

Vytěžování Dat

Přednáška 5 – Self Organizing Map

Miroslav Čepek

Katedra počítačů, Computational Intelligence Group



Evropský sociální fond Praha & EU:
Investujeme do vaší budoucnosti

16.10.2012

- Jaké znáte shlukovací algoritmy?
 - KMeans
 - Hierarchické shlukování
 - KMeans – nedopadne pokaždé stejně, musím zkoušet počet centroidů.
 - Hierarchické shlukování – musím spočítat N^2 vzdáleností. Což pro větší N není jednoduché.

- Jaké znáte shlukovací algoritmy?
- KMeans
- Hierarchické shlukování
- KMeans – nedopadne pokaždé stejně, musím zkoušet počet centroidů.
- Hierarchické shlukování – musím spočítat N^2 vzdáleností. Což pro větší N není jednoduché.

Shlukovací algoritmy a nevýhody

- Jaké znáte shlukovací algoritmy?
- KMeans
- Hierarchické shlukování
- KMeans – nedopadne pokaždé stejně, musím zkoušet počet centroidů.
- Hierarchické shlukování – musím spočítat N^2 vzdáleností. Což pro větší N není jednoduché.

- Existuje spousta dalších algoritmů pro shlukování dat.
- Ukáži vám ještě jeden – **Self Organizing Map (SOM)**.

- Jedinci (reprezentanti, centroidy, neurony, jedinci) spolu soutěží – o něco :).
- Nepotřebuji žádného arbitra (učitele), který by říkal, kam se mají jedinci přesunout. Každý jedinec to umí zjistit sám.
- Jedinci se učí z příkladů.
- Systém (populace jedinců) se v průběhu času **samoorganizuje** sám.
- A teď to zkusíme použít na shlukování.

- Jedinci (reprezentanti, centroidy, neurony, jedinci) spolu soutěží – o něco :).
- Nepotřebuji žádného arbitra (učitele), který by říkal, kam se mají jedinci přesunout. Každý jedinec to umí zjistit sám.
- Jedinci se učí z příkladů.
- Systém (populace jedinců) se v průběhu času **samoorganizuje** sám.
- A teď to zkusíme použít na shlukování.

Kompetitivní učení v pohádce

- Vzpomínáte si na pohádku o Králích z minulé přednášky?
- Do země s N domy přijelo K králů a někde se usídlili. A každý král zabral domy, které mu byli nejbliž a z nich vybíral daně. A protože lidé chtěli, aby jim byl král co nejbliž, král se přestěhoval do geometrického středu domů.
- Tím se ale některé domy ocitly blíže jinému králi a tak z nich daně začal vybírat jiný král. Králové se opět přesunout a tak dále.
- Nejbližší král tedy získá všechny daně z domů, které jsem mu nejbliž.

- KMeans také používá kompetitivní učení. Jak?
- KMeans je trochu skromější.
- Reprezentanti (centroidy) soutěží o data. A nejbližší reprezentant vyhraje – zabere celou instanci a jiného reprezentant k ní nepustí. "Bere vše".

- KMeans také používá kompetitivní učení. Jak?
- KMeans je trochu skromější.
- Reprezentanti (centroidy) soutěží o data. A nejbližší reprezentant vyhraje – zabere celou instanci a jiného reprezentant k ní nepustí. "Bere vše".

Kvantizační chyba

- Minule jsem v souvislosti s KMeans mluvil o optimalizaci (minimalizaci) chyby.
- Této chybě se říká **kvantizační chyba**. A vyjadřuje průměrnou vzdálenost mezi daty a odpovídajícími reprezentanty.
- Průměrná vzdálenost mezi krály a jejich poddanými.

$$\text{kvantizační chyba} = \frac{1}{\text{počet instancí}} \sum_{i=0}^k \sum_{x \in \text{nearest}(r_i)} \text{dist}(r_i, x)$$

- r_i je i -tý reprezentant. A $\text{nearest}(r_i)$ je množina instancí, které jsou mu nejbliž.
- x je jedna z instancí.

Vektorová kvantizace

- A cílem (nejen) KMeans je minimalizovat tuto chybu.
- Tím že minimalizuji kvantizační chybu tlačím reprezentanty do míst, kde se nachází hodně instancí.
- Snažím se tím aproximovat hustotu instancí pomocí (menší hustoty) reprezentantů.
- Do míst, kde je vysoká hustota instancí, se snažím dostat hodně reprezentantů a naopak – do míst s málo instancemi dávám málo reprezentantů.
- Cílem kvantizace vektorů je aproximace hustoty pravděpodobnosti $p(x)$ výskytu instancí x pomocí konečného počtu reprezentantů w_i .

