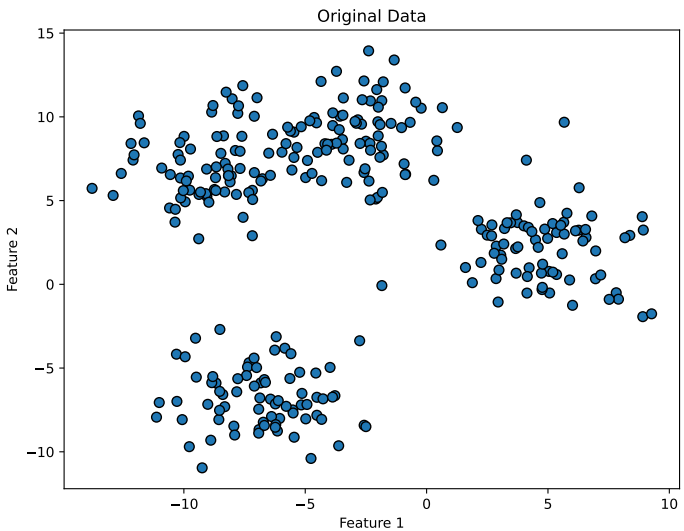
Učení bez učitele, unsupervised learning, samoorganizace

- 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

Učení bez učitele, unsupervised learning, samoorganizace

Shluk (klastr)

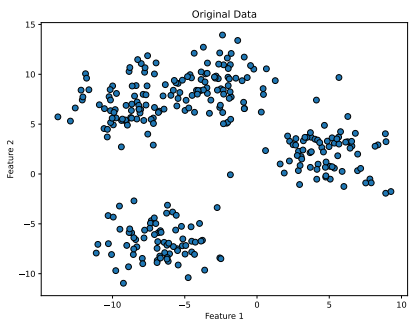
- Skupina vzorů s malým rozptylem a velkou vzdáleností od ostatních shluků

Shlukování (klastrování)

- Disjunktní rozdělení dat na shluky

Problémy:

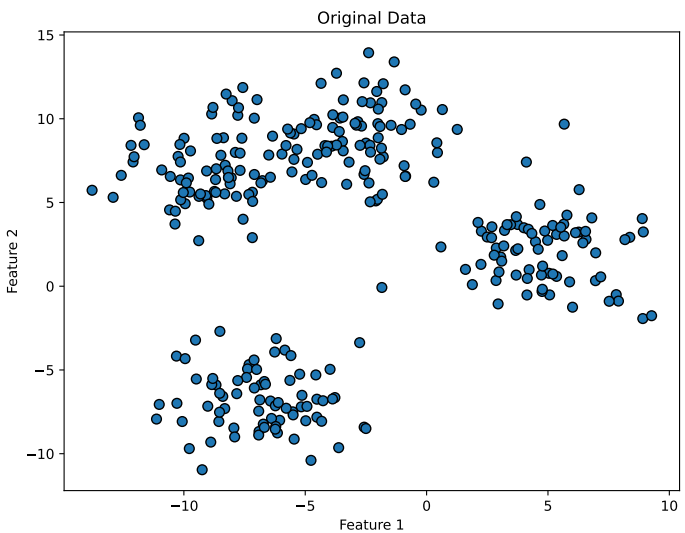
- Jak určit počet a rozložení shluků v příznakovém prostoru?
- Jak vybrat reprezentanta/y shluku?
  - Vhodně vybrané / všechny trénovací vzory patřící do shluku
  - Střed shluku (*těžiště, centroid*)

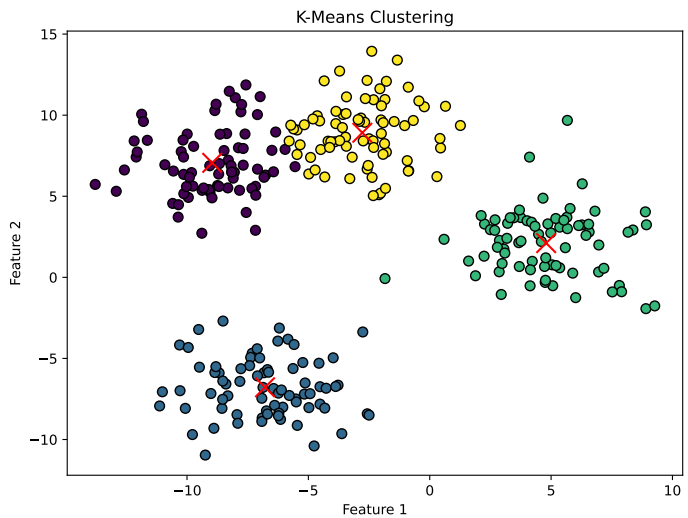


Odbočka: Jednoduché metody strojového učení pro klasifikaci a shlukování

Dva algoritmy na k:

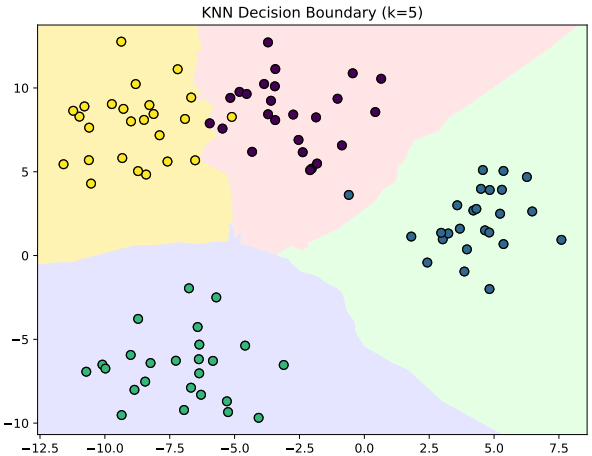
- algoritmus k nejbližších sousedů
- algoritmus k středů





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

Odbočka: Jednoduché metody strojového učení pro klasifikaci a shlukování

Jak počítat vzdálenost?

