Opakování - co jsme probírali minule

Učení bez učitele, unsupervised learning, samoorganizace

- Jednoduché metody strojového učení pro klasifikaci a shlukování
  - algoritmus k nejbližších sousedů
  - algoritmus k středů
- Modely mělkých umělých neuronových sítí učené bez učitele
  - Kompetitivní modely
  - Kohonenovy mapy (SOM)

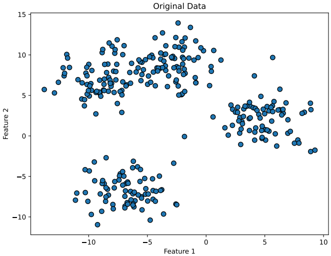
Dnešní hodina

1. Shlukování - prohloubení látky
   - algoritmus k středů
   - hierarchické shlukování
2. Hybridní modely umělých neuronových sítí (kombinace učení s učitelem a bez učitele):
   - LVQ (Učení vektorové kvantizace)
   - Sítě se vstřícným šířením (Counter-propagation)
   - RBF-sítě
   - ART (Adaptive Resonance Theory)

   ...

Učení bez učitele, unsupervised learning, samoorganizace

- trénovací množina T tvaru $T = \{\vec{x}_1, ..., \vec{x}_N\}$
- $\vec{x}_i \in R^n$ je (i-tý) trénovací vstupní vzor, požadovaný výstup neznáme
- **Myšlenka:** model sám rozhodne, jaká odezva je pro daný vzor nejlepší a podle toho nastaví své váhy $\rightarrow$ samoorganizace (self-organisation)



- máme data a neznáme jejich strukturu
- snažíme se strukturu a vlastnosti dat odhalit, najít v nich vzory, popř. shluky

Shlukování (klastrování)

Shluk (klastr)

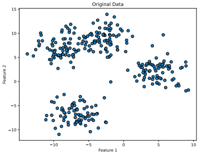
- Skupina vzorů s malým rozptylem a velkou vzdáleností od ostatních shluků

Shlukování (klastrování)

- Disjunktní rozdělení dat na shluky

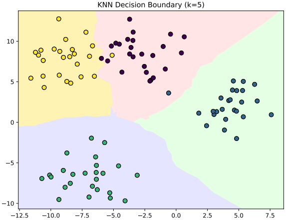
Problémy:

- Jak určit počet a rozložení shluků v příznakovém prostoru?
- Jak vybrat reprezentanta/y shluku?
  - Vhodně vybrané / všechny trénovací vzory patřící do shluku
  - Střed shluku (*těžiště, centroid*)



Odbočka: metoda k nejbližších sousedů

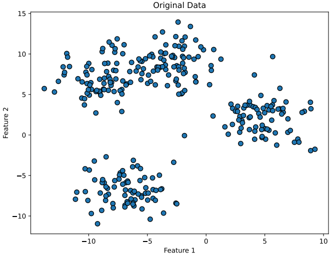
- Klasifikační metoda, učení s učitelem:
  Vzory z trénovací množiny jsou uloženy a klasifikovány do jedné z l různých tříd
- Neznámý vstupní vektor je zařazen do té třídy, ke které patří většina z k nejbližších vektorů z uložené množiny

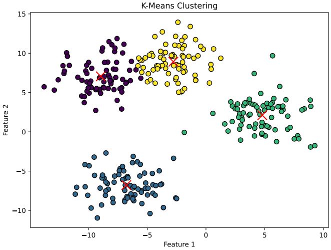


- Nejjednodušší varianta: 1 nejbližší soused,
  klasifikační model = uložená data

Algoritmus k středů (k-means clustering)

- Učení bez učitele
- Vstupní vzory jsou klasifikovány do k různých shluků, každý shluk l je reprezentován svým centroidem (středem, těžištěm) $\vec{c}_l$
- Nový vektor $\vec{x}$ je zařazen k tomu shluku i, jehož centroid $\vec{c}_l$ je mu nejblíže





Algoritmus k středů (k-means clustering)

1. Je dána trénovací množina $T = \{\vec{x}_1, ..., \vec{x}_N\}, \vec{x}_i \in R^n$
2. Zvolíme k náhodných vektorů $\vec{c}_l, l = 1, ..., k$ (z R^n nebo z T) za středy (centroidy) shluků
3. Opakujeme:
   - Přiřadíme každý vektor z T k nejbližšímu středu shluku
   - Přepočítáme středy shluků na základě přiřazených vzorů:

$$\vec{c}_l = \frac{1}{n_l} \sum_{l_i=1}^{n_l} (\vec{x}_{l_i})$$

     n_l ... počet vektorů přiřazených k l-tému shluku
     l_i ... indexuje vektory přiřazené k l-tému shluku
   - Předchozí dva kroky opakujeme, dokud se mění příslušnost trénovacích vzorů ke shlukům

→ **Vektorová kvantizace**

Algoritmus k středů (k-means clustering)

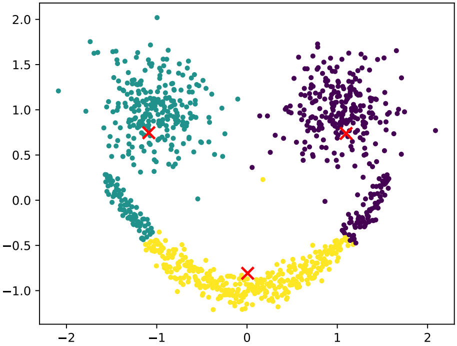
Výhody

- Rychlý algoritmus, jednoduchý na implementaci
- Vhodný pro rychlý náhled na strukturu dat

Nevýhody

- Nutnost zvolit počet shluků předem
- Dávkové zpracování (problém pro velká data nebo online učení)
- Vysoká citlivost k počáteční volbě centroidů *(... obrázek)*
- Citlivost k odlehlým vzorům
- Pro složitá data nemusí být úspěšný: vyhledává sférické shluky *(... obrázek)*
- Problém, pokud je vysoká dimenze vstupních dat (*prokletí dimenzionality*), popř. navzájem silně korelované příznaky

Algoritmus k středů (k-means clustering)



Algoritmus k středů (k-means clustering)

Nevýhody a jejich řešení

- Nutnost zvolit počet shluků předem
- Dávkové zpracování (problém pro velká data nebo online učení)
- Vysoká citlivost k počáteční volbě centroidů
- Citlivost k odlehlým vzorům

  $\rightarrow$ vhodná počáteční volba centroidů (např. podle náhodně vybraných vzorů)

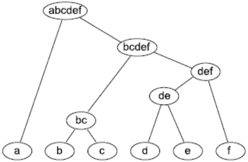
  $\rightarrow$ normalizace dat:
  - zajistí také invarianci vůči změně měřítka a posunutí
  - ale ne vždy pomůže
- Pro složitá data nemusí být úspěšný: vyhledává sférické shluky

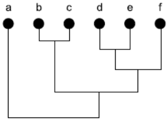
  $\rightarrow$ volba jiné metriky
- Problém, pokud je vysoká dimenze vstupních dat (*prokletí dimenzionality*), popř. navzájem silně korelované příznaky

  $\rightarrow$ PCA předzpracování vstupních příznaků

Hierarchické shlukování

- není nutné předem znát předpokládaný počet shluků
- na začátku učení každý trénovací vzor reprezentuje jeden shluk
- spočítáme matici vzdáleností mezi jednotlivými trénovacími vzory
- v průběhu učení se iterativně slučují dva nejbližší shluky
- **Vizualizace:** dendrogram





Zdroj: Kateřina Horaisová: slidy k předmětu Neuronové sítě 2, FJFI ČVUT Děčín

Hierarchické shlukování

Jak určit vzdálenost mezi shluky C_i ,$C_j \subset R^n$?

- **Metoda nejbližšího souseda**

$$d(C_i, C_j) = \min\{d(\vec{x}, \vec{y}) | \vec{x} \in C_i, \vec{y} \in C_j, i \neq j\}$$

- **Metoda nejvzdálenějšího souseda**

$$d(C_i, C_j) = \max\{d(\vec{x}, \vec{y}) | \vec{x} \in C_i, \vec{y} \in C_j, i \neq j\}$$

- **Průměrná vzdálenost shluků**

$$d(C_i, C_j) = \frac{1}{m_i m_j} \sum_{\vec{x} \in C_i} \sum_{\vec{x'} \in C_j} d(\vec{x}, \vec{y})$$

- **Vzdálenost středů shluků**

$$d(C_i, C_j) = d(\vec{\mu}_i, \vec{\mu}_j)$$

- **Wardova metoda minimálního rozptylu**

$$d(C_i, C_j) = \frac{m_i m_j}{m_i + m_j} d(\mu_i, \mu_j)$$

Hierarchické shlukování

- **Metoda nejbližšího souseda**
  - je citlivá na šum a na změny v poloze bodů
- **Metoda nejvzdálenějšího souseda**
  - omezuje vznik podélných shluků, citlivá na šum
- **Průměrná vzdálenost shluků**
  - méně citlivá na šum
- **Vzdálenost středů shluků**
  - málo citlivá na šum a přitom méně náročná na výpočet
- **Wardova metoda minimálního rozptylu**
  - upřednostňuje slučování malých shluků s velkými oproti slučování stejně velkých shluků

Hierarchické shlukování

Připomenutí ... jak spočítat vzdálenost u shlukovacích metod

- Euklidovská vzdálenost (metrika): $d(\vec{p},\vec{q}) = \sqrt{\sum_{i=1}^{n}(p_i - q_i)^2}$
- pokud jen porovnáváme vzdálenosti: $d(\vec{p},\vec{q}) = \sum_{i=1}^{n}(p_i - q_i)^2$

Ale existují i alternativní metriky, např.:

- Manhattan (městská) metrika: $d(\vec{p},\vec{q}) = \sum_{i=1}^{n}|p_i - q_i|$
- Čebyševova metrika: $d(\vec{p},\vec{q}) = \max_i |p_i - q_i|$
- Minkewského metrika: $d(\vec{p},\vec{q}) = \left(\sum_{i=1}^{n}|p_i - q_i|^r\right)^{\frac{1}{r}}$
- Kosínová podobnost: $\cos(\vec{p},\vec{q}) = \frac{\vec{p}\cdot\vec{q}}{||\vec{p}||||\vec{q}||}$

...