Vektorová kvantizace

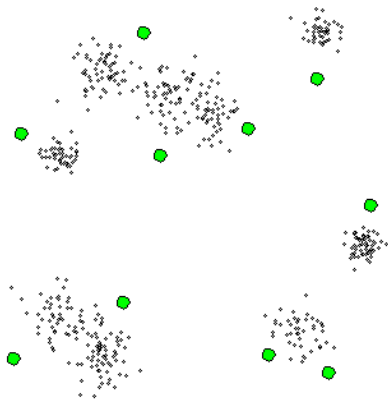
- A cílem (nejen) KMeans je minimalizovat tuto chybu.
- Tím že minimalizuji kvantizační chybu tlačím reprezentanty do míst, kde se nachází hodně instancí.
- Snažím se tím aproximovat hustotu instancí pomocí (menší hustoty) reprezentantů.
- Do míst, kde je vysoká hustota instancí, se snažím dostat hodně reprezentantů a naopak – do míst s málo instancemi dávám málo reprezentantů.
- Cílem kvantizace vektorů je aproximace hustoty pravděpodobnosti $p(x)$ výskytu instancí x pomocí konečného počtu reprezentantů w_i .

- U KMeans z domu vybírá daně jen nejbližší král. "Winner takes all".
- Co když platí, že část daní může vybírat i jiný blízký král?
- Pak už neplatí, že vítěz bere vše a něco zbude i na ostatní.
- Zde je důležité okolí – tj. jak daleko se králi ještě vyplatí jet pro svůj díl daní.
 - ▶ Malé okolí – vítěz bere vše - daně vybírá jen jeden král
 - ▶ Velké okolí – komunismus - každý král dostane z každého domu kousek.

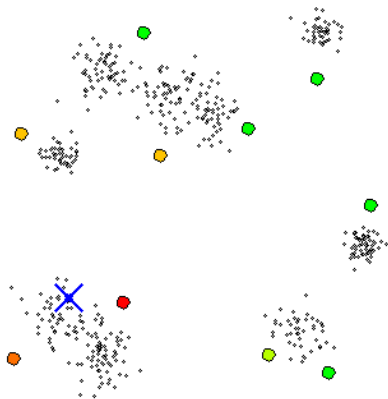
Neuronový plyn

- Jiný způsob, jakým lze minimalizovat kvantizační chybu.
 - Na rozdíl od KMeans používám okolí a jinak počítám nové pozice středů.
- 1 Náhodně rozmístí reprezentanty a zvol velké okolí.
 - 2 Vyber *nějakou* vstupní instanci x_j .
 - 3 Spočítej vzdálenost mezi x_j a všemi reprezentanty $w_i \forall i$.
 - 4 Uprav pozice všech středů v závislosti na vzdálenosti od instance a okolí.
 - 5 Zmenši okolí.
 - 6 Pokud ještě chceš pokračovat, pokačuj bodem 2.

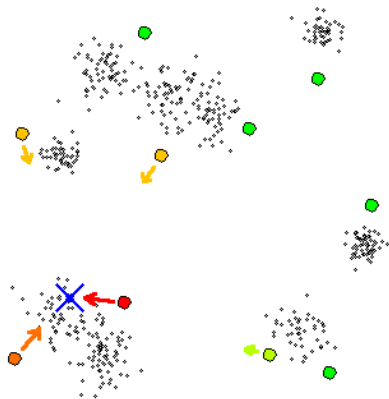
Ilustrace iterace Neuronového plynu



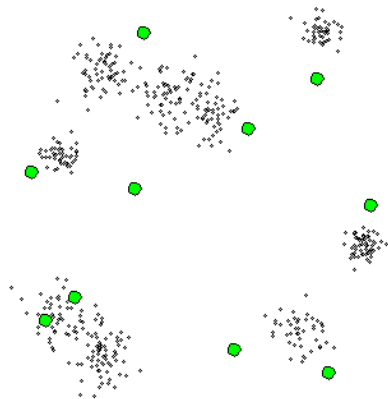
Ilustrace iterace Neuronového plynu



Ilustrace iterace Neuronového plynu



Ilustrace iterace Neuronového plynu



Neuronový plyn (II)

- V algoritmu je několik stupňů volnosti.
- "Vyber *nějakou* vstupní instanci"
 - ▶ Procházíme postupně jednotlivé instance postupně v pevném pořadí.
 - ★ Nevhodné protože výstup může záviset na pořadí předkládání instancí.
 - ▶ Projde všechny instance jednou, pak v jiném pořadí podruhé, atd...
 - ▶ Vybírá skutečně náhodně. Čili nezaručuje, že počet předložení sítí bude pro všechny instance stejný.
 - ★ Nepoužívá se, protože není zaručeno, že nepředložím x_1 10x, pak x_2 12x, atd...

