

Neuronové sítě

Doc. RNDr. Iveta Mrázová, CSc.

Katedra teoretické informatiky

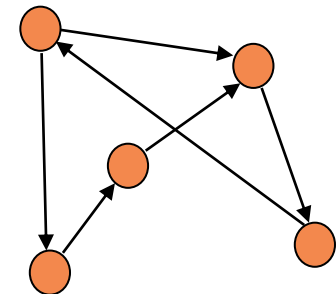
Matematicko-fyzikální fakulta

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
 - $w: C \rightarrow \mathbf{R}$ je váhová funkce
 - $t: N \rightarrow \mathbf{R}$ je prahová funkce
- (\mathbf{R} označuje množinu reálných čísel)



Vrstevnaté neuronové sítě (2)

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

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

$$E = \frac{1}{2} \sum_p \sum_j (y_{j,p} - d_{j,p})^2$$

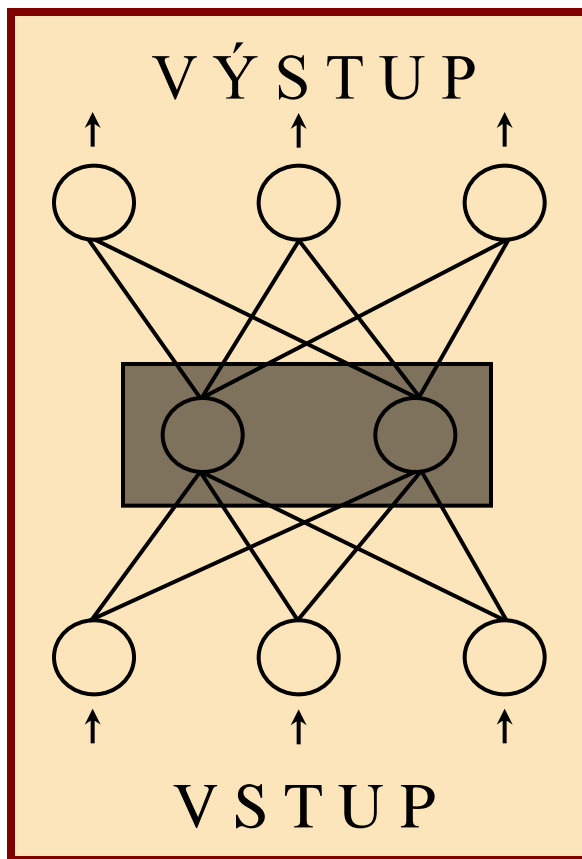
Diagrammatic annotations for the equation above:

- Arrows point from the text "vzory" to the summation index p .
- An arrow points from the text "výstupní neurony" to the summation index j .
- An arrow points from the text "požadovaná odezva" to the term $d_{j,p}$.
- An arrow points from the text "skutečná odezva" to the term $y_{j,p}$.

- 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 w_{ij} přispívající k minimalizaci E
chyba na výstupu sítě

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

skutečný výstup váha spoje potenciál neuronu j

BP-sítě: adaptační pravidla (2)

Aktualizace synaptických vah pro výstupní vrstvu:

$$\begin{aligned}\Delta_E w_{ij} &\cong - \frac{\partial E}{\partial w_{ij}} = - \frac{\partial E}{\partial y_j} \frac{\partial y_j}{\partial \xi_j} \frac{\partial \xi_j}{\partial w_{ij}} = \\&= - \frac{\partial E}{\partial y_j} \frac{\partial y_j}{\partial \xi_j} \frac{\partial}{\partial w_{ij}} \sum_{i'} w_{i'j} y_{i'} = \\&= - \frac{\partial E}{\partial y_j} \frac{\partial y_j}{\partial \xi_j} y_i = - \frac{\partial E}{\partial y_j} f'(\xi_j) y_i = \\&= - (y_j - d_j) f'(\xi_j) y_i = \delta_j y_i\end{aligned}$$

BP-sítě: adaptační pravidla (3)

Aktualizace synaptických vah pro skryté vrstvy:

$$\begin{aligned}\Delta_E w_{ij} &\cong - \frac{\partial E}{\partial w_{ij}} = - \left(\sum_k \frac{\partial E}{\partial \xi_k} \frac{\partial \xi_k}{\partial y_j} \right) \frac{\partial y_j}{\partial \xi_j} y_i = \\ &= - \left(\sum_k \frac{\partial E}{\partial \xi_k} \frac{\partial}{\partial y_j} \sum_{j'} w_{j'k} y_{j'} \right) \frac{\partial y_j}{\partial \xi_j} y_i = \\ &= - \left(\sum_k \frac{\partial E}{\partial \xi_k} w_{jk} \right) \frac{\partial y_j}{\partial \xi_j} y_i = \\ &= \left(\sum_k \delta_k w_{jk} \right) f'(\xi_j) y_i = \delta_j y_i\end{aligned}$$

BP-sítě: adaptační pravidla (4)

Výpočet derivace sigmoidální přenosové funkce podle:

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

Aktualizace vah podle:

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

■ 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: Vypočtete skutečný výstup

aktivita neuronů v každé vrstvě je dána pomocí:

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

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:

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

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

$w_{ij}(t)$ 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

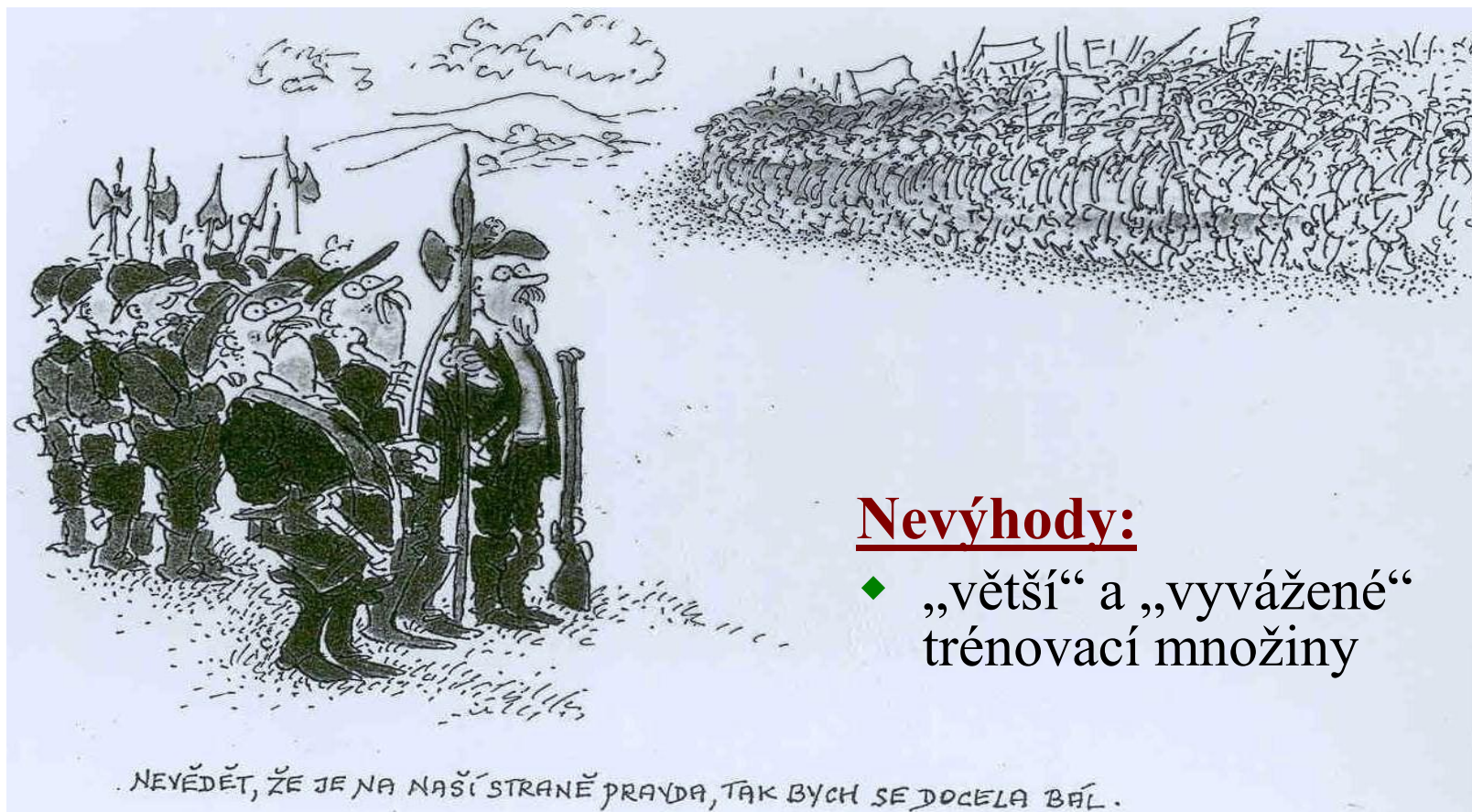
λ strmost přenosové funkce

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:

$$\begin{aligned}\sigma_{\xi}^2 &= E\{(\xi_j)^2\} - E^2\{(\xi_j)\} = E\left\{\left(\sum_{i=0}^n w_{ij} x_i\right)^2\right\} - 0 = \\ &= \sum_{i,k=0}^n E\{w_{ij} w_{kj} x_i x_k\} = \leftarrow \text{vzájemná nezávislost pro všechna } j \\ &= \sum_{i=0}^n E\{w_{ij}^2\} E\{x_i^2\}\end{aligned}$$