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

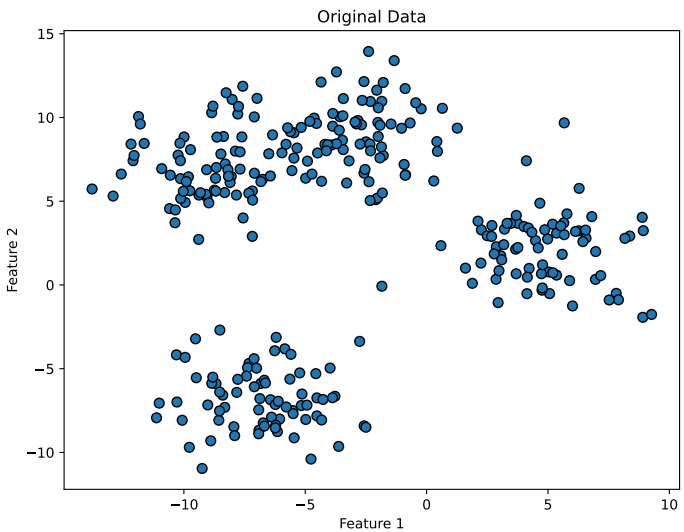
Ale existují i alternativní metriky, např.:

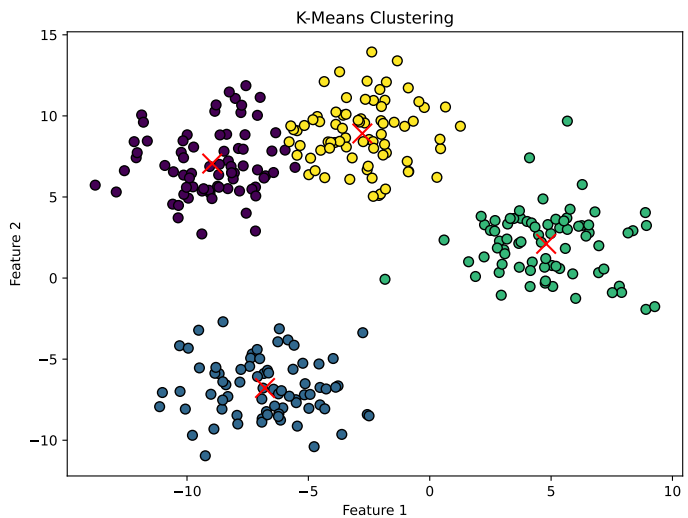
- Manhattan (městská) metrika: $d(\vec{p}, \vec{q}) = \sum_{i=1}^{n} |p_i - q_i|$
- Čebyševova metrika: $d(\vec{p}, \vec{q}) = \max_i |p_i - q_i|$
- Minkewského metrika: $d(\vec{p}, \vec{q}) = (\sum_{i=1}^{n} |p_i - q_i|^r)^{\frac{1}{r}}$
- Kosínová podobnost: $\cos(\vec{p}, \vec{q}) = \frac{\vec{p}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

...

Algoritmus k středů (k-means clustering)

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





Odbočka: Jednoduché metody strojového učení pro klasifikaci a shlukování

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**

Odbočka: Jednoduché metody strojového učení pro klasifikaci a shlukování

Vektorová kvantizace:

- snažíme se co nejlépe pokrýt vstupní prostor pomocí reprezentantů a respektovat přitom statistické rozdělení vzorů (je to blízké odhadování hustoty dat ve statistice)
- reprezentace množiny vektorů pomocí menší podmnožiny jejich reprezentantů
- ztrátová komprese dat

Učení bez učitele, unsupervised learning, samoorganizace

- Modely mělkých umělých neuronových sítí učené bez učitele
  - Kompetitivní modely
  - Kohonenovy mapy (1989)
- Modely hlubokých sítí (mimo rámec tohoto předmětu)
  - Generative Adversarial Networks ,GANs
  - Autoencodery
- Hybridní mělké 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

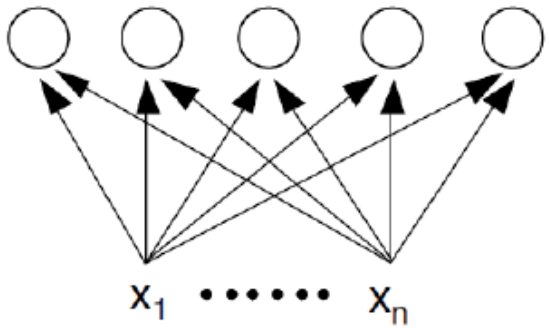
Kompetiční učení – motivace:

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

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



Princip kompetičního učení

1. Předložím trénovací 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
  $\rightarrow$ vítězný neuron se snaží co nejvíce přiblížit k danému trénovacímu vzoru

(obrázek)

Kompetiční učení

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

Možná interpretace:

- Pro předložený vzor je vždy jeden neuron excitován, ostatní inhibovány
  $\rightarrow$ odpovídá to výběru jednoho z k neuronů
- Aktivita neuronu signalizuje příslušnost předloženého vstupu ke shluku vektorů reprezentovaných tímto neuronem
  $\rightarrow$ síť funguje i jako klasifikátor

Kompetiční učení

Jak implementovat laterální inhibici ve výstupní vrstvě?

1. Iterativním výpočtem:

$$y_i = f\left(\sum_{l=1}^{k} t_{li} y_l\right)$$

   - obvykle: f je sigmoidální funkce, $t_{ii} = (k-1)$ a $t_{ij} = -1$ pro $i \neq j$
   - 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

Kompetiční učení

Adaptace ... diferenční pravidlo:

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

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

Cíl učení

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

(obrázek)

Kompetiční učení

Adaptace ... diferenční pravidlo

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

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

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 ... $\alpha \to 0$

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
Formální algoritmus pro variantu „vítěz bere vše”:

- Cíl učení: postupně váhy jednotlivých neuronů natavit tak, aby odpovídaly středům shluků v datech
- Během učení se pro každý neuron minimalizuje energetická funkce

$$E = \sum_{j_i=1}^{n_i} ||\vec{x_{j_i}} - \vec{w_i}||^2$$

  n_i ... počet vektorů přiřazených k i-tému neuronu
  j_i ... indexuje vektory přiřazené k i-tému neuronu
- dávková varianta předchozího algoritmu: stabilnější
- rozmyslete si souvislost s algoritmem k středů

Kompetiční učení - kosínová podobnost

Kosínová podobnost (cosine similarity)

- vzdálenost jako skalární součin: $d(\vec{w}, \vec{x}) = \vec{w} \cdot \vec{x}$
- alternativa k Euklidovské vzdálenosti
- pro normované vektory odpovídá kosinu úhlu mezi těmito dvěma vektory $\cos(\vec{x}, \vec{w}) = \frac{\vec{x} \cdot \vec{w}}{\|\vec{x}\| \|\vec{w}\|}$
- na rozdíl od Euklidovské vzdálenosti se maximalizuje

Alternativní adaptační pravidlo

- Vítězný neuron m zadaptuje své váhy podle:

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

- jedná se o variantu Hebbova učení
- Dávková (batch) aktualizace $\rightarrow$ stabilnější proces učení

(obrázek)

Kompetiční učení - kosínová podobnost
Formální algoritmus

Vstup

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

Inicializace

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

Opakuj – Test:

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

Kompetiční učení

Výhody:

- Učení bez učitele (samoorganizace)
- Počet shluků je volitelný
- Snadné zjištění reprezentanta shluku i ... $\vec{w}_i$

Aplikace:

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

Kompetiční učení a PCA analýza

PCA (Principal component analysis)