Hierarchické shlukování

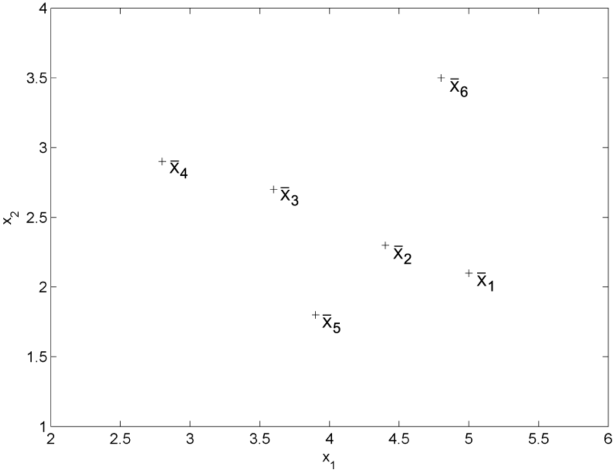
Výhody

- Hierarchická struktura, snadná interpretace
- Nevyžaduje předem stanovit počet shluků, nízké nároky na parametry
- Oproti algoritmu k středů si poradí i se shluky různých tvarů

Nevýhody

- Výpočetní náročnost
- Nelze učit online (a změnit, pokud se později objeví nová data)
- Citlivost na metriku vzdálenosti a na metodu určení vzdálenosti mezi shluky
- Citlivost k odlehlým vzorům
- Horší interpretovatelnost pro velký počet shluků

Hierarchické shlukování - příklad



x_1	x_2
5,0	2,1
4,4	2,3
3,6	2,7
2,8	2,9
3,9	1,8
4,8	3,5

Zdroj: Kateřina Horaisová: slidy k předmětu Neuronové sítě 2, FJFI ČVUT Děčín

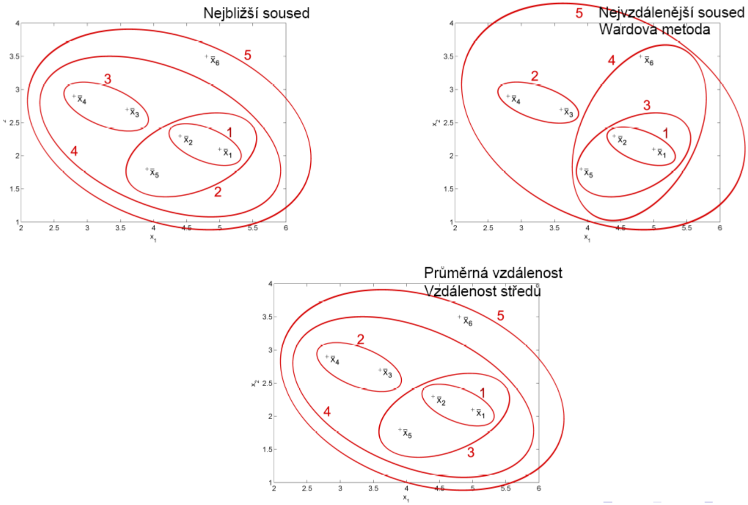
Hierarchické shlukování - příklad

Matice vzdáleností mezi trénovacími vzory:

	$\overline{x}_1$	$\overline{x}_2$	$\overline{x}_3$	$\overline{x}_4$	$\overline{x}_5$	$\overline{x}_6$
$\overline{x}_1$	0,0000	0,6325	1,5232	2,3409	1,1402	1,4142
$\overline{x}_2$	0,6325	0,0000	0,8944	1,7088	0,7071	1,2649
$\overline{x}_3$	1,5232	0,8944	0,0000	0,8246	0,9487	1,4422
$\overline{x}_4$	2,3409	1,7088	0,8246	0,0000	1,5556	2,0881
$\overline{x}_5$	1,1402	0,7071	0,9487	1,5556	0,0000	1,9235
$\overline{x}_6$	1,4142	1,2649	1,4422	2,0881	1,9235	0,0000

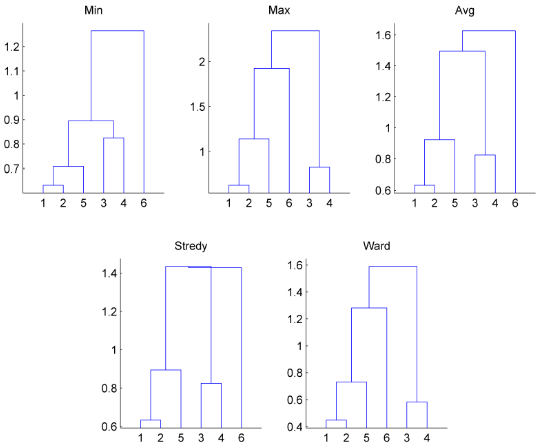
Zdroj: Kateřina Horaisová: slidy k předmětu Neuronové sítě 2, FJFI ČVUT Děčín

Hierarchické shlukování - příklad



Zdroj: Kateřina Horaisová: slidy k předmětu Neuronové sítě 2, FJFI ČVUT Děčín

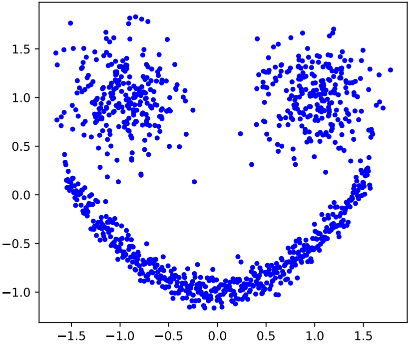
Hierarchické shlukování - příklad



Zdroj: Kateřina Horaisová: slidy k předmětu Neuronové sítě 2, FJFI ČVUT Děčín

Hierarchické shlukování - příklad

Na rozmyšlenou: Která varianta hierarchického shlukování si nejlépe poradí s následujícím příkladem?



Metriky pro hodnocení kvality shlukování

- kvantifikují, jak jsou shluky kompaktní a jak dobře jsou od sebe navzájem odděleny
- **Silueta (Silhouette)**
  - měří, jak jsou si navzájem vzdálené body uvnitř svého shluku ve srovnání s ostatními shluky.
- **Davies-Bouldin index**
  - určuje „odlehlost" shluků tím, že porovnává vzdálenosti mezi středy shluků a průměrné vzdálenosti uvnitř shluků
- **Calinski-Harabasz index**
  - měří podobnost objektů uvnitř stejného shluku a odlišnost mezi různými shluky pomocí rozptylu.
- **Vnitřní suma čtverců (WCSS)**
  - Součet čtverců vzdáleností mezi jednotlivými body a středy jejich shluků.
- ...

Metriky pro hodnocení kvality shlukování

- **Silueta (Silhouette)**
  - měří, jak jsou si navzájem vzdálené body uvnitř svého shluku ve srovnání s ostatními shluky.
  - oblíbená míra pro určení optimálního počtu shluků (maximalizuje se)

$$S(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$

  - $S(i)$ je silueta pro i-tý bod
  - $a(i)$ je průměrná vzdálenost bodu i k bodům ve stejném shluku
  - $b(i)$ je průměrná vzdálenost bodu i k bodům v nejbližším sousedním shluku.

