Učení bez učitele

Minule: Učení bez učitele (unsupervised learning)

- Kompetitivní modely
- Klastrování (shlukování)

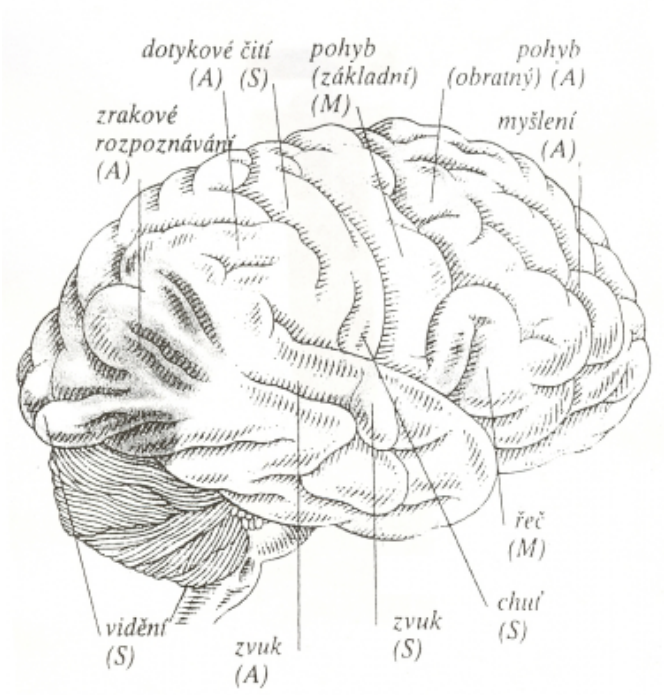
Zbývá:

- Kohonenovy mapy (1989)
- Hybridní modely (kombinace učení s učitelem a bez učitele):
  - LVQ (Učení vektorové kvantizace)
  - Sítě se vstřícným šířením (Counterpropagation)
  - RBF-sítě
  - ART (Adaptive Resonance Theory)

Kohonenovy mapy

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

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

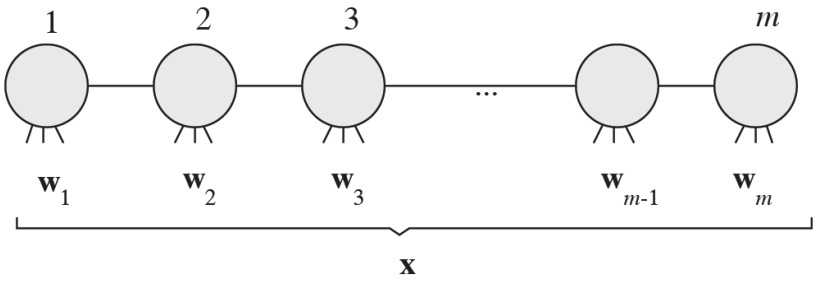


Biologická motivace

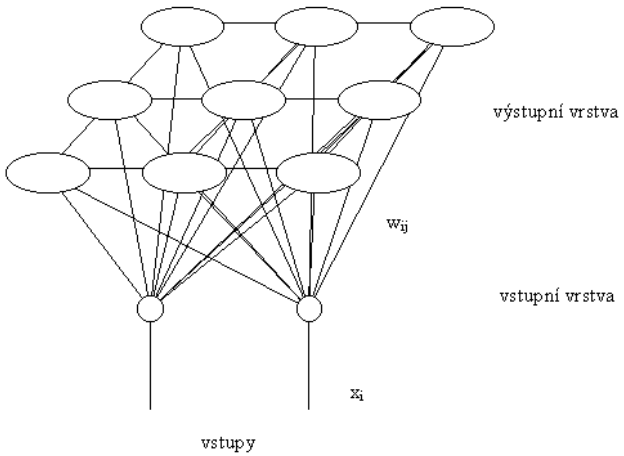
- Mozková kůra: specializovzné oblasti neuronů - více citlivé na určitý druh podnětů
- Fyzicky blízké neurony reagují podobně - laterální vazby vedou k excitaci blízkých a k inhibici vzdálených neuronů

Kohonenovy mapy

Architektura 1D - řetízek:



Architektura 2D - mřížka:



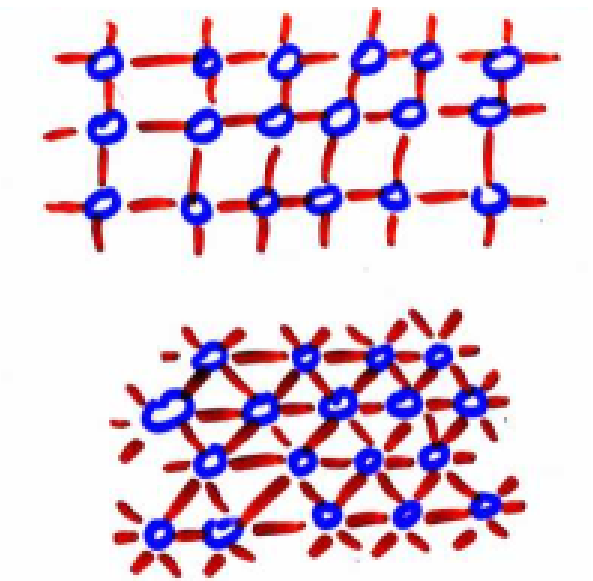
- Učení bez učitele
- Rozpoznávání
- Redukce dimenzionality dat
- Vizualizace dat
- Ekonomické aplikace

Kohonenovy mapy

Architektura

- Výstupní neurony jsou uspořádány do mřížky
- Na mřížce je definovaná sousednost fyzických neuronů
  (× logická sousednost daná blízkostí váhových vektorů)
- **Cíl:**
  - sousední neurony by měly také reagovat na velmi podobné signály
  - specializace každého neuronu na jinou část vstupního prostoru

Různé topologie mřížky
(pro dvě dimenze)

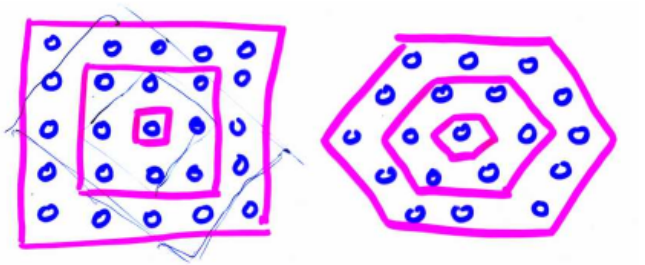


Kohonenovy mapy – učení

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

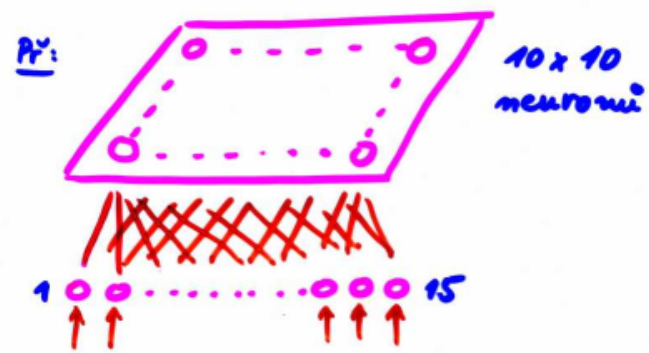
Dvě možné interpretace z pohledu aplikací

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

Kohonenovy mapy

Snížení dimenze dat při zachování topologie – příklad:

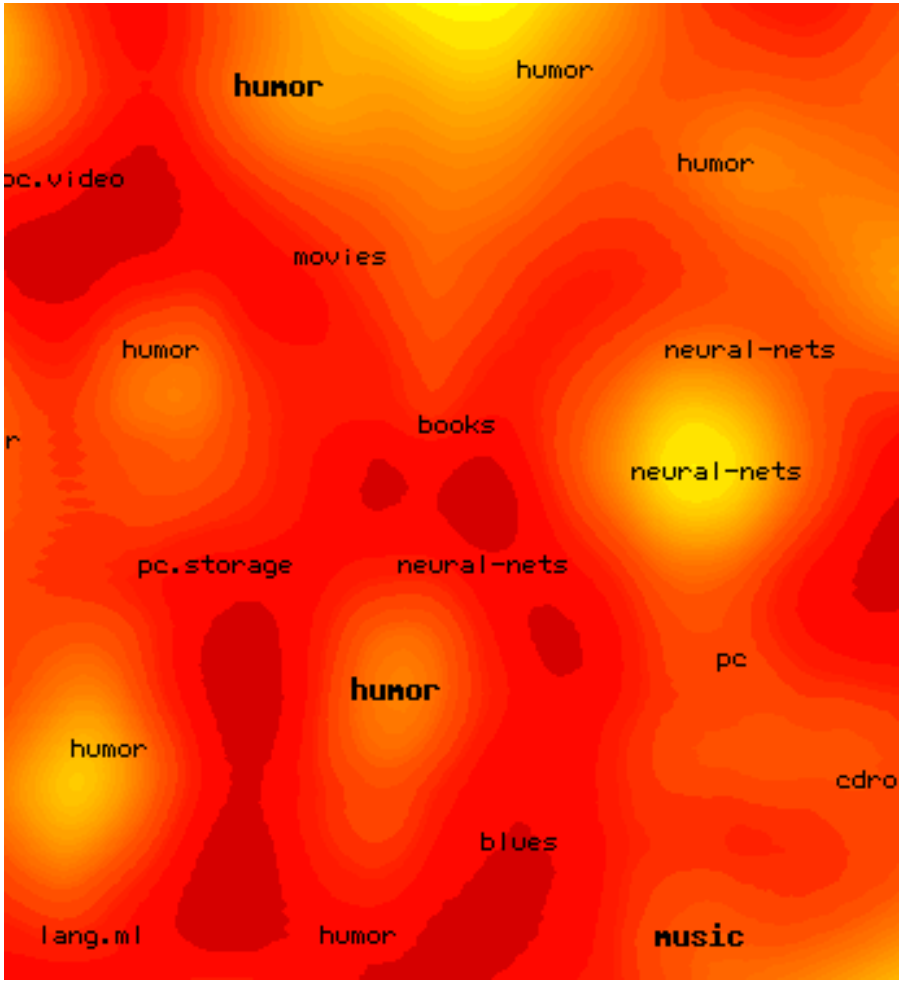
- Síť zobrazí 15-dimenzionální vstupní prostor (jeho část) do 2-dimenzionálního výstupního prostoru, navíc bude zachována sousednost obrazů tohoto zobrazení
- Pro velmi hustou síť je navíc transformace spojitá
- Vizualizace dat



Kohonenovy mapy

Snížení dimenze dat při zachování topologie

- Příklad aplikace: WEBSOM (http://websom.hut.fi)
- 2D-vizualizace blízkosti (podobnosti) webových dokumentů



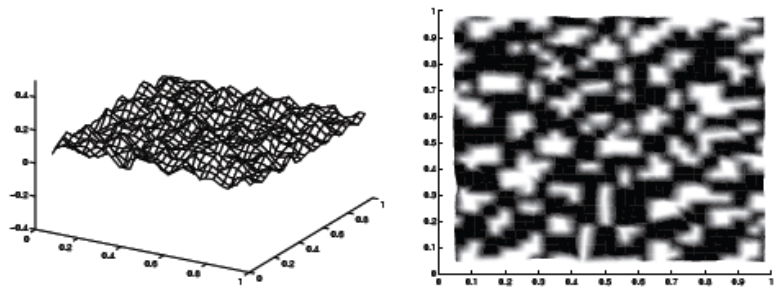
Kohonenovy mapy

Shlukování (klastrování)

- Na vektor vah do výstupního neuronu se lze dívat jako na bod vstupního prostoru
- Výstupní neurony se snaží co nejlépe pokrýt vstupní prostor (jeho část) a respektovat statistické rozdělení vektorů *(vector quantization)*
- Výstupní neurony jsou reprezentanti vstupních dat (shluků)
- Navíc se zachovává struktura fyzické sousednosti

Příklad

- 2-dimenzionální síť ve 3-dimenzionálním vstupním prostoru



Kohonenovy mapy

Definice okolí - v jedné dimenzi (řetízek)