- redukce dimenzionality vstupních dat
  - použít méně příznaků bez ztráty podstatné informace
  - výběr nejdůležitějších příznaků
- Mějme množinu vektorů $T = \{\vec{x}_1, ..., \vec{x}_N\}$
- Nejdůležitějším příznakem (tj. 1. PC) pro tuto množinu je vektor $\vec{w}$, který maximalizuje výraz $\frac{1}{N} \sum_{i=1}^{N} ||\vec{w} \cdot \vec{x}_i||^2$ (pro kosínovou podobnost)

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)

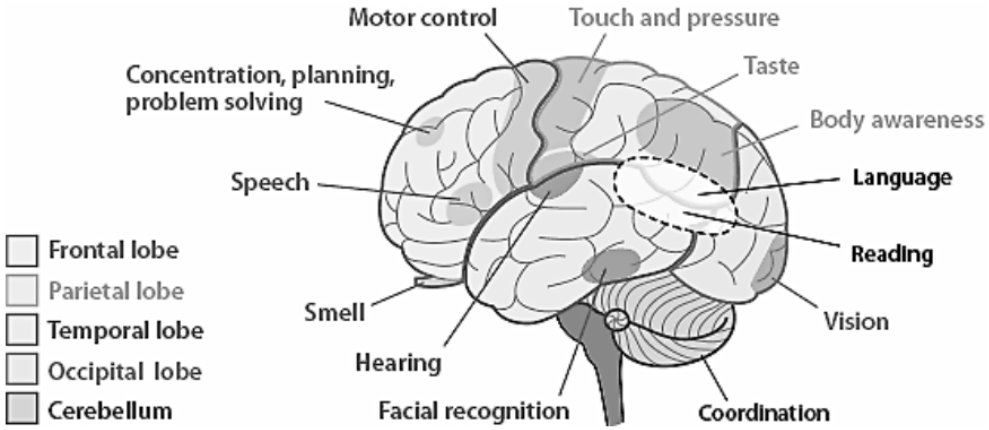
Odbočka: jak je to v Matlabu

- *competlayer* ... vytvoření kompetiční sítě
  - net = competlayer(počet_neuronů,parametr_učení);
  - net = configure(net,x);
  - net.IW{1,1} ... matice vah
  - net.b{1,1} ... prahy neuronů
  - net.trainFcn ... učící funkce ... *trainru*
  - net.trainParam ... parametry učící funkce
- *train* ... učení
  - net = train(net,X).
- *sim* ... rozpoznávání
  - Y = sim(net,X).
  - ac = vec2ind(Y)

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

- původní aplikace: fonetický psací stroj (finština: řeč $\rightarrow$ 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/

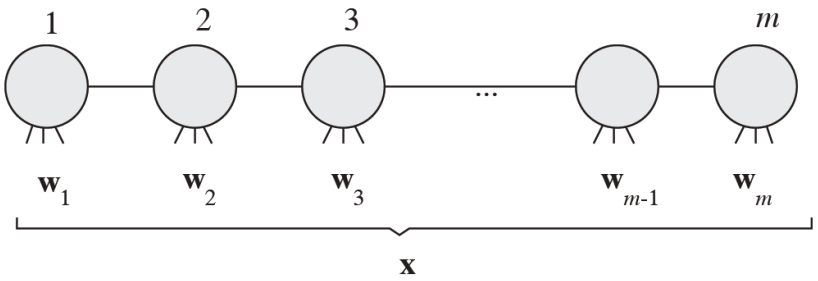
Kohonenovy mapy

Dvě možné interpretace z pohledu aplikací

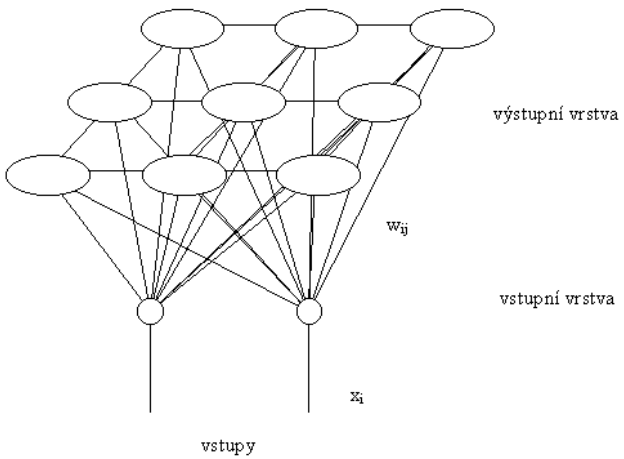
1. Snížení dimenze dat při zachování topologie, vizualizace
2. Shlukování (klastrování)

Kohonenovy mapy - architektura (topologie)

Architektura 1D - řetízek:



Architektura 2D - mřížka:

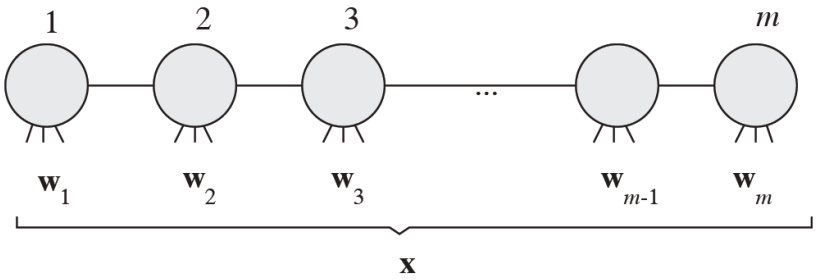


- Neurony jsou topologicky uspořádány do mřížky
- Na mřížce je definovaná sousednost fyzických neuronů
  ($\times$ logická sousednost daná blízkostí váhových vektorů)

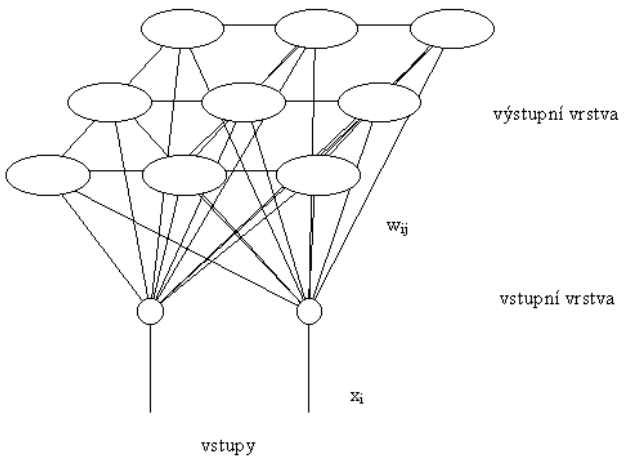
Paul Rojas: Neural Networks - A Systematic Introduction, Springer, 1996

Kohonenovy mapy - architektura (topologie)

Architektura 1D - řetízek:



Architektura 2D - mřížka:



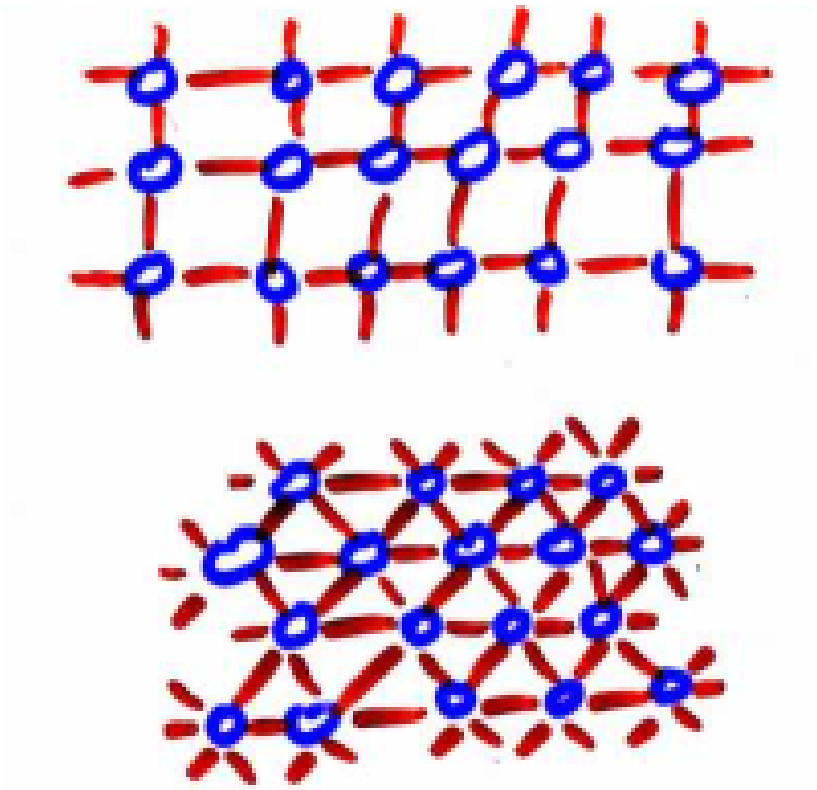
Cíl:

- mohu identifikovat nejbližší sousedy neuronu
- sousední neurony by měly reagovat na velmi podobné signály
- specializace každého neuronu na jinou část vstupního prostoru (zde excitují)

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

Kohonenovy mapy - architektura (topologie)

Různé topologie mřížky (pro dvě dimenze)

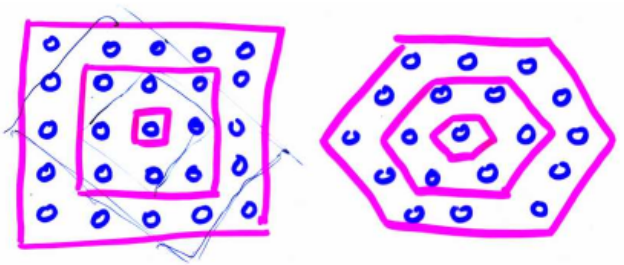


Kohonenovy mapy – učení

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

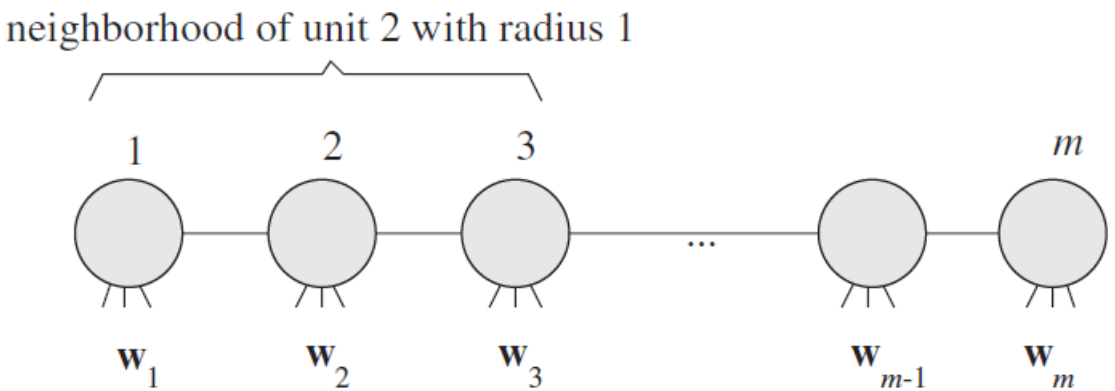
Topologické okolí neuronu



Kohonenovy mapy - 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)

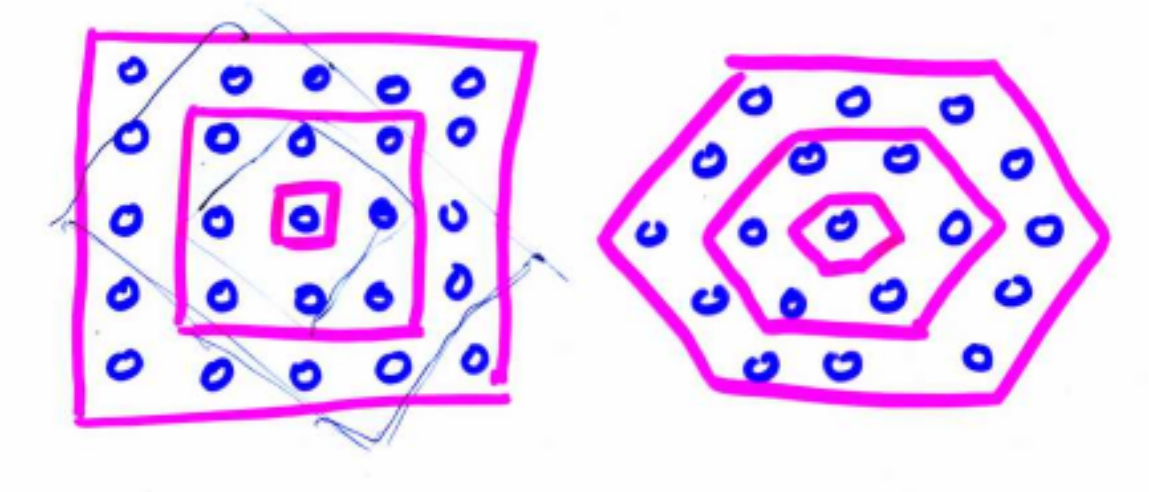


Paul Rojas: Neural Networks - A Systematic Introduction, Springer, 1996

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,...)