Metriky pro hodnocení kvality shlukování

- **Davies-Bouldin index**
  - měří kompaktnost shluků a separaci mezi shluky
  - určuje ,,odlehlost" shluků tím, že porovnává vzdálenosti mezi středy shluků a průměrné vzdálenosti uvnitř shluků

$$DB = \frac{1}{k} \sum_{i=1}^{k} \max_{j \neq i} \left(\frac{s_i + s_j}{d(c_i, c_j)} \right)$$

  - k je počet shluků
  - s_i je míra vnitřní podobnosti i-tého shluku
  - $d(c_i, c_j)$ je vzdálenost mezi středy i-tého a j-tého shluku

Metriky pro hodnocení kvality shlukování

- **Calinski-Harabasz index**
  - měří podobnost objektů uvnitř stejného shluku a odlišnost mezi různými shluky pomocí rozptylu.

$$CH = \frac{B(k)}{W(k)} \times \frac{n-k}{k-1}$$

  - $B(k)$ je součet čtverců vzdáleností mezi středem shluku a globálním středem dat,
  - $W(k)$ je celková vnitřní suma čtverců pro shluk
  - n je počet bodů
- **Vnitřní suma čtverců (WCSS)**
  - Součet čtverců vzdáleností mezi jednotlivými body a středy jejich shluků.

$$WCSS = \sum_{i=1}^{k} \sum_{x \in C_i} ||x - \mu_i||^2$$

Opakování - co jsme probírali minule

Modely mělkých umělých neuronových sítí učené bez učitele

- Kompetitivní modely
- Kohonenovy mapy (SOM)

Kompetitivní modely a kompetiční učení

- **Myšlenka:** neurony odpovídají bodům ve vstupním prostoru, každý reprezentuje jeden shluk (nebo jeho část)
- **Kompetice:** neurony bojují o ,,právo reprezentovat předložený vzor"
- **Inhibice:** ,,potlačování (aktivity) soupeřů"
- Pravidlo ,,vítěz bere vše" *(Winner takes all)*

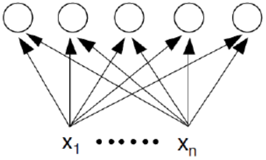
Cíl učení

- Umístit neurony do středu shluků vzorů (*těžiště, centroid*)
- Zachovat již dříve vytvořenou strukturu sítě

Kompetitivní modely

Architektura

- neuronová síť s jednou vrstvou
- vstupní vrstva odpovídá jednotlivým vstupním příznakům
- počet neuronů ve výstupní vrstvě odpovídá (předpokládanému) počtu shluků



- každý vstupní neuron je propojen hranou s každým výstupním neuronem
- mohou být i „laterální" vazby mezi jednotlivými neurony ve výstupní vrstvě

Kompetitivní modely - použití

1. Předložím vzor $\vec{x}$
2. Neurony počítají (např. Euklidovskou) vzdálenost mezi předloženým vzorem a svým váhovým vektorem
3. V kompetici „vítězí" neuron, který je k předloženému vzoru nejblíže
4. Vítězný neuron bude nejaktivnější a bude potlačovat **(inhibovat)** aktivitu ostatních neuronů
   $\rightarrow$ pomocí „laterálních" spojů … **laterální inhibice**

- **Výsledek:** jeden neuron je excitován, ostatní inhibovány
- Aktivita neuronu signalizuje příslušnost předloženého vstupu ke shluku vektorů reprezentovaných tímto neuronem
  $\rightarrow$ síť funguje i jako klasifikátor

Kompetitivní modely

Jak implementovat laterální inhibici ve výstupní vrstvě?

1. Pomocí laterálních vazeb a iterativního výpočtu
   - nemusí vždy vést k dobrým výsledkům (hrozí vyhlazení rozdílů mezi aktivitami neuronů)
2. Prostým výběrem neuronu s max. odezvou ... **vítěz bere vše**
   - jednodušší na implementaci, stabilnější

Kompetitivní modely - učení

Adaptace ... diferenční pravidlo:

- Vítězný neuron *i* se snaží co nejvíce přiblížit k danému trénovacímu vzoru
  $\rightarrow$ zadaptuje své váhy směrem k předloženému vzoru $\vec{x}$:

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

Parametr učení .... plasticita sítě ... α

- $\alpha = 1$ ... úplné přitažení vektoru vah ke vzoru
- $0 < \alpha < 1$ ... částečné přitažení vektoru vah ke vstupu
- $\alpha = 0$ ... ignoruje vstup (ustálený stav)

Pro pevné α síť obvykle nekonverguje ... $\alpha \rightarrow 0$

Kompetiční učení
Formální algoritmus pro variantu „vítěz bere vše":

Vstup

- Trénovací množina $T = \{\vec{x}_1, ..., \vec{x}_N\}$ vstupních vzorů v n-rozměrném prostoru, které chceme klasifikovat do k shluků
- Jednovrstvá neuronová síť s k neurony ve výstupní vrstvě.

Inicializace

- Náhodně vygeneruj váhové vektory $\vec{w}_1, .., \vec{w}_k$ (nebo náhodně vyber k vzorů z T)

Opakuj:

- Náhodně vyber další $\vec{x} \in X$
- Spočítej $d(\vec{x}, \vec{w}_i) = ||\vec{x} - \vec{w}_i||^2 = \sum_{j=1}^{n}(x_j - w_{ji})^2$ pro $i = 1, ..., k$
- Vyber $\vec{w}_m$ tž. $d(\vec{x}, \vec{w}_m) \leq d(\vec{x}, \vec{w}_i)$ pro všechna $i = 1, ..., k$
- **Aktualizace**
    - Nahraď vektor $\vec{w}_m$ vektorem $\vec{w}_m + \alpha(\vec{x} - \vec{w}_m)$

Kompetiční učení

Výhody a nevýhody:

- Online učení („online verze algoritmu k-středů")
- Snadné zjištění reprezentanta shluku i ... $\vec{w}_i$
- Počet shluků je daný předem
- Oproti algoritmu k-středů robustní vůči šumu v datech

Aplikace:

- Shlukování
- Komprese, extrakce příznaků, redukce dimenzionality
- Detekce anomálií
- Rozpoznávání vzorů

Kompetiční učení

Problémy:

- Nutnost volit rychlost klesání α
- Nutnost vhodně inicializovat váhy ... ovlivňuje rychlost učení
  - např. podle náhodně vybraných vzorů
- Hustota reprezentantů nemusí odpovídat hustotě dat
- Mrtvé (nevyužité) neurony

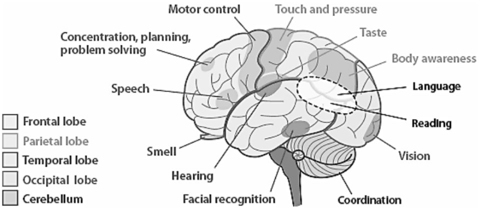
  $\rightarrow$
  - Normalizace vektorů
  - Řízená kompetice a mechanismus svědomí
  - Topologické okolí neuronu, např. mřížka v Kohonenově vrstvě,

(obrázek)

Kohonenovy mapy (SOM, Self-organizing feature maps) (Teuvo Kohonen, 1981)

- původní aplikace: fonetický psací stroj (finština: řeč → písmo)

Biologická motivace



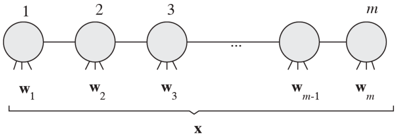
- Mozková kůra: specializované oblasti neuronů - více citlivé na určitý druh podnětů

- Fyzicky blízké neurony reagují podobně - laterální vazby vedou k excitaci blízkých a k inhibici vzdálenějších neuronů

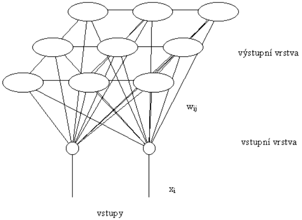
https://cybernetist.com/2017/01/13/self-organizing-maps-in-go/

Kohonenovy mapy - architektura (topologie)

Architektura 1D - řetízek:



Architektura 2D - mřížka:

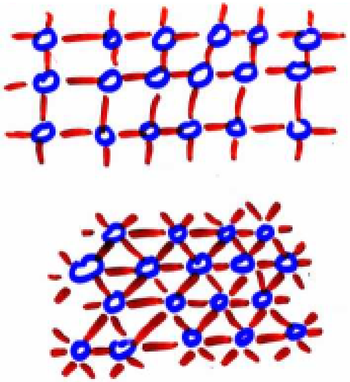


- Neurony jsou topologicky uspořádány do mřížky
- Na mřížce je definovaná sousednost fyzických neuronů ($\times$ logická sousednost daná blízkostí váhových vektorů)
- Sousední neurony by měly reagovat na velmi podobné signály

Paul Rojas: Neural Networks - A Systematic Introduction, Springer, 1996

Kohonenovy mapy - architektura (topologie)

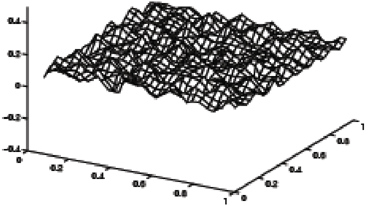
Různé topologie mřížky (pro dvě dimenze)

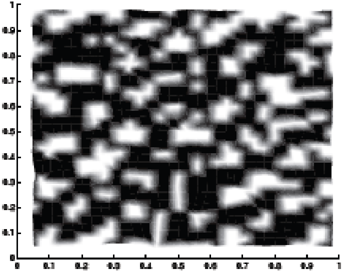


Kohonenovy mapy

Dvě možné interpretace z pohledu aplikací

1. Snížení dimenze dat při zachování topologie, vizualizace
2. Shlukování (klastrování)

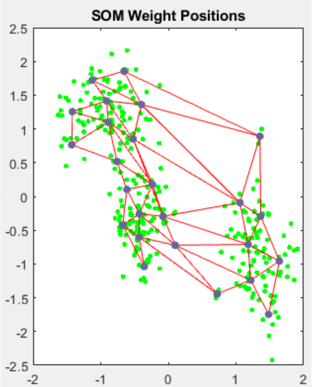




Kohonenovy mapy

Dvě možné interpretace z pohledu aplikací

1. Snížení dimenze dat při zachování topologie, vizualizace
2. Shlukování (klastrování), vektorová kvantizace

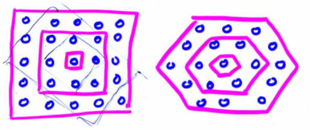


Kohonenovy mapy – učení

Princip

1. Předložím trénovací vzor $\vec{x}$
2. Neurony počítají (Euklidovskou) vzdálenost mezi předloženým vzorem a svým váhovým vektorem
3. V kompetici ,,vítězí" neuron, který je k předloženému vzoru nejblíže
4. V průběhu učení se aktualizují váhy vítězného neuronu, ale i jeho nejbližších sousedů
   - Sousední neurony by měly také reagovat na velmi podobné signály
     $\rightarrow$ zobrazení zachovává topologii

