Učení bez učitele

Učení bez učitele (unsupervised learning)

- samoorganizace, shlukování
- trénovací množina T tvaru $T = \{x^1, ..., x^N\}$
- $x^i \in \Re^n$ je (i-tý) trénovací vstupní vzor, požadovaný výstup neznáme
- **Myšlenka:** 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

Učení bez učitele (unsupervised learning)

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

Kompetitivní modely

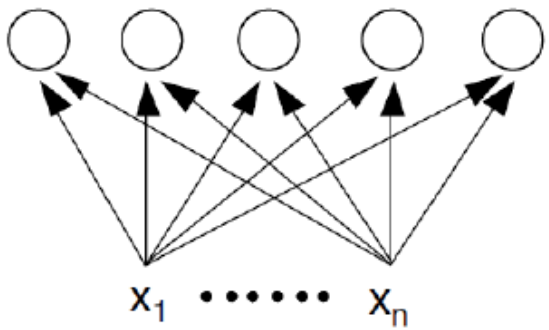
Kompetiční učení – motivace

- **Kompetice:** boj o ,,právo reprezentovat předložený vzor”
- **Inhibice:** ,,potlačování (aktivity) soupeřů”
- Pravidlo ,,vítěz bere vše” *(Winner takes all)*

Kompetitivní modely

Architektura

- dvě vrstvy neuronů
- neurony ve vstupní vrstvě odpovídají 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

Kompetiční 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. Vítězný neuron bude nejaktivnější a bude potlačovat **(inhibovat)** aktivitu ostatních neuronů
   $\rightarrow$ pomocí ,,laterálních" spojů ... **laterální inhibice**

- **Výsledek:** jeden neuron neuron je excitován, ostatní inhibovány

Kompetiční učení

- Pro rozhodnutí, zda bude neuron aktivní nebo ne, je nutná globální informace o stavu všech neuronů v síti

Jak implementovat laterální inhibici ve výstupní vrstvě?

1. Iterativním výpočtem (...)
2. Prostým výběrem neuronu s max. odezvou ... winner takes all

- Aktivita neuronu signalizuje příslušnost předloženého vstupu ke shluku vektorů reprezentovaných tímto neuronem
  $\rightarrow$ síť funguje i jako klasifikátor

Kompetiční učení

Adaptace ... diferenční pravidlo

- Vítězný neuron i zadaptuje své váhy směrem k předloženému vzoru $\vec{x}$:

$$\Delta\vec{w}_i = \alpha(\vec{x} - \vec{w}_i)$$

Cíl učení

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

Kompetiční učení

Adaptace ... diferenční pravidlo

- Vítězný neuron i zadaptuje své váhy směrem k předloženému vzoru $\vec{x}$:

$$\Delta \vec{w}_i = \alpha(\vec{x} - \vec{w}_i)$$

Plasticita sítě ... α

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

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

Kompetiční učení

Alternativa k Euklidovské vzdálenosti – skalární součin $\vec{w}.\vec{x}$

- pro normované vektory odpovídá kosinu úhlu mezi těmito dvěma vektory
- na rozdíl od Euklidovské vzdálenosti se maximalizuje

Alternativní adaptační pravidlo

- Vítězný neuron i zadaptuje své váhy podle:

$$\Delta \vec{w}_i = \alpha \vec{x}$$

- Dávková aktualizace $\rightarrow$ stabilnější proces učení

Kompetiční učení – formální algoritmus

Vstup

- Množina $X = \{\vec{x}_1, ..., \vec{x}_N\}$ normovaných vstupních vektorů v n-rozměrném prostoru, které chceme klasifikovat do k shluků
- Síť s k neurony (s nulovým prahem) ve výstupní vrtsvě.

Inicializace

- Náhodně generuj a normuj váhové vektory $\vec{w}_1, .., \vec{w}_k$

Opakuj – Test:

- Náhodně vyber $\vec{x} \in X$
- Spočítej $\vec{w}_i.\vec{x}$ pro $i = 1, ..., k$
- Vyber $\vec{w}_m$ tž. $\vec{w}_m.\vec{x} \geq \vec{w}_i.\vec{x}$ pro všechna $i = 1, ..., k$
- **Aktualizace**
  - Nahraď vektor $\vec{w}_m$ normovaným vektorem $\vec{w}_m + \alpha\vec{x}$

Kompetiční učení

Výhody:

- Při učení (adaptaci) není třeba učitel (samoorganizace)
- Počet shluků je volitelný
- snadné zjištění reprezentanta shluku i ... $\vec{w}_i$

