Neuronové sítě

Doc. RNDr. Iveta Mrázová, CSc.

Katedra teoretické informatiky
Matematicko-fyzikální fakulta
Univerzity Karlovy v Praze

Neuronové sítě

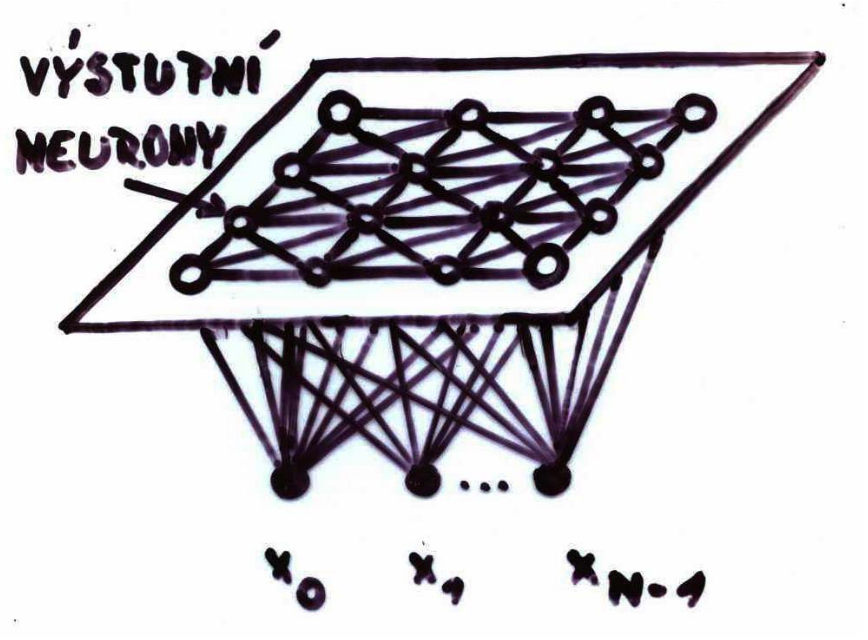
– Kohonenovy mapy a hybridní modely –

Doc. RNDr. Iveta Mrázová, CSc.

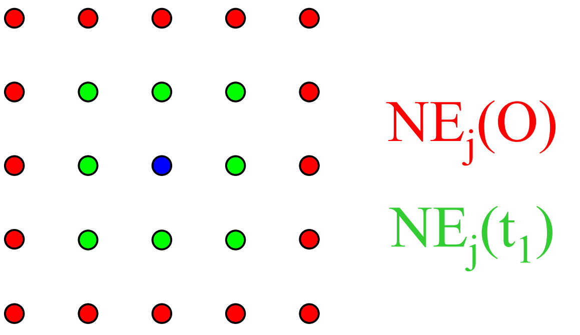
Katedra teoretické informatiky
Matematicko-fyzikální fakulta
Univerzity Karlovy v Praze