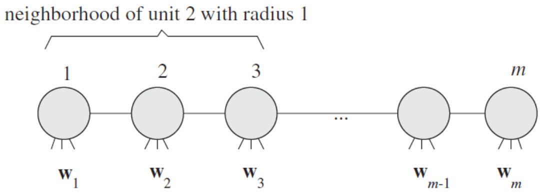
Topologické okolí neuronu



Kohonenovy mapy - definice okolí:

Definice okolí - v jedné dimenzi (řetízek)

- Neurony tvoří posloupnost a mohou být očíslované $1, \ldots, m$
- Do okolí neuronu k s poloměrem 1 patří neurony $k-1$ a $k+1$ (až na kraje)

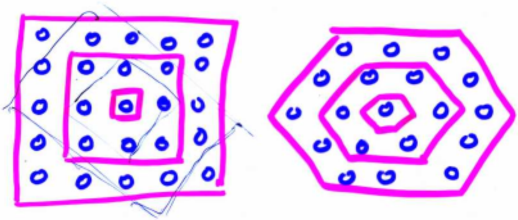


Paul Rojas: Neural Networks - A Systematic Introduction, Springer, 1996

Kohonenovy mapy - definice okolí:

Definice okolí - ve více dimenzích (mřížka)

- Obdobně - do okolí neuronu k s poloměrem 1 patří neurony propojené s k laterální vazbou
- Na mřížce můžeme definovat libovolnou metriku (čtvercová, hexagonální,...)



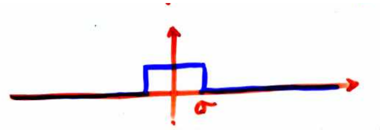
Kohonenovy mapy

Funkce okolí = funkce laterální interakce

- $\Lambda(i, k)$ ... síla laterální vazby mezi neurony i a k během učení
- Měla by klesat s rostoucí vzdáleností neuronů i a k

Příklady

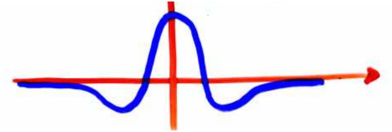
- **Diskrétní okolí**
  - $\Lambda(i, k) = 1$ pro všechny i z okolí k s poloměrem nejvýše σ, $\Lambda(i, k) = 0$ pro ostatní
  - efektivní z hlediska implementace, minimální režie (stačí adaptovat neurony z okolí)
  - σ je šířka okolí (nejjednodušeji $\sigma = 1$)



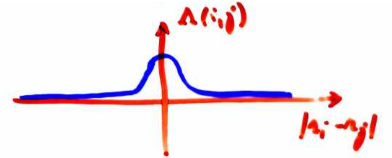
Kohonenovy mapy

Funkce okolí = funkce laterální interakce

- **Funkce mexického klobouku**
  - biologicky nejvěrnější



- **Gaussovská funkce**
  - $\Lambda(i,k) = e^{-\frac{|\vec{w}_i - \vec{w}_k|^2}{\sigma^2}}$, kde σ je šířka okolí (obvykle $\sigma \to 0$)



Kohonenovy mapy

Adaptace

- Nechť k je vítězný neuron pro předložený vstupní vektor $\vec{x}$
- Každý neuron i zadaptuje své váhy podle pravidla:

$$\Delta \vec{w}_i = \alpha \Lambda(i, k)(\vec{x} - \vec{w}_i)$$

Nastavitelné parametry

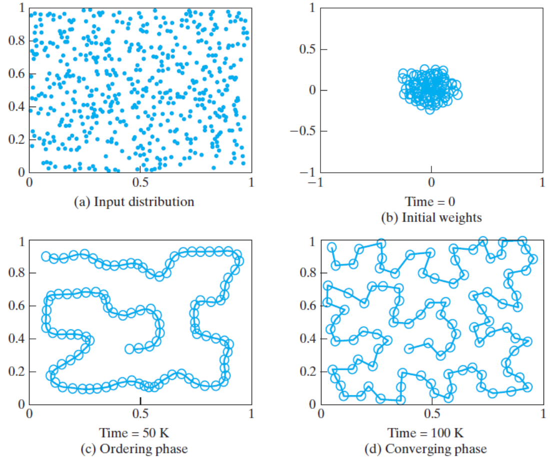
- **Parametr učení ... vigilanční (bdělostní) koeficient** ... $\alpha \in < 0, 1 >$
  - Pro pevné α síť obvykle nekonverguje ... $\alpha \to 0$
- **Šířka okolí** σ
  - obvykle $\sigma \to 0$

Kohonenovy mapy - algoritmus učení

1. **Inicializace**:
   Zvol hodnoty vah mezi vstupy a výstupními neurony jako malé náhodné hodnoty. Zvol počáteční vigilanční koeficient α, poloměr okolí σ a funkci laterální interakce Λ.
2. **Opakuj:**
   1. Předlož další trénovací vzor $\vec{x}$
   2. Spočítej vzdálenosti d_i mezi $\vec{x}$ a $\vec{w}_i$ pro každý výstupní neuron i:
      $$d_i = \sum_j (x_j - w_{ji})^2$$
   3. Vyber výstupní neuron k s minimální vzdáleností d_k jako ,,vítěze"
   4. Aktualizuj váhy všech neuronů i (popř. jen neuronů z okolí k) podle:
      $$\vec{w}_i(t+1) = \vec{w}_i(t) + \alpha(t)\Lambda(i,k)(\vec{x} - \vec{w}_i(t))$$

Kohonenovy mapy

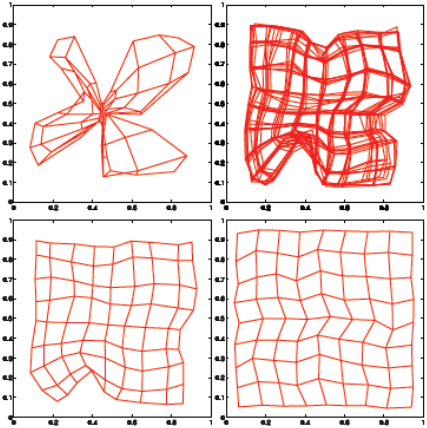
Příklad – rovnoměrně rozdělená data a řetízek



Simon Haykin: Neural Networks and Learning Machines, 3rd edition, 2008

Kohonenovy mapy

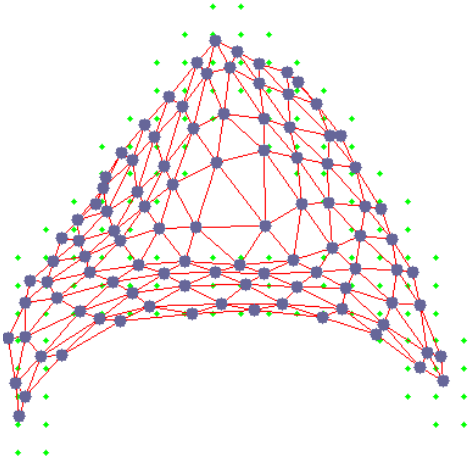
Příklad – rovnoměrně rozdělená data a mřížka



Paul Rojas: Neural Networks - A Systematic Introduction, Springer, 1996

Kohonenovy mapy

Příklad – nerovnoměrně rozdělená data a mřížka

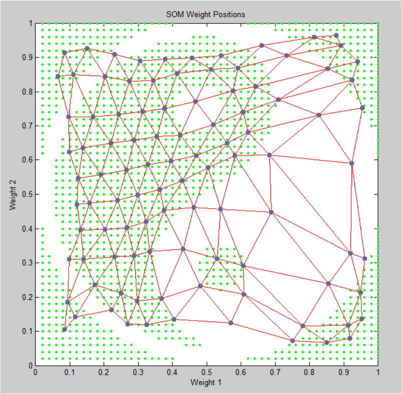


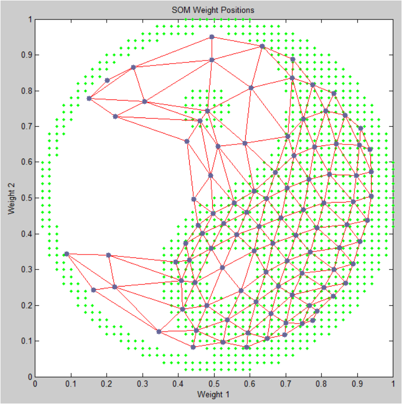
Kohonenovy mapy – analýza algoritmu učení

Otázka – jak dobře se Kohonenova síť naučila?