Kompetiční učení

Problémy:

- Nutnost volit rychlost klesání α
- Nutnost vhodně inicializovat váhy ... ovlivňuje rychlost učení
  - Např. podle náhodně vybraných vzorů
- Mrtvé (nevyužité) neurony
  - Normalizace vektorů
  - Mřížka v Kohonenově vrstvě
  - Topologické okolí neuronu
  - Řízená kompetice a mechanismus svědomí

Jak je to v Matlabu

- *newc* ... vytvoření kompetiční sítě
  - net = newc(Rozsah_hodnot,Pocet_neuronu)
  - net.IW{1,1} ... matice vah
  - net.b{1,1} ... prahy neuronů
  - net.trainFcn ... učící funkce ... *trainr*
  - net.trainParam ... parametry učící funkce
- *train* ... učení
  - net = train(net,T).
- *sim* ... rozpoznávání
  - Y = sim(net,T).

Odbočka: Klastrovací metody pro empirická vícerozměrná data

Dva základní přístupy:

- algoritmus k nejbližších sousedů
- algoritmus k středů

K nejbližších sousedů

- 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

Odbočka: Klastrovací metody pro empirická vícerozměrná data

Algoritmus k středů (k-means clustering)

- Učení bez učitele
- Vstupní vektory jsou klasifikovány do k různých shluků, každý shluk i je reprezentován svým centroidem $\vec{c}_i$
- Na začátku učení obsahuje každý shluk právě 1 vektor
- Nový vektor $\vec{x}$ je zařazen k tomu shluku i, jehož centroid $\vec{c}_i$ je nejblíže tomuto vzoru

Odbočka: Klastrovací metody pro empirická vícerozměrná data

Algoritmus k středů (k-means clustering) - procedura:

- Nový vektor $\vec{x}$ je zařazen k tomu shluku i, jehož centroid $\vec{c}_i$ je nejblíže tomuto vzoru
- Centroid $\vec{c}_i$ je pak aktualizován pomocí:

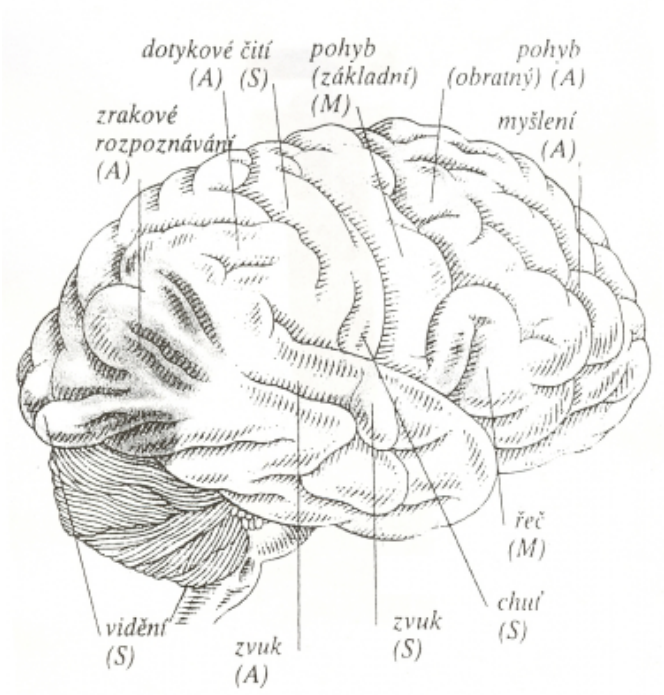
$$\vec{c}_i(new) = \vec{c}_i(old) + 1/n_i(\vec{x} - \vec{c}_i(old))$$

  - n_i ... počet vektorů již přiřazených ke shluku i
- Tato procedura se iterativně opakuje pro celou množinu dat (jejich strukturu pak vystihují centroidy $\vec{c}_i, i = 1, ..., k$)
  $\rightarrow$ **Vektorová kvantizace**