Kohonenovy mapy

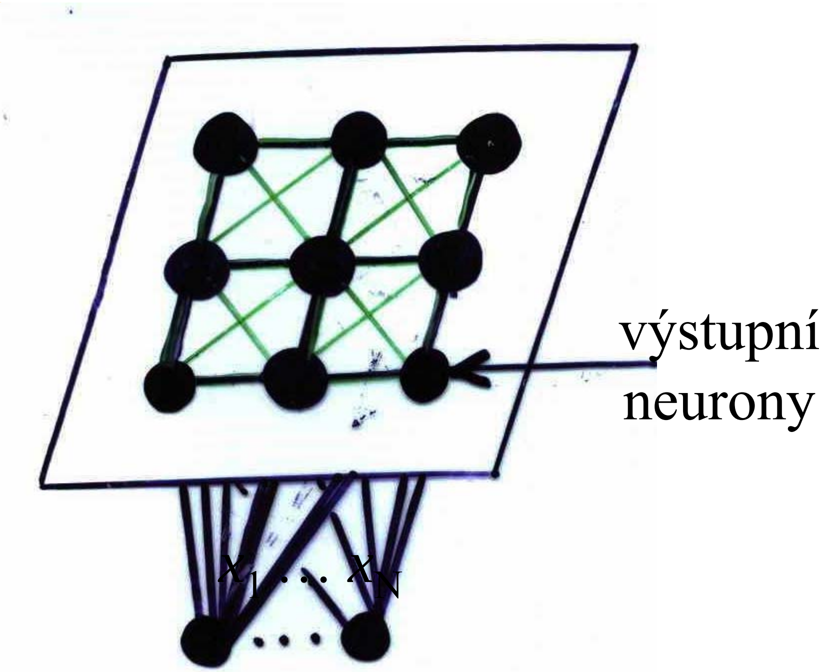
- Teuvo Kohonen – fonetický psací stroj



Topologické okolí



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

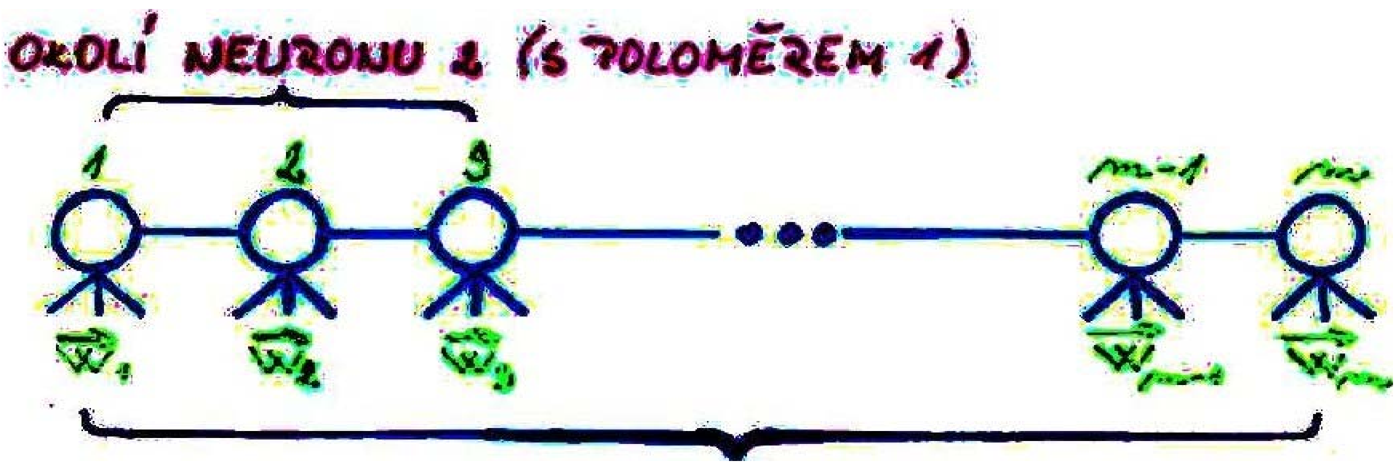
  → v průběhu učení se aktualizují váhy příslušných neuronů i jejich sousedů

  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***



$\vec{X}$

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, …)

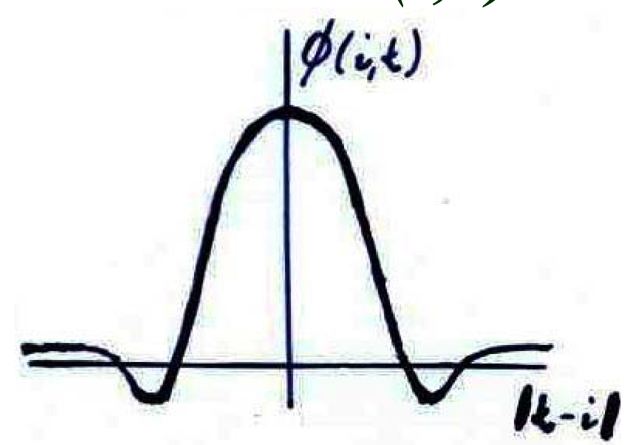
Kohonenův model – algoritmus učení (6)

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

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

Příklad:

- $\Phi(i,k)=1$ $\forall i$ 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 $\boldsymbol{\Phi}$.