- konvergence algoritmu?, jak dlouho se učí?
- do jaké míry je zachována topologie zobrazení?
- je zobrazení správné?
- vliv parametrů α, σ

Kohonenovy mapy - příklad





Kohonenovy mapy – analýza algoritmu učení

Jak dobře se Kohonenova síť naučila?

- **Vnitřní suma čtverců (WCSS)**
  - Součet čtverců vzdáleností mezi jednotlivými body a jim příslušejícími neurony.

$$WCSS = \sum_{i=1}^{k} \sum_{x \in C_i} ||\vec{x} - \mu_i||^2$$

- **Topografická chyba**
  - Popisuje kvalitu ,,natažení" mřížky sítě na vstupní data
  - Procento vzorů, pro které první a druhý nejbližší neuron nejsou sousedy v mřížce

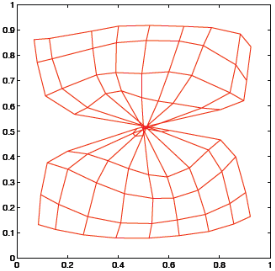
$$E = \frac{1}{n} \sum_{i=1}^{n} u(\vec{x}_i)$$

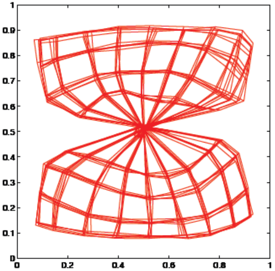
$u(\vec{x}_i) = 1$ p.k. první a druhý nejbližší neuron nejsou sousedy v mřížce

Kohonenovy mapy

Zásadní vliv má volba parametrů α, σ

- závisí na úloze
- rychlé klesání σ ... topologické zvraty v počáteční fázi učení (překroucení, či zborcení mřížky)





- rychlé klesání α ... zamrznutí sítě v některém z mělkých lokálních minim nebo dokonce mimo lokální minima

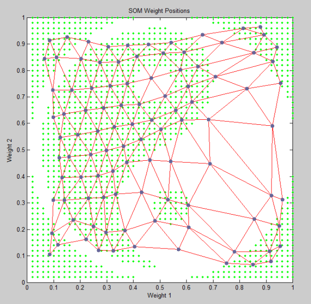
Kohonenovy mapy

Řešení: dynamické změny parametrů
adaptace ve dvou fázích:

- **Organizace (určení oblastí):**
  - široká okolí (zpočátku celá síť), pozvolna se zužují
  - α je relativně velká (blízko 1), téměř neměnná
- **Ustálení:**
  - malá okolí (v závěru jen jeden neuron),
  - α rychle klesá k nule

Kohonenovy mapy - vizualizace

- 2D data - snadno

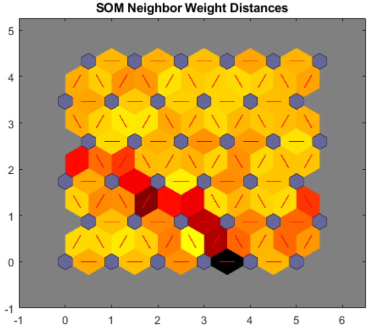


- U-matice (matice vzdáleností)
- Projekce do 2D
  - S využitím PCA analýzy
  - Sammonova projekce

Kohonenovy mapy - vizualizace

U-matice (matice vzdáleností)

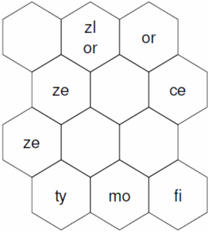
- matice vzdáleností mezi váhovými vektory jednotlivých neuronů
- čím větší vzdálenost, tím tmavší barva
- jak odhalit shluky? ... podle pohoří (tmavých čar)

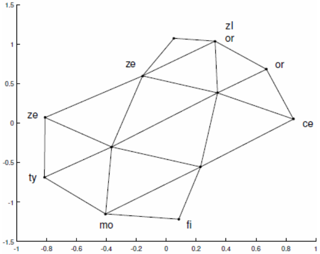


Kohonenovy mapy - vizualizace

Sammonova projekce

- netransformuje osy, ale znovu umísťuje body v novém (méně-dimenzionálním) prostoru
- při umísťování se snaží zachovat vztahy v datech (data, která si byla blízko v původním prostoru, budou blízko i v novém prostoru)





Zdroj: Kateřina Horaisová: slidy k předmětu Neuronové sítě 2, FJFI ČVUT Děčín

Hybridní modely

Kombinace učení s učitelem a učení bez učitele:

- LVQ (Učení vektorové kvantizace)
- Sítě se vstřícným šířením (Counter-propagation)
- RBF-sítě
- ART (Adaptive Resonance Theory)
- ...

LVQ (učení vektorové kvantizace)

- LVQ = Learning Vector Quantization
- varianta Kohonenovy mapy (SOM)
- hybridní neuronová síť - kombinace učení s učitelem a bez učitele

Použití

- klasifikace (do více tříd)
- komprese dat, např. pro přenos dat v digitálním kanálu (video)
- obecně pro snížení počtu vzorků
- možnost adaptivního zvyšování počtu tříd

LVQ (učení vektorové kvantizace)

Vektorová kvantizace

- je dáno: množina trénovacích vzorů (bodů)
- výsledek: množina reprezentantů ve vstupním prostoru
    - snaha o co nejlepší pokrytí vstupního prostoru
    - respektují statistické rozdělení vektorů (hustotu dat) $\rightarrow$ každému reprezentantovi by měl odpovídat cca. stejný počet bodů, které jsou k němu nejblíže

Příklady (už bylo)

- Algoritmus k středů
- Kompetitivní síť, Kohonenova mapa - na vektory vah k výstupním neuronům se lze dívat jako na reprezentanty

LVQ (učení vektorové kvantizace)

LVQ - základní princip

- Je dáno: množina trénovacích vzorů ve tvaru $(\vec{x}, \vec{y})$ (požadovaný vstup, požadovaný výstup)

1. Vypočteme reprezentanty (centroidy) pomocí samoorganizace
2. Ke každému centroidu přiřadíme ty vzory, ke kterým je nejblíže a spočteme četnost jednotlivých tříd
3. Centroidu přiřadíme nejčetnější třídu
4. Síť pak můžeme použít pro klasifikaci (rozpoznávání)

LVQ (učení vektorové kantizace) - varianty

LVQ1 - motivace

- **Adaptační pravidlo pro standardní Kohonenovu mapu:**
    - Nechť k je vítězný neuron pro předložený vstupní vektor $\vec{x}$:

$$k = argmin_i d_i = argmin_i \|\vec{x} - \vec{w}_i\|^2$$

    - Každý neuron i zadaptuje své váhy podle pravidla:

$$\vec{w}_i(t+1) = \vec{w}_i(t) + \alpha(t)\Lambda(i,k)(\vec{x} - \vec{w}_i(t))$$

- **Idea:** $\vec{x}$ by měl patřit do stejné třídy, jako nejbližší (vítězný) neuron k ($\vec{w}_k$)

LVQ (učení vektorové kvantizace) - varianty

LVQ1 - adaptační pravidlo

- Minimalizace stupně chybné klasifikace
- Vítězný neuron k zadaptuje své váhy podle pravidla:

$$\vec{w}_k(t+1) = \begin{cases} \vec{w}_k(t) + \alpha(t)(\vec{x} - \vec{w}_k(t)) \\ \text{pokud jsou } \vec{x} \text{ a } \vec{w}_k \text{ klasifikovány stejně} \\ \vec{w}_k(t) - \alpha(t)(\vec{x} - \vec{w}_k(t)) \\ \text{pokud jsou } \vec{x} \text{ a } \vec{w}_k \text{ klasifikovány jinak} \end{cases}$$

- Ostatní neurony $i \neq k$ své váhy nemění:

$$\vec{w}_i(t+1) = \vec{w}_i(t)$$

- $0 < \alpha(t) < 1$ ... parametr učení (vigilanční / bdělostní koeficient)

LVQ (učení vektorové kvantizace) - varianty

LVQ2 a LVQ2.1

- Minimalizace počtu bodů blízko hranic
- **Idea:** adaptace dvou nejbližších sousedů k a l současně
  - k musí patřit ke správné třídě a l k nesprávné
  - $\vec{x}$ musí být z ,,okénka" ... z dělící nadplochy mezi $\vec{w}_k$ a $\vec{w}_l$
- **Definice okénka:**

$$min\left(\frac{d_k}{d_l}, \frac{d_l}{d_k}\right) > s, \;\; s = \frac{1-w}{1+w}$$

  - d_k ... vzdálenost $\vec{w}_k$ a $\vec{x}$
  - $0.2 < w < 0.3$ ... relativní šířka okénka (určeno autory experiementálně)

LVQ (učení vektorové kvantizace) - varianty

LVQ2.1 - adaptační pravidlo

- Neurony zadaptují své váhy podle pravidel:

$$\begin{aligned}\vec{w}_k(t+1) &= \vec{w}_k(t) + \alpha(t)(\vec{x}(t) - \vec{w}_k(t)) \\ \vec{w}_l(t+1) &= \vec{w}_l(t) - \alpha(t)(\vec{x}(t) - \vec{w}_l(t)) \\ \vec{w}_i(t+1) &= \vec{w}_i(t)\end{aligned}$$