Kohonenovy mapy

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

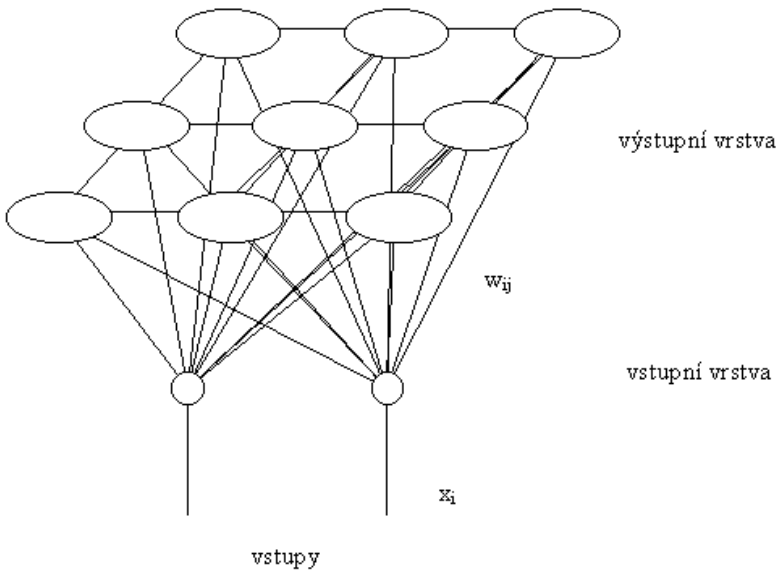
- původní aplikace: fonetický psací stroj (finština: řeč → písmo)



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



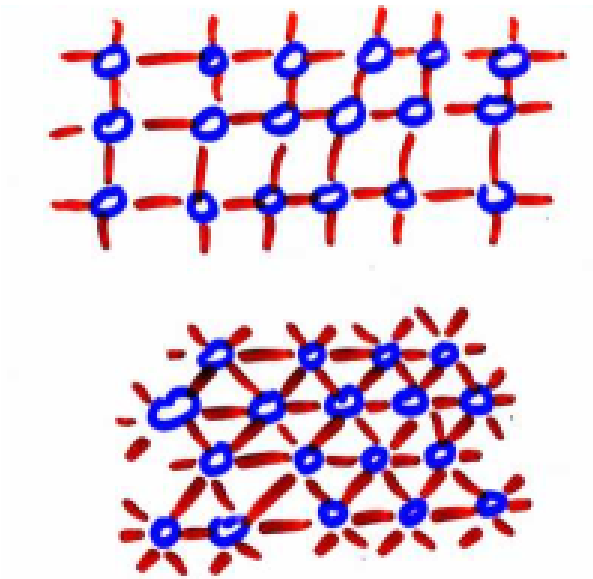
- Učení bez učitele
- Rozpoznávání
- Redukce dimenzionality dat
- Vizualizace dat
- Ekonomické aplikace

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

Různé topologie mřížky
(pro dvě dimenze)

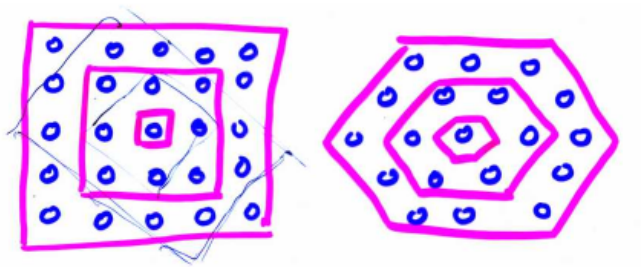


Kohonenovy mapy – učení

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

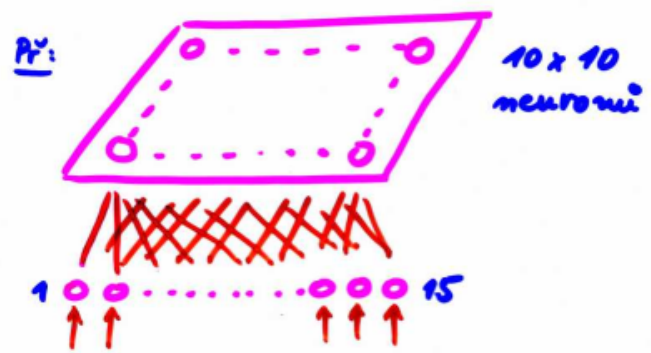
Dvě možné interpretace z pohledu aplikací

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



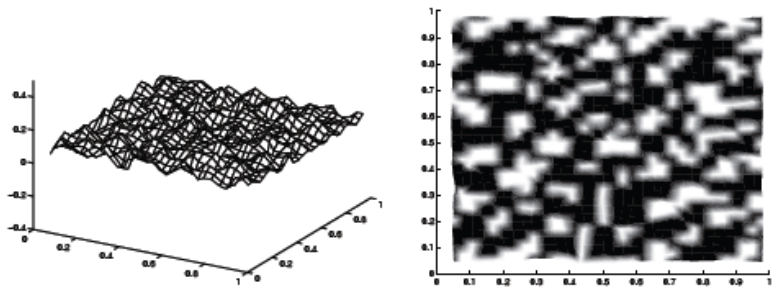
Kohonenovy mapy

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,...)