Algoritmus zpětného šíření: volba počátečních vah (5)

IDEA - pokračování:

- ♦ **Další předpoklad:** trénovací vzory jsou normalizované a leží v intervalu $\langle 0, 1 \rangle$. Potom:

$$E \{ (x_i)^2 \} = \int_0^1 x_i^2 \, dx = \frac{x^3}{3} \Big|_0^1 = \frac{1}{3}$$

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

$$E \{ (w_{ij})^2 \} = \int_{-a}^a w_{ij}^2 \, dw_{ij} = \frac{w_{ij}^3}{3} \Big|_{-a}^a = \frac{2a^3}{3}$$

- ♦ N ... počet vah vedoucích k danému neuronu

Algoritmus zpětného šíření: volba počátečních vah (6)

IDEA - pokračování:

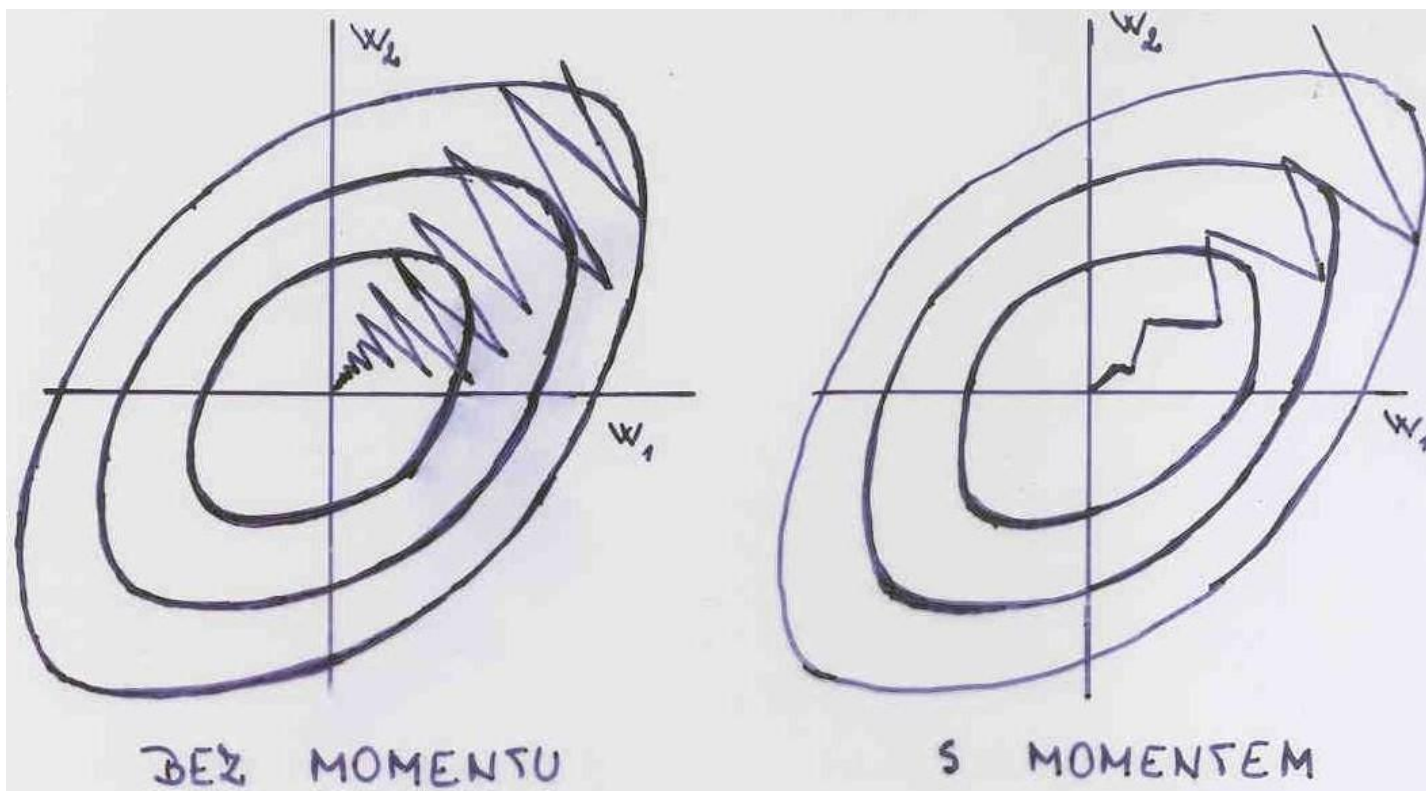
- ♦ Směrodatná odchylka tedy bude odpovídat:

$$A = \sigma_{\xi} = \sqrt{N} \frac{a}{3} \sqrt{2a} \quad \left(\rightarrow a\sqrt{2a} = A \frac{3}{\sqrt{N}} \right)$$

- ♦ 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:
$$\left[-\frac{3}{\sqrt{N}} \cdot 0.8 \cdot A, \frac{3}{\sqrt{N}} \cdot 0.8 \cdot A \right], \quad \text{resp.} \quad \left[-\frac{1.65}{\sqrt[3]{N}} \cdot \sqrt[3]{A^2}, \frac{1.65}{\sqrt[3]{N}} \cdot \sqrt[3]{A^2} \right]$$
- ♦ 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, \dots, w_n je změna váhy w_k v kroku $i + 1$ dána vztahem

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

kde: α parametr učení

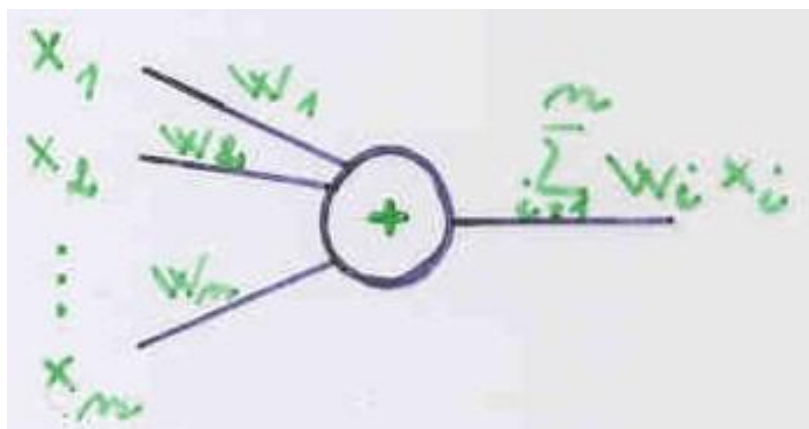
α_m ... moment učení

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

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

→ MINIMALIZACE E :

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

Algoritmus zpětného šíření s momentem (6)

- ♦ E je kvadratická funkce \rightarrow pomocí gradientní metody lze nalézt minimum

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

- \rightarrow Gradientní metoda dává nejlepší výsledky pro „nejdůležitější“ osy „stejně délky“ (principal axes)
- \rightarrow 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:

$$c_1^2 w_1^2 + c_2^2 w_2^2 + \dots + c_n^2 w_n^2 + \sum_{i \neq j} d_{ij} w_i w_j + C$$

- ♦ Krok v i –tém směru minimalizuje

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

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

Algoritmus Silvy & Almeidy (2)

Heuristika:

- ◆ **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 ($i = 1, \dots, n$)
inicializace malými náhodnými hodnotami

Algoritmus Silvy & Almeidy (3)

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

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

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

$u, d \dots$ předem zvolené pevné konstanty

$$\delta_i^{(k)} = (1 - \Phi) \nabla_i E^{(k)} + \Phi \delta_i^{(k-1)}, \quad \text{kde } \Phi \text{ je konstanta}$$

- ♦ 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: proved' Krok (t) algoritmu zpětného šíření s momentem

Krok 3: pokud se nezměnilo znaménko derivace (podle w_{ij}), zvětši parametr učení (proved' $\forall w_{ij}$) : $\alpha_{ij}(t+1) = \alpha^+ \cdot \alpha_{ij}(t)$

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

$\vec{w} = (w_1, \dots, w_n)$... 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)

$\nabla^2 E(\vec{w})$ Hessianová matice ($n \times n$) parciálních derivací druhého řádu:

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

Algoritmy druhého řádu (3)

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

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

→ 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 Hessianové matice

Algoritmy druhého řádu (4)

Pseudonewtonovské metody:

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

- ◆ Adaptace vah podle:

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

Algoritmy druhého řádu (5)

Pseudonewtonovské metody:

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

◆ Variety algoritmů:

- Quickprop
- Levenberg-Marquardtův algoritmus

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:

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

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

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:

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

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

Levenberg-Marquardtův algoritmus

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

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

gradient

Hessovská matice

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

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

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 w_i byla přičtena malá perturbace β) – $E(\vec{w}')$
- ♦ Aktualizace vah pomocí:
$$\Delta w_i = -\alpha \frac{E(\vec{w}') - E(\vec{w})}{\beta}$$
- ♦ Aktualizace se iterativně opakuje pro vždy náhodně zvolenou váhu

Relaxační metody (2)

Alternativa s rychlejší konvergencí:

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

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

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

Relaxační metody (3)

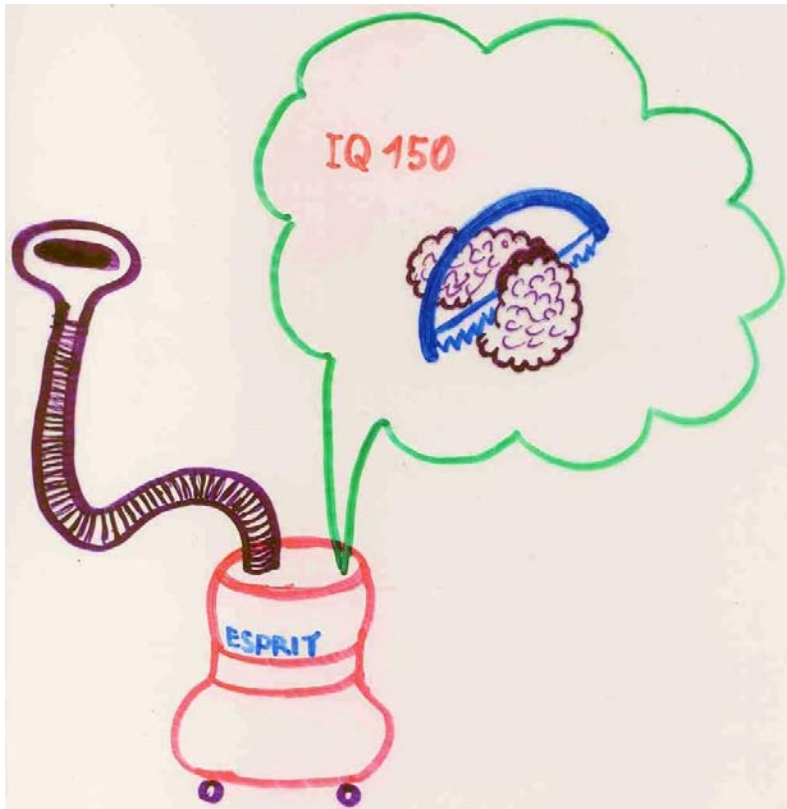
- ◆ Pokud byl předchozí potenciál $\sum_{k=1}^m w_k x_k$, jsou nové váhy dány pomocí:

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

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

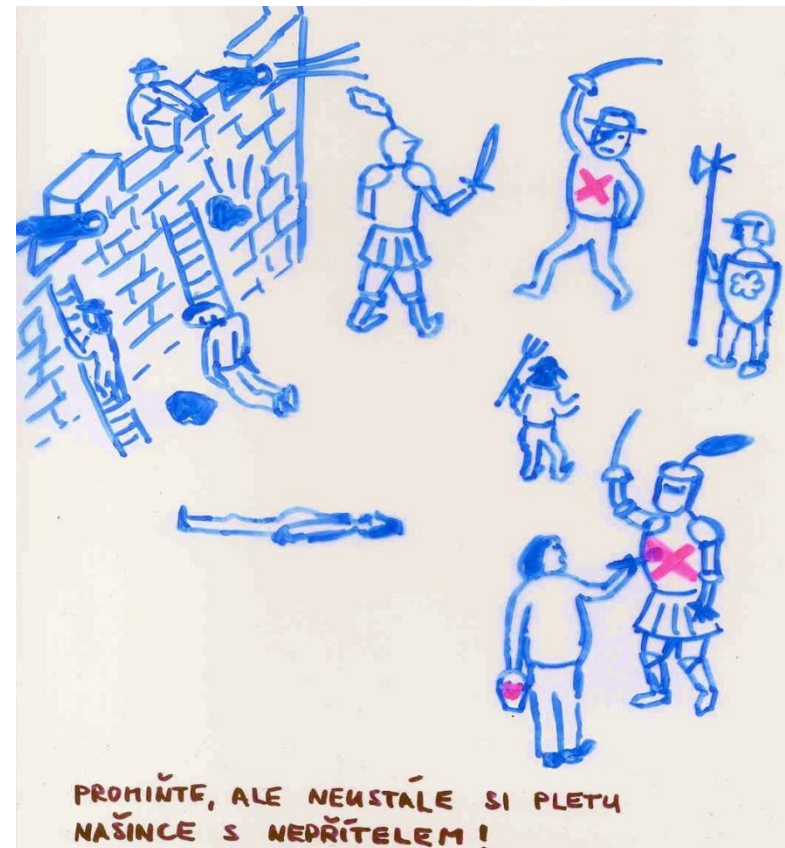
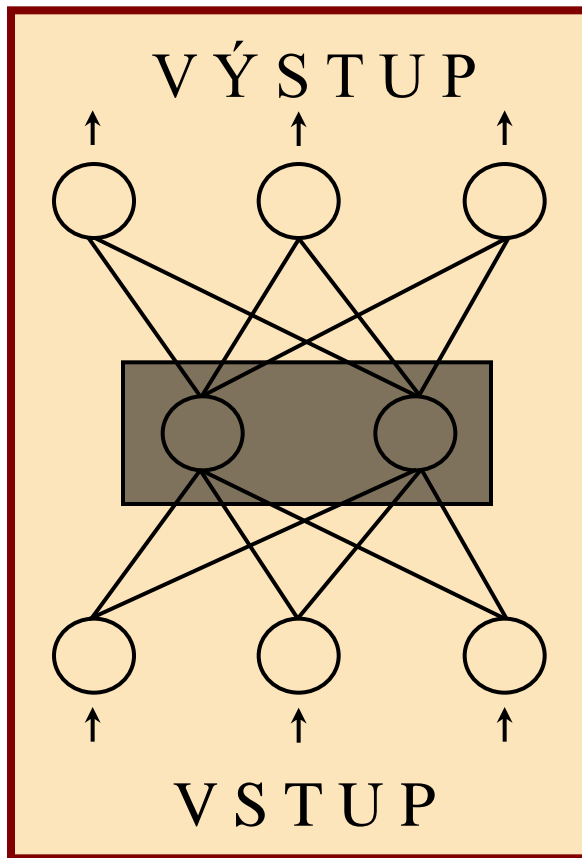
(to lze poněkud omezit zavedením stochastických faktorů anebo kombinací s metodou pro perturbaci vah)

Interní reprezentace znalostí

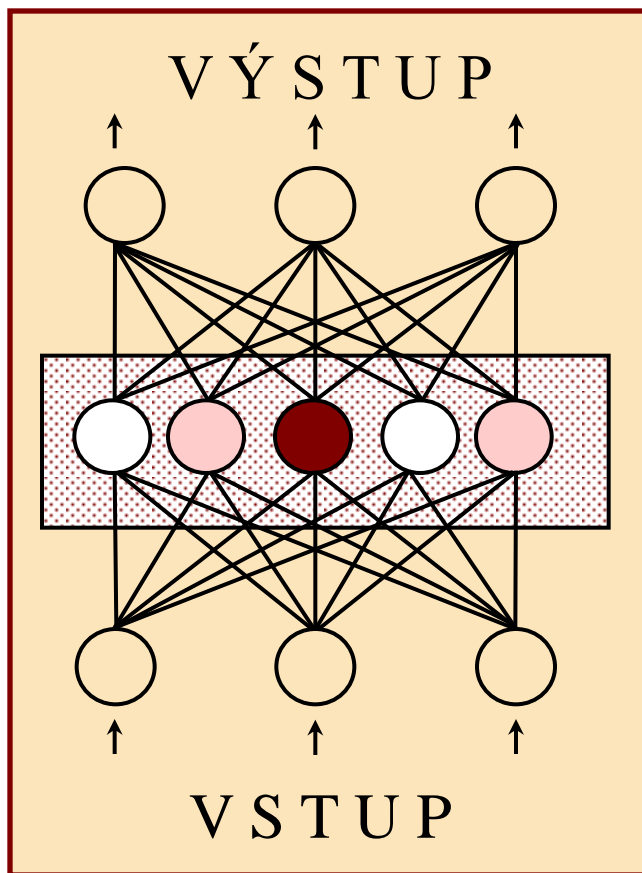


- počet neuronů a generalizační schopnosti sítě
→ **prořezávání a doučování**

Interní reprezentace znalostí



Kondenzovaná interní reprezentace



♦ interpretace aktivity skrytých neuronů:

- 1 \longleftrightarrow aktivní \longleftrightarrow ANO
- 0 \longleftrightarrow pasivní \longleftrightarrow NE
- $\frac{1}{2}$ \longleftrightarrow tichý \longleftrightarrow
 \longleftrightarrow „nelze rozhodnout“