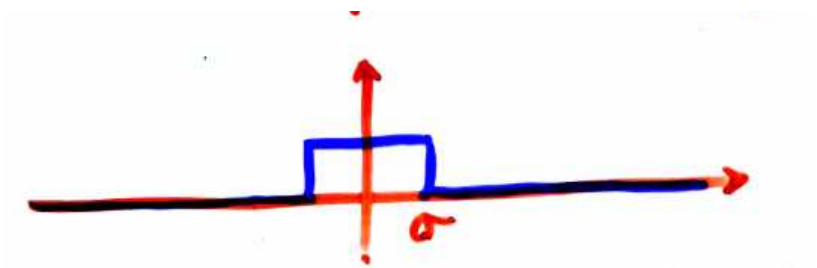
Kohonenovy mapy

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

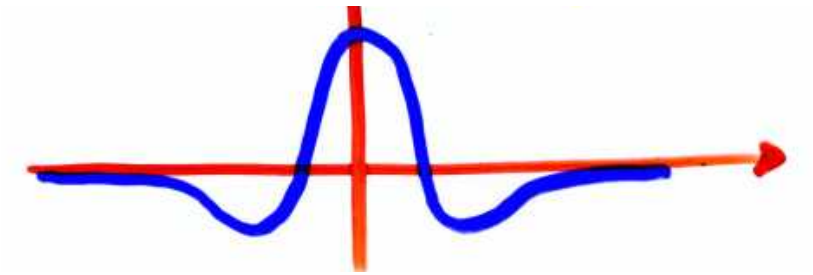
- **Diskrétní okolí**
  - $\Lambda(i, k) = 1$ pro všechny i z okolí k s poloměrem nejvýše σ, $\Lambda(i, k) = 0$ pro ostatní
  - efektivní z hlediska implementace, minimální režie (stačí adaptovat neurony z okolí)
  - σ je šířka okolí (nejjednodušeji $\sigma = 1$)



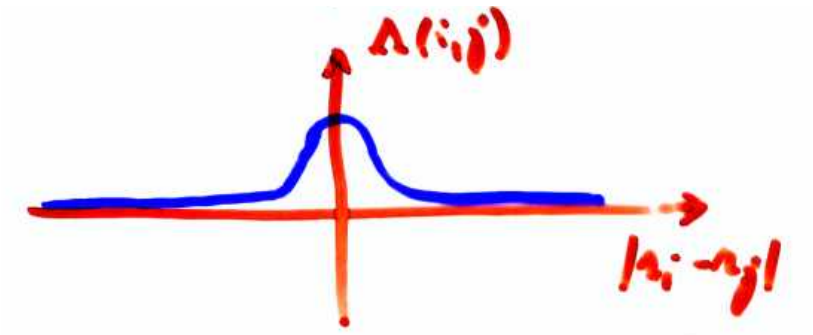
Kohonenovy mapy

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

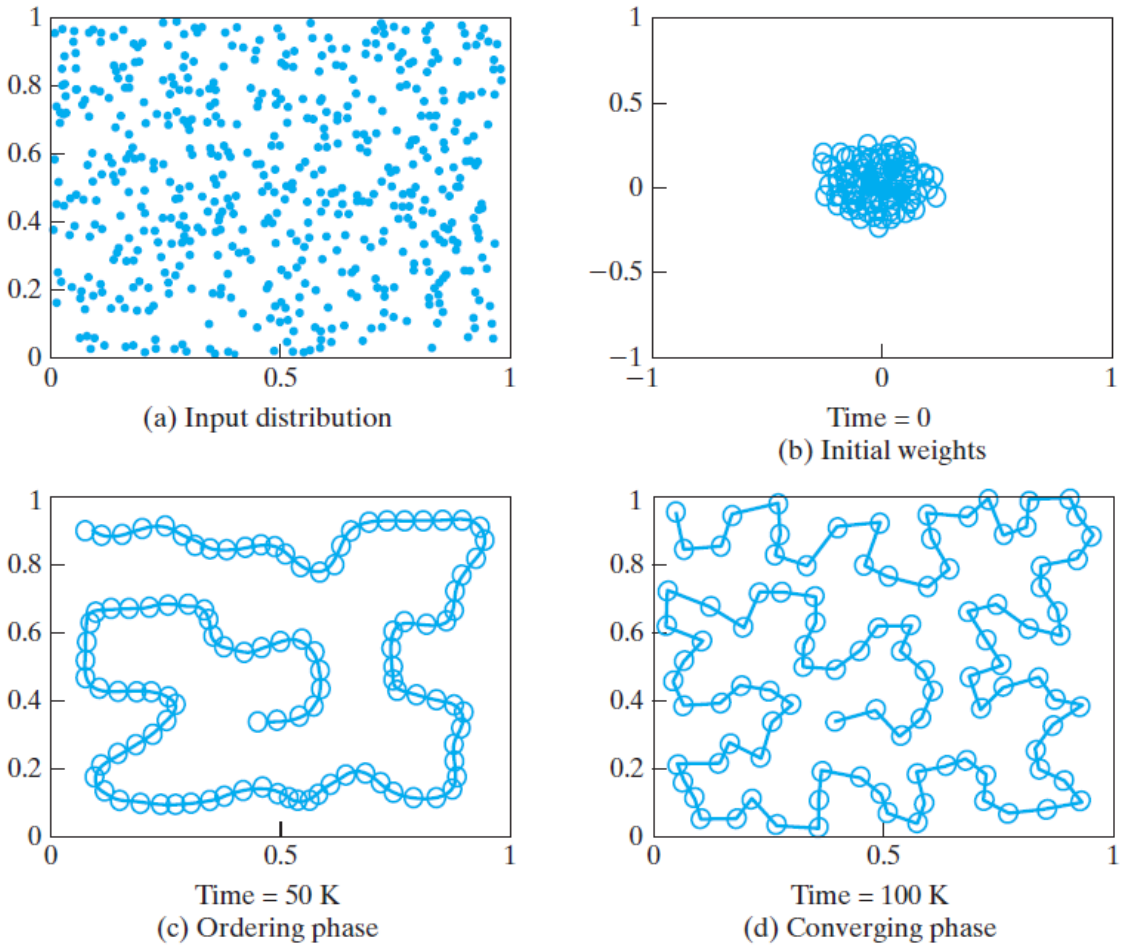
- **Parametr učení ... vigilanční (bdělostní) koeficient** ... $\alpha \in < 0, 1 >$
  - Pro pevné α síť obvykle nekonverguje ... $\alpha \to 0$
- **Šířka okolí** σ
  - obvykle $\sigma \to 0$

