Dobývání znalostí

Doc. RNDr. Iveta Mrázová, CSc.

Katedra teoretické informatiky
Matematicko-fyzikální fakulta
Univerzity Karlovy v Praze

Dobývání znalostí

– Umělé neuronové sítě –

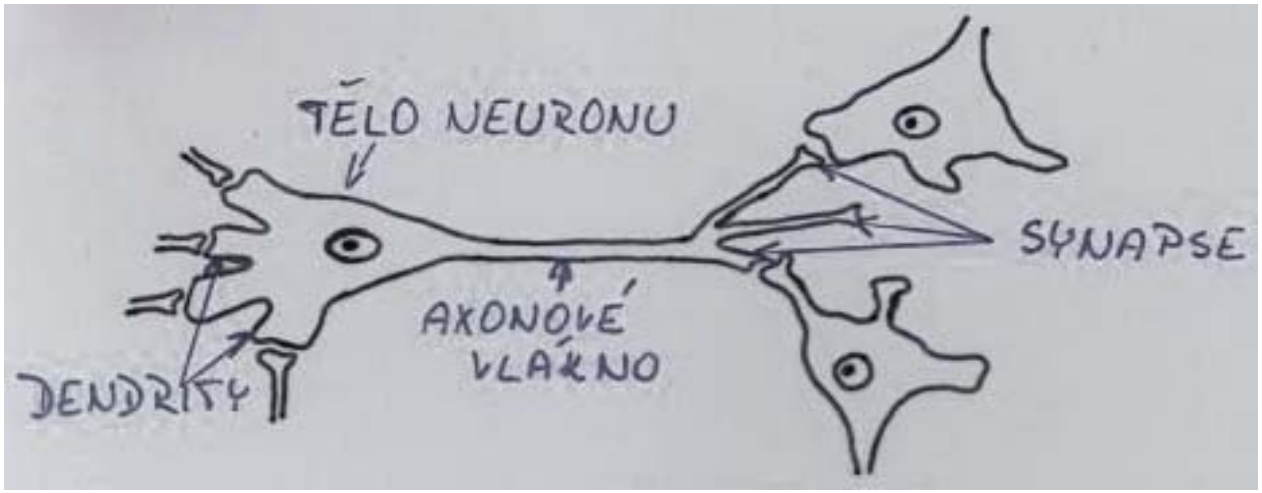
Doc. RNDr. Iveta Mrázová, CSc.

Katedra teoretické informatiky
Matematicko-fyzikální fakulta
Univerzity Karlovy v Praze

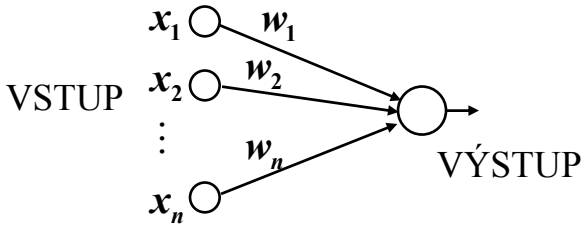
Základní biologické poznatky (1)

- **Model neuronu**
  - základní „výpočetní jednotka“ složitějšího celku – neuronové sítě
  - biologický neuron se skládá z:

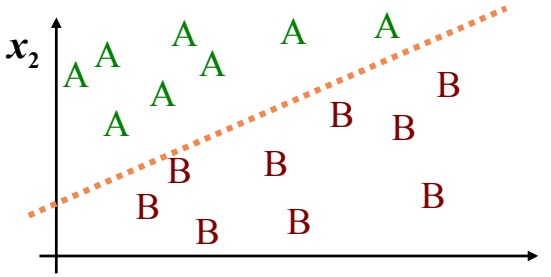
    těla (somatu), dendritů, axonu a synapsí



Formální neuron



$$y = f_h\left(\sum_{i=1}^{n} w_i x_i + \vartheta\right)$$



$$x_2 = -\frac{w_1}{w_2} x_1 - \frac{\vartheta}{w_2}$$

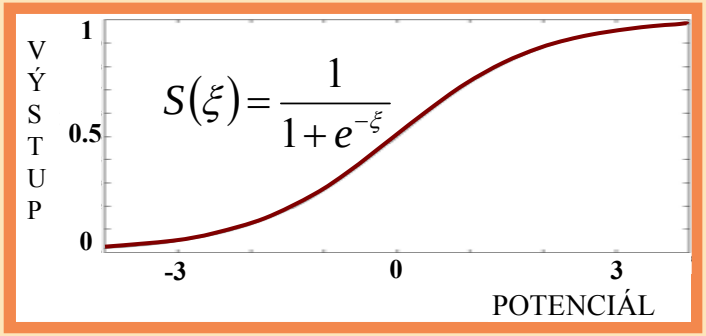
$$y = \begin{cases} 1 & if \sum_{i=1}^{n} w_i x_i + \vartheta \geq 0 : \text{ TŘÍDA A} \\ 0 & if \sum_{i=1}^{n} w_i x_i + \vartheta < 0 : \text{ TŘÍDA B} \end{cases}$$

Typy přenosových funkcí

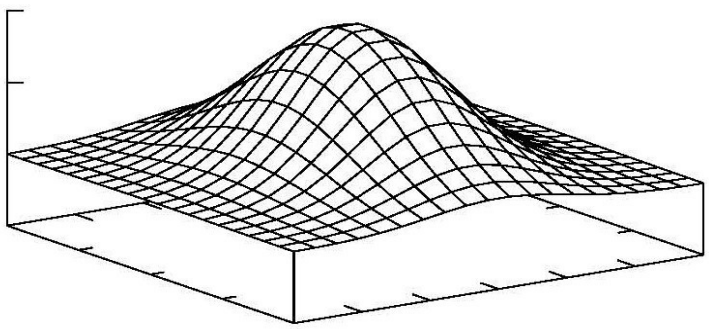
Skoková

$$y = \begin{cases} 1 & if \sum_{i=1}^{n} w_i x_i + \vartheta \geq 0 : \text{TŘÍDA A} \\ 0 & if \sum_{i=1}^{n} w_i x_i + \vartheta < 0 : \text{TŘÍDA B} \end{cases}$$

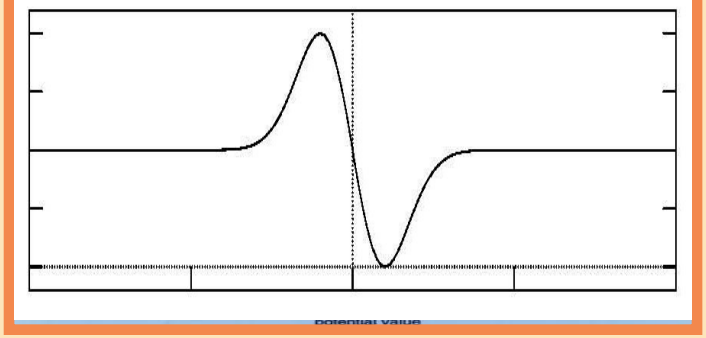
Sigmoidální



Radiální (RBF)



Waveletová



Definice formálního neuronu

Neuron s vahami $(w_1, \ldots, w_n) \in R^n$, prahem $\vartheta \in R$ a přenosovou funkcí $f : R^{n+1} \times R^n \longrightarrow R$ počítá pro libovolný vstup $\vec{z} \in R^n$ svůj výstup $y \in R$ jako hodnotu přenosové funkce v $\vec{z}$, $f[\vec{w}, \vartheta](\vec{z})$.

Nejčastěji se uvažuje tzv. sigmoidální přenosová funkce:

$$y = f[\vec{w}, \vartheta](\vec{z}) = f(\xi) = \frac{1}{1 + e^{-\xi}}$$

$$\xi = \sum_{i=1}^{n} z_i w_i + \vartheta$$ označuje tzv. potenciál neuronu, R množinu reálných čísel.

Definice stavů neuronu

Nechť $\vec{z}$ označuje vstup neuronu.

- Jestliže $f[\vec{w}, \vartheta](\vec{z}) = 1$, říkáme, že je neuron aktivní;
- Jestliže $f[\vec{w}, \vartheta](\vec{z}) = \frac{1}{2}$, říkáme, že je neuron tichý;
  Tato skutečnost znamená, že příslušný vstup leží v dělicí nadrovině určené tímto neuronem.
- Jestliže $f[\vec{w}, \vartheta](\vec{z}) = 0$, říkáme, že je neuron pasivní.

Učení a rozpoznávání

- **Učení:**
  - **S učitelem - trénovací množina** tvaru [vstup/požadovaný výstup]
  - **Bez učitele (samoorganizace)** - chybí požadovaný výstup

  ⇒ **Cíl: nastavení (adaptace) synaptických vah**

  (např. minimalizací střední kvadratické odchylky)

  Cílová funkce: např. $\sum_{p} \sum_{j} \left(y_{p,j} - d_{p,j} \right)^2$

  y je skutečný a d je požadovaný výstup

- **Rozpoznávání:**
  - **nově předkládaných vstupních vzorů**

  => Cíl: získat odezvu (výstup) neuronové sítě

Perceptron a lineární separabilita (1)

D: **Jednoduchý perceptron** je výpočetní jednotka s prahem ϑ, která pro n reálných vstupů $x_1, x_2, \ldots, x_n$ a váhy $w_1, \ldots, w_n$ dává výstup 1, jestliže platí nerovnost

$$\sum_{i=1}^{n} w_i \, x_i \geq \vartheta$$ (tzn. pokud $\vec{w} \cdot \vec{x} \geq \vartheta$) a 0 jinak.

Pozn.: Obdobně pro tzv. **rozšířený váhový a vstupní vektor**:

$$\vec{w} = (w_1, w_2, \ldots, w_n, w_{n+1}) \quad ; \quad w_{n+1} = -\vartheta$$
$$\vec{x} = (x_1, x_2, \ldots, x_n, 1)$$

=> výstup 1, jestliže $\vec{w} \cdot \vec{x} \geq 0$

Perceptron a lineární separabilita (2)

Lineární separabilita:

D: Dvě množiny ***A*** a ***B*** se nazývají **lineárně separabilní** v ***n***-rozměrném prostoru, pokud existuje ***n+1*** reálných čísel $w_1, \ldots, w_n, \vartheta$ takových, že každý bod $(x_1, x_2, \ldots, x_n) \in A$ splňuje $\sum_{i=1}^{n} w_i x_i \geq \vartheta$ a každý bod $(x_1, x_2, \ldots, x_n) \in B$ splňuje $\sum_{i=1}^{n} w_i x_i < \vartheta$

Perceptron a lineární separabilita (4)

Absolutní lineární separabilita:

D: Dvě množiny A a B se nazývají **absolutně lineárně separabilní** v n-rozměrném prostoru, pokud existuje $n+1$ reálných čísel $w_1, \ldots, w_n, \vartheta$ takových, že každý bod $(x_1, x_2, \ldots, x_n) \in A$ splňuje $\sum_{i=1}^{n} w_i x_i > \vartheta$ a každý bod $(x_1, x_2, \ldots, x_n) \in B$ splňuje $\sum_{i=1}^{n} w_i x_i < \vartheta$

Dělicí nadrovina – pro rozšířený váhový, resp. příznakový prostor (2)

D: Dělicí nadrovina určená ***n*** – rozměrným váhovým vektorem $\vec{w}$ je množina všech bodů $\vec{x} \in R^n$, pro které $\vec{w} \cdot \vec{x} = 0$