Krok 2: Předlož nový trénovací vzor.

Krok 3: Spočítej vzdálenosti $\boldsymbol{d_j}$ 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 $\boldsymbol{x_i(t)}$ je vstupem neuronu ***i*** v čase ***t*** a $\boldsymbol{w_{ij}(t)}$ 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 $j \in N_c$; $0 \leq i \leq N-1$

$\alpha(t)$ 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.

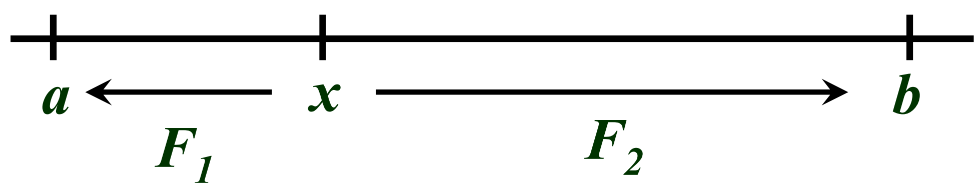
Krok 6: Přejdi ke Kroku 2.

Analýza konvergence ~ stabilita řešení a uspořádaný stav

Stabilita řešení za předpokladu, že síť už dospěla do jistého uspořádaného stavu:

1) **Jednorozměrný případ:**

   a) interval *[a,b]* , *1* neuron s váhou *x*, neuvažováno okolí:



→ **konvergence** x **do středu** *[a,b]*

Analýza konvergence ~ stabilita řešení a uspořádaný stav (2)

- Adaptační pravidlo: $x_n = x_{n-1} + \eta\,(\,\xi - x_{n-1}\,)$

  x_n, x_{n-1} …… váhy v čase n a $n-1$
  ξ …………..…. náhodně zvolené číslo z intervalu $[\,a\,,\,b\,]$

- Pro $0 < \eta < 1$ nemůže posloupnost $x_1, x_2, \ldots$ „opustit“ $[\,a\,,\,b\,]$
- Omezená je i očekávaná hodnota $\langle x \rangle$ váhy x
- Očekávaná hodnota derivace x v čase je nulová: $\left\langle \frac{dx}{dt} \right\rangle = 0$

  (jinak by mohlo být $\langle x \rangle < a$ anebo $\langle x \rangle > b$)

- Protože: $\left\langle \frac{dx}{dt} \right\rangle = \eta\,\big(\langle \xi \rangle - \langle x \rangle\big) = \eta \left(\frac{a+b}{2} - \langle x \rangle \right)$

  bude: $\langle x \rangle = (\,a + b\,)\,/\,2$

Analýza konvergence ~ stabilita řešení a uspořádaný stav (3)

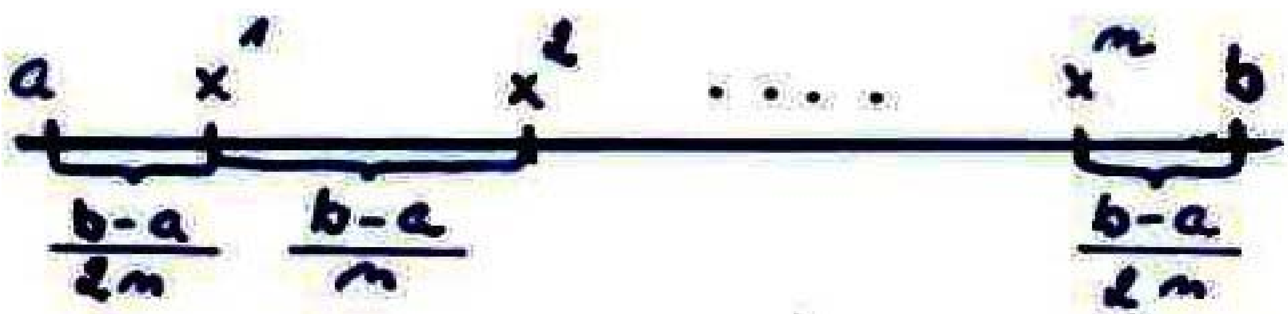
b) interval $[a,b]$, n neuronů s vahami $x^1, x^2, \ldots, x^n$