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

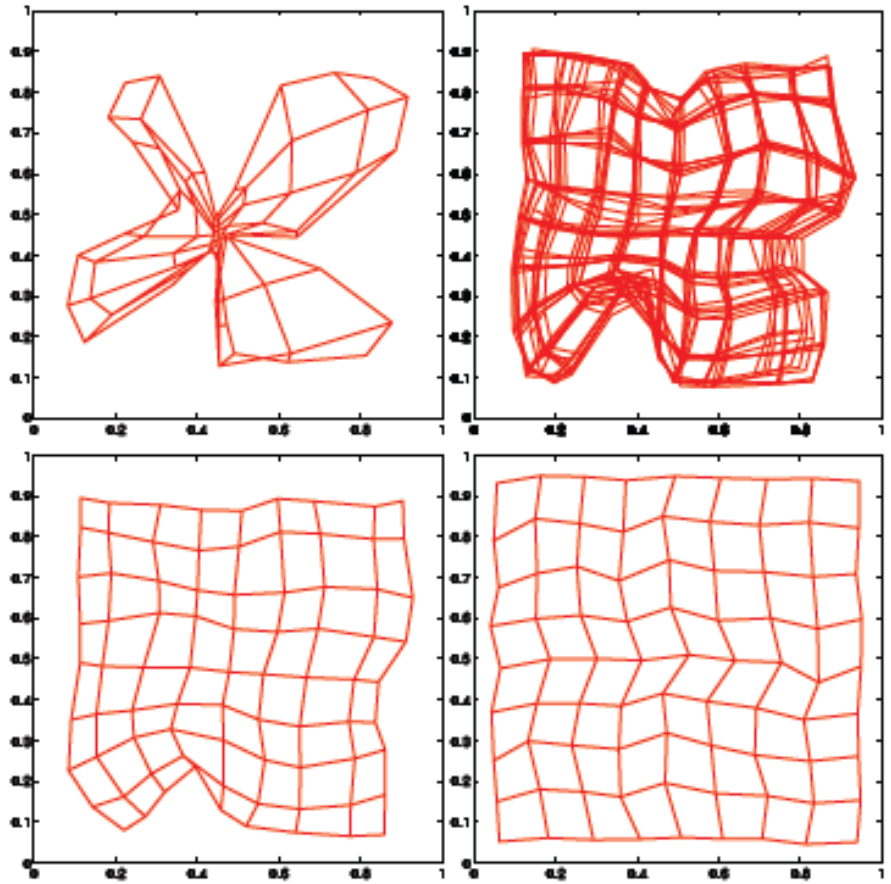
Příklad – rovnoměrně rozdělená data a řetízek



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

Kohonenovy mapy

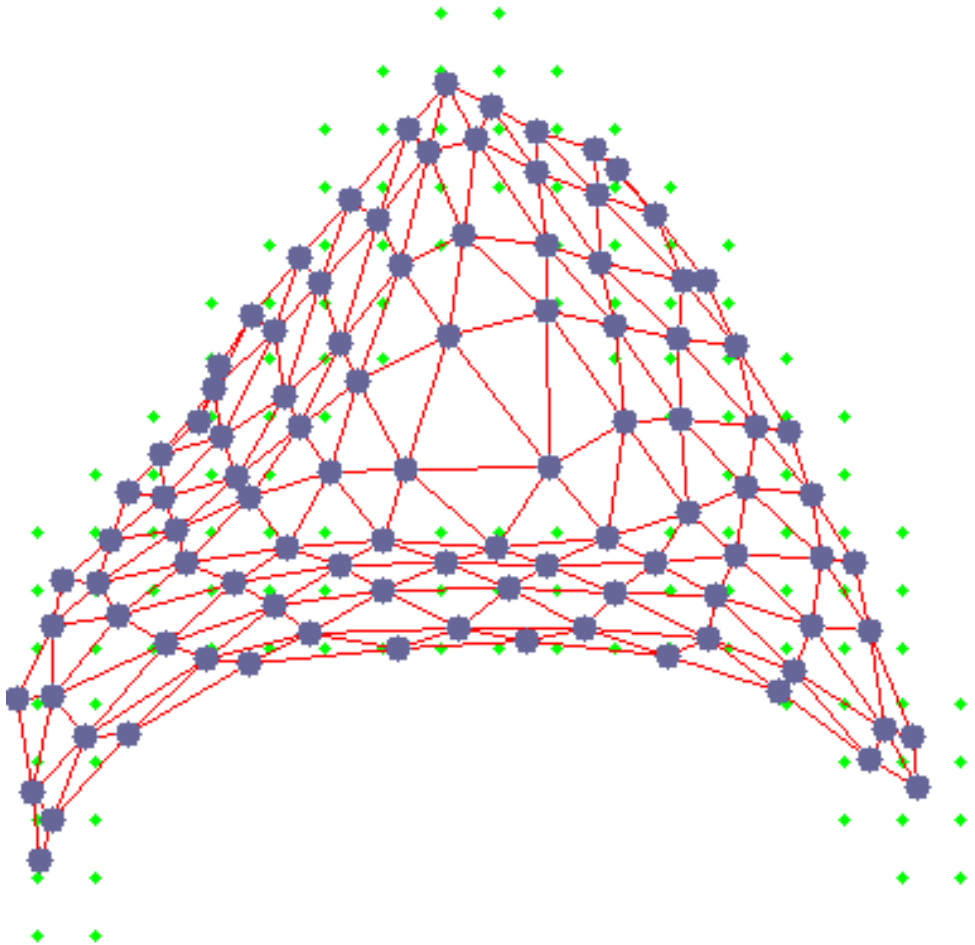
Příklad – rovnoměrně rozdělená data a mřížka



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

Kohonenovy mapy

Příklad – nerovnoměrně rozdělená data a mřížka



Kohonenovy mapy

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 – analýza algoritmu učení

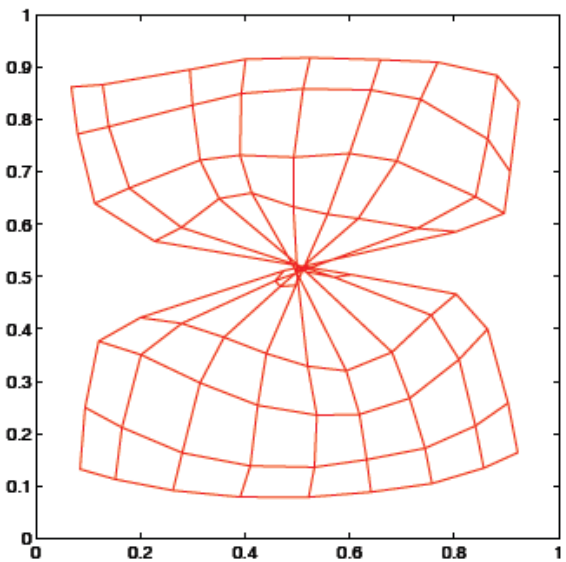
Otázka – jak dobře se Kohonenova síť naučila?