- ♦ průhledná struktura sítě
- ♦ detekce nadbytečných neuronů a prořezávání
- ♦ lepší generalizace

Kondenzovaná interní reprezentace

D: Pro vrstevnatou síť B zpracovávající vstupní vzor \vec{x} :

- Skrytý neuron s vahami (w_1, \dots, w_n) , prahem θ , vstupním vzorem \vec{z} a přenosovou funkcí $f[\vec{w}, \theta](\vec{z})$ vytváří **reprezentaci** $r : r = y = f[\vec{w}, \theta](\vec{z})$
- Vektor \vec{r} reprezentací vytvořených vrstvou skrytých neuronů se nazývá **interní reprezentace** \vec{x}

Kondenzovaná interní reprezentace

D: Pro vrstevnatou síť ***B***:

- Interní reprezentace $\vec{r} = (r_1, \dots, r_m)$ je **binární**, jestliže $r_i \in \{0, 1\}; \quad 1 \leq i \leq m$
- Interní reprezentace $\vec{r} = (r_1, \dots, r_m)$ je **kondenzovaná**, jestliže $r_i \in \{0, 0.5, 1\}; \quad 1 \leq i \leq m$

Požadavky na vynucování kondenzované interní reprezentace

- ♦ formulace „požadovaných vlastností“ ve formě cílové funkce:

$$\mathbf{G} = \mathbf{E} + \mathbf{c}_s \mathbf{F}$$

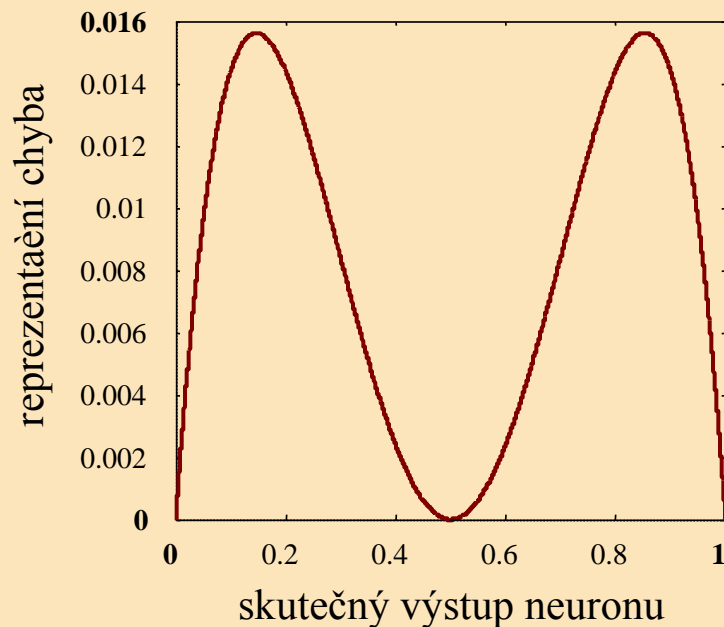
standardní chybová funkce
reprezentační chybová funkce
velikost vlivu F na G

- ♦ lokální minima reprezentační chybové funkce odpovídají aktivním, pasivním a tichým stavům:

$$F = \sum_p \sum_h y_{h,p}^s (1 - y_{h,p})^s (y_{h,p} - 0.5)^2$$

vzory p skryté neurony h pasivní stav $y_{h,p}^s$ aktivní stav $(1 - y_{h,p})^s$ tvar F $(y_{h,p} - 0.5)^2$ tichý stav

Vliv parametrů na vytváření kondenzované interní reprezentace



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

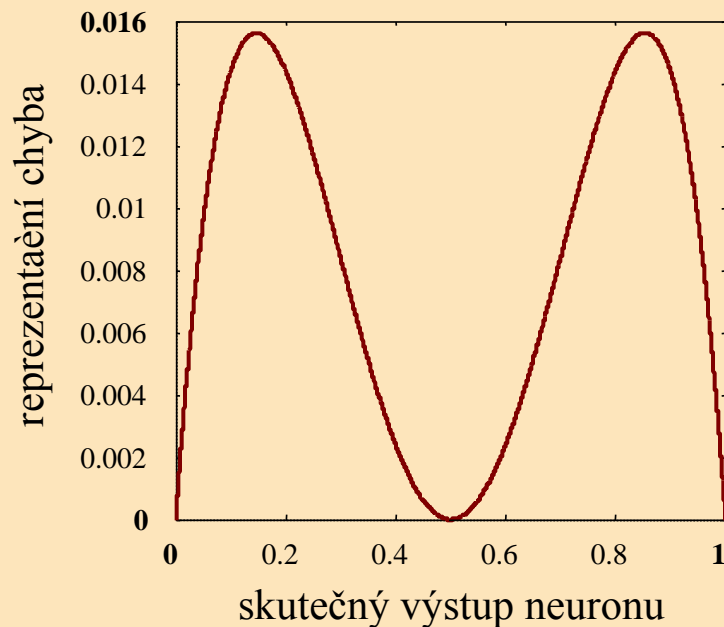
- ◆ pomalejší vytváření interní reprezentace a požadovaná funkce sítě
- ◆ stabilita vytvářené interní reprezentace a optimální architektura sítě
- ◆ tvar reprezentační chybové funkce, rychlost vytváření interní reprezentace a její forma
- ◆ časová náročnost při adaptaci vah

Chybový člen pro posilování kondenzované interní reprezentace

Kondenzovaná interní reprezentace ($y_j^s (1 - y_j)^s (y_j - 0.5)^2$):

$$\rho_j = \begin{cases} 0 & \text{pro výstupní neurony} \\ - \left[2(s+1)y_j (1 - y_j) - \frac{s}{2} \right] \cdot y_j^s (1 - y_j)^s (y_j - 0.5) & \text{pro neurony z nejvyšší skryté vrstvy} \\ \left(\sum_k \rho_k w_{jk} \right) y_j (1 - y_j) & \text{pro ostatní skryté neurony} \end{cases}$$

Vliv parametrů na vytváření kondenzované interní reprezentace

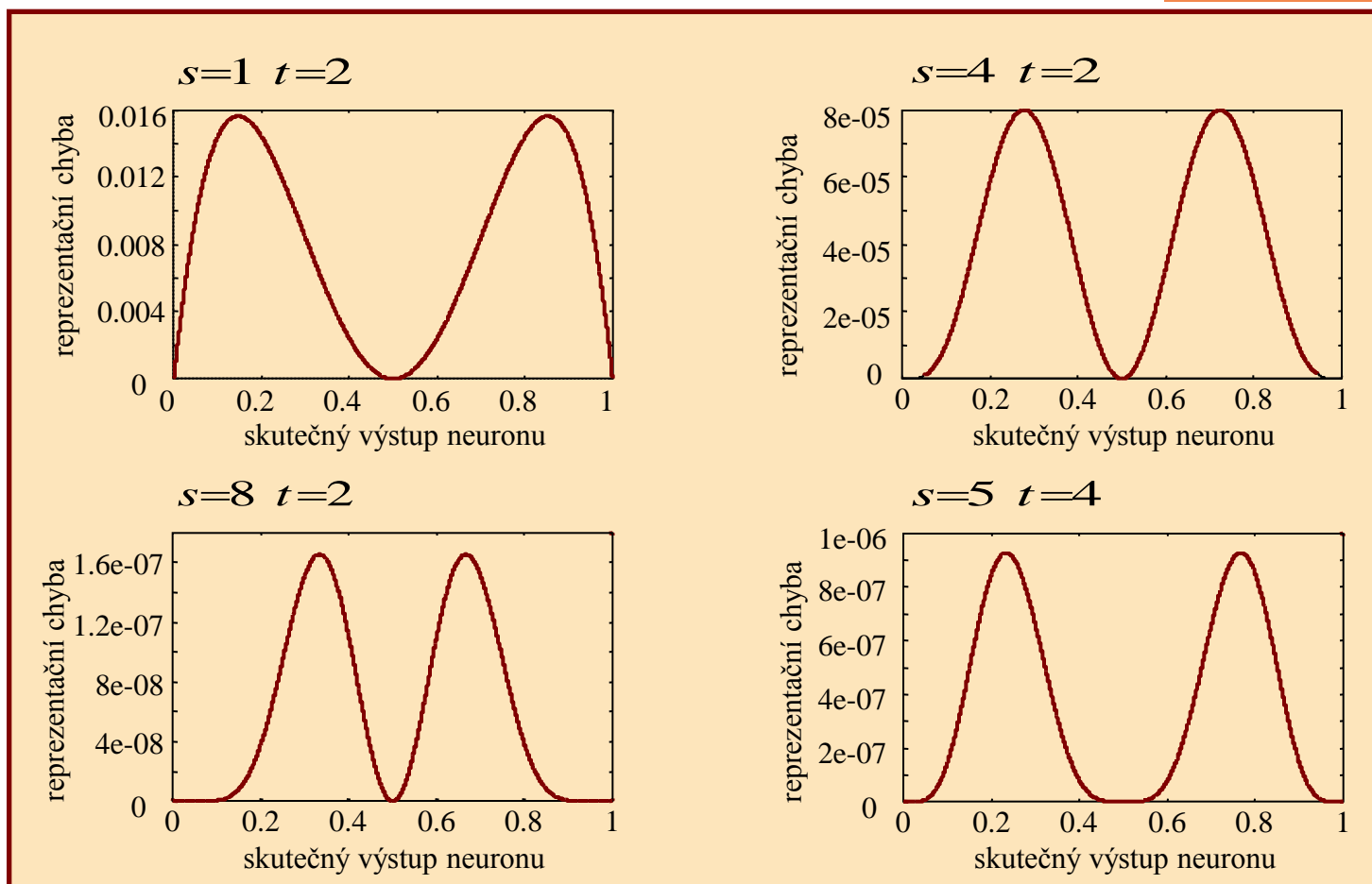


$$w_{ij}(t+1) = w_{ij}(t) + \alpha \delta_j y_i + \alpha_r \rho_j y_i + \alpha_m (w_{ij}(t) - w_{ij}(t-1))$$