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

Definice okolí - ve více dimenzích

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

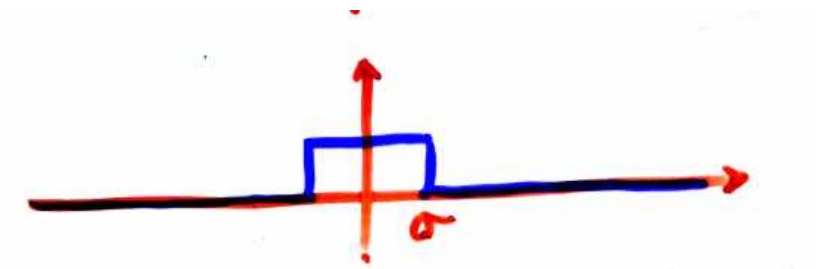
Kohonenovy mapy

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

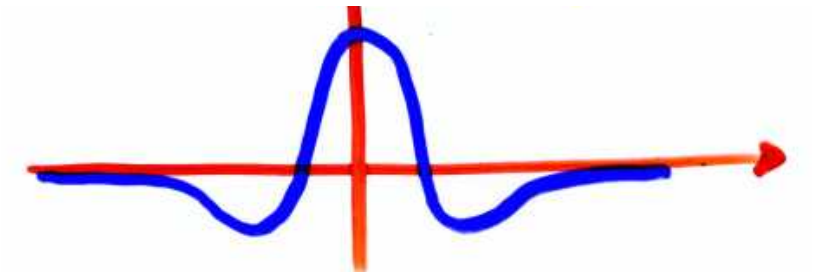
- **Diskrétní okolí**
  - $\Lambda(i, k) = 1$ pro všechny i z okolí k s poloměrem nejvýše σ, $\Lambda(i, k) = 0$ pro ostatní
  - efektivní z hlediska implementace, minimální režie (stačí adaptovat neurony z okolí)
  - σ je šířka okolí (nejjednodušeji $\sigma = 1$)



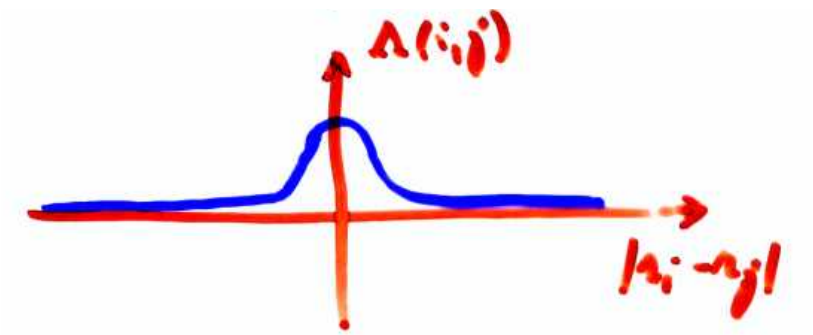
Kohonenovy mapy

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 paametry

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

Kohonenovy mapy

Algoritmus

1. Zvol hodnoty vah mezi vstupními a výstupními neurony jako malé náhodné hodnoty. Zvol počáteční vigilanční koeficient α, poloměr okolí σ a funkci laterální interakce Λ.
2. Předlož nový trénovací vzor $\vec{x}$
3. Spočítej vzdálenosti d_i mezi $\vec{x}$ a $\vec{w}_i$ pro každý výstupní neuron i:
$$d_i = \sum_j (x_j - w_{ji})^2$$
4. Vyber výstupní neuron k s minimální vzdáleností d_k jako ,,vítěze"
5. Aktualizuj váhy (...)
6. Přejdi ke kroku 2

Kohonenovy mapy

Algoritmus

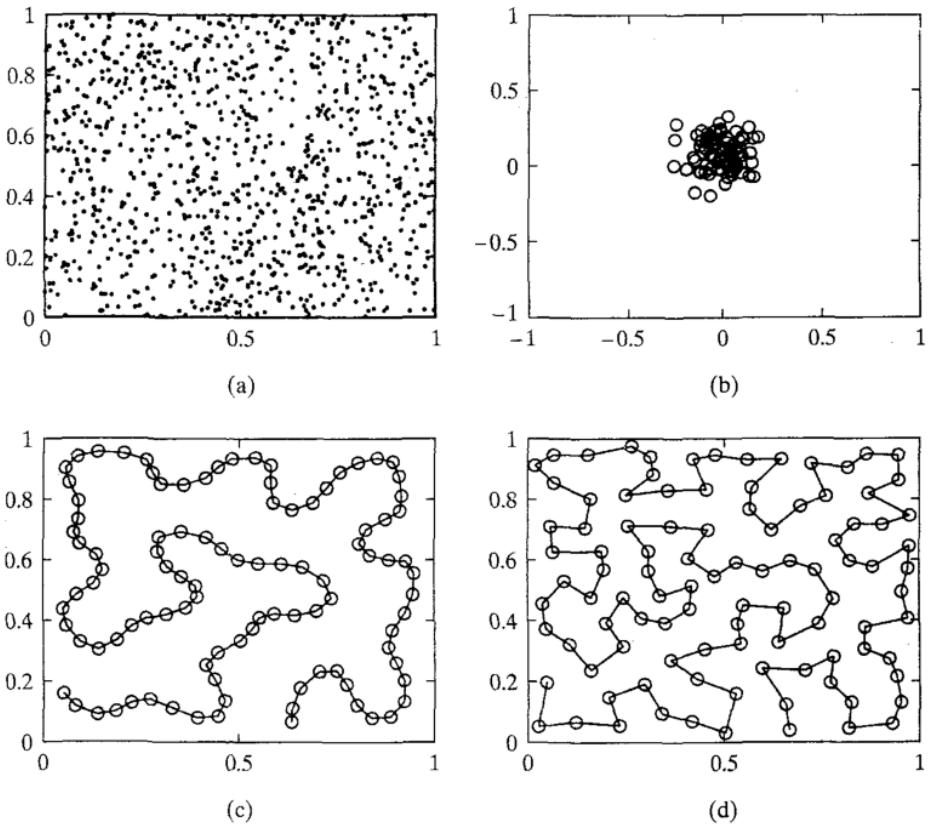
1. Inicializace
2. Předlož nový trénovací vzor $\vec{x}$
3. Spočítej vzdálenosti d_i
4. Vyber výstupní neuron k s minimální vzdáleností d_k jako „vítěze"
5. Aktualizuj váhy všech neuronů i (popř. jen neuronů z okolí k) podle:

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

6. Přejdi ke kroku 2

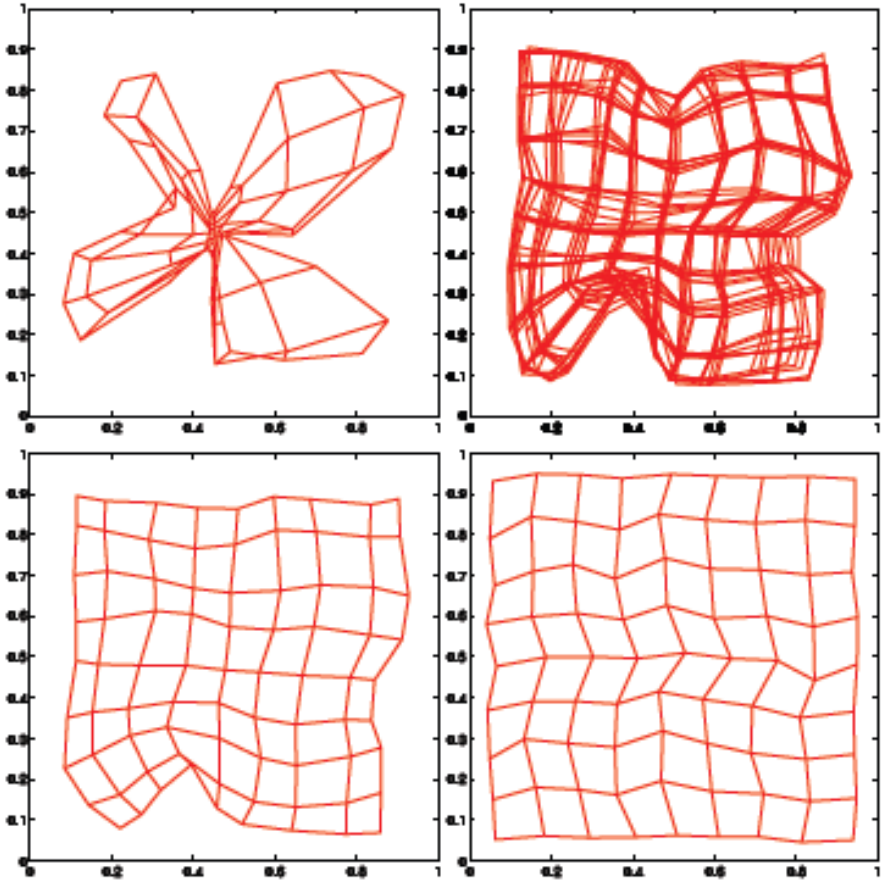
Kohonenovy mapy

Příklad – rovnoměrně rozdělená data a řetízek



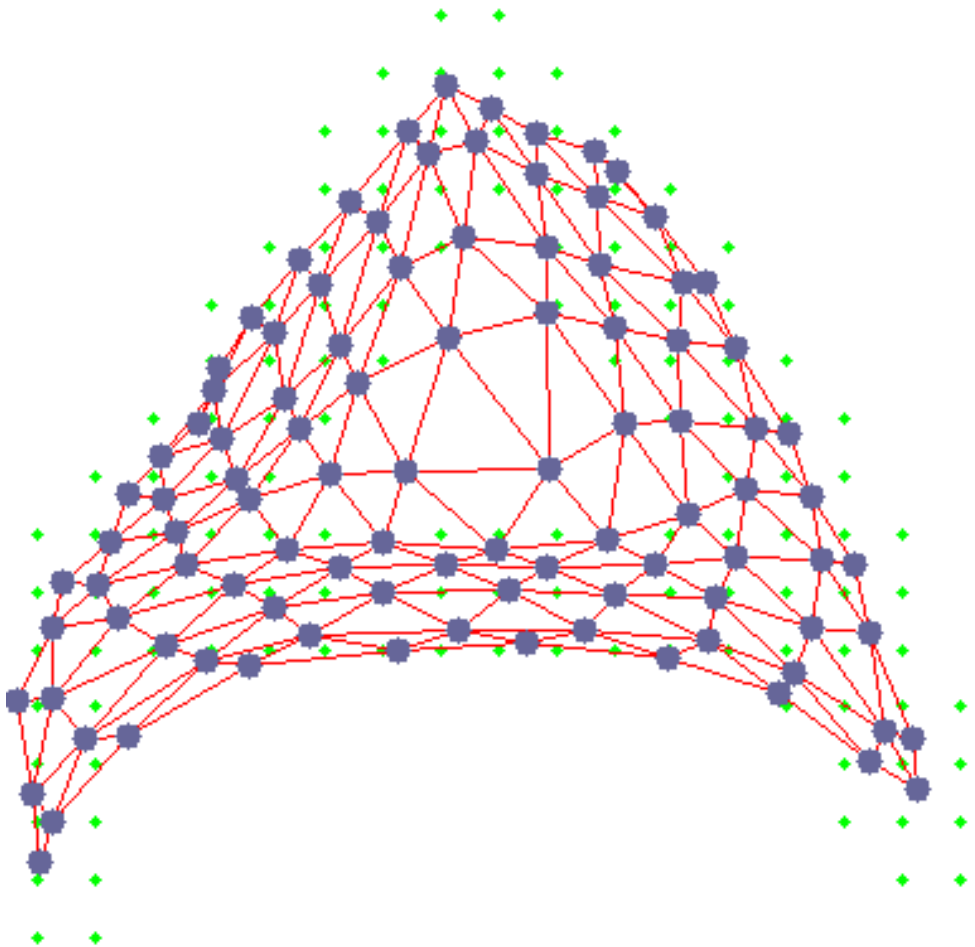
Kohonenovy mapy

Příklad – rovnoměrně rozdělená data a mřížka



Kohonenovy mapy

Příklad – nerovnoměrně rozdělená data a mřížka



Kohonenovy mapy – analýza algoritmu učení