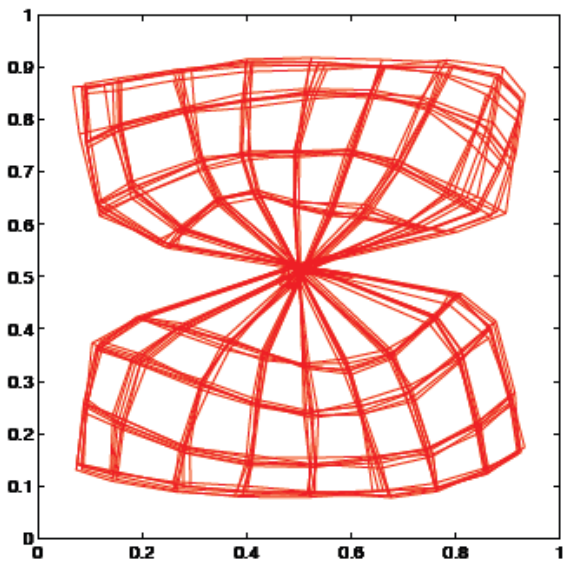
- konvergence algoritmu?, jak dlouho se učí?
- do jaké míry je zachována topologie zobrazení?
- optimalita zobrazení
- vliv parametrů α, σ

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

Dvě možné interpretace z pohledu aplikací

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

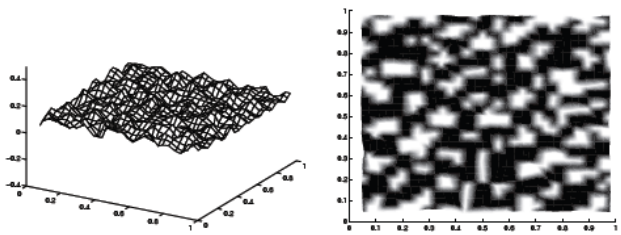
Kohonenovy mapy

Snížení dimenze dat při zachování topologie – příklad:

- Síť zobrazí n-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

Příklad

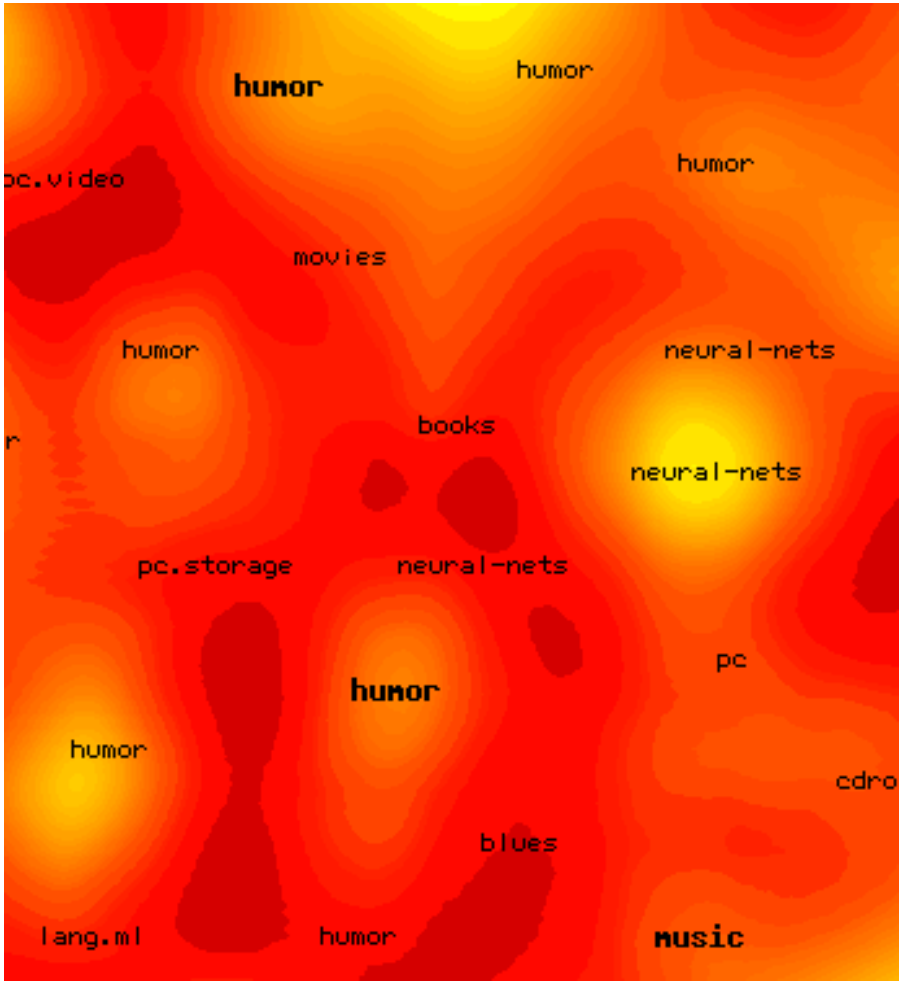
- 2-dimenzionální síť ve 3-dimenzionálním vstupním prostoru



Kohonenovy mapy

Snížení dimenze dat při zachování topologie

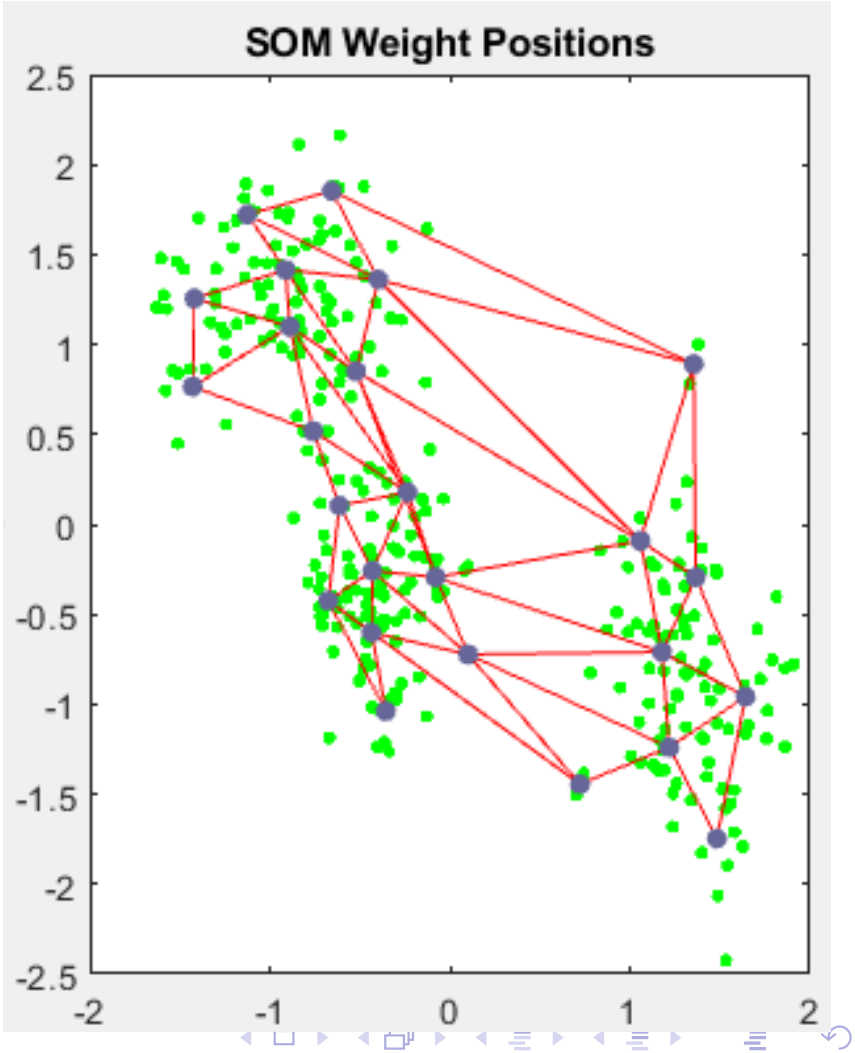
- Příklad aplikace: WEBSOM (1998)
- 2D-vizualizace blízkosti (podobnosti) webových dokumentů