- ◆ pomalejší vytváření interní reprezentace a požadovaná funkce sítě
- ◆ stabilita vytvářené interní reprezentace a optimální architektura sítě
- ◆ tvar reprezentační chybové funkce, rychlost vytváření interní reprezentace a její forma
- ◆ časová náročnost při adaptaci vah

Tvar reprezentační chybové funkce

$$F = y^s (1 - y)^s (y - 0.5)^t$$



Chybový člen pro posilování binární interní reprezentace

Binární interní reprezentace ($y_j (1 - y_j)$):

$$\rho_j = \begin{cases} 0 & \text{pro výstupní neurony} \\ - (1 - 2y_j) y_j (1 - y_j) & \text{pro neurony z nejvyšší skryté vrstvy} \\ \left(\sum_k \rho_k w_{jk} \right) y_j (1 - y_j) & \text{pro ostatní skryté neurony} \end{cases}$$

Jednoznačná interní prezentace

- ♦ Hodně odlišným výstupům by měly odpovídat hodně odlišné interní reprezentace
- ♦ Formulace požadavků ve formě modifikované cílové funkce: $G = E + F + H$
- ♦ Kritérium pro jednoznačnost IR:

$$H = -\frac{1}{2} \sum_p \sum_{q \neq p} \sum_j \sum_o (d_{o,p} - d_{o,q})^2 (y_{j,p} - y_{j,q})^2$$

Diagrammatic annotations for the formula above:

- Arrows from p and $q \neq p$ point to the text "vzory skryté neurony".
- An arrow from j points to the text "výt. neurony".
- An arrow from o points to the text "výst. neurony".
- An arrow from $d_{o,p}$ points to the text "= konst. pro dané p".
- An arrow from $d_{o,q}$ points to the text "= konst. pro dané p".
- An arrow from $y_{j,p}$ points to the text "= konst. pro dané p".
- An arrow from $y_{j,q}$ points to the text "= konst. pro dané p".

Prořezávání podle interní reprezentace (1)

D: Pro danou vrstevnatou síť B a množinu S vstupních vzorů určujících vstupní vektory \vec{z} :

- Skrytý neuron s vahami (w_1, \dots, w_n) , prahem θ a přenosovou funkcí $f[\vec{w}, \theta](\vec{z})$ vytváří **uniformní reprezentaci** r , jestliže:

$$r = f[\vec{w}, \theta](\vec{z}) = \text{const} \quad \text{pro všechny vstupní vzory } \vec{x} \in S$$

Prořezávání podle interní reprezentace (2)

D: Pro danou vrstevnatou síť B a množinu S vstupních vzorů určujících vstupní vektory \vec{z} :

- Skrytý neuron $i \in N$ s vahami (w_{i1}, \dots, w_{in}) , prahem θ_i a přenosovou funkcí $f_i[\vec{w}_i, \theta_i](\vec{z})$ vytváří reprezentaci r_i **identickou** k reprezentaci r_j vytvářené skrytým neuronem $j \in N$ s vahami (w_{j1}, \dots, w_{jn}) , prahem θ_j a přenosovou funkcí $f_j[\vec{w}_j, \theta_j](\vec{z})$, jestliže:

$$f_i[\vec{w}_i, \theta_i](\vec{z}) = f_j[\vec{w}_j, \theta_j](\vec{z}) \quad \text{pro všechny v vstupní vzory } \vec{x} \in S$$

Prořezávání podle interní reprezentace (3)

D: Pro danou vrstevnatou síť B a množinu S vstupních vzorů určujících vstupní vektory \vec{z} :

- Skrytý neuron $i \in N$ s vahami (w_{i1}, \dots, w_{in}) , prahem ϑ_i a přenosovou funkcí $f_i[\vec{w}_i, \vartheta_i](\vec{z})$ vytváří reprezentaci r_i **inverzní** k reprezentaci r_j vytvářené skrytým neuronem $j \in N$ s vahami (w_{j1}, \dots, w_{jn}) , prahem ϑ_j a přenosovou funkcí $f_j[\vec{w}_j, \vartheta_j](\vec{z})$, jestliže:

$$f_i[\vec{w}_i, \vartheta_i](\vec{z}) = 1 - f_j[\vec{w}_j, \vartheta_j](\vec{z}) \quad \text{pro všechny vstupní vzory } \vec{x} \in S$$

Prořezávání podle interní reprezentace (4)

D: Pro danou vrstevnatou síť B a množinu vstupních vzorů S :

redukovaná vrstva je vrstva, pro kterou platí, že:

- žádný neuron nevytváří uniformní reprezentaci,
- žádný neuron i nevytváří reprezentaci identickou k reprezentaci vytvářené jiným neuronem j a
- žádný neuron i nevytváří reprezentaci inverzní k reprezentaci vytvářené jiným neuronem j .

Interní reprezentace vytvářené redukovanou vrstvou se nazývá **redukovaná**.

Prořezávání podle interní reprezentace (5)

D: Pro danou množinu vstupních vzorů S :

- vrstevnatá síť B je **redukovaná**, jestliže jsou všechny její skryté vrstvy redukovány.
- vrstevnatá síť B je **ekvivalentní** k vrstevnaté síti B' , jestliže je pro libovolný vstupní vzor $\vec{x} \in S$ skutečný výstup \vec{y}_B sítě B roven skutečnému výstupu $\vec{y}_{B'}$ sítě B' : $\vec{y}_B = \vec{y}_{B'}$

Prořezávání podle interní reprezentace (6)

V: Ke každé vrstevnaté síti B a množině vstupních vzorů S existuje ekvivalentní redukovaná vrstevnatá síť B' .

Důkaz (idea):

Popis konstrukce redukované vrstevnaté sítě B' :

Nechť $B = (N, C, I, O, w, t)$ je původní vrstevnatá síť.

1. Postupná eliminace všech takových neuronů j , které vytvářejí uniformní reprezentaci r_j^k a přičtení součinu $w_{ij} r_j^k$ ke všem prahům θ_j v následující vrstvě.

Prořezávání podle interní reprezentace (7)

Důkaz (pokračování):

2. Postupná eliminace všech takových neuronů j , které vytvářejí reprezentaci r_j^{id} identickou k reprezentaci r_k vytvářené jiným neuronem k a přičtení vah w_{ij} ke každé váze w_{ik} , kde i je neuron v následující vrstvě.
3. Postupná eliminace všech takových neuronů j , které vytvářejí reprezentaci r_j^{in} inverzní k reprezentaci r_k vytvářené jiným neuronem k a nahrazení všech vah w_{ik} , kde i označuje neuron z následující vrstvy, rozdílem $w_{ik} - w_{ij}$ a přičtení váhy w_{ij} k prahu θ_j každého neuronu i .

Prořezávání podle interní reprezentace (8)

Důkaz (pokračování):

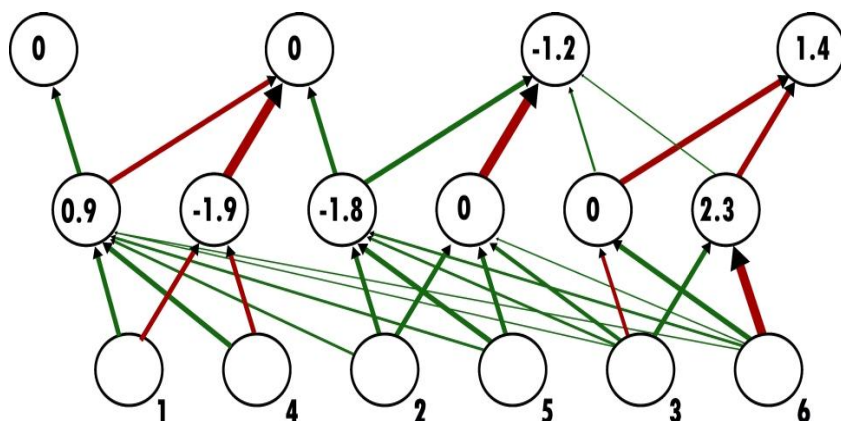
Potom bude pro libovolný vstupní vzor \vec{x} skutečný výstup $\vec{y}_{B'}$ vrstevnaté sítě B' roven skutečnému výstupu \vec{y}_B vrstevnaté sítě B .