- k a l jsou nejblíže k $\vec{x}$, $i \neq k, l$
- k a $\vec{x}$ patří do stejné třídy, l patří do jiné,
- $\vec{x}$ je z okénka
- $0 < \alpha(t) < 1$ ... rychlost učení

LVQ (učení vektorové kvantizace) - varianty

LVQ3

- Snaha o optimální umístění váhových vektorů
- Aproximuje rozložení tříd a stabilizuje řešení
- LVQ3 - adaptační pravidlo:
  - k a l jsou nejblíže k $\vec{x}$, k a $\vec{x}$ patří do stejné třídy, l patří do jiné, $\vec{x}$ je z okénka:

$$\begin{aligned} \vec{w}_k(t+1) &= \vec{w}_k(t) + \alpha(t)(\vec{x}(t) - \vec{w}_k(t)) \\ \vec{w}_l(t+1) &= \vec{w}_l(t) - \alpha(t)(\vec{x}(t) - \vec{w}_l(t)) \end{aligned}$$

  - k a l jsou nejblíže k $\vec{x}$, k, l a $\vec{x}$ patří do stejné třídy:

$$\begin{aligned} \vec{w}_k(t+1) &= \vec{w}_k(t) + \epsilon\alpha(t)(\vec{x}(t) - \vec{w}_k(t)) \\ \vec{w}_l(t+1) &= \vec{w}_l(t) + \epsilon\alpha(t)(\vec{x}(t) - \vec{w}_l(t)) \end{aligned}$$

  - experimentálně: $0.1 < \epsilon < 0.5$, $0.2 < w \leq 0.3$

LVQ (učení vektorové kvantizace)

Přesnost klasifikace záleží na

- Vhodném počtu neuronů
- Na jejich vhodné inicializaci ... pomocí SOM
- Na vhodném $\alpha(t)$ a dalších parametrech
- Na vhodném kritériu ukončení učení, počtu iterací

Doporučované použití

- Začít samoorganizací
- Použít LVQ1
- Doladit pomocí LVQ2.1 či LVQ3

LVQ - Jak je to v Matlabu

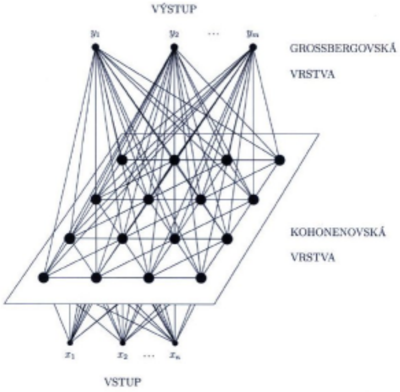
- *newlvq* ... vytvoření modelu
  - lvqnet(hiddenSize,lvqLR,lvqLF)
  - lvqLR ... parametr učení, implicitně 0.01
  - lvqLF ... učící funkce, defaultně *'learnlv1'*, další je *'learnlv2'*
- *train* ... učení
  - net = train(net,x, t).
- *sim* ... rozpoznávání
  - y = net(x);
  - perf = perform(net,y,t)
  - classes = vec2ind(y);

Sítě se vstřícným šířením (Counter-propagation)

(Hecht-Nielsen, 1987)

Architektura

- Tři vrstvy neuronů:
  - Vstupní vrstva
  - Kohonenovská (klastrovací) vrstva
  - Grossbergovská vrstva
- Učení s učitelem i bez učitele
- Rozpoznávání vzorů
- Fungují jako vyhledávací tabulka

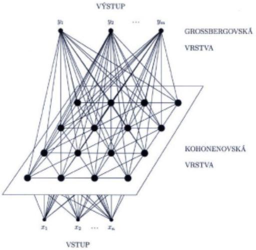


obrázek převzat z I.Mrázová, Neuronové sítě

Sítě se vstřícným šířením (Counter-propagation)

Terminologie

- Vstupní vrstva … n neuronů
- Kohonenovská vrstva … mřížka s K neurony
- Grossbergovská vrstva … m neuronů
- Síť zobrazí vstupní vektor $\vec{x} \in R^n$ na výstupní vektor $\vec{y} \in R^m$
  - z_i … výstupy (aktivity) neuronů v Kohonenovské vrstvě
  - y_l … výstupy (aktivity) neuronů v Grossbergovské vrstvě
  - w_{ji} … váhy hran mezi vstupní a Kohonenovskou vrstvou
  - v_{il} … váhy hran mezi Kohonenovskou a Grossbergovskou vrstvou



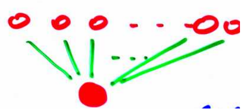
Sítě se vstřícným šířením (Counter-propagation)

Režim vybavování

- zobrazení $f : R^n \to R^m$:
- Vstupní vektor $\vec{x}$ vybudí jeden neuron v Kohonenovské vrstvě (vítězný).. k-tý
- Grossbergovská (výstupní) vrstva:
  - Provádí standardní skalární součin:

$$y_l = \sum_{i=1}^{K} v_{il} z_i = v_{kl}$$

$\to$ výstup sítě: $\vec{y} = \vec{v_k}$



Grossbergova (výstupní) hvězda

- **vyhledávací tabulka:** Grossbergovská vrstva provádí výběr jednoho vektoru z k vektorů ($\sim$ odpovídá vahám hran od vítězného k-tého neuronu v Kohonenově mřížce)

Sítě se vstřícným šířením (Counter-propagation)

Algoritmus

1. **Inicializace:** Zvolíme náhodné hodnoty synaptických vah
2. Předložíme nový trénovací vzor ve tvaru $(\vec{x}, \vec{t})$ = (vstup, požadovaný výstup).
3. Spočítáme vzdálenosti d_i mezi $\vec{x}$ a $\vec{w}_i$ pro každý neuron i v Kohonenovské vrstvě. Použijeme např. upravenou euklidovskou metriku:

$$d_i = \sum_j (x_j - w_{ji})^2$$

4. Vyber neuron k s minimální vzdáleností d_k jako ,,vítěze"

$$k = argmin_i d_i$$

Sítě se vstřícným šířením (Counter-propagation)

Algoritmus - pokračování

5. Aktualizujeme váhy w_{ji} mezi vstupním neuronem j a neurony i Kohonenovské vrstvy, které se nacházejí v okolí vítězného neuronu k tak, aby lépe odpovídaly předloženému vzoru:

$$\vec{w}_i(t+1) = \vec{w}_i(t) = \alpha(t)\Lambda(i,k)(t)(\vec{x} - \vec{w}_i(t)),$$

   - $\Lambda(i,k)$ ... funkce okolí
   - $0 < \alpha(t) < 1$ ... parametr učení pro váhy mezi vstupní a Kohonenovskou vrstvou, klesá v čase.

Sítě se vstřícným šířením (Counter-propagation)

Algoritmus - dokončení

6. Aktualizujte váhy v_{kl} mezi ,,vítězným" neuronem k z Kohonenovské vrstvy a neurony l Grossbergovské vrstvy tak, aby výstupní vektor lépe odpovídal požadované odezvě:

$$v_{kl}(t+1) = v_{kl}(t) + \gamma(t_l - v_{kl}(t)),$$

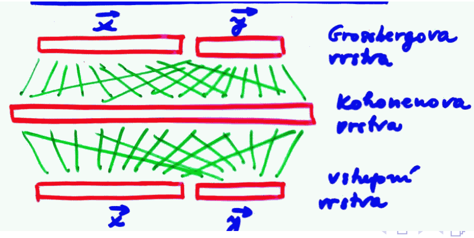
   - $0 < \gamma < 1$ ... parametr učení pro váhy mezi Kohonenovskou a Grossbergovskou vrstvou,
   - t_l ... označuje požadovanou aktivitu neuronu l Grossbergovské vrstvy

7. Pokračujeme krokem (2)

Sítě se vstřícným šířením (Counter-propagation)

Příklady použití

- Rozpoznávání vzorů
- Komprese dat, redukce dimenzionality
  - např. přenos obrazů, videa
- Podobně jako vrstevnatá neuronová síť
  - efektivnější výpočet, rychlejší adaptace
  - nižší přesnost
  - citlivé na volbu parametrů
- Původní využití: reprezentace zobrazení f a f^{-1} zároveň:

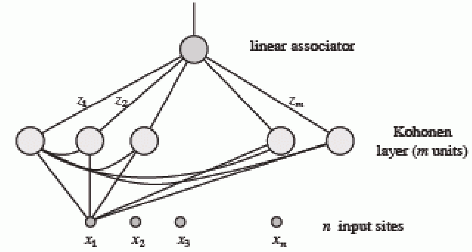


RBF-sítě (Sítě s lokálními jednotkami)

Radial basis functions

(Moody, Darken, 1989)

- Hybridní architektura
- Učení s učitelem
- Rozdíl od counter-propagation: Gaussovské jednotky v Kohonenovské vrstvě
- Zjednodušený model:



Paul Rojas: Neural Networks - A Systematic Introduction, Springer, 1996

RBF-sítě (Sítě s lokálními jednotkami)