Otázka – jak dobře se Kohonenova síť naučila?

- konvergence algoritmu?, její doba
- do jaké míry je zachována topologie zobrazení
- optimalita zobrazení
- vliv parametrů α, σ

Kohonenovy mapy

Energetická funkce Kohonenovy mapy

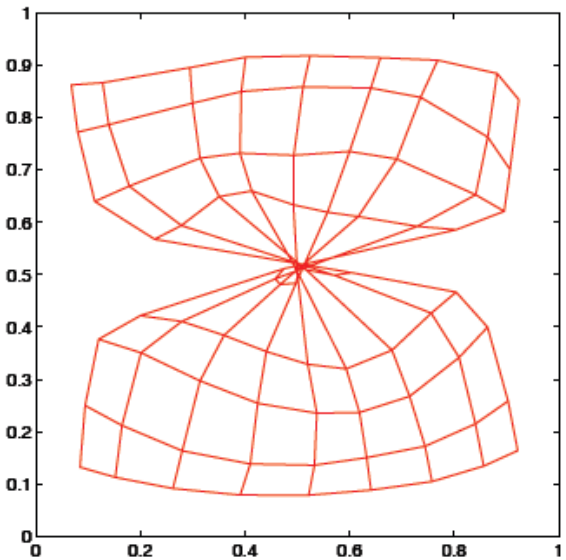
$$E(\vec{w}_i) = \frac{1}{2} \sum_i \sum_{\vec{x}} \Lambda(i, k_{\vec{x}}) |\vec{x} - \vec{w}_i|^2$$