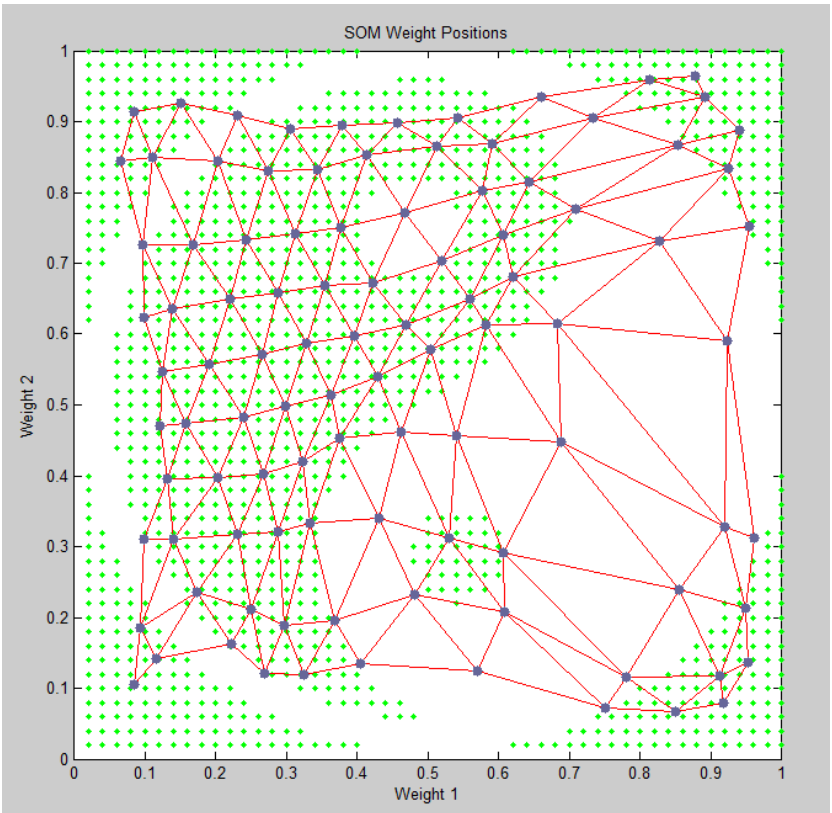
Kohonenovy mapy

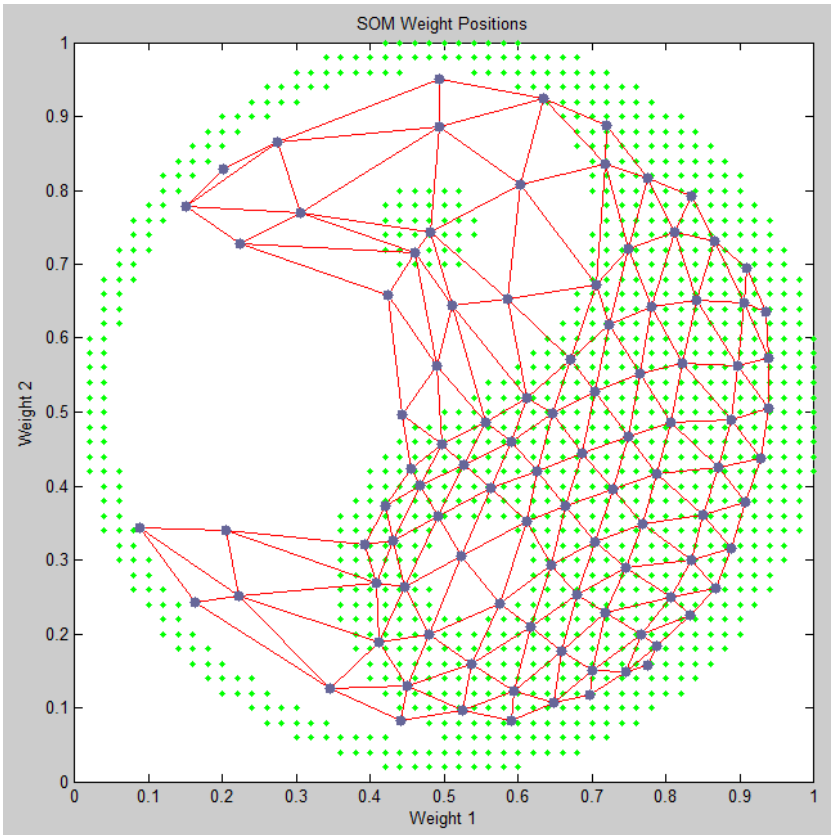
Shlukování (klastrování)

- 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 *(vektorová kvantizace)*
- Výstupní neurony jsou reprezentanti vstupních dat (shluků)
- Navíc se zachovává struktura fyzické sousednosti



Kohonenovy mapy -příklad





Kohonenovy mapy - Jak je to v Matlabu

- *selforgmap* ... vytvoření kohonenovy mapy
  - net = selforgmap(dimensions)
  - net = selforg-map(dimensions,coverSteps,initNeighbor,topologyFcn,distanceFcn)
  - dimensions ... rozměry sítě, implicitně [8 8]
  - coverSteps ... počet kroků učení, než se nastaví nulové okolí, implicitně 100
  - initNeighbor ... počáteční velikost okolí, implicitně 3
  - topologyFcn... topologie, implicitně *'hextop'*
  - distanceFcn ... funkce vzdálenosti, implicitně *'linkdist'*
- Možné topologie
  - *'hextop'*, *'gridtop'*, *'randtop'*, *'tritop'*
- Funkce vzdálenosti
  - *'dist'* (Euklidova), *'linkdist'*, *'boxdist'* (čtverec), *'manlist'* (Manhattan)