- neuvažováno okolí,
- váhy jsou uspořádány monotónně:

$$a < x^1 < x^2 < \ldots < x^n < b$$

→ **konvergence vah k**

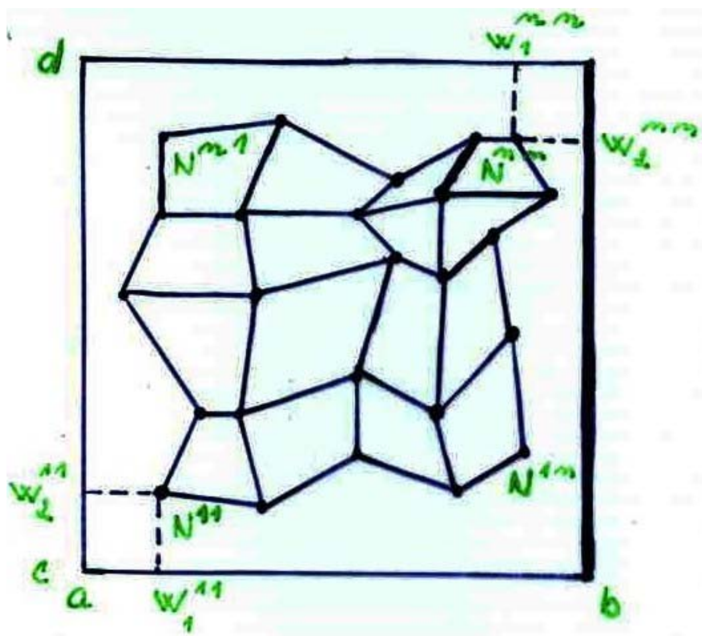
$$\langle x^i \rangle = a + (2i-1)\,\frac{b-a}{2n}$$



Analýza konvergence ~ stabilita řešení a uspořádaný stav (4)

2) Dvourozměrný případ:

- interval ***[a , b]* × *[c , d]*** , ***n* × *n*** neuronů
- neuvažováno okolí, monotónní uspořádání vah:



$$w_1^{ij} < w_1^{ik} \qquad \text{pro} \quad j < k$$

$$w_2^{ij} < w_2^{kj} \qquad \text{pro} \quad j < k$$

→ **Redukce problému na dva jednorozměrné**

Analýza konvergence ~ stabilita řešení a uspořádaný stav (5)

2) **Dvourozměrný případ** (pokračování):

- Nechť $w_j^1 = \frac{1}{n}\sum_{i=1}^{n} w_1^{ij}$ označuje průměr vah neuronů v ***j*** – tém sloupci
- Protože $w_1^{ij} < w_1^{ik}$ pro $j < k$, budou w_1^j monotónně uspořádané: $a < w_1^1 < w_1^2 < \ldots < b$
- Průměr vah bude v prvním sloupci oscilovat kolem očekávané hodnoty $\langle w_1^1 \rangle$
- Podobně pro neurony v každém řádku

→ **konvergence ke stabilnímu stavu**
(pro dostatečně malý parametr učení)

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 $\vec{x}$ ($\boldsymbol{c}$ ~ „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"

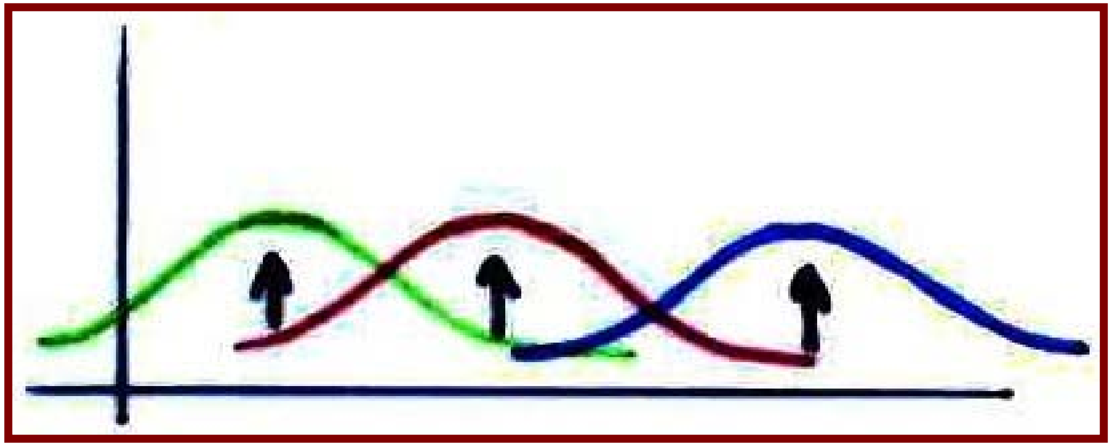
- $\vec{w}_k(t+1) = \vec{w}_k(t) + \varepsilon\alpha(t)\left[\vec{x}(t) - \vec{w}_k(t)\right]$