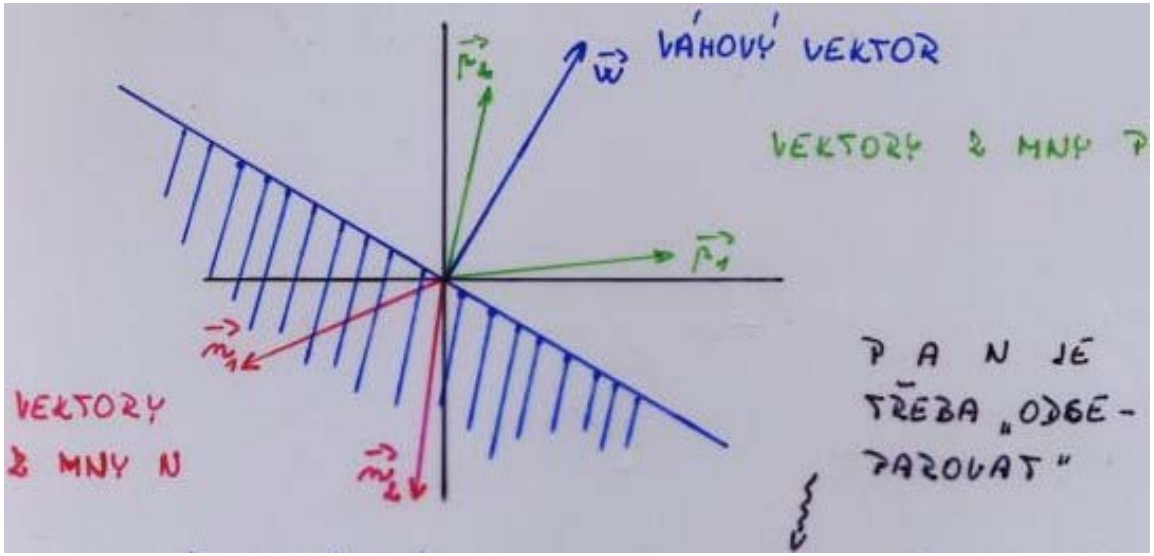
Problém: Nalézt takové váhy, resp. práh, které by umožnily separaci (oddělení) dvou množin vzorů

=> např. **PERCEPTRONOVÝ ALGORITMUS UČENÍ**

Předpoklad:

- ***A*** … množina vstupních vektorů v ***n*** –rozměrném prostoru
- ***B*** … množina vstupních vektorů v ***n*** –rozměrném prostoru

Perceptronový algoritmus učení (1)



Hledáme váhový vektor $\vec{w}$ s pozitivním skalárním součinem pro všechny vektory reprezentované body v ***P*** a se záporným skalárním součinem pro všechny vektory reprezentované body v ***N***

Perceptronový algoritmus učení (2)

⇒ **OBECNĚ:** za předpokladu, že ***P*** a ***N*** jsou množiny ***n*** – rozměrných vektorů chceme nalézt takový váhový vektor $\vec{w}$, že:

$$\vec{w} \cdot \vec{x} > 0 \quad \forall \vec{x} \in P$$

$$\vec{w} \cdot \vec{x} < 0 \quad \forall \vec{x} \in N$$

- Perceptronový algoritmus učení začíná s náhodně zvoleným váhovým vektorem $\vec{w}_0$
- Pokud existuje vektor $\vec{x} \in P$ takový, že $\vec{w} \cdot \vec{x} < 0$, znamená to, že úhel mezi těmito dvěma vektory je větší než 90°

  → Váhový vektor je nutné zadaptovat (~ otočit) ve směru $\vec{x}$ (tak, aby se tento vektor dostal do „pozitivního“ poloprostoru definovaného $\vec{w}$

Perceptronový algoritmus učení (3)

→ Otočení ve směru $\vec{x}$ lze provést přičtením $\vec{x}$ k vektoru $\vec{w}$

- Pokud existuje vektor $\vec{x} \in N$ takový, že $\vec{w} \cdot \vec{x} > 0$, znamená to, že úhel mezi těmito dvěma vektory je menší než 90°
  - → Váhový vektor je nutné zadaptovat (~ otočit) směrem od $\vec{x}$ (tak, aby se tento vektor dostal do „negativního“ poloprostoru definovaného $\vec{w}$
  - → Otočení směrem od $\vec{x}$ lze provést odečtením $\vec{x}$ od vektoru $\vec{w}$
- vektory z ***P*** tedy otáčejí váhový vektor opačným směrem než vektory z ***N***
- Pokud existuje řešení, lze ho nalézt v konečném počtu kroků

Perceptronový algoritmus učení (4)

Krok 1: Inicializace vah malými náhodnými hodnotami $w_i(0)$

$w_i(0)$ … váha vstupu i v čase 0 ; $(1 \le i \le n+1)$

Krok 2: Předložení trénovacího vzoru ve tvaru

$(x_1, \ldots, x_{n+1})$ … vstupní vzor a

$d(t)$ …….. požadovaný výstup (pro předložený vstup)

Krok 3: Výpočet skutečného výstupu (odezvy sítě)

$$y(t) = \operatorname{sgn}\left(\sum_{i=1}^{n+1} w_i(t)\, x_i(t) \right)$$

Krok 4: Adaptace vah podle:

$w_i(t+1) = w_i(t)$	výstup je správný
$w_i(t+1) = w_i(t) + x_i(t)$	výstup je 0 a měl být 1
$w_i(t+1) = w_i(t) - x_i(t)$	výstup je 1 a měl být 0

Krok 5: Pokud t nedosáhl požadované hodnoty, přejdi ke Kroku 2

Perceptronový algoritmus učení (5)

- **Heuristika pro počáteční nastavení vah:**
  Začít s průměrem „pozitivních“ vstupních vektorů minus průměr „negativních“ vektorů

- **Modifikace:** **parametr učení** η $(0 \leq \eta \leq 1)$
  (stupeň adaptivity vah ~ plasticita sítě)

  - Adaptace vah podle:

$w_i(t+1) = w_i(t)$	výstup je správný
$w_i(t+1) = w_i(t) + \eta\, x_i(t)$	výstup je 0 a měl být 1
$w_i(t+1) = w_i(t) - \eta\, x_i(t)$	výstup je 1 a měl být 0

Konvergence perceptronového algoritmu učení (Rosenblatt, 1959)

V: Nechť P a N jsou konečné a lineárně separabilní množiny. Potom provede perceptronový algoritmus učení **konečný počet aktualizací** váhového vektoru $\vec{w}_t$.

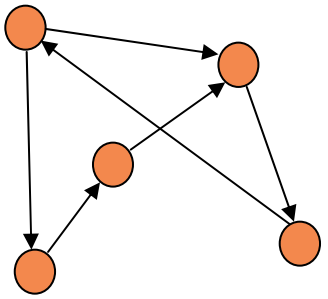
(Pokud se budou cyklicky testovat jeden po druhém vzory z P a N, najde perceptronový algoritmus učení po provedení konečného počtu aktualizací váhový vektor, pomocí něhož lze navzájem separovat P a N.)

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)** ***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)

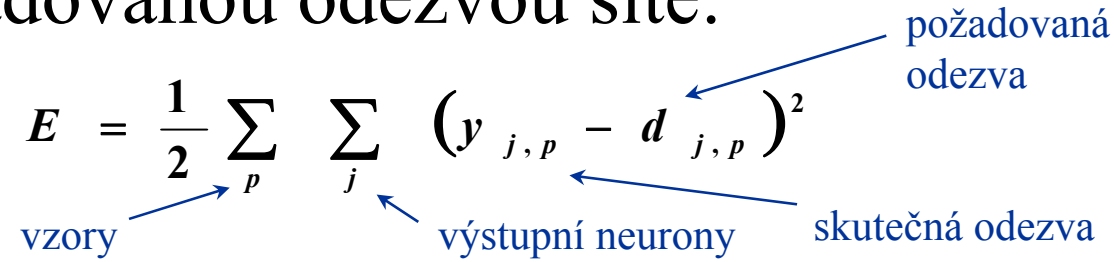
Cíl: najít takovou matici vah, která by zaručovala pro všechny vstupní vzory z trénovací množiny to, že skutečný výstup neuronové sítě bude stejný jako její požadovaný výstup

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

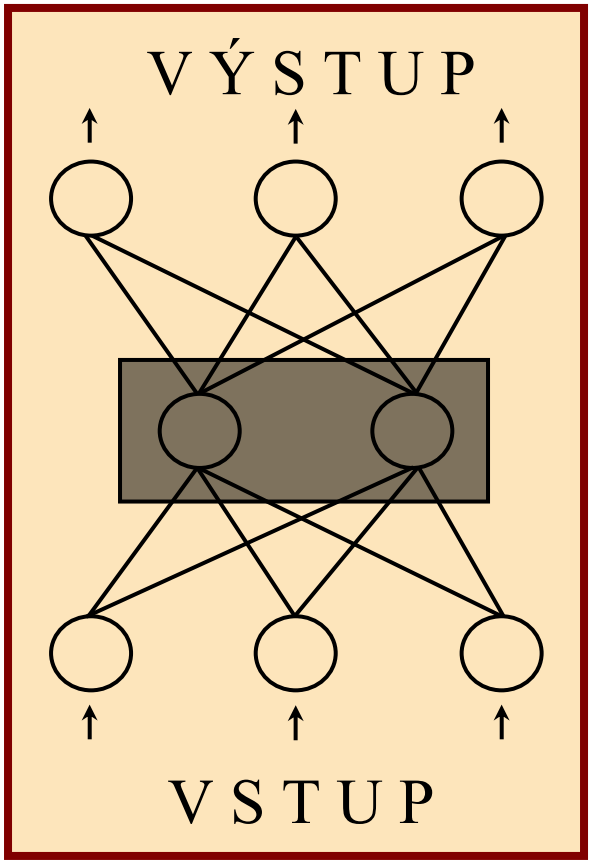
- vyjadřuje odchylku mezi skutečnou a požadovanou odezvou sítě:



- 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

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

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:

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

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 ($0 \le \alpha , \alpha_m \le 1$)