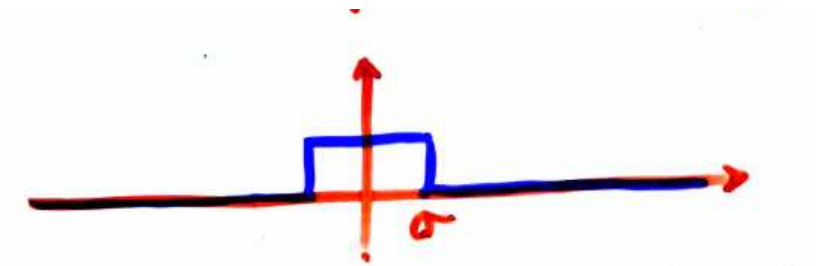
Kohonenovy mapy

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

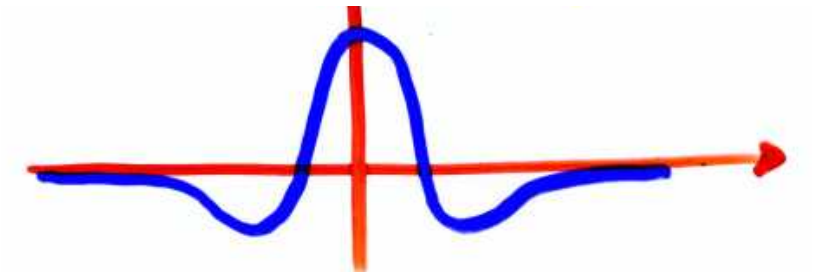
- **Diskrétní okolí**
  - $\Lambda(i, k) = 1$ pro všechny i z okolí k s poloměrem jedna (obecně nejvýše σ), $\Lambda(i, k) = 0$ pro ostatní
  - efektivní z hlediska implementace, minimální režie (stačí adaptovat neurony z okolí)



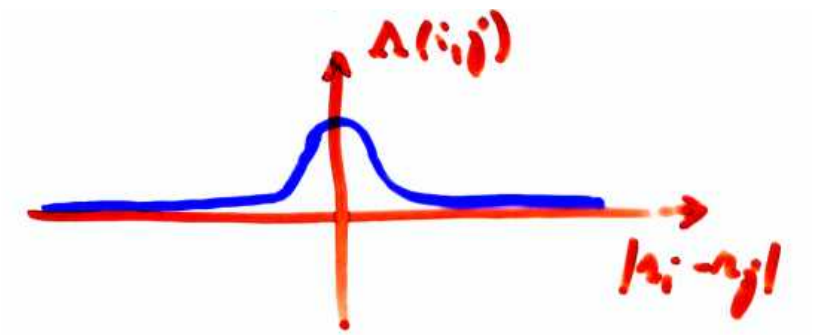
Kohonenovy mapy

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:

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

Vigilanční (bdělostní) koeficient ... $\alpha \in < 0, 1 >$

- Pro pevné α síť obvykle nekonverguje ... $\alpha \to 0$

Kohonenovy mapy

Algoritmus

1. Zvol hodnoty vah mezi vstupními a výstupními neurony jako malé náhodné hodnoty. Zvol počáteční 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