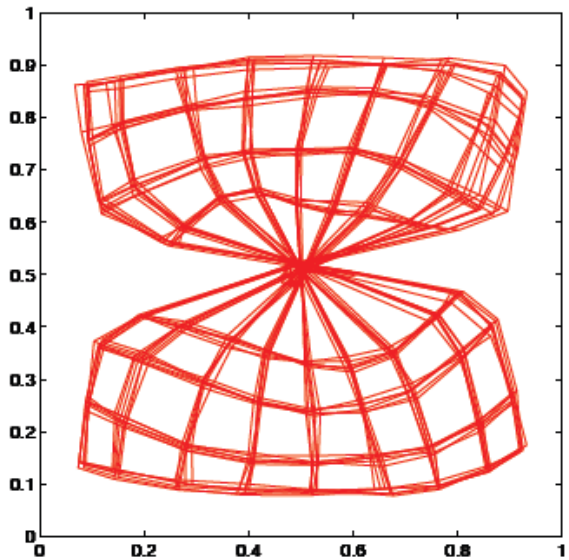
- Kohonenův algoritmus učení je (za určitých předpokladů – vhodná volba α, σ) gradientní metoda pro E

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

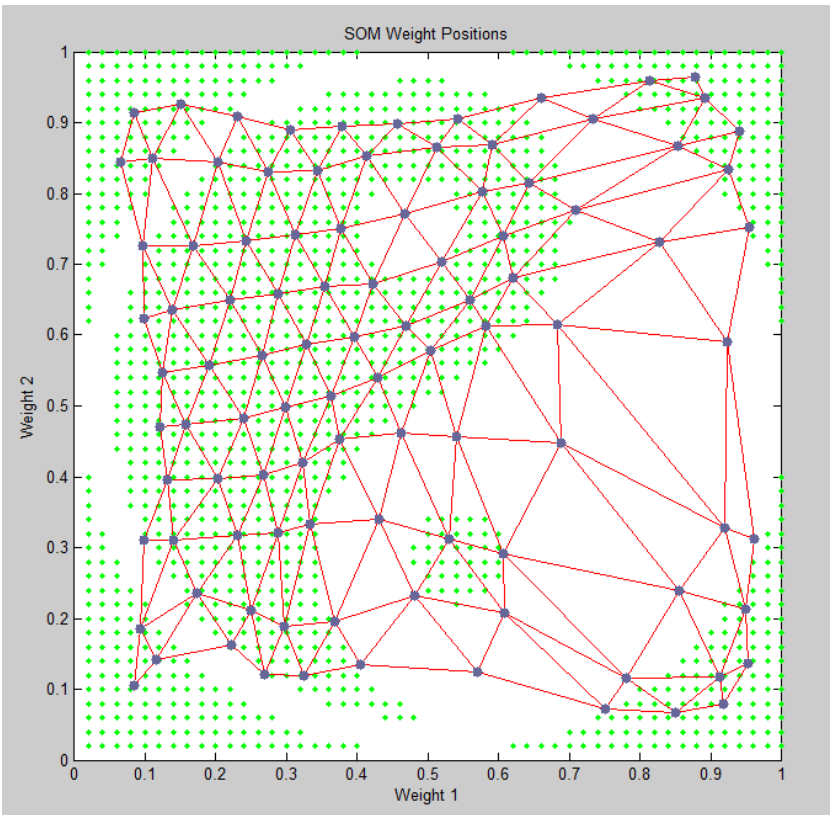
- **Organizace:**
  - široká okolí (zpočátku celá síť), pozvolna se zužují
  - α je relativně velká (blízko 1), téměř neměnná
- **Ustálení:**
  - malá okolí (v závěru jen jeden neuron),
  - α rychle klesá k nule