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

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

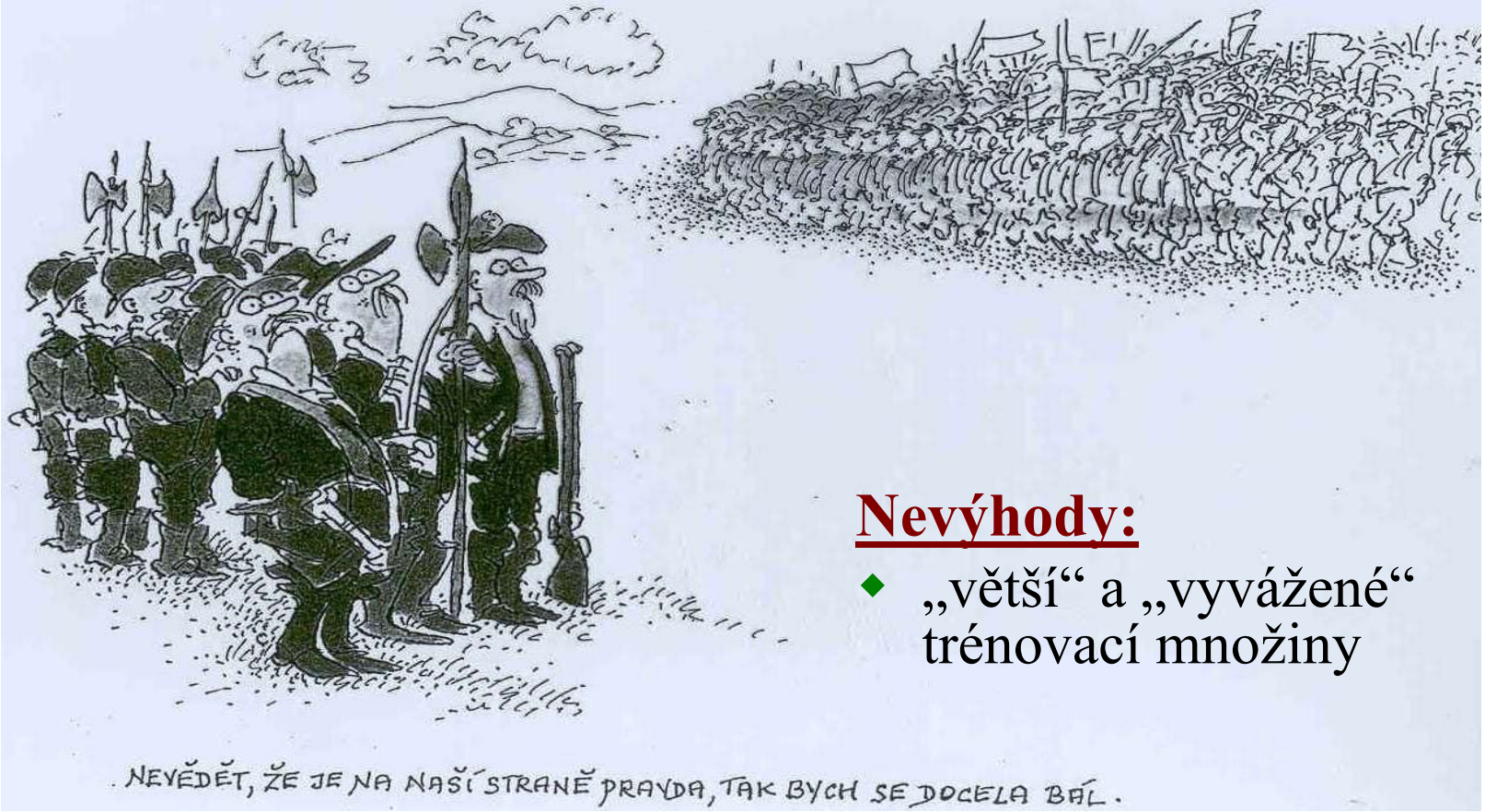
λ ..... strmost přenosové funkce

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

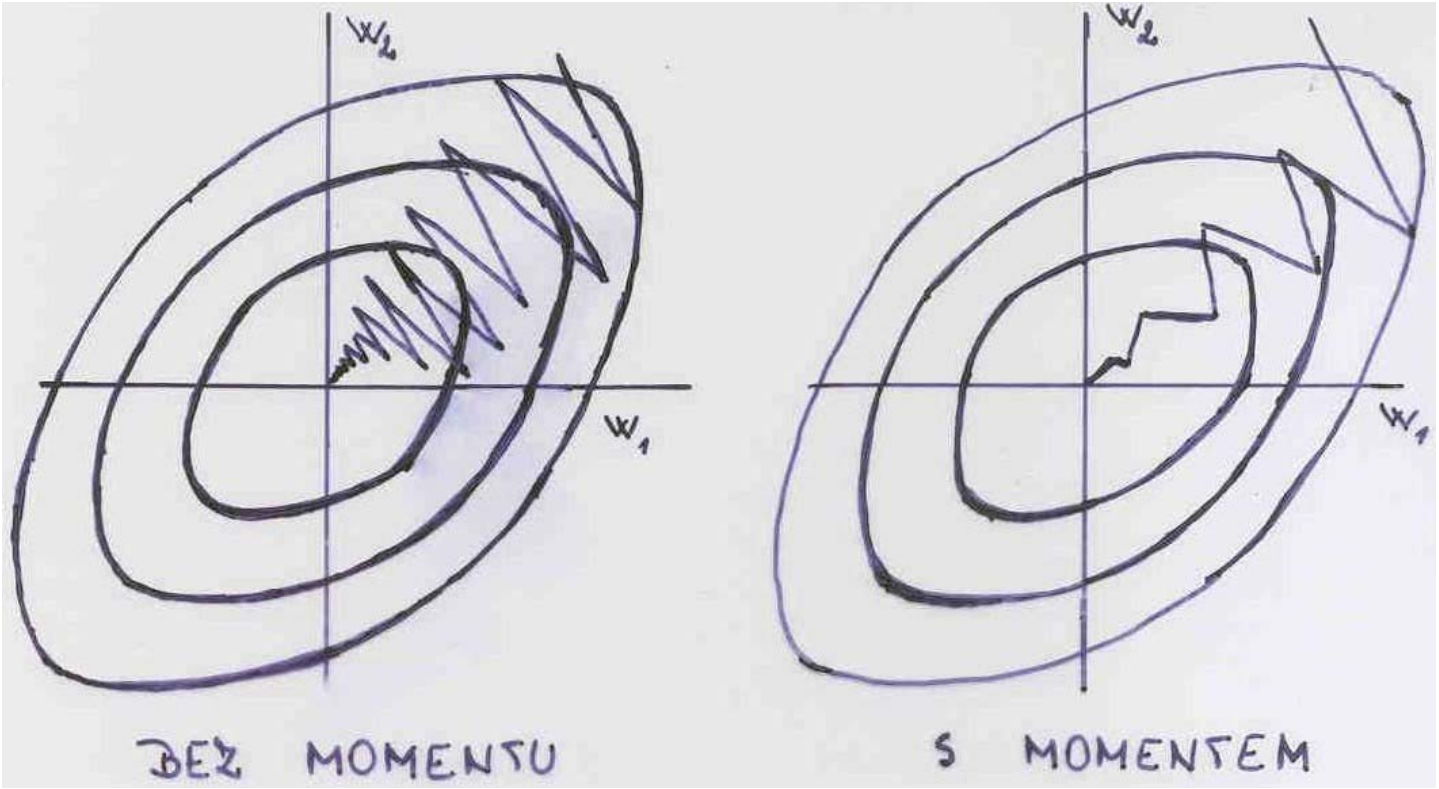
Algoritmus zpětného šíření a urychlení učení (2)

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 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: 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 (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 $(i = 1, \ldots, n)$ inicializace malými náhodnými hodnotami

Algoritmus Silvy & Almeidy (3)

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

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

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

2. Algoritmy druhého řádu:

- berou v úvahu více informací o tvaru chybové funkce než jen gradient → 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

→ 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 $\nabla E(\vec{w}+\vec{h})$):

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

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

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

==> **Newtonovské metody:**

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

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

- **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: $\left(\frac{\partial^2 E(\vec{w})}{\partial w_i^2} \right)$
- Ostatní prvky Hessovské matice jsou vynulovány
- Adaptace vah podle:

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

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:

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

- 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

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

gradient

Hessovská matice

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

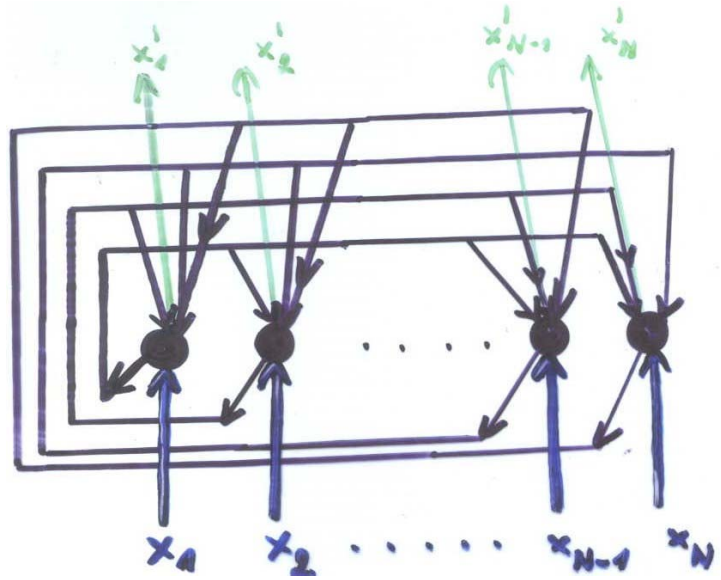
$$\frac{\partial^2 e}{\partial w_i \partial w_j} = 2\left[\frac{\partial y}{\partial w_i}\frac{\partial y}{\partial w_j} + \left(y - d\right)\frac{\partial^2 y}{\partial w_i \partial w_j}\right]$$

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

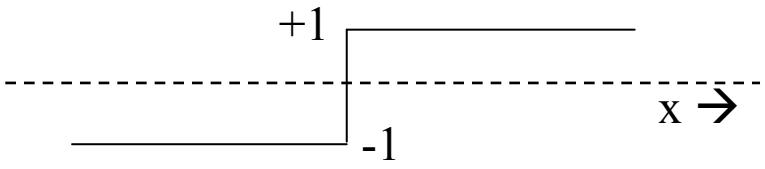
3. **Relaxační metody** – perturbace vah:

- 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

Hopfieldovy sítě



Skoková přenosová funkce: f_h



- ***n*** neuronů se skokovou přenosovou funkcí
- Bipolární vstupy i výstupy ***{ + 1 , - 1 }***
- Synaptické váhy w_{ij} (mezi všemi neurony navzájem)
- ***m*** trénovacích vzorů (tříd)
- Učení s učitelem
- Rozpoznávání
- Použití:
  - Asociativní paměť
  - Optimalizační úlohy

Hopfieldův model (bipolární)

Krok 1: Učení - nastavte hodnoty synaptických vah

$$w_{ij} = \begin{cases} \sum_{s=1}^{m} x_i^s x_j^s & \text{pro} \quad i \neq j \\ 0 & \text{pro} \quad i = j \end{cases}$$

w_{ij} …. Váha synapse mezi neurony i a j

$x_i^s \in \{-1,+1\}$ …. i – tá složka s – tého vzoru

$1 \leq i,j \leq n$

Hopfieldův model (bipolární) (2)

Krok 2: Inicializace - předložte neznámý vstupní vzor: $y_i(0) = x_i \qquad 1 \le i \le n$

$y_i(t)$ …. Výstup neuronu i v čase t

$x_i \in \{-1,+1\}$ …. i – tá složka předloženého vzoru

Krok 3: Iterace

$$y_j(t+1) = f_h\left[\sum_{i=1}^{n} w_{ij}\, y_i(t)\right] \qquad 1 \le j \le n$$

f_h …. Skoková přenosová funkce

Hopfieldův model (bipolární) (3)

Iterativní proces se při rozpoznávání opakuje, dokud se výstupy neuronů neustálí. Výstupy neuronů pak repre-zentují ten trénovací vzor, který nejlépe odpovídá předloženému (neznámému) vzoru.

Krok 4: Přejděte ke Kroku 2.