Neurony v Kohonenovské vrstvě

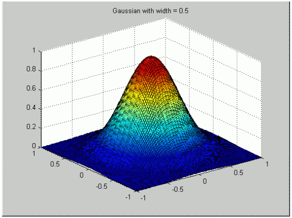
- Lokální výpočetní jednotky (RBF-jednotky)
- Neuron spočte svůj vnitřní potenciál ξ a výstup y podle:

$$\xi = \frac{\|\overrightarrow{x} - \overrightarrow{w}\|}{h}$$

- Gaussovská (radiální) přenosová funkce:

$$z = f(\xi) = e^{-\frac{\xi^2}{\alpha}} = e^{-\frac{\|\overrightarrow{x} - \overrightarrow{w}\|^2}{\alpha h^2}}$$

- $\vec{x} \in R^n$ ... vstupní vektor
- $\vec{w} \in R^n$ ... váhový vektor neuronu
- h ... konstanta (pro daný neuron)
  ... šířka okolí
- α ... konstanta



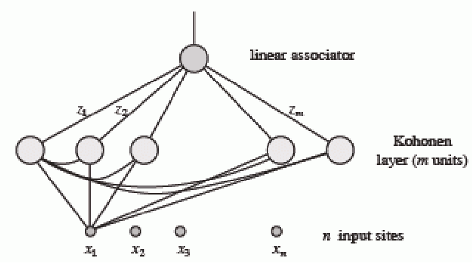
RBF-sítě (Sítě s lokálními jednotkami)

Celková funkce sítě

- $f : R^n \to R^m$:

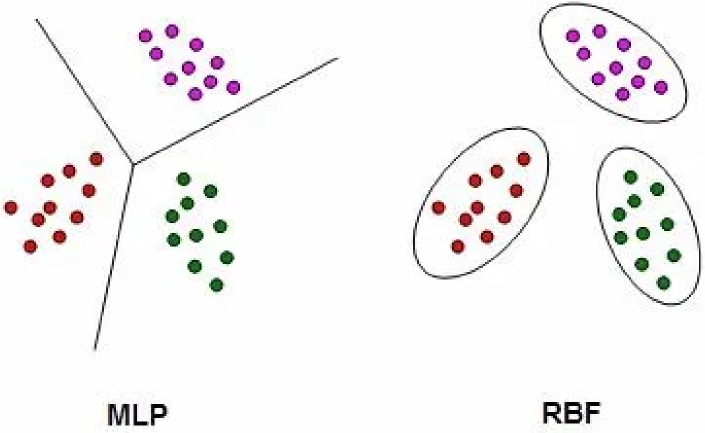
$$f_l(x_1, \ldots, x_n) = \sum_{i=1}^{K} v_{il} z_i = \sum_{i=1}^{K} v_{il} e^{-\frac{\|\vec{x} - \vec{w}_i\|^2}{\alpha h_i^2}}$$

- $\vec{v}_l \in R^K$ ... váhový vektor ze skrytých neuronů do výstupního neuronu l
- Výstupní neurony jsou lineární jednotky



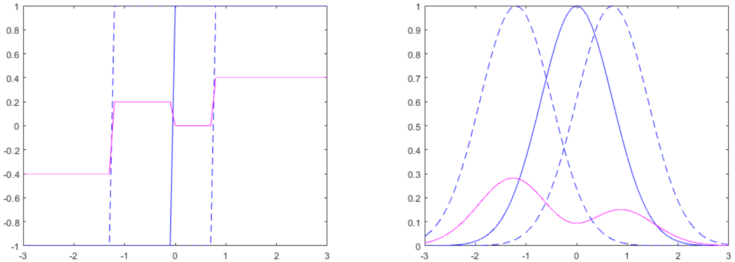
RBF-sítě (Sítě s lokálními jednotkami)

RBF-sítě a klasifikace



RBF-sítě (Sítě s lokálními jednotkami)

RBF-sítě a lineární regrese



RBF-sítě (Sítě s lokálními jednotkami)

Algoritmus učení

- **Vstup:** trénovací množina s N vzory ve tvaru $(\vec{x}_p, \vec{d}_p) =$ (vstup, požadovaný výstup).
- **Výstup:** parametry sítě - váhy hran a parametry neuronů

Algoritmus učení má tři fáze:

1. Přepočítáme středy centroidů (RBF-jednotek) ... váhy w_{ji} ze vstupní do Kohonenovské vrstvy
2. Přepočítáme šířky okolí centroidů h_i a další parametry
3. Přepočítáme váhy do výstupní vrstvy ... v_{il}

RBF-sítě (Sítě s lokálními jednotkami)

Algoritmus učení - má tři fáze

1. Spočítáme středy centroidů ... váhy w_{ji} ze vstupní do Kohonenovské vrstvy
   - samoorganizace (učení bez učitele)
   - viz. counter-propagation
2. Přepočítáme šířky okolí centroidů h_i a další parametry
   - např. podle vzdálenosti nejbližších sousedů (není třeba znova předkládat trénovací vzory)
3. Přepočítáme váhy do výstupní vrstvy ... v_{il}
   - např. pomocí algoritmu zpětného šíření (učení s učitelem)

RBF-sítě (Sítě s lokálními jednotkami)

Algoritmus učení - výpočet vah do výstupní vrstvy... v_{il}

- Pomocí algoritmu zpětného šíření (učení s učitelem)
- N trénovacích vzorů ve tvaru $(\vec{x}_p, \vec{d}_p)$ = (vstup, požadovaný výstup)
- Chybová funkce:

$$E = \frac{1}{2}\sum_{p=1}^{N}\sum_{l=1}^{m}(\sum_{i=1}^{K} v_{il}z_i - d_{pl})^2 = \frac{1}{2}\sum_{p=1}^{N}\sum_{l=1}^{m}(\sum_{i=1}^{K} v_{il}e^{-\frac{\|\vec{x}_p-\vec{w}_i\|^2}{\alpha h_i^2}} - d_{pl})^2$$

- Adaptační pravidlo pro jeden trénovací vzor

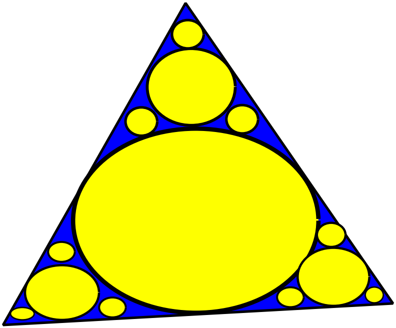
$$\Delta v_{il} \sim -\frac{\partial E}{\partial v_{il}} \sim \gamma e^{-\frac{\|\vec{x}_p-\vec{w}_i\|^2}{\alpha h_i^2}}(d_l - \sum_{i=1}^{K} v_{il}e^{-\frac{\|\vec{x}_p-\vec{w}_i\|^2}{\alpha h_i^2}})$$

$$= \gamma z_i(d_l - \sum_{i=1}^{K} v_{il}z_i) = \gamma z_i(d_l - y_l)$$

RBF-sítě (Sítě s lokálními jednotkami)

Analýza modelu

- Univerzální aproximátor (stačí jedna skrytá vrstva)
  ... ale potřebný počet lokálních jednotek roste exponenciálně
- potřebuje mnohem víc neuronů než vrstevnatá neuronová síť s jednou skrytou vrstvou:



RBF-sítě (Sítě s lokálními jednotkami)

Analýza modelu

- Univerzální aproximátor (stačí jedna skrytá vrstva)
  ... ale potřebný počet lokálních jednotek roste exponenciálně
- Silný v regresních a interpolačních úlohách
- Alternativa vrstevnatých neuronových sítí, pro některé typy problémů se hodí více, pro některé méně
- Rychlé učení (až o dva řády rychlejší než vrstevnaté neuronové sítě)
- Neumí si poradit s irelevantními/odlehlými vstupy.
- Obtížně se učí (kombinace učení bez učitele a s učitelem), citlivé k volbě parametrů (např. počet neuronů)

RBF - Jak je to v Matlabu

- *newrbe* ... vytvoření modelu
  - počet výpočetních jednotek je roven počtu trénovacích vzorů
  - net = newrbe(X,T,SC)
  - X ... vstupní vzory
  - T ... výstupní vzory
  - SC ... šířka okolí
- *newrb* ... vytvoření modelu
  - přidává výpočetní jednotky, dokud MSE není menší než daná mez (EG)
  - net = newrb(X,T,EG,SC)
  - X ... vstupní vzory
  - T ... výstupní vzory
  - EG ... požadovaná MSE
  - SC ... šířka okolí
- rozpoznávání
  - Y = net(X);

Doplnění: modulární neuronové sítě

- ART neuronové sítě (Adaptive resonance theory)
- Kaskádová korelace
- ...