Neuronový plyn (II)

- V algoritmu je několik stupňů volnosti.
- "Vyber *nějakou* vstupní instanci"
 - ▶ Procházíme postupně jednotlivé instance postupně v pevném pořadí.
 - ★ Nevhodné protože výstup může záviset na pořadí předkládání instancí.
 - ▶ Projde všechny instance jednou, pak v jiném pořadí podruhé, atd...
 - ▶ Vybírá skutečně náhodně. Čili nezaručuje, že počet předložení sítí bude pro všechny instance stejný.
 - ★ Nepoužívá se, protože není zaručeno, že nepředložím x_1 10x, pak x_2 12x, atd...

Neuronový plyn (II)

- V algoritmu je několik stupňů volnosti.
- "Vyber *nějakou* vstupní instanci"
 - ▶ Procházíme postupně jednotlivé instance postupně v pevném pořadí.
 - ★ Nevhodné protože výstup může záviset na pořadí předkládání instancí.
 - ▶ Projde všechny instance jednou, pak v jiném pořadí podruhé, atd...
 - ▶ Vybírá skutečně náhodně. Čili nezaručuje, že počet předložení sítí bude pro všechny instance stejný.
 - ★ Nepoužívá se, protože není zaručeno, že nepředložím x_1 10x, pak x_2 12x, atd...

Neuronový plyn (II)

- V algoritmu je několik stupňů volnosti.
- "Vyber *nějakou* vstupní instanci"
 - ▶ Procházíme postupně jednotlivé instance postupně v pevném pořadí.
 - ★ Nevhodné protože výstup může záviset na pořadí předkládání instancí.
 - ▶ Projde všechny instance jednou, pak v jiném pořadí podruhé, atd...
 - ▶ Vybírá skutečně náhodně. Čili nezaručuje, že počet předložení sítí bude pro všechny instance stejný.
 - ★ Nepoužívá se, protože není zaručeno, že nepředložím x_1 10x, pak x_2 12x, atd...

Neuronový plyn (II)

- V algoritmu je několik stupňů volnosti.
- "Vyber *nějakou* vstupní instanci"
 - ▶ Procházíme postupně jednotlivé instance postupně v pevném pořadí.
 - ★ Nevhodné protože výstup může záviset na pořadí předkládání instancí.
 - ▶ Projde všechny instance jednou, pak v jiném pořadí podruhé, atd...
 - ▶ Vybírá skutečně náhodně. Čili nezaručuje, že počet předložení sítí bude pro všechny instance stejný.
 - ★ Nepoužívá se, protože není zaručeno, že nepředložím x_1 10x, pak x_2 12x, atd...

Neuronový plyn (II)

- V algoritmu je několik stupňů volnosti.
- "Vyber *nějakou* vstupní instanci"
 - ▶ Procházíme postupně jednotlivé instance postupně v pevném pořadí.
 - ★ Nevhodné protože výstup může záviset na pořadí předkládání instancí.
 - ▶ Projde všechny instance jednou, pak v jiném pořadí podruhé, atd...
 - ▶ Vybírá skutečně náhodně. Čili nezaručuje, že počet předložení sítí bude pro všechny instance stejný.
 - ★ Nepoužívá se, protože není zaručeno, že nepředložím x_1 10x, pak x_2 12x, atd...

Neuronový plyn (III)

- "Uprav pozice všech středů v závislosti..."
- Čím vzdálenější reprezentant, tím se posouvá méně.
- $w_i^{t+1} = w_i^t + \eta^t e^{-k/\lambda^t} (x - w_i^t)$
 - ▶ η^t je adaptační krok v kroku t a určuje o kolik se maximálně může reprezentant posunout. (Typicky o dost menší než 1 a s rostoucím t klesá k 0).
 - ▶ k pořadí ve vzdálenosti reprezentanta od instance.
 - ▶ λ^t definuje velikost okolí a s rostoucím t klesá.

Neuronový plyn (III)

- "Uprav pozice všech středů v závislosti..."
- Čím vzdálenější reprezentant, tím se posouvá méně.
- $w_i^{t+1} = w_i^t + \eta^t e^{-k/\lambda^t} (x - w_i^t)$
 - ▶ η^t je adaptační krok v kroku t a určuje o kolik se maximálně může reprezentant posunout. (Typicky o dost menší než 1 a s rostoucím t klesá k 0).
 - ▶ k pořadí ve vzdálenosti reprezentanta od instance.
 - ▶ λ^t definuje velikost okolí a s rostoucím t klesá.

Neuronový plyn (IV)

- "Zmenši okolí"

- ▶ Okolí (λ^t) se typicky postupně zmenšuje o nějaký násobek. Např.
 $\lambda^{t+1} = \lambda^t * 0.95$
- ▶ Při zmenšování okolí se podobným způsobem zmenšuje i adaptační krok. $\eta^{t+1} = \eta^t * 0.95$.