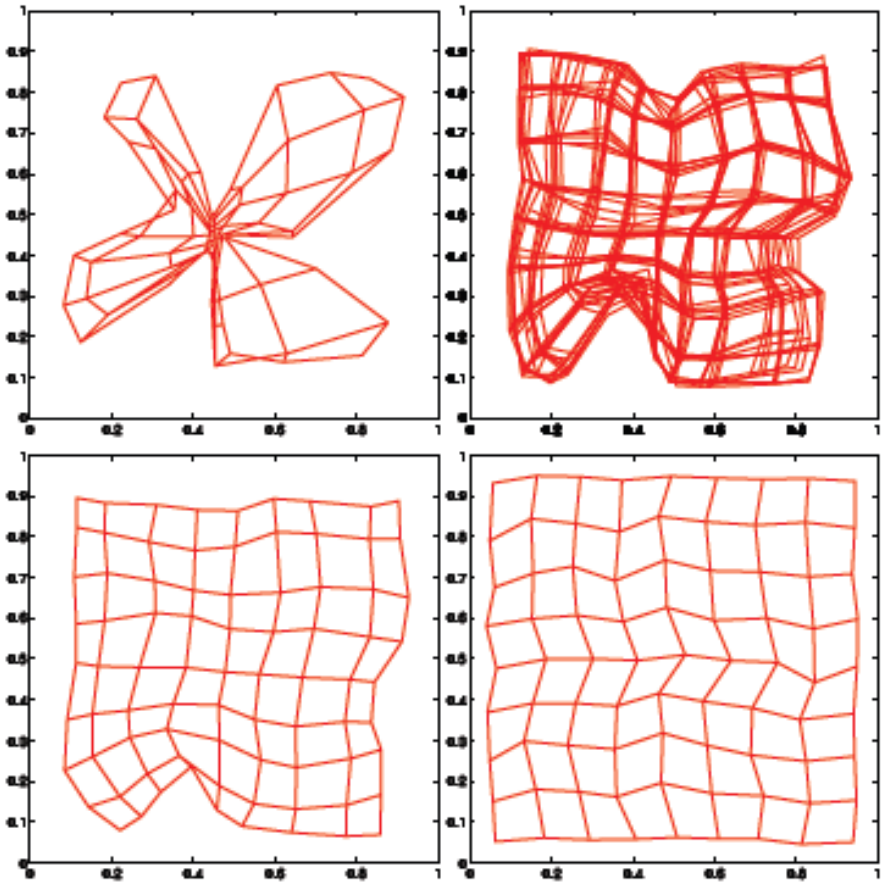
1. Inicializace
2. Předlož nový trénovací vzor $\vec{x}$
3. Spočítej vzdálenosti d_i
4. Vyber výstupní neuron k s minimální vzdáleností d_k jako ,,vítěze"
5. Aktualizuj váhy všech neuronů i (popř. jen neuronů z okolí k) podle:

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

6. Přejdi ke kroku 2

Kohonenovy mapy

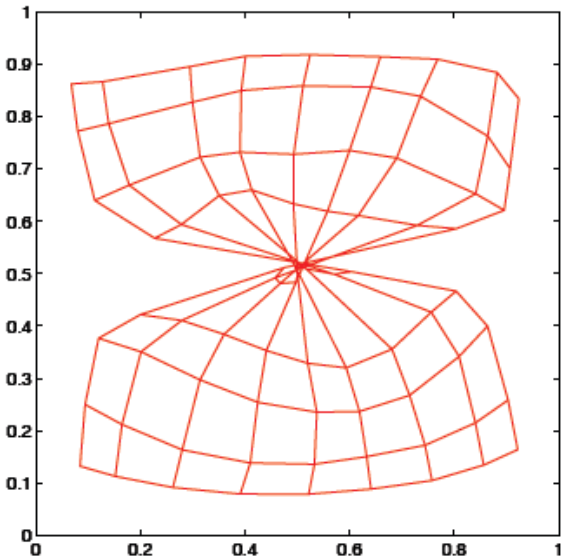
Příklad

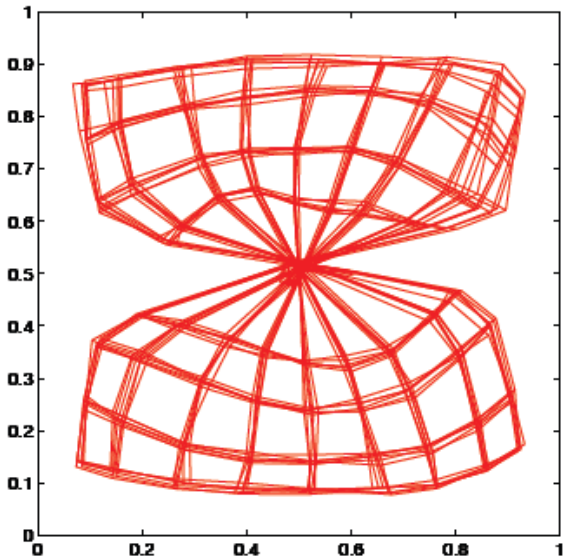


Kohonenovy mapy

Zásadní vliv má volba parametrů α, σ

- 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 1
- Možné topologie
  - *'hextop'*, *'gridtop'*, *'randtop'*
- Funkce vzdálenosti
  - *'dist'* (Euklidova), *'linkdist'*, *'boxdist'* (čtverec), *'manlist'* (Manhattan)

Kohonenovy mapy -příklad

- cv10_som_bitmapa.m