Hopfieldův model (bipolární) (4)

Konvergence (Hopfield):

- Symetrické váhy: $w_{ij} = w_{ji}$
- Asynchronní aktualizace výstupu jednotlivých neuronů

Nevýhody:

- Kapacita *(m < 0.15 n) n / 2 log n*
- Stabilita (→ ortogonalizace)

Hopfieldův model – příklad

Učení:

- Vzory: [-1, -1, 1, 1]
  [1, -1, 1, -1]

- Nastavení vah:

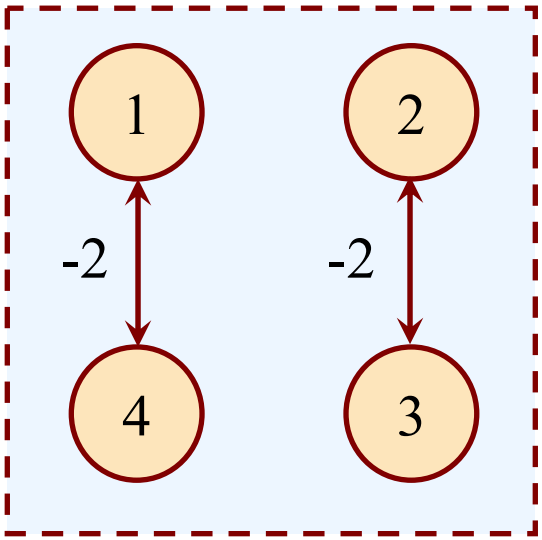
$$w_{ij} = \sum_{m=1}^{M} x_i^{(m)} \, x_j^{(m)} \qquad i \neq j$$

$$w_{ij} = 0 \qquad i = j$$

Hopfieldův model – příklad (2)

- Nastavení vah:

$$W = \begin{bmatrix} 0 & 0 & 0 & -2 \\ 0 & 0 & -2 & 0 \\ 0 & -2 & 0 & 0 \\ -2 & 0 & 0 & 0 \end{bmatrix}$$



Rozpoznávání:

- Vzor: [-1, -1, 1, -1]

[1, -1, 1, -1] [-1, -1, 1, 1]

Hopfieldův model - rozpoznávání

Po předložení vzoru $\vec{x}_1$ bude vektor potenciálů sítě

$$\vec{\xi} = \vec{x}_1 \cdot W = \vec{x}_1 \cdot \left(\vec{x}_1^T \vec{x}_1 + \ldots + \vec{x}_m^T \vec{x}_m - mI \right) =$$

$$= \underbrace{\vec{x}_1 \vec{x}_1^T}_{=n} \vec{x}_1 + \underbrace{\vec{x}_1 \vec{x}_2^T}_{=\alpha_{12}} \vec{x}_2 + \ldots + \underbrace{\vec{x}_1 \vec{x}_m^T}_{=\alpha_{1m}} \vec{x}_m - m\vec{x}_1 I =$$

$$= \left(n - m\right)\vec{x}_1 + \underbrace{\sum_{j=2}^{m} \alpha_{1j} \vec{x}_j}_{PERTURBACE}$$

Hopfieldův model – rozpoznávání (2)

$\alpha_{12}, \ldots, \alpha_{1m}$ …. Skalární součin $\vec{x}_1$ s každým dalším vektorem $\vec{x}_2, \ldots, \vec{x}_m$

→ **Stav $\vec{x}_1$ je stabilní, jestliže $m < n$ a perturbace $\sum_{j=2}^{m} \alpha_{1j} \vec{x}_j$ je malá**

$$\left(\Rightarrow \operatorname{sgn}\left(\vec{\xi} \right) = \operatorname{sgn}\left(\vec{x}_1 \right) \right)$$

➔ **Malý počet ortogonálních vzorů**

Hopfieldův model – rozpoznávání (3)

- Stav neuronů zachován, dokud nejsou vybrány k aktualizaci
- Výběr pro aktualizaci se provádí náhodně
- Neurony jsou navzájem plně propojeny
- Symetrické váhy: $w_{ij} = w_{ji}$
- $w_{ii} = 0$

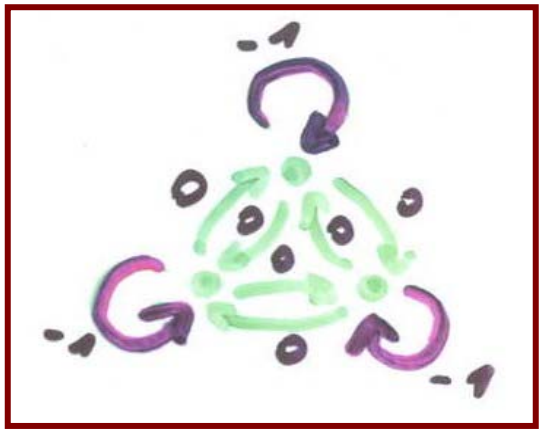
→ Konvergence ke stabilnímu řešení při rozpoznávání
- nutná podmínka:

symetrická váhová matice s nulovou diagonálou a asynchronní dynamikou

Hopfieldův model - příklady

- Váhová matice s nenulovou diagonálou nemusí vést ke stabilním stavům

$$W = \begin{pmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{pmatrix}$$

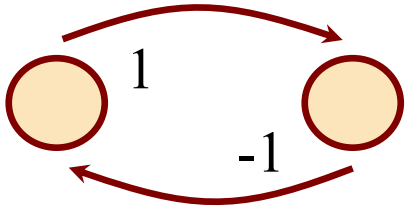


  - Synchronní dynamika: (-1, -1, -1) ↔ (1, 1, 1)
  - Asynchronní dynamika:
    - Náhodný výběr jednoho z osmi možných vzorů

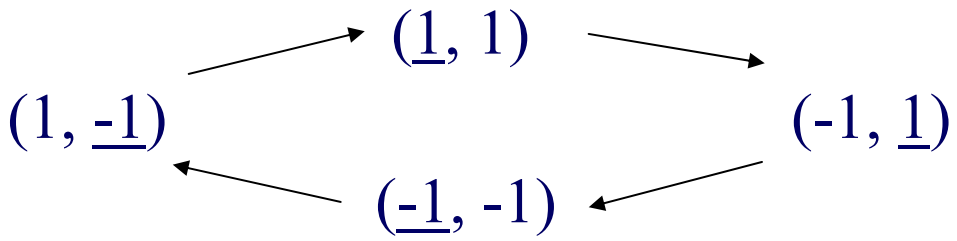
Hopfieldův model – příklady (2)

- Nesymetrická matice:

$$W = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$$



  - Asynchronní dynamika:



cyklické změny

Energetická funkce

Energetická funkce $E(\vec{x})$ Hopfieldovy sítě s *n* neurony a váhovou maticí *W* vyjadřuje energii sítě ve stavu $\vec{x}$:

$$E(\vec{x}) = -\frac{1}{2}\,\vec{x}\,W\,\vec{x}^{T} = -\frac{1}{2}\sum_{j=1}^{n}\sum_{i=1}^{n} w_{ij}\,x_i\,x_j$$

(Obdobně i pro sítě s prahovými neurony:

$$E(\vec{x}) = -\frac{1}{2}\,\vec{x}\,W\,\vec{x}^{T} + \vec{\theta}\,\vec{x}^{T} =$$

$$= -\frac{1}{2}\sum_{j=1}^{n}\sum_{i=1}^{n} w_{ij}\,x_i\,x_j + \sum_{i=1}^{n}\vartheta_i\,x_i \qquad)$$

Energetická funkce (2)

Věta:

Hopfieldova síť s asynchronní dynamikou dosáhne z libovolného počátečního stavu sítě stabilního stavu v lokálním minimu energetické funkce.

Idea důkazu:

- Počáteční stav
  - Předložený vzor: $\vec{x} = (x_1, \ldots, \underline{x_k}, \ldots, x_n)$

Energetická funkce (3)

Idea důkazu (pokračování):

$$E(\vec{x}) = -\frac{1}{2}\sum_{j=1}^{n}\sum_{i=1}^{n} w_{ij}\, x_i\, x_j$$

K aktualizaci vybrán neuron ***k***

- ***k*** nezmění svůj stav $\rightarrow$ $E(\vec{x})$ se nezmění
- ***k*** změní svůj stav $\rightarrow$ $\vec{x}' = \left(x_1, \ldots, \underline{x_k}', \ldots, x_n \right)$

Energetická funkce (4)

Idea důkazu (pokračování):

$$E(\vec{x}') = -\frac{1}{2}\sum_{\substack{j=1\\ j\neq k}}^{n}\sum_{\substack{i=1\\ i\neq k}}^{n} w_{ij}\, x_i\, x_j -$$

symetrie vah

$$-\sum_{i=1}^{n} w_{ik}\, x_i\, x_k' = \begin{cases} -\frac{1}{2}\sum_{j=1}^{n} w_{kj}\, x_k'\, x_j \\ -\frac{1}{2}\sum_{i=1}^{n} w_{ik}\, x_i\, x_k' \end{cases}$$

Energetická funkce (5)

Idea důkazu (pokračování):

- Rozdíl energií:

$$E(\vec{x}) - E(\vec{x}') = -\sum_{i=1}^{n} w_{ik}\, x_i\, x_k - \left(-\sum_{i=1}^{n} w_{ik}\, x_i\, x_k' \right) =$$

$$= -\underbrace{\left(x_k - x_k' \right)}_{\neq 0} \underbrace{\sum_{i=1}^{n} w_{ik}\, x_i}_{\text{POTENCIÁL}} > 0$$

$w_{kk} = 0$

různé znaménko (jinak by nedošlo ke změně stavu)

Energetická funkce (6)

Idea důkazu (pokračování):

→ **Vždy, když dojde ke změně stavu neuronu, sníží se celková energie sítě**

- Konečný počet možných stavů

→ **Stabilní stav, kdy energii sítě už nelze snižovat**

QED

Stochastické modely neuronových sítí

- Hopfieldův model se používá k řešení optimalizačních problémů, které lze vyjádřit ve formě minimalizované energetické funkce (i když není zaručeno nalezení globálního optima)
- **Problém:** zabránit „uvíznutí“ v lokálním minimu energetické funkce

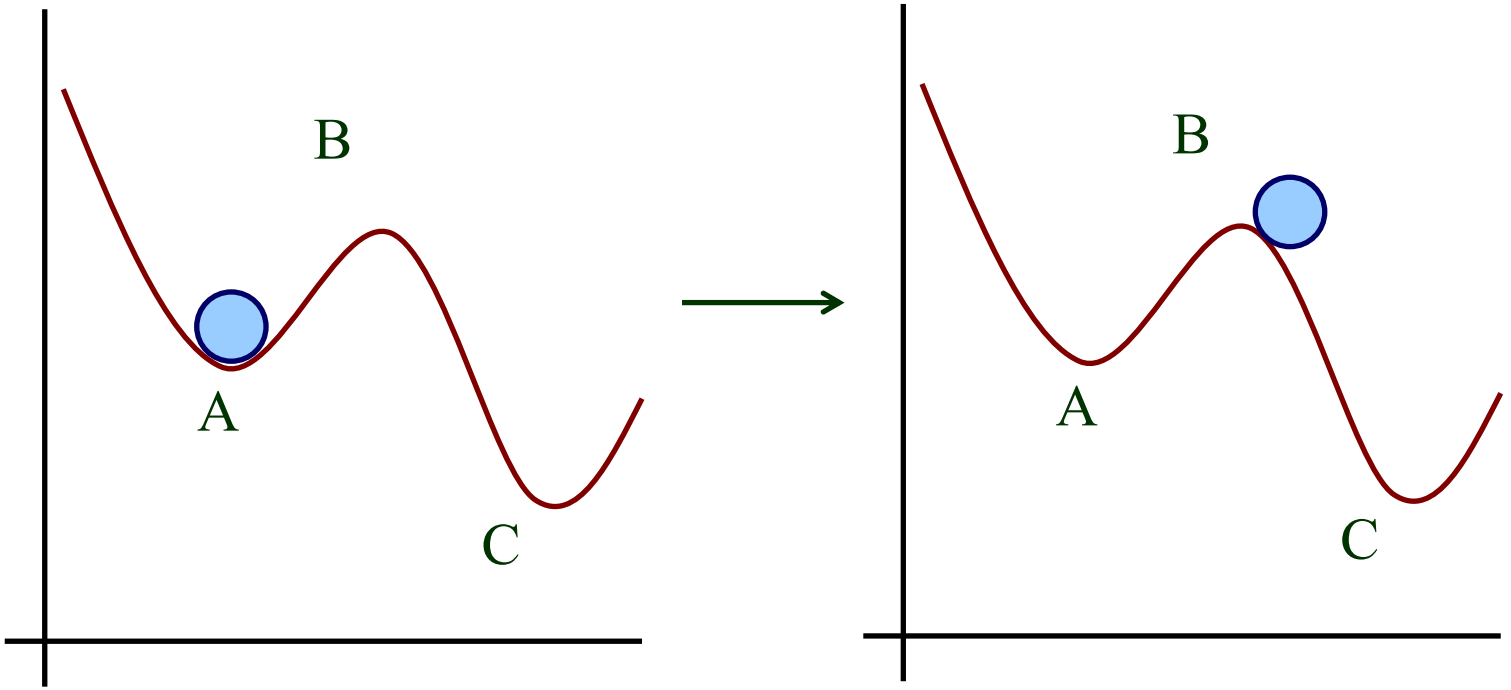
Stochastické modely neuronových sítí (2)