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

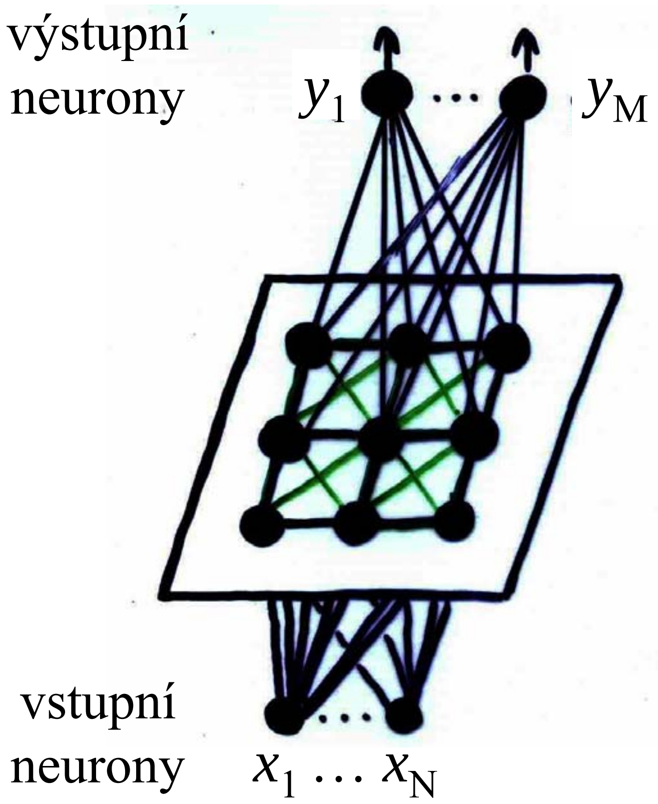


Varianty algoritmu učení pro Kohonenovy mapy (7)

<u>Další varianty:</u>

- 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 $\vec{x}(t)$:

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

$\alpha(t)$, 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 $\vec{d}$:

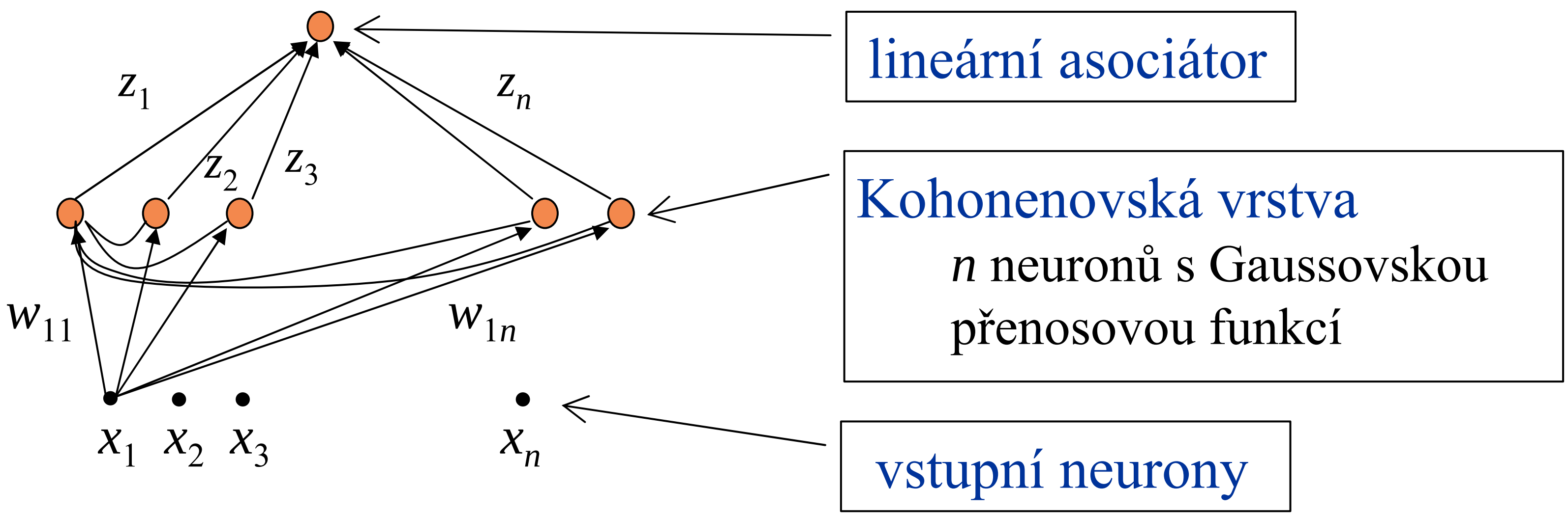
$$w_{cj}(t+1) = (1-\beta)\, w_{cj}(t) + \gamma\, z_c d_j$$

$w_{cj}(t)$ 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:

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

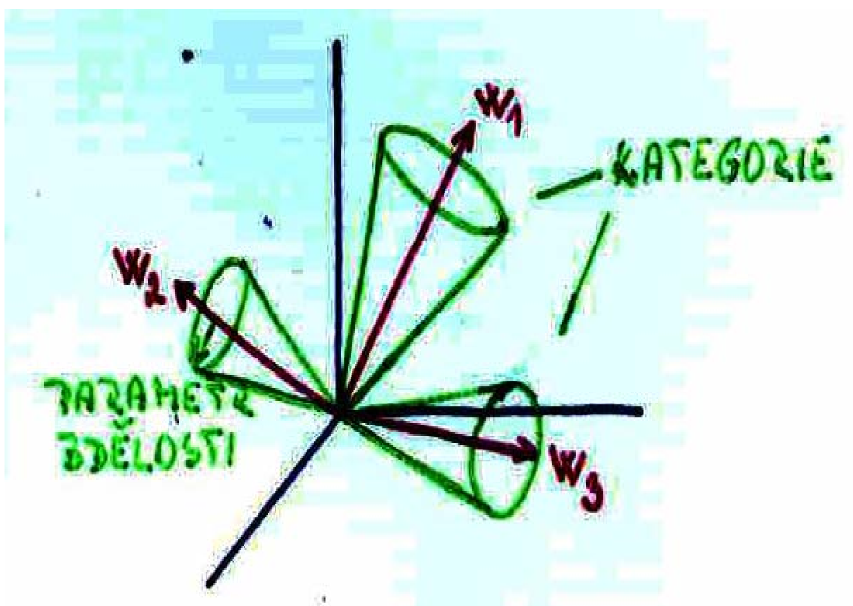
$$\Delta z_i \cong -\frac{\partial E}{\partial z_i} = \gamma\, g_i(\vec{x}) \left(d - \sum_{i=1}^{n} g_i(\vec{x})\, z_i \right)$$