Vrstevnatá síť B' konstruovaná ze sítě B popsáním způsobem je redukováná a ekvivalentní k B .

QED

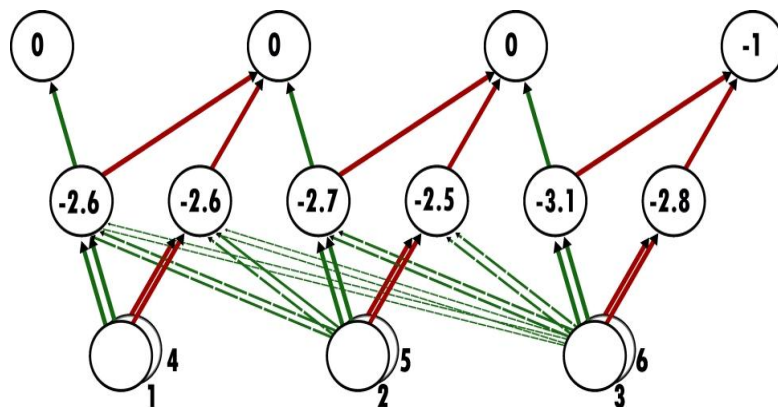
Výsledky experimentů: binární sčítání

[$5(\approx(1,-1,1)) + 3(\approx(-1,1,1)) = 8(\approx(1,-1,-1,-1))$]



◆ SCG-s náповědou (přenos na 2. výstupní neuron)

- ‘přenos’ první a druhý výstupní bit – skryté neurony 1 a 3
- funkce ostatních skrytých neuronů není tak zřejmá



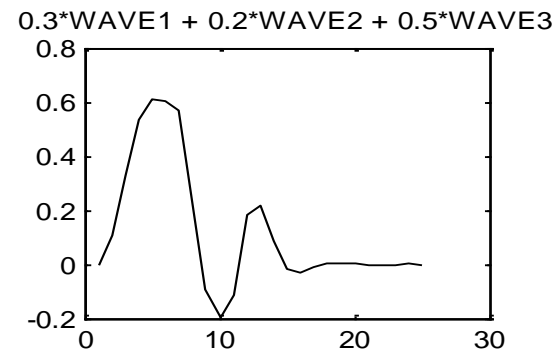
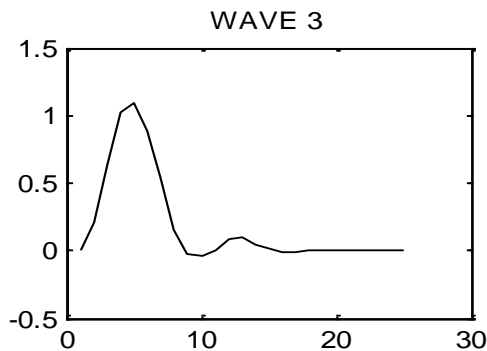
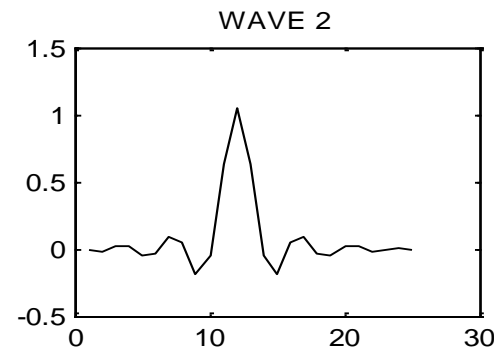
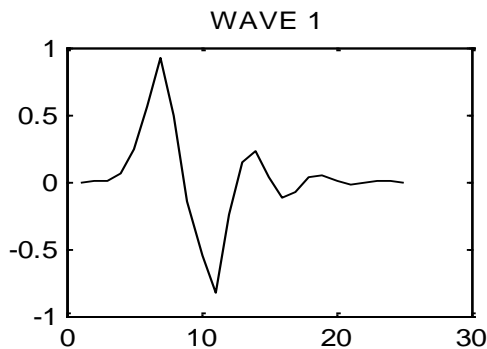
◆ SCGIR-s náповědou (přenos na 2. výstupní neuron)

- ‘přenos’ pro vyšší výstupní bity – skryté neurony 1, 3, 5
- podobná funkce je zřejmá pro jednotlivé výstupní neurony

Akustická emise: simulace (s M. Chladou a Z. Převorovským)

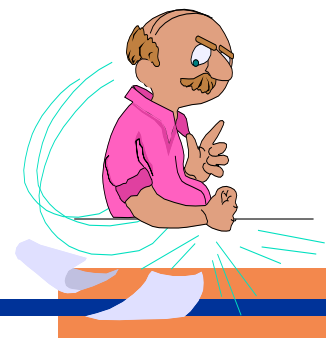


MODELOVANÝ SIGNÁL

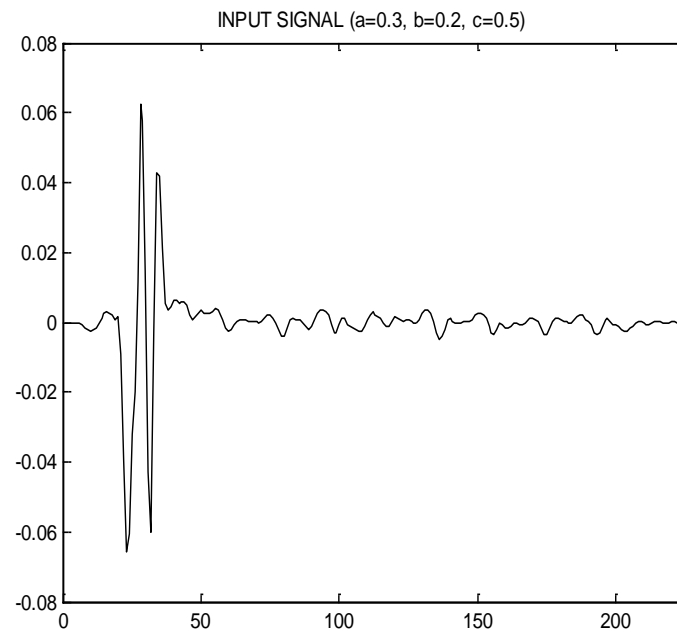
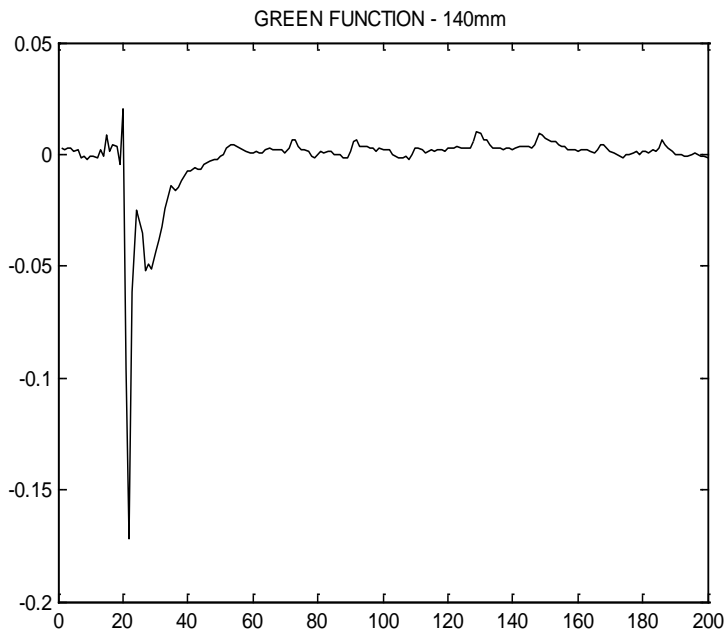


Simulovaná AE-data

(s M. Chladou a Z. Převorovským)