Modifikace Hopfieldova modelu:

1. strategie: zvětšení počtu možných cest k řešení ve stavovém prostoru
   → dovolit i stavy ve formě reálných hodnot (sigmoidální přenosová funkce)
   ==> **spojitý model**

2. strategie: omezení lokálních minim energetické funkce pomocí „zašuměné dynamiky sítě“
   → dočasné povolení aktualizace stavu sítě i za cenu přechodného zvýšení energetické hladiny
   ==> **simulované žíhání, Boltzmannův stroj**

Simulované žíhání



Simulované žíhání (2)

- Při minimalizaci energetické funkce E se tento jev simuluje následujícím způsobem:
  - Hodnota proměnné x se změní vždy, když může aktualizace Δx zmenšit hodnotu energetické funkce E
  - Pokud by se při aktualizaci x naopak hodnota E zvýšila o ΔE, bude nová hodnota x (tj. $x + \Delta x$) přijata s pravděpodobností $p_{\Delta E}$:

$$p_{\Delta E} = \frac{1}{1 + e^{\Delta E / T}}$$

kde T je tzv. teplotní konstanta

Simulované žíhání (3)

- Pro velké hodnoty T bude: $p_{\Delta E} \approx \frac{1}{2}$

  a aktualizace stavu nastane zhruba v polovině těchto případů

- Pro $T = 0$ bude docházet pouze k takovým aktualizacím, kdy se hodnota E sníží

- **Postupná změna hodnot T z velmi vysokých hodnot směrem k nule odpovídá zahřátí a postupnému ochlazování v procesu žíhání**

Simulované žíhání (4)

- **Navíc lze ukázat, že touto strategií lze dosáhnout (asymptoticky) globálního minima energetické funkce**
- **Sigmoida nejlépe odpovídá funkcím používaným v termodynamice** (pro analýzu teplotní rovnováhy)

Boltzmannův stroj

Definice:

Boltzmannův stroj je Hopfieldova síť, která se skládá z n neuronů se stavy $x_1, x_2, \ldots, x_n$.

Stav neuronu i se aktualizuje asynchronně podle pravidla:

$$x_i = \begin{cases} 1 & \text{s pstí } p_i \\ 0 & \text{s pstí } 1 - p_i \end{cases}$$

kde $$p_i = \frac{1}{1 + e^{-\left(\sum_{j=1}^{n} w_{ij} x_j - \vartheta_i \right) / T}}$$

Boltzmannův stroj (2)

Ve vztahu: $$p_i = \frac{1}{1 + e^{-\left(\sum_{j=1}^{n} w_{ij} x_j - \vartheta_i \right) / T}}$$

označuje T kladnou teplotní konstantu, w_{ij} váhy sítě a ϑ_i prahy neuronů

Energetická funkce Boltzmannova stroje:

$$E = -\frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij} x_i x_j + \sum_{i=1}^{n} \vartheta_i x_i$$

Boltzmannův stroj (3)

- Rozdíl mezi Boltzmannovým strojem a Hopfieldovým modelem spočívá ve stochastické aktivaci neuronů
- Pokud je ***T*** **velmi malé**, bude $p_i \sim 1$, jestliže je

$$\sum_{j=1}^{n} w_{ij}\, x_j - \vartheta_i \; > \; 0$$

  - × pokud je excitace neuronu záporná, bude $p_i \sim 0$

→ dynamika Boltzmannova stroje aproximuje dynamiku diskrétní Hopfieldovy sítě a Boltzmannův stroj najde lokální minimum energetické funkce

Boltzmannův stroj (4)

- Pro je $T > 0$ je pravděpodobnost změny anebo posloupnosti změn ze stavu $x_1, \ldots, x_n$ do jiného stavu vždy nenulová

  → Boltzmannův stroj nezůstane v jediném stavu

  – snižování a zároveň možnost zvyšování energie systému

- Pro **veliké hodnoty** T projde síť téměř celý stavový prostor

  × V ochlazovací fázi má síť tendenci zůstávat déle v oblastech blízkých atraktorům lokálních minim

Boltzmannův stroj (5)

Pokud se teplota snižuje správným způsobem, můžeme očekávat, že systém dosáhne globálního minima *s pravděpodobností 1*

Učení bez učitele

- **Učení bez učitele:**
  - Samoorganizace a shlukování
- **Motivace:**
  - Síť sama rozhodne, která odezva je pro daný vzor nejlepší a podle toho nastaví své váhy
- **Problém:**
  - Určit počet a rozložení shluků v příznakovém prostoru

Učení bez učitele (2)

Učení bez učitele (3)

Učení bez učitele (4)

Kompetiční učení:

- Boj o „právo reprezentovat předložený vzor“
- „Potlačování soupeřů“ → **INHIBICE**
- Pravidlo „**vítěz bere vše**“
  (WTA – Winner_takes_all)

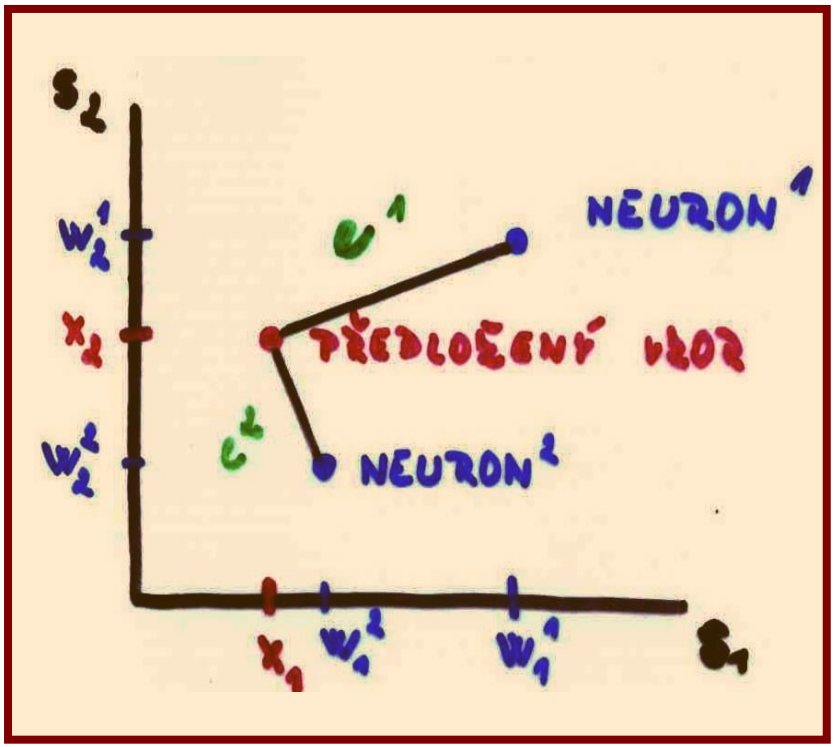
Posilované učení (reinforcement):

- Důraz na co nejlepší reprodukci vstupů

Kompetiční učení bez učitele

- ***n*** – rozměrný vstupní vzor je zpracováván pomocí takového počtu neuronů, který odpovídá (předpokládanému) počtu shluků
- Neurony v tomto případě počítají (Euklidovskou) **vzdálenost mezi předloženým vzorem a** svým **váhovým vektorem**

Kompetiční učení bez učitele (2)



- V kompetici „vítězí“ neuron, který ke nejblíže předloženému vzoru
- Vítězný neuron bude nejaktivnější a bude potlačovat – **inhibovat** – aktivitu ostatních neuronů

Kompetiční učení bez učitele (2a)

- Inhibice pomocí „laterálních spojů“

  ==> **laterální inhibice**

- Pro rozhodnutí, zda bude neuron aktivní nebo ne, je nutná globální informace o stavu všech neuronů v síti
- Aktivita neuronu signalizuje příslušnost předloženého vstupu ke shluku vektorů reprezentovaných tímto neuronem

Kompetiční učení bez učitele (3)

- Vítězný neuron zadaptuje své váhy směrem k předloženému vzoru:

$$\Delta \vec{w} = \alpha \cdot \left(\vec{x} - \vec{w} \right)$$

plasticita sítě (během učení pomalu klesá)

Cíl:

- Umístit neurony do středu shluků vzorů
- Zachovat již vytvořenou strukturu sítě

Kompetiční učení bez učitele (4)

- **Urychlení procesu učení:**
  - Vhodná inicializace vah
  - Např. podle náhodně vybraných vzorů
- **Problémy:**
  - Mrtvé (nevyužité) neurony
    - Mřížka v Kohonenově vrstvě
    - Topologické okolí neuronu
    - Řízená kompetice a mechanismus svědomí

Kompetiční učení bez učitele (5)

- Během učení by se měly váhy jednotlivých neuronů nastavit tak, aby odpovídaly „těžišti příslušného shluku“
- Energetická funkce množiny ***n***– rozměrných normovaných vstupních vzorů $X = \{\vec{x}_1, \ldots, \vec{x}_m\}; (n \geq 2)$ je pro ***1*** neuron s váhovým vektorem $\vec{w}$ dána pomocí:

$$E_X(\vec{w}) = \sum_{i=1}^{m} \left\| \vec{x}_i - \vec{w} \right\|^2 ; \quad \vec{w} \in R^n$$

Kompetiční učení bez učitele (6)

==> lze ukázat, že v optimálním případě je vektor vah umístěn v těžišti shluku vstupních vzorů

$$E_X(\vec{w}) = \sum_{i=1}^{m} \left\| \vec{x}_i - \vec{w} \right\|^2 = \sum_{i=1}^{m} \sum_{j=1}^{n} \left(x_{ij} - w_j \right)^2 =$$

$$= \sum_{i=1}^{m} \sum_{j=1}^{n} \left(x_{ij}^2 - 2 x_{ij} w_j + w_j^2 \right) =$$

$$= m \left(\sum_{j=1}^{n} w_j^2 - \frac{2}{m} \sum_{j=1}^{n} w_j \left(\sum_{i=1}^{m} x_{ij} \right) \right) + \sum_{i=1}^{m} \sum_{j=1}^{n} x_{ij}^2 =$$