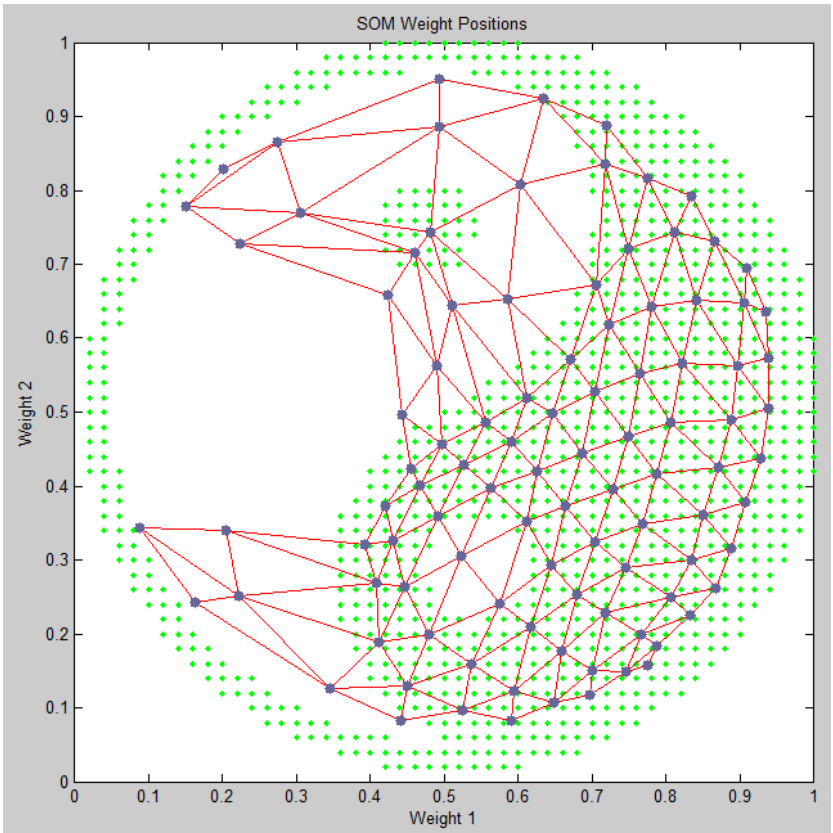
Kohonenovy mapy - Jak je to v Matlabu

- *newsom* ... vytvoření kohonenovy mapy
  - net = newsom(R,[D1,D2,...],TPLG,DFCN,ST,IN)
  - R ... rozsahy hodnot vstupních vzorů
  - [D1,D2,...] ... rozměry sítě, implicitně [5 8]
  - TPLG ... topologie, implicitně *'hextop'*
  - DFCN ... funkce vzdálenosti, implicitně *'linkdist'*
  - ST ... počet kroků učení, než se nastaví nulové okolí, implicitně 100
  - IN ... počáteční velikost okolí, implicitně 3
- Možné topologie
  - *'hextop'*, *'gridtop'*, *'randtop'*
- Funkce vzdálenosti
  - *'dist'* (Euklidova), *'linkdist'*, *'boxdist'* (čtverec), *'manlist'* (Manhattan)

Kohonenovy mapy -příklad

- cv10_som_bitmapa.m





Učení bez učitele – kombinace učení s učitelem a učení bez učitele

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

LVQ (učení vektorové kvantizace)

- LVQ = Learning Vector Quantization
- varianta SOM (Kohonenovy mapy)
- 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 (vektorů)
- 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 vektorů, 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 kvantizace) - varianty

LVQ1 - motivace

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

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

  - Každý neuron i zadaptuje své váhy podle pravidla:

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

- **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:

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

- Každý neuron $i \neq k$ své váhy nemění:

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

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

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:**

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

  - d_k ... vzdálenost $\vec{w}_k$ a $\vec{x}$
  - $0.2 < w < 0.3$ ... relativní šířka okénka

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 nejbliže 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

- Optimální umístění váhových vektorů
- Aproximace rozložení tříd a stabilizace řešení
- LVQ3 - adaptační pravidlo:
  - k a l jsou nejbliže 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 nejbliže 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: $0.1 < \epsilon < 0.5$, $0.2 < w < 0.3$

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

Doporčované použití

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

LVQ - Jak je to v Matlabu

- *newlvq* ... vytvoření modelu
  - net = newlvq(R,S,PC,LR,LF)
  - R ... rozsahy hodnot vstupních vzorů
  - S ... počet neuronů
  - PC ... procentuální zastoupení tříd
  - LR ... parametr učení, implicitně 0.01
  - LF ... učící funkce, defaultně *'learnlv1'*, další je *'learnlv2'*
- *train* ... učení
  - Tc = ind2vec(T).
  - net = train(net,P,Tc).
- *sim* ... rozpoznávání
  - Yc = sim(net,P).
  - Y = vec2ind(Yc).