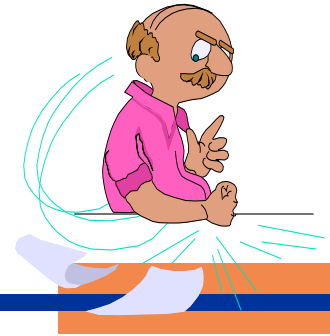


KONVOLUCE S GREENOVOU FUNKCÍ



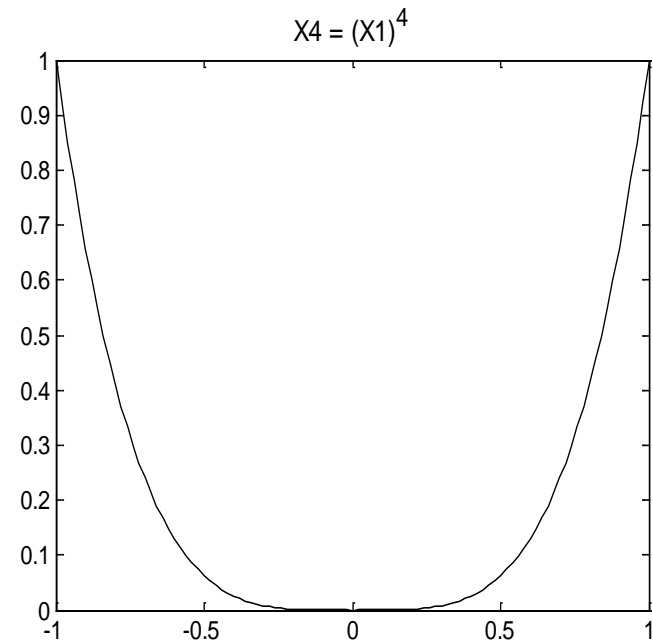
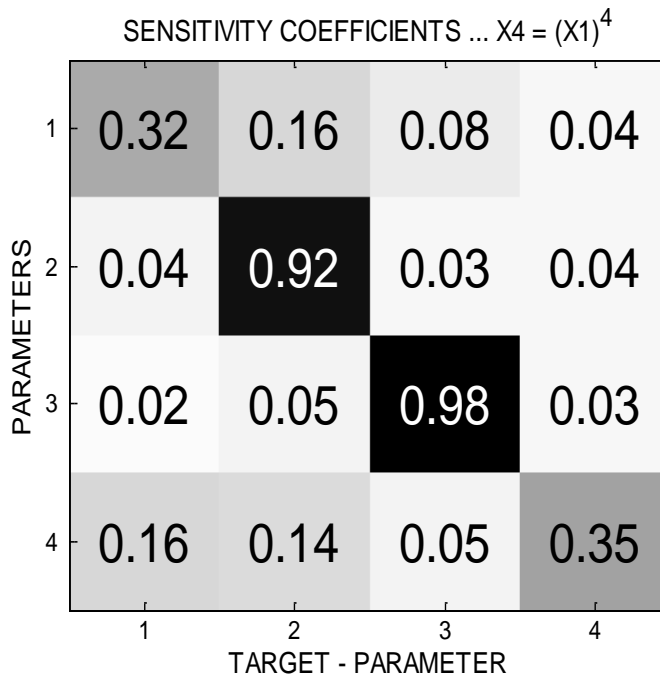
Model závislosti

(s M. Chladou a Z. Převorovským)



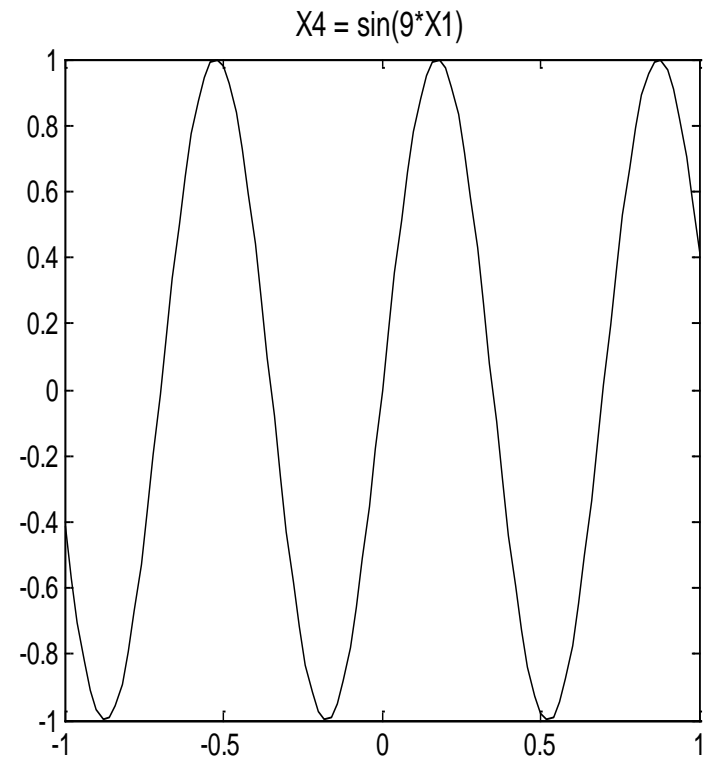
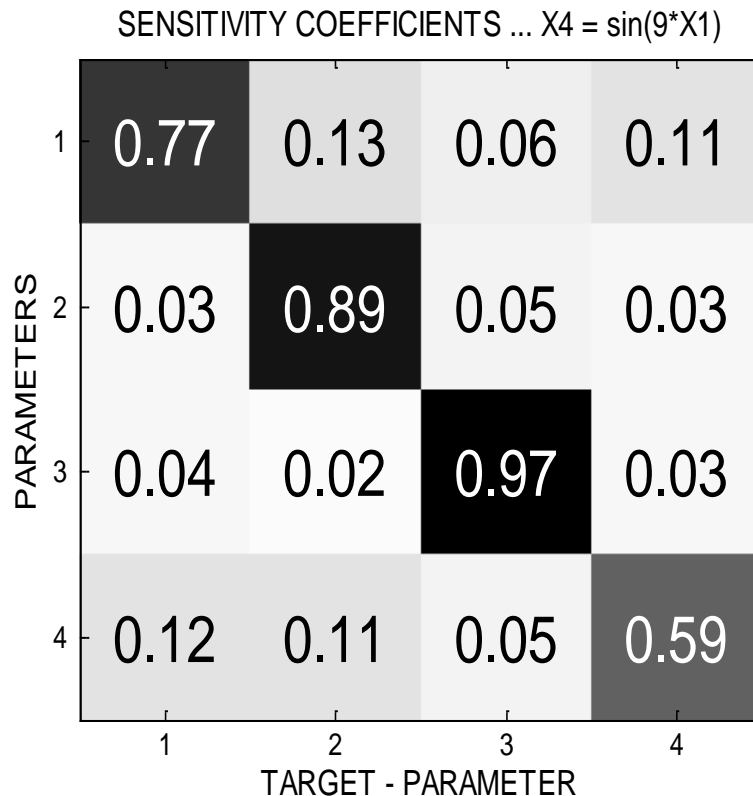
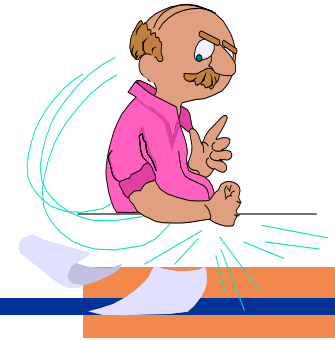
Celková **citlivost sítě** (přes Q vzorů) s -tého výstupu na r -tý vstup:

$$sens_r = 1/Q \sum_q \sum_s \left| \partial y_{q,s} / \partial y_{q,r} \right|$$



Model závislosti

(s M. Chladou a Z. Převorovským)



Faktorová vs. citlivostní analýza vstupních parametrů

(s M. Chladou a Z. Převorovským)



vstupní parametry	1	0.04	0.91	0.16	0.10	0.01	0.07	0.06	0.01	0.23
	2	0.09	0.02	0.01	0.19	0.03	0.03	0.95	0.03	0.16
	3	0.10	0.96	0.15	0.00	0.03	0.00	0.02	0.05	0.07
	4	0.13	0.91	0.02	0.03	0.05	0.06	0.04	0.06	0.20
	5	0.30	0.05	0.04	0.41	0.06	0.49	0.17	0.07	0.66
	6	0.26	0.00	0.04	0.03	0.08	0.93	0.02	0.10	0.20
	7	0.29	0.06	0.03	0.27	0.03	0.17	0.16	0.04	0.86
	8	0.12	0.06	0.01	0.88	0.02	0.02	0.24	0.03	0.36
	9	0.90	0.15	0.09	0.08	0.15	0.17	0.06	0.21	0.18
	10	0.93	0.14	0.06	0.08	0.10	0.17	0.06	0.09	0.19
	11	0.25	0.10	0.12	0.03	0.25	0.11	0.03	0.90	0.05
	12	0.20	0.07	0.14	0.02	0.93	0.09	0.03	0.23	0.04
	13	0.04	0.09	0.97	0.00	0.05	0.00	0.00	0.06	0.02
	14	0.08	0.18	0.95	0.02	0.09	0.04	0.01	0.06	0.02
		1	2	3	4	5	6	7	8	9

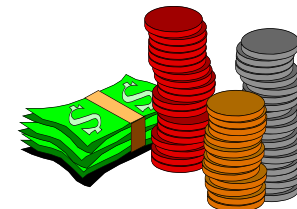
vybrané faktory

- ♦ Vybráno 9 faktorů (“vysvětlují” 98.4% proměnných)
- ♦ redukce lineárně závislých vstupních parametrů