Kompetiční učení bez učitele (7)

$$E_X(\vec{w}) = m\sum_{j=1}^{n}\left(w_j^2 - \frac{2}{m} w_j \left(\sum_{i=1}^{m} x_{ij}\right) + \frac{1}{m^2}\left(\sum_{i=1}^{m} x_{ij}\right)\left(\sum_{i=1}^{m} x_{ij}\right)\right) -$$

$$\underbrace{- \frac{1}{m}\sum_{j=1}^{n}\left(\sum_{i=1}^{m} x_{ij}\right)\left(\sum_{i=1}^{m} x_{ij}\right) + \sum_{i=1}^{m}\sum_{j=1}^{n} x_{ij}^2}_{= K} =$$

$$= m\left(\sum_{j=1}^{n}\left(w_j - \frac{1}{m}\sum_{i=1}^{m} x_{ij}\right)^2\right) + K =$$

$$= m \left\| \vec{w} - \vec{x}^* \right\|^2 + K$$

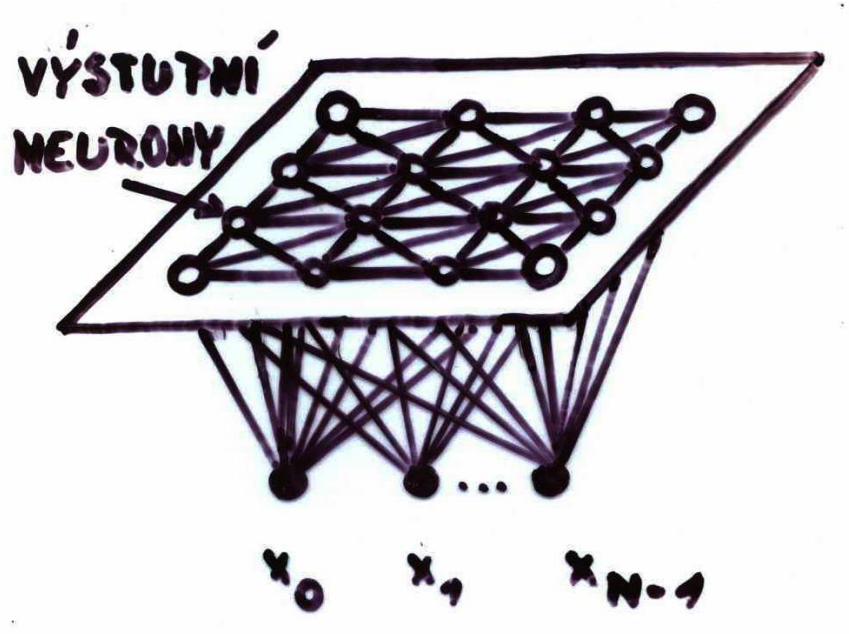
Kompetiční učení bez učitele (8)

→ vektor $\vec{x}^*$ představuje centroid shluku $\{\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_m\}$ a K je konstanta

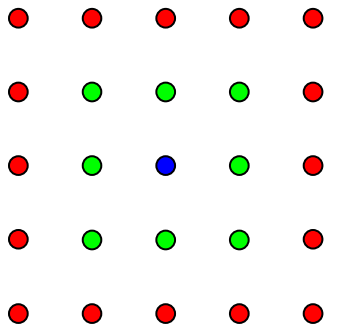
→ energetická funkce má globální minimum v $\vec{x}^*$

Kohonenovy mapy

- Teuvo Kohonen – fonetický psací stroj



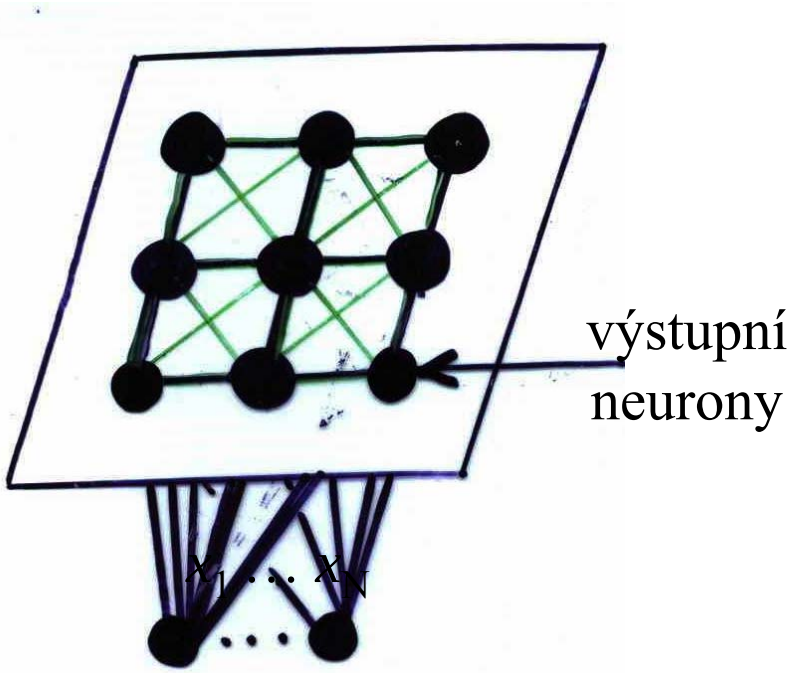
Topologické okolí



$NE_j(O)$

$NE_j(t_1)$

Kohonenovy mapy (2)



- **Učení: bez učitele**
- **Rozpoznávání**
- **Použití:**
  - Fonetický psací stroj
  - Ekonomie

Kohonenův model – algoritmus učení

Motivace:

- Mřížka, na níž jsou uspořádané neurony, umožňuje identifikaci nejbližších sousedů daného neuronu

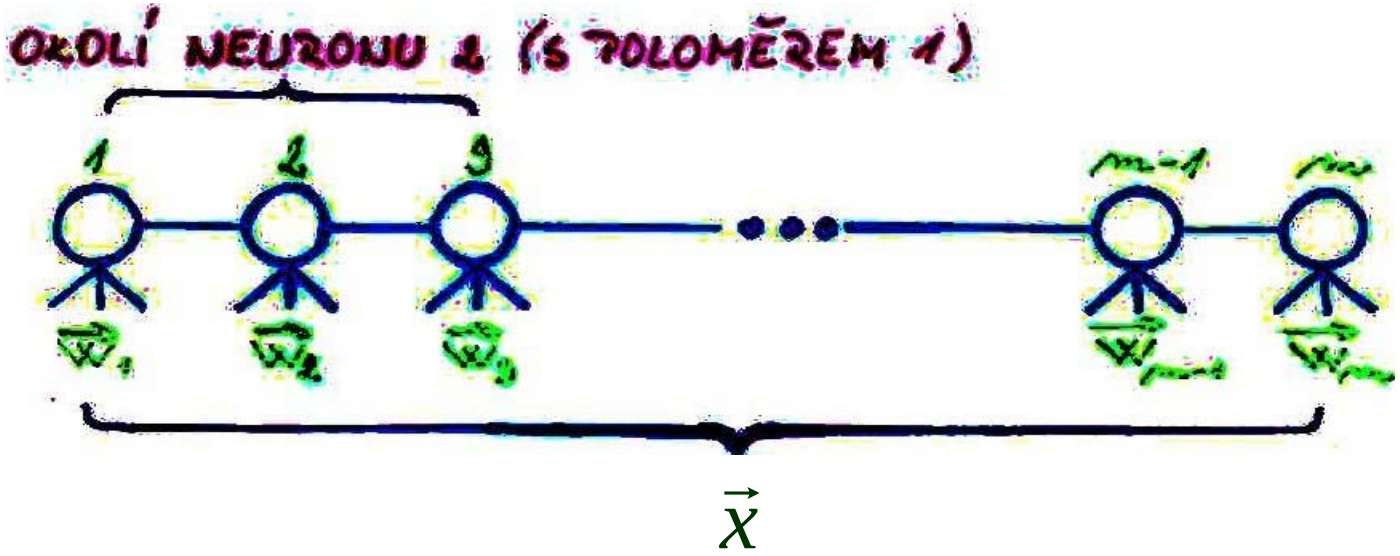
  → v průběhu učení se aktualizují váhy příslušných neuronů i jejich sousedů

  Cíl: sousední neurony by měly také reagovat na velmi podobné signály

Kohonenův model – algoritmus učení (2)

Problém (1-dim):

- Rozčlenění ***n*** – rozměrného prostoru pomocí jednorozměrného řetězce „Kohonenovských neuronů“
- Neurony uspořádané do posloupnosti a označené od ***1*** do ***n***



Kohonenův model – algoritmus učení (3)

Problém (1-dim – pokračování):

- Jednorozměrná mřížka neuronů:
  - Každý neuron „dostává“ ***n***–rozměrný vstup $\vec{x}$ a na základě ***n***– rozměrného váhového vektoru $\vec{w} = (w_1, \ldots, w_n)$ spočítá svou excitaci

Cíl: „specializace“ každého neuronu na jinou oblast vstupního prostoru (tuto „specializaci“ charakterizuje maximální excitace příslušného neuronu pro vzory z dané oblasti)

Kohonenův model – algoritmus učení (4)

Problém (1-dim – pokračování):

→ „Kohonenovské“ neurony počítají Euklidovskou vzdálenost mezi vstupem $\vec{x}$ a příslušným váhovým vektorem $\vec{w}$

→ „nejbližšímu“ neuronu bude odpovídat maximální excitace

Kohonenův model – algoritmus učení (5)

Definice okolí:

- V jednorozměrné Kohonenově mapě patří do okolí neuronu ***k*** s poloměrem ***1*** neurony ***k – 1*** a ***k + 1***
- Neurony na obou koncích jednorozměrné Kohonenovy mapy mají asymetrické okolí
- V ***1*** – rozměrné Kohonenově mapě patří do okolí neuronu ***k*** o poloměru ***r*** všechny neurony, které jsou od ***k*** vzdáleny až o ***r*** pozic doleva či doprava
- Obdobně pro vícerozměrné Kohonenovy mapy a zvolenou metriku na mřížce (čtvercová, hexagonální, …)

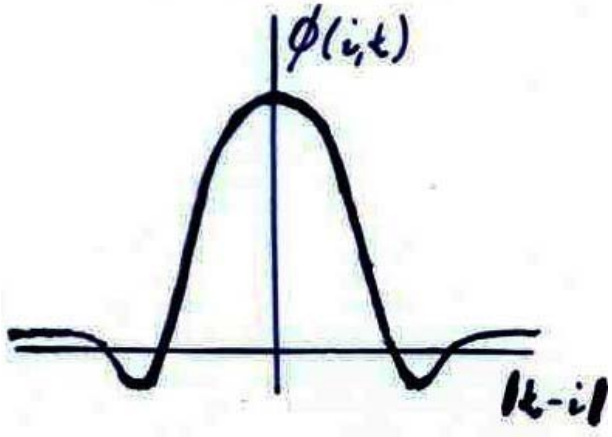
Kohonenův model – algoritmus učení (6)

Funkce laterální interakce $\Phi(i,k)$:

~ „síla laterální vazby“ mezi neurony ***i*** a ***k*** během učení

Příklad:

- $\Phi(i,k)=1 \quad \forall i$ z okolí k s poloměrem r a $\Phi(i,k)=0$ $\forall$ ostatní i
- Funkce „mexického klobouku“
- … a další …



Kohonenovy samoorganizující se příznakové mapy: algoritmus

Krok 1: Zvol hodnoty vah mezi N vstupmíni a M výstupními neurony jako malé náhodné hodnoty. Zvol počáteční poloměr okolí a funkci laterální interakce Φ.

Krok 2: Předlož nový trénovací vzor.

Krok 3: Spočítej vzdálenosti d_j mezi vstupním a váhovým vektorem pro každý výstupní neuron j pomocí:

$$d_j = \sum_{i=0}^{N-1} \left(x_i(t) - w_{ij}(t) \right)^2$$

Kde $x_i(t)$ je vstupem neuronu i v čase t a $w_{ij}(t)$ je váhou synapse ze vstupního neuronu i k výstupnímu neuronu j v čase t. Tuto vzdálenost lze upravit váhovým koeficientem a předat kompetiční vrstvě.

Kohonenovy samoorganizující se příznakové mapy: algoritmus (2)

Krok 4: Vyber (např. pomocí laterální interakce) takový výstupní neuron c, který má minimální d_j a označ ho jako „vítěze“.

