkce, lze požadovaný potenciál neuronu ***i*** určit pomocí:

$$\sum_{k=1}^{m} w'_k \, x_k = s^{-1}\left(o_i + \Delta o_i \right)$$

$$\left(\text{pro } y = s(\xi) = \frac{1}{1+e^{-\xi}} \quad \text{je} \quad \xi = s^{-1}(y) = \ln \frac{y}{1-y} \right)$$

Relaxační metody (3)

- Pokud byl předchozí potenciál $\sum_{k=1}^{m} w_k \, x_k$, jsou nové váhy dány pomocí:

$$w'_k = w_k \cdot \frac{s^{-1}\left(o_i + \Delta o_i\right)}{\sum_{k=1}^{m} w_k \, x_k}$$

- Váhy se adaptují proporcionálně ke své velikosti: $\dfrac{w'_k}{\xi'} = \dfrac{w_k}{\xi}$

(to lze poněkud omezit zavedením stochastických faktorů anebo kombinací s metodou pro perturbaci vah)