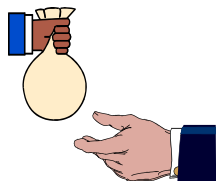
SENSITIVITY COEFFICIENTS				
INPUTS	1	0.173	0.266	0.149
	2	0.093	0.068	0.047
	3	0.320	0.193	0.184
	4	0.301	0.178	0.196
	5	0.564	0.250	0.206
	6	0.196	0.322	0.158
	7	0.099	0.063	0.043
	8	0.065	0.015	0.030
	9	0.022	0.014	0.016
	10	0.053	0.020	0.012
	11	0.035	0.012	0.032
	12	0.039	0.050	0.022
	13	0.081	0.134	0.082
	14	0.260	0.172	0.109
	1	2	3	
OUTPUTS				

- ♦ Vybráno 7 příznaků
- ♦ detekce nelineární závislosti vstupních parametrů (1, 3, 4, 5, 6, 13, 14)

Analýza dat ze Světové banky



WDI-indikátory (indikátory vývoje ve světě)



- každoročně zveřejňovány Světovou bankou
 - pomoc rozvojovým zemím při půjčkách / investicích
 - odhad stavu ekonomik a jejich vývoje v jednotlivých zemích
- původ údajů - neúplné a nepřesné údaje

♦ používané techniky

- regresní analýza - lineární závislosti
- kategorizace států používaná v rozvinutých zemích (G. Ip, Wall Street Journal)
- kategorizace zemí podle HDP (Světová banka)
- Kohonenovy mapy (T. Kohonen, S. Kaski, G. Deboeck)

Analýza dat ze Světové banky: použité WDI-indikátory



- ◆ Implicitní deflace HDP
- ◆ Vnější zadluženost (% HNP)
- ◆ Celkové náklady na zadlužení (% z exportu zboží a služeb)
- ◆ Export high-tech technologií (% z vyvážených výrobků)
- ◆ Výdaje na armádu a zbrojení (% HNP)
- ◆ Výdaje na výzk. a výv. (% HNP)
- ◆ Celk. výd. na zdrav. (% HDP)
- ◆ Veř. výd. na školst. (% HNP)
- ◆ Očekávaná délka života u mužů
- ◆ Plodnost
- ◆ GINI-index (rozdělení příjmů a spotřeby)
- ◆ Uživ. internetu na 10000 obyvatel
- ◆ Počet mobilních telefonů na 1000 obyvatel
- ◆ HNP na obyvatele podle parity kupní síly (PPP)
- ◆ HNP na obyvatele (v USD)
- ◆ Růst HDP (% na obyvatele)

Analýza dat ze Světové banky: předzpracování



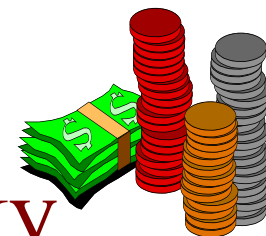
- ◆ 99 států se 16 WDI-indikátory
- ◆ po složkách transformace vzorů do intervalu (0,1) pomocí:

$$x' = \frac{x - x_{\min}}{x_{\max} - x_{\min}} \quad \text{a} \quad x'' = \frac{1}{1 + e^{-4(x' - 1/2)}}$$

↑
maximum přes všechny vzory minimum přes všechny vzory

- ◆ FCM-klastrování: 7 shluků, $s = 1.4$
- ◆ řízené učení a iterativní rozpoznávání:
 - 99 (90+9) států s 14 (13+1) WDI-indikátory
 - GREN-sít' 14-12-1, BP-sít' 13-10-1; 500-600 cyklů učení

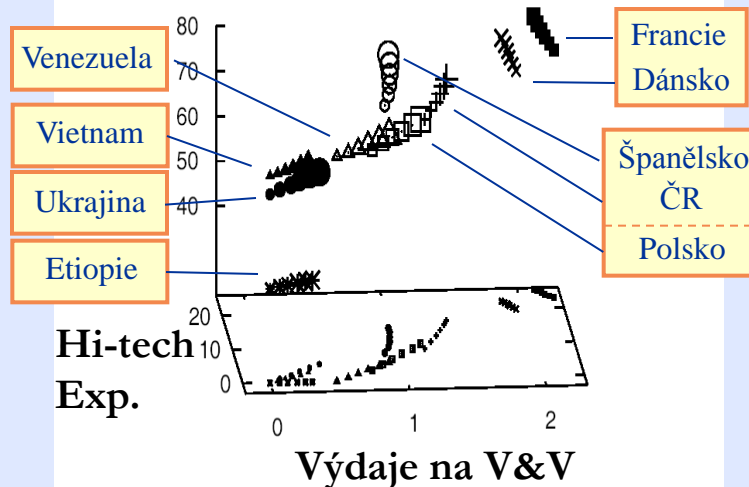
Analýza dat ze Světové banky: vliv indikátorů na stav ekonomiky



Indikátor	Síť 1	Síť 2
GDP defl.	0.0	0.0
Vněj. dluh	5.6	10.9
Celk. nákl. na dluh	5.5	8.1
Export high-tech	12.2	6.6
Vojenské výdaje	5.4	6.1
Výdaje na výzk. a výv.	16.0	12.0
Uživ. internetu	11.1	12.4
Mobily	8.3	10.0
GINI-index	7.1	3.9
Oček. délka života	12.3	7.6
Plodnost	4.4	5.0
Výdaje na zdrav.	6.1	10.9
Veř. výd. na školstv.	6.1	6.1

Relativní citlivost GREN-sítí

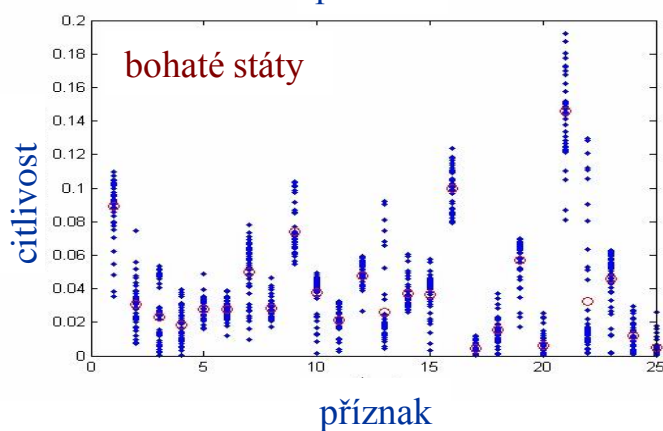
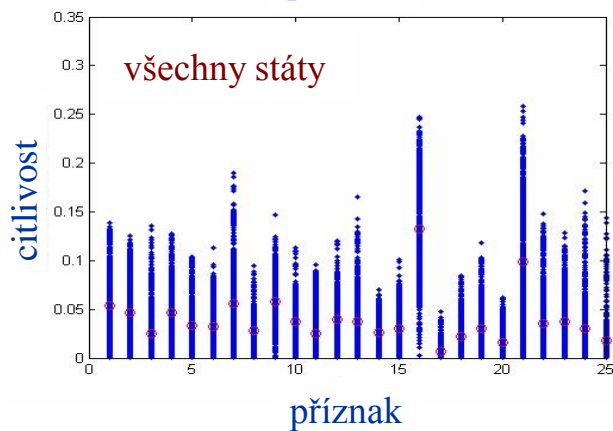
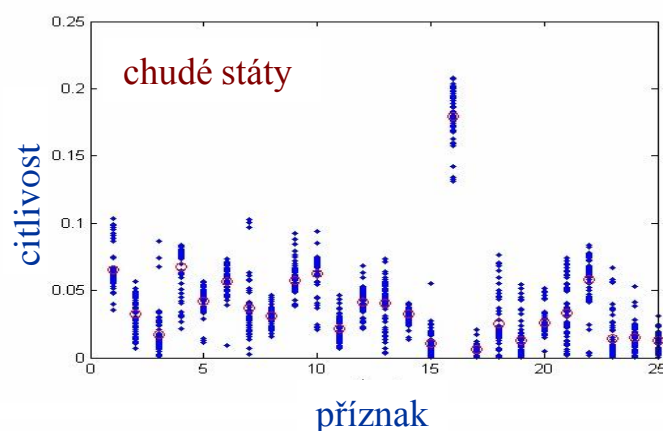
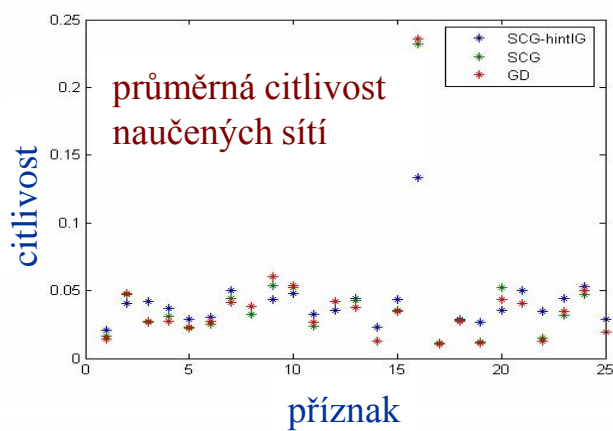
Očekávaná délka života



Iterativní rozpoznávání – vyšší
HNP podle PPP (Síť 1)

Citlivost na vstupní příznaky

(se Z. Reitermanovou)



Vzájemná závislost parametrů

(se Z. Reitermanovou)