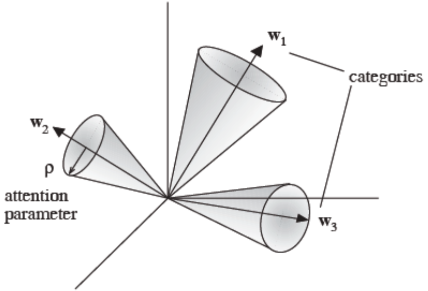
ART-sítě (Adaptive resonance theory)

(Grossberg, Carpenter, 1986)

- Hybridní architektura bližší biologickým principům - částečně modulární
- Učení bez učitele i s učitelem
- Online učení

Použití

- Shlukování
- Rozpoznávání znaků, řečových segmentů apod.



w_i ... vektor vah, reprezentuje všechny vzory z kuželu

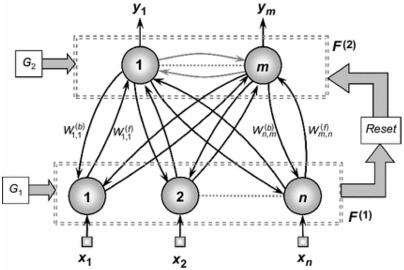
ρ ... parametr bdělosti (poloměr kuželu)

Paul Rojas: Neural Networks - A Systematic Introduction, Springer, 1996

ART-sítě (Adaptive resonance theory)

Architektura ART-1

- Dvouvrstvá rekurentní síť
  - Porovnávací (vstupní) vrstva ... n neuronů
  - Rozpoznávací (výstupní) vrstva ... m neuronů
- ART-1 ... binární vstupy
- ART-2 ... reálné vstupy



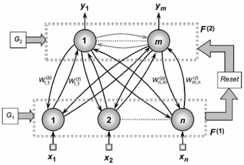
Základní vlastnosti

- plasticita (reaguje na nové vstupy)
- stabilita (nezapomene, co už ví)

ART-sítě (Adaptive resonance theory)

Vazby mezi neurony:

- ve výstupní vrstvě ... laterální inhibice
- ze vstupní do výstupní vrstvy
  (váhy $w_{ij}, i = 1, ..., n, j = 1, ..., m,$)
- z výstupních neuronů ke vstupním
  (váhy $t_{ij}, i = 1, ..., n, j = 1, ..., m,$)
  ... pro porovnání skutečné podobnosti s předloženým vzorem (založena na skalárním součinu)
- Řídící signály ... G1, G2, Reset



ART-sítě (Adaptive resonance theory)

Test bdělosti

- práh bdělosti ρ … určuje, jak blízko musí být předložený vzor k uloženému, aby mohly patřit do stejné kategorie
- Mechanismus vypnutí (zablokování) neuronu s maximální odezvou
    - pokud neuron vyhrál v kompetici, ale je daleko $\rightarrow$ vypnu ho a zkusím to znova
    - pokud nenaleznu žádný neuron dostatečně blízko $\rightarrow$ přidám nový neuron

$\rightarrow$ stabilita $\times$ plasticita sítě
$\rightarrow$ síť má velké problémy i při jen trochu zašuměných vzorech (příliš narůstá počet uložených vzorů)

ART-sítě (Adaptive resonance theory)

Algoritmus učení – má 5 fází:

1. inicializační - nastavení počátečního stavu sítě
2. rozpoznávací - dopředný výpočet - naleznu vítězný neuron v rozpoznávací vrstvě
3. porovnávací - zpětný výpočet - provedu test bdělosti
4. vyhledávací - hledám jiný vítězný neuron
5. adaptační - adaptace vah u vítězného neuronu

ART-sítě (Adaptive resonance theory)

Algoritmus učení – inicializační fáze

1. **Počáteční inicializace vah:**

$$
\begin{aligned}
t_{ij}(0) &= 1, \\
w_{ij}(0) &= \frac{1}{1+n}, \\
i = 1, \dots, n \quad &, \quad j = 1, \dots, m, \\
0 \leq \quad \rho \quad &\leq 1
\end{aligned}
$$

- $w_{ij}(t)$ ... 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 (vzor specifikovaný výstupním neuronem j)
- ρ ... práh bdělosti

ART-sítě (Adaptive resonance theory)

Algoritmus učení – inicializační a rozpoznávací fáze

2. Předlož nový vstupní vzor: $\vec{x}(t) = \{x_1, \ldots, x_n\}$
3. Spočti odezvu (aktivitu) neuronů ve výstupní (rozpoznávací) vrstvě:

$$y_j(t) = \sum_{i=1}^{n} w_{ij}(t)x_i, \; j = 1, \ldots, m$$

   - $y_j(t)$ ... aktivita výstupního neuronu j v čase t

4. Vyber neuron k, který nejlépe odpovídá předloženému vzoru (např. pomocí laterální interakce):

$$k = argmax\{y_j\}$$

ART-sítě (Adaptive resonance theory)

Algoritmus učení – porovnávací a vyhledávací fáze

5. **Test bdělosti:**
   - Výpočet bdělosti μ vítězného neuronu k podle:

$$\mu = \frac{\|T.\vec{x}\|}{\|\vec{x}\|},$$

$$\|T.\vec{x}\| = \sum_{i=1}^{n} t_{ik}(t)x_i, \ \|\vec{x}\| = \sum_{i=1}^{n} x_i,$$

   - Pokud platí $\mu > \rho$, pokračuj krokem 7, jinak pokračuj krokem 6.
6. **Zmraz (zablokuj) neuron k s největší odezvou:**
   - Nastav výstup neuronu k dočasně na nulu.
   - Opakuj krok 3 (neuron k se neúčastní maximalizace).

ART-sítě (Adaptive resonance theory)

Algoritmus učení – adaptační fáze

7. Pokud nebyl nalezen vhodný neuron, přidej do sítě nový neuron jako ,,vítězný".
8. **Adaptace vah u ,,vítězného" neuronu k:**

$$
\begin{aligned}
t_{ik}(t+1) &= t_{ik}(t)x_i, \\
w_{ik}(t+1) &= \frac{t_{ik}(t)x_i}{0.5 + \sum_{l=1}^{n} t_{lk}(t)x_l}
\end{aligned}
$$

9. Odblokuj všechny zmražené neurony a opakuj krok 2.

ART-sítě (Adaptive resonance theory)

Analýza modelu

- Hlavní výhody: Stabilita a plasticita sítě
- Síť sama určí správný počet neuronů
- Velká citlivost na počáteční volbu parametrů:
    - práh bdělosti
    - pořadí předkládání vzorů
- Velká citlivost na šum v datech

Příklady aplikací

- Shlukování
- Rozpoznávání znaků, řečových segmentů apod.

Konstrukční algoritmy - Kaskádová korelace

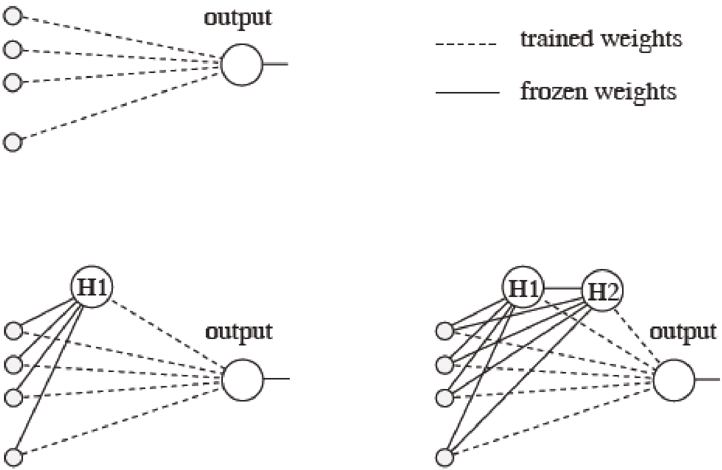
(Fahlman, Labiere, 1990)

- robustní rostoucí architektura vrstevnaté neuronové sítě

Princip

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

Kaskádová korelace



Paul Rojas: Neural Networks - A Systematic Introduction, Springer, 1996

Kaskádová korelace

Algoritmus učení

- Minimalizace MSE na výstupu sítě

Učení probíhá ve dvou fázích:

- **První fáze:** Adaptace sítě, např. pomocí algoritmu Quickprop
    - pokud je MSE na výstupu dostatečně nízká, KONEC
    - jinak přidáme nový neuron
- **Druhá fáze:** Přidání nového neuronu
    - nový neuron je adaptován tak, aby maximalizoval korelaci mezi svým výstupem a chybou na výstupu sítě
      $\rightarrow$ přidávaný neuron se ,,naučí" nějaký příznak, který vysoce koreluje s aktuální (zbývající) chybou
    - Váhy do nově přidaného neuronu jsou zmrazeny a v dalších fázích se doučují jen váhy na výstup

Kaskádová korelace

Algoritmus učení

- Cílem učení skrytých neuronů je maximalizace S:

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

- p ... počet trénovacích vzorů
- V_i ... výstup přidávaného neuronu pro i-tý vzor
- $\overline{V}$ ... průměrný výstup přidávaného neuronu
- E_i ... MSE pro i-tý vzor
- $\overline{E}$ ... průměrná chyba

Kaskádová korelace

Algoritmus učení

- Cílem učení skrytých neuronů je maximalizace S:

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

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

  - σ ... znaménko korelace mezi výstupem a přidávaným neuronem
  - f_i' ... derivace přenosové funkce pro i-tý vzor
  - I_{ik} ... k-tý vstup přidávaného neuronu pro i-tý vzor

Kaskádová korelace

Analýza algoritmu

- Snadné rozšíření na více výstupů
- Síť sama určí správný počet neuronů ... uživatel ho nemusí specifikovat
- Rychlé učení ... v každém kroku se adaptuje jen jeden neuron, váhy do stávajících neuronů už se neadaptují → stabilita
- Nebezpečí přeučení ... saturace neuronů