d ... požadovaný výstup

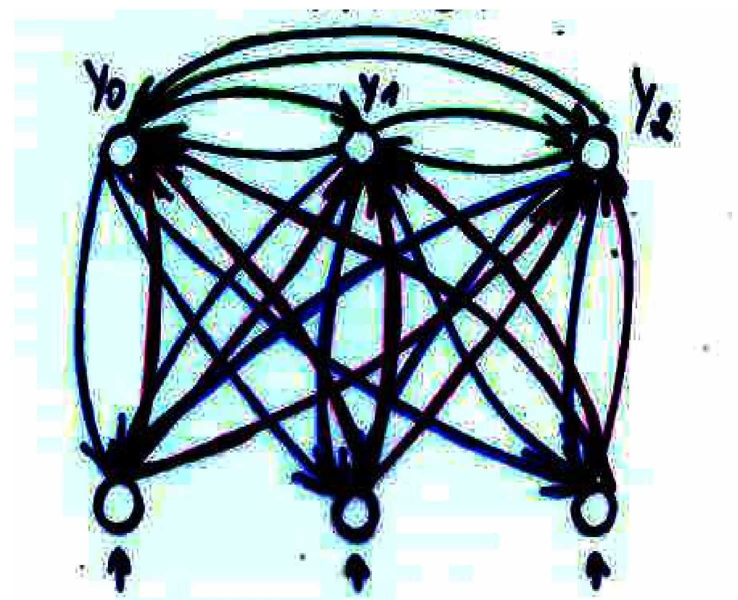
p ... počet trénovacích vzorů

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 zašumě-ných vzorech“

→ **narůstá počet ukládaných vzorů**

ART 1 – algoritmus učení

Krok 1: **Inicializace**

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

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

$\rho \qquad\qquad 0 \leq \rho \leq 1$

$b_{ij}(t)$ ~ 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**

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

$\boldsymbol{\mu_j}$ ~ výstup výstupního neuronu $\boldsymbol{j}$

$\boldsymbol{x_i}$ ~ $\boldsymbol{i}$ – 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**

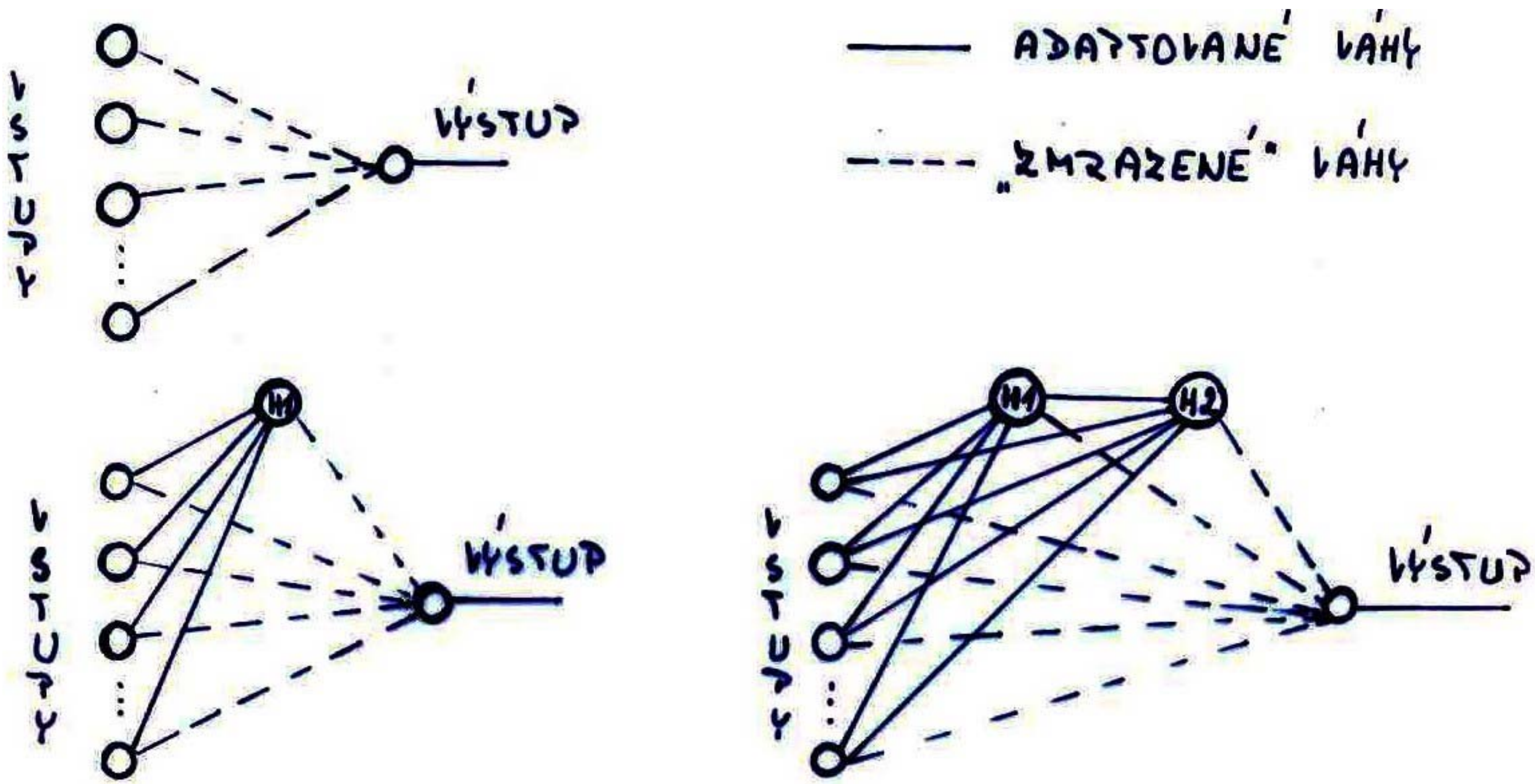
(Předtím znovu „zapoj“ všechny neurony „zmrazené v Kroku 6)