Krok 5: Váhy se aktualizují pro neuron c a všechny neurony v okolí definovaném pomocí N_c. Nové váhy jsou:

$$w_{ij}(t+1) = w_{ij}(t) + \alpha(t)\, \Phi(c,j)\, (x_i(t) - w_{ij}(t))$$

Pro $j \in N_c$; $0 \le i \le N-1$

$\alpha(t)$ je vigilanční koeficient ($0 < \alpha(t) < 1$) , který klesá v čase (vigilance ~ bdělost) .

Kohonenovy samoorganizující se příznakové mapy: algoritmus (3)

Pro volbu $\alpha(t)$ by mělo platit:

$$\sum_{t=1}^{\infty} \alpha(t) = \infty \quad \wedge \quad \sum_{t=1}^{\infty} \alpha^2(t) < \infty$$

Při procesu učení tak vítězný neuron upraví svůj váhový vektor směrem k aktuálnímu vstupnímu vektoru.
Totéž platí pro neurony v okolí „vítěze“.
Hodnota funkce $\Phi(c,j)$ klesá s rostoucí vzdáleností neuronů od středu okolí N_c.

Krok 6: Přejdi ke Kroku 2.

Analýza konvergence ~ stabilita řešení a uspořádaný stav (6)

PROBLÉMY:

- „rozvinutí“ planární mřížky a podmínky, za kterých k němu dojde
- „metastabilní stavy“ a nevhodná volba funkce laterální interakce (příliš rychlý pokles)

→ **konvergence pro 1-dimenzionální případ za předpokladu rovnoměrného rozložení a adaptačního plavidla**

$$w_k^{new} = w_k^{old} + \gamma \left(\xi - w_k^{old} \right)$$

kde k označuje vítězný neuron a jeho dva sousedy (Cottrell & Fort, 1986)

Varianty algoritmu učení pro Kohonenovy mapy

Učení s učitelem:

(LVQ ~ Learning Vector Quantization)

LVQ1:

- Motivace:

  -) $\vec{x}$ by měl patřit ke stejné třídě jako nejbližší $\vec{w}_i$

- nechť $c = \arg\min_i \left\{ \left\| \vec{x} - \vec{w}_i \right\| \right\}$ je index $\vec{w}_i$ ležícího nejblíž k $\vec{x}$ (c ~ „vítězný“ neuron)

Varianty algoritmu učení pro Kohonenovy mapy (2)

LVQ1 (pokračování)**:**

→ **adaptační pravidla** ($0 < \alpha(t) < 1$):

$$\vec{w}_c(t+1) = \vec{w}_c(t) + \alpha(t)\left[\vec{x}(t) - \vec{w}_c(t)\right]$$

jestliže $\vec{x}$ a $\vec{w}_c$ jsou klasifikovány stejně

$$\vec{w}_c(t+1) = \vec{w}_c(t) - \alpha(t)\left[\vec{x}(t) - \vec{w}_c(t)\right]$$

jestliže $\vec{x}$ a $\vec{w}_c$ jsou klasifikovány jinak

$$\vec{w}_i(t+1) = \vec{w}_i(t) \quad \text{jestliže } i \neq c$$

Varianty algoritmu učení pro Kohonenovy mapy (3)

LVQ2.1:

- Motivace: adaptace dvou nejbližších sousedů $\vec{x}$ současně
  - Jeden z nich musí patřit ke správné třídě a druhý k nesprávné
  - Navíc musí být $\vec{x}$ z okolí dělicí nadplochy mezi $\vec{w}_i$ a $\vec{w}_j$ (~ z „okénka“)
  - Je-li d_i (resp. d_j) Euklidovská vzdálenost mezi $\vec{x}$ a $\vec{w}_i$ (resp. mezi $\vec{x}$ a $\vec{w}_j$), lze „okénko“ definovat pomocí vztahu:

$$\min\left(\frac{d_i}{d_j}, \frac{d_j}{d_i}\right) > s \quad , \quad \text{kde} \quad s = \frac{1-w}{1+w}$$

(doporučované hodnoty w (~ šířky „okénka“): ***0.2 – 0.3***)

Varianty algoritmu učení pro Kohonenovy mapy (4)

LVQ2.1 (pokračování):

→ **adaptační pravidla** ($0 < \alpha(t) < 1$):

- $\vec{w}_i(t+1) = \vec{w}_i(t) - \alpha(t)\left[\vec{x}(t) - \vec{w}_i(t)\right]$
- $\vec{w}_j(t+1) = \vec{w}_j(t) + \alpha(t)\left[\vec{x}(t) - \vec{w}_j(t)\right]$

- $\vec{w}_i$ a $\vec{w}_j$ leží nejblíže k $\vec{x}$
- přitom $\vec{x}$ a $\vec{w}_j$ patří ke stejné třídě
- a $\vec{x}$ a $\vec{w}_i$ patří k různým třídám
- $\vec{x}$ je z „okénka“

Varianty algoritmu učení pro Kohonenovy mapy (5)

LVQ3 (motivace):

- aproximace rozložení tříd a stabilizace řešení

→ **adaptační pravidla** ($0 < \alpha(t) < 1$):

- $\vec{w}_i(t+1) = \vec{w}_i(t) - \alpha(t)\left[\vec{x}(t) - \vec{w}_i(t)\right]$
- $\vec{w}_j(t+1) = \vec{w}_j(t) + \alpha(t)\left[\vec{x}(t) - \vec{w}_j(t)\right]$

$\vec{w}_i$ a $\vec{w}_j$ leží nejblíže k $\vec{x}$; přitom $\vec{x}$ a $\vec{w}_j$ patří ke stejné třídě a $\vec{x}$ a $\vec{w}_i$ patří k různým třídám a $\vec{x}$ je z „okénka“

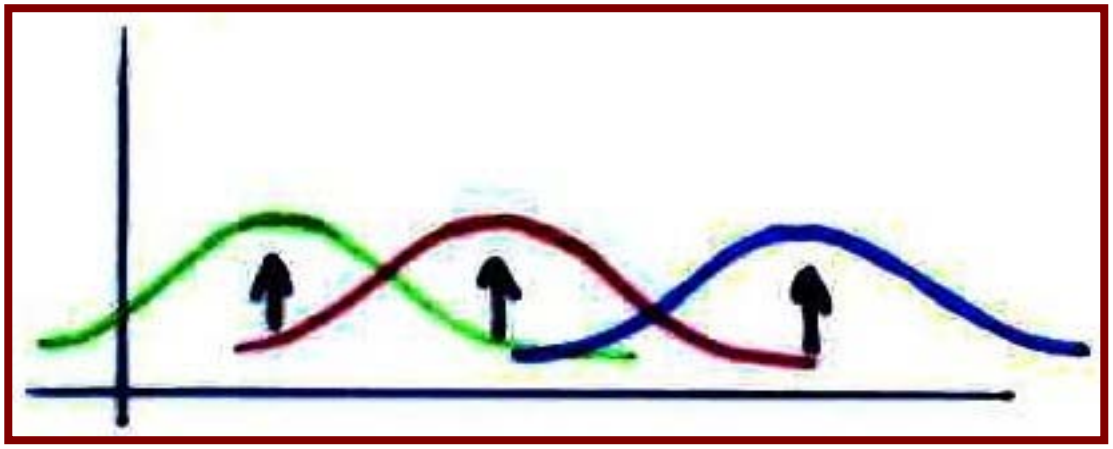
- $\vec{w}_k(t+1) = \vec{w}_k(t) + \varepsilon\alpha(t)\left[\vec{x}(t) - \vec{w}_k(t)\right]$

pro $k \in \{i, j\}$ jestliže $\vec{x}$, $\vec{w}_i$ i $\vec{w}_j$ patří do stejné třídy

Varianty algoritmu učení pro Kohonenovy mapy (6)

LVQ3 (pokračování)**:**

- volba parametrů:
  - $0.1 \leq \varepsilon \leq 0.5$
  - $0.2 \leq w \leq 0.3$

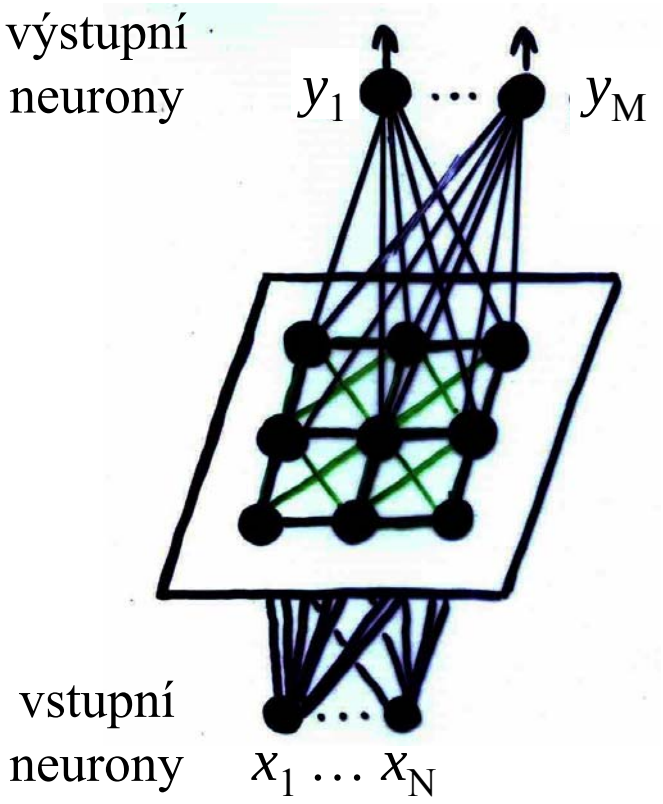


Varianty algoritmu učení pro Kohonenovy mapy (7)

Další varianty:

- Vícevrstvé Kohonenovy mapy
  - Strom abstrakce
- Sítě se vstřícným šířením (Counterpropagation)
  - Učení s učitelem – dvě fáze učení
    - Kohonenovská (klastrovací) vrstva
    - Grossbergovská vrstva (adaptace vah jen pro vítězné neurony z Kohonenovské vrstvy

Sítě se vstřícným šířením



Učení: s učitelem

Rozpoznávání

Použití:

- Heteroasociativní paměť

Učící algoritmus pro sítě se vstřícným šířením (1)

Krok 1: Zvolte náhodné hodnoty synaptických vah.

Krok 2: Předložte nový trénovací vzor ve tvaru

(vstup, požadovaný výstup).

Krok 3: Vyberte v Kohonenově vrstvě neuron *c*, jehož synaptické váhy nejlépe odpovídají předloženému vzoru $\vec{x}(t)$. Pro tento neuron tedy bude platit, že vzdálenost e_k mezi příslušným váhovým vektorem $\vec{v}_k(t)$ a předloženým vzorem $\vec{x}(t)$ je minimální. Použít lze např. Euklidovskou metriku, potom:

$$e_c = \min_k e_k = \min_k \sqrt{\sum_i \left(x_i(t) - v_{ik}(t)\right)^2}$$

Učící algoritmus pro sítě se vstřícným šířením (2)

Krok 4: Aktualizujeme váhy v_{ik} mezi vstupním neuronem i a neurony Kohonenovské vrstvy, které se nacházejí v okolí N_c neuronu c tak, aby lépe odpovídaly předloženému vzoru $\vec{x}(t)$:

$$v_{ik}(t+1) = v_{ik}(t) + \alpha(t)(x_i(t) - v_{ik}(t))$$

$\alpha(t)$, kde $0 < \alpha(t) < 1$, je parametr učení pro váhy mezi vstupní a Kohonenovskou vrstvou, který klesá v čase. t představuje současný a $t + 1$ následující krok učení.