Kaskádová korelace
(Fahlman & Lebiere, 1990)

~ robustní rostoucí architektura

- Systém začíná proces učení s přímým propojením vstupů na výstup
- Postupně jsou přidávány skryté neurony
- Vstupy každého nového neuronu jsou propojeny se všemi původními vstupy i se všemi dříve vytvořenými neurony

Kaskádová korelace (2)
(Fahlman & Lebiere, 1990)



Kaskádová korelace (3)
(Fahlman & Lebiere, 1990)

Učení sítě (probíhá ve dvou fázích):

a) V první fázi se již existující síť adaptuje pomocí algoritmu Quickprop

- Jestliže se do určité doby chyba na výstupu sítě nijak podstatně nezmenší, přidá se síti nový neuron
- Jestliže je aktuální hodnota chyby dostatečně nízká, algoritmus končí

Kaskádová korelace (4)
(Fahlman & Lebiere, 1990)

Učení sítě (pokračování):

b) Nově přidávaný neuron je ze skupiny „kandidátů“ adaptovaných tak, aby maximalizovali korelaci mezi svým výstupem a chybou na výstupu sítě

→ **přidávaný neuron „se naučil“ nějaký příznak, který vysoce koreluje se „zbytkovou“ chybou**

- Vstupní váhy přidávaného neuronu budou zmrazeny
- „Doučeny“ budou váhy od přidávaného neuronu na výstup

Kaskádová korelace (5)
(Fahlman & Lebiere, 1990)

Učení sítě (pokračování):

Cílem při učení skrytých neuronů je maximalizovat ***S***:

$$S = \left| \sum_{i=1}^{p} \left(V_i - \overline{V} \right)\left(E_i - \overline{E} \right) \right|$$

$\boldsymbol{p}$ ….. počet trénovacích vzorů

$\boldsymbol{V_i}$ ….. výstup přidávaného neuronu pro $\boldsymbol{i}$ – tý vzor

$\overline{V}$ ….. průměrný výstup přidávaného neuronu

$\boldsymbol{E_i}$ ….. chyba pro $\boldsymbol{i}$ – tý vzor

$\overline{E}$ ….. průměrná chyba

Kaskádová korelace (6)
(Fahlman & Lebiere, 1990)

Učení sítě (pokračování):

$$\frac{\partial S}{\partial w_k} = \sum_{i=1}^{p} \sigma \left(E_i - \overline{E} \right) f_i' I_{i,k}$$

$\boldsymbol{\sigma}$ ….. znaménko korelace mezi přidávaným neuronem a výstupem

f_i' ….. derivace přenosové funkce pro vzor ***i***

$I_{i,k}$….. ***k*** – tý vstup přidávaného neuronu pro vzor ***i***