Učící algoritmus pro sítě se vstřícným šířením (3)

Krok 5: Aktualizujte váhy w_{ci} mezi „vítězným“ neuronem c z Kohonenovské vrstvy a neurony Grossbergovské vrstvy tak, aby výstupní vektor $\vec{y}$ lépe odpovídal požadované odezvě $\vec{d}$:

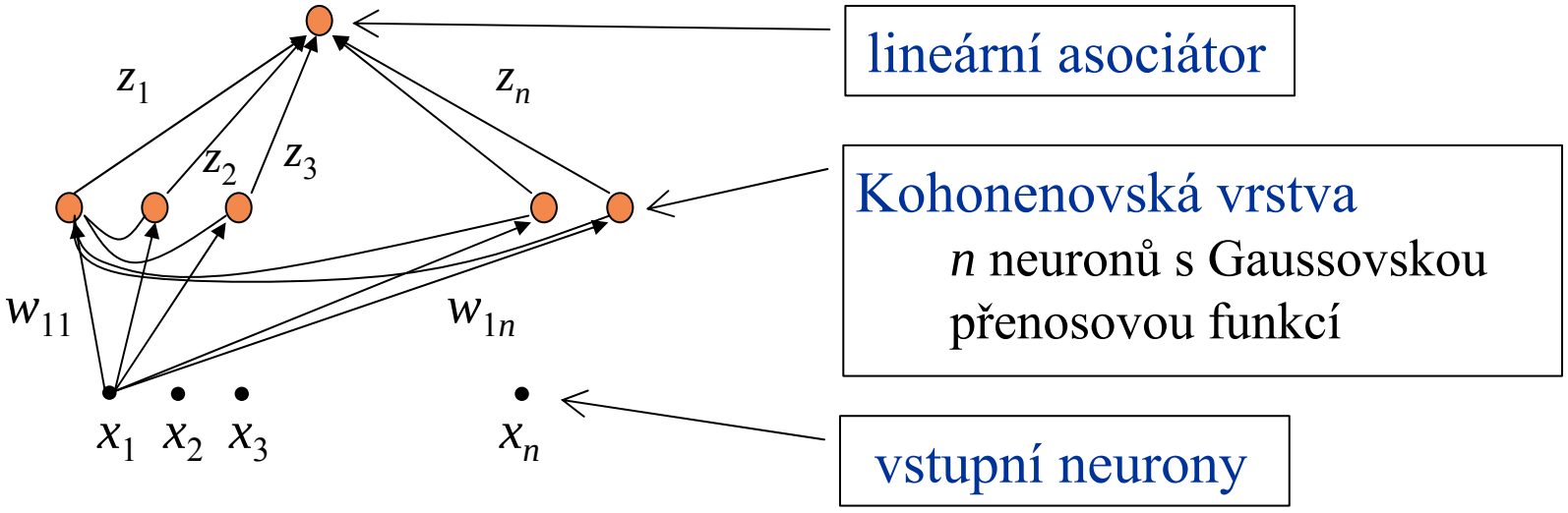
$$w_{cj}(t+1) = (1-\beta)w_{cj}(t) + \gamma z_c d_j$$

$w_{cj}(t)$ je váha synaptického spoje mezi c-tým neuronem Kohonenovské vrstvy a j-tým neuronem Grossberovské vrstvy v čase t,$w_{cj}(t+1)$ označuje hodnotu této synaptické váhy v čase $t+1$. β je kladná konstanta ovlivňující závislost nové hodnoty synaptické váhy na její hodnotě v předchozím kroku učení. Kladná konstanta γ představuje parametr učení vah mezi Kohonenovskou a Grossbergovskou vrstvou, z_c označuje aktivitu „vítězného“ neuronu Kohonenovské vrstvy.

Krok 6: Přejděte ke kroku 2.

RBF-sítě (Radial Basis Functions)

- **Hybridní architektura (Moody & Darken)**
- **Učení s učitelem**



RBF-sítě (Radial Basis Functions)

- Každý neuron j počítá svůj výstup $g_j(t)$ podle:

$$g_j(\vec{x}) = \frac{\exp\left(\frac{(\vec{x} - \vec{w}_j)^2}{2\sigma_j^2}\right)}{\sum_k \exp\left(\frac{(\vec{x} - \vec{w}_k)^2}{2\sigma_k^2}\right)}$$

$\vec{x}$ ... vstupní vektor

$\vec{w}_1, \ldots, \vec{w}_m$ ... váhové vektory skrytých neuronů

$\sigma_1, \ldots, \sigma_m$ ... konstanty (nastavené např. podle vzdálenosti mezi příslušným váhovým vektorem a jeho nejbližším sousedem)

RBF-sítě (Radial Basis Functions)

- výstup každého srytého neuronu je normován
  - vzájemné propojení všech neuronů
- váhy $z_1, \ldots, z_m$ lze nastavit např. pomocí algoritmu zpětného šíření:

$$E = \frac{1}{2}\sum_{p}\left(\sum_{i=1}^{n} g_i(\vec{x}_p)\, z_i - d_p\right)^2$$

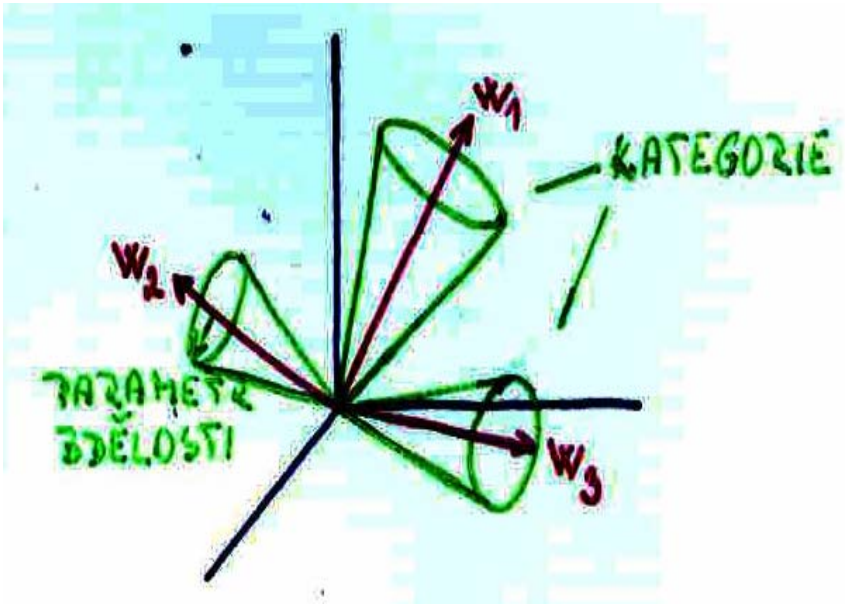
$$\Delta z_i \cong -\frac{\partial E}{\partial z_i} = \gamma\, g_i(\vec{x})\left(d - \sum_{i=1}^{n} g_i(\vec{x})\, z_i\right)$$

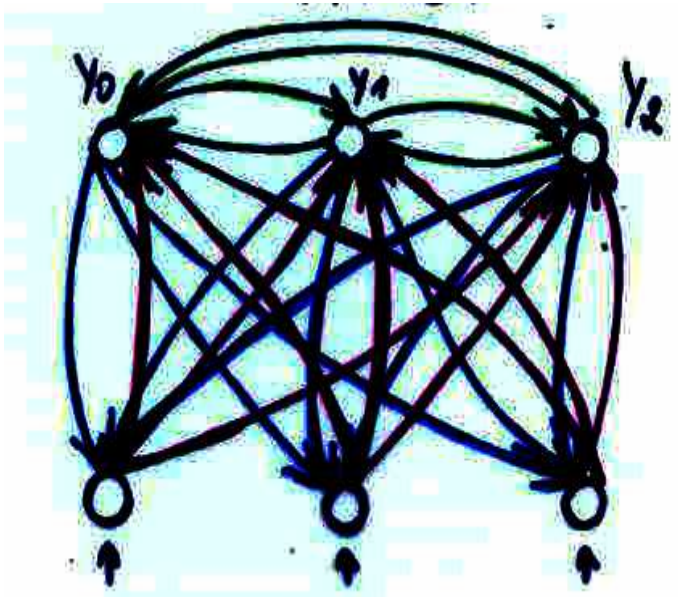
d ... požadovaný výstup

p ... počet trénovacích vzorů

γ ... parametr učení

ART – Adaptive Resonance Theory
(Carpenter & Grossberg)





výstup

x_0 x_1 x_2

vstup

ART – Adaptive Resonance Theory (2)
(Carpenter & Grossberg)

ART 1:

- Binární vstupy, učení bez učitele
- Laterální inhibice pro určení výstupního neuronu s maximální odezvou
- Váhy i pro zpětnou vazbu (z výstupních neuronů směrem ke vstupním) pro porovnání skutečné podobnosti s rozpoznaným vzorem

ART – Adaptive Resonance Theory (3)
(Carpenter & Grossberg)

ART 1 (pokračování):

- Vigilanční test – parametr bdělosti
- Mechanismus pro „vypnutí“ výstupního neuronu s maximální odezvou

→ **stabilita × plasticita sítě**

× síť má velké problémy i při „jen trochu zašuměných vzorech“

→ **narůstá počet ukládaných vzorů**

ART 1 – algoritmus učení

Krok 1: **Inicializace**

$$t_{ij}(0) = 1 \qquad 0 \le i \le N-1$$

$$b_{ij}(0) = 1/(1+N) \qquad 0 \le j \le M-1$$

$$\rho \qquad 0 \le \rho \le 1$$

$b_{ij}(t)$ ~ váha mezi vstupním neuronem i a výstupním neuronem j v čase t

$t_{ij}(t)$ ~ váha mezi výstupním neuronem j a vstupním neuronem i v čase t (určují vzor specifikovaný výstupním neuronem j)

ρ ~ práh bdělosti (určuje, jak blízko musí být předložený vstup k uloženému vzoru - aby mohly patřit do stejné kategorie)

ART 1 – algoritmus učení (2)

Krok 2: **Předlož nový vstup**

Krok 3: **Spočítej aktivaci výstupních neuronů**

$$\mu_j = \sum_{i=0}^{N-1} b_{ij}(t)\, x_i \quad ; \quad 0 \le j \le M-1$$

$\mu_j \sim$ výstup výstupního neuronu j

$x_i \sim$ i – tá složka vstupního vektoru ($\in \{0, 1\}$)

Krok 4: **Vyber uložený vzor, který nejlépe odpovídá předloženému vzoru** (např. pomocí laterální inhibice):

$$\mu_{j^*} = \max_j \{ \mu_j \}$$

ART 1 – algoritmus učení (3)

Krok 5: **Test bdělosti**

$$\|\vec{x}\| = \sum_{i=0}^{N-1} x_i \quad \text{a} \quad \|T \cdot \vec{x}\| = \sum_{i=0}^{N-1} t_{ij^*}\, x_i$$

jestliže $\dfrac{\|T \cdot \vec{x}\|}{\|\vec{x}\|} > \rho$ přejdi ke Kroku 7

jinak přejdi ke Kroku 6

Krok 6: **Zmrazení nejlépe odpovídajícího neuronu**

výstup neuronu ***j**** vybraného v Kroku 4 je dočasně nastaven na nulu (a neúčastní se maximalizace v Kroku 4). Poté přejdi ke Kroku 4.

ART 1 – algoritmus učení (4)

Krok 7: **Aktualizace nejlépe odpovídajícího neuronu**

$$t_{ij^*}(t+1) = t_{ij^*}(t) \cdot x_i$$

$$b_{ij^*}(t+1) = \frac{t_{ij^*}(t) \cdot x_i}{0.5 + \sum_{i=0}^{N-1} t_{ij^*}(t) \cdot x_i}$$

Krok 8: **Přejdi ke Kroku 2 a opakuj**

(Předtím znovu „zapoj“ všechny neurony „zmrazené v Kroku 6)