- "Pokud ještě chceš pokračovat..."

- ▶ Dopředu určím, že chci pokračovat dokud je $\lambda > 0.05$ nebo skončí poté, co předložíš všechny instance 10x.

Kontrolní otázka: Za jakých podmínek se přesune nejbližší reprezentant na pozici právě předložené instance?

Neuronový plyn (IV)

- "Zmenši okolí"
 - ▶ Okolí (λ^t) se typicky postupně zmenšuje o nějaký násobek. Např.
 $\lambda^{t+1} = \lambda^t * 0.95$
 - ▶ Při zmenšování okolí se podobným způsobem zmenšuje i adaptační krok. $\eta^{t+1} = \eta^t * 0.95$.
- "Pokud ještě chceš pokračovat..."
 - ▶ Dopředu určím, že chci pokračovat dokud je $\lambda > 0.05$ nebo skončí poté, co předložíš všechny instance 10x.

Kontrolní otázka: Za jakých podmínek se přesune nejbližší reprezentant na pozici právě předložené instance?

Neuronový plyn (IV)

- "Zmenši okolí"
 - ▶ Okolí (λ^t) se typicky postupně zmenšuje o nějaký násobek. Např.
 $\lambda^{t+1} = \lambda^t * 0.95$
 - ▶ Při zmenšování okolí se podobným způsobem zmenšuje i adaptační krok. $\eta^{t+1} = \eta^t * 0.95$.
- "Pokud ještě chceš pokračovat..."
 - ▶ Dopředu určím, že chci pokračovat dokud je $\lambda > 0.05$ nebo skončí poté, co předložíš všechny instance 10x.

Kontrolní otázka: Za jakých podmínek se přesune nejbližší reprezentant na pozici právě předložené instance?

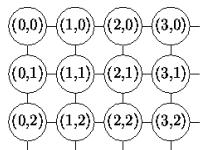
Neuronový plyn (IV)

- "Zmenši okolí"
 - ▶ Okolí (λ^t) se typicky postupně zmenšuje o nějaký násobek. Např.
 $\lambda^{t+1} = \lambda^t * 0.95$
 - ▶ Při zmenšování okolí se podobným způsobem zmenšuje i adaptační krok. $\eta^{t+1} = \eta^t * 0.95$.
- "Pokud ještě chceš pokračovat..."
 - ▶ Dopředu určím, že chci pokračovat dokud je $\lambda > 0.05$ nebo skončí poté, co předložíš všechny instance 10x.

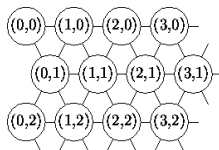
Kontrolní otázka: Za jakých podmínek se přesune nejbližší reprezentant na pozici právě předložené instance?

Vylepšení Neuronového plynu

- Jak by se dal neuronový plyn vylepšit dál?
- Co kdyby se neposouvali všichni reprezentanti blízko instance?
- Vytvoříme "přátelské" vztahy mezi reprezentanty. A budou se posouvat jen "kamarádi" vítězného reprezentanta.
- Když vizualizujeme "přátelství" mezi reprezentanty získáme pravidelnou mřížku (síť).
- Typicky se používá čtvercová nebo hexagonální síť.



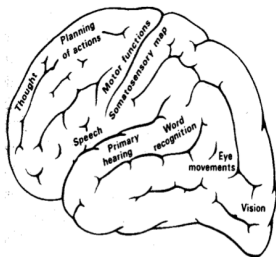
Rectangular



Hexagonal

Inspirace pro SOM

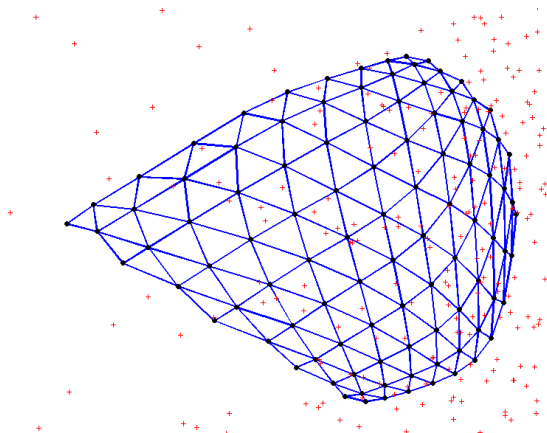
- Inspirací nejsou králové, ale oblasti v lidském mozku.
- Řídící centra jednotlivých končetin spolu souvisí a navzájem se ovlivňují.



- Neuronová síť SOM je vynálezem prof. Kohonena z Finska.
- Původně vznikla jako model motorického cortexu a její první aplikace byl fontetický psací stroj.
- A protože se prof. Kohonen zabýval umělými neuronovými sítěmi, převzal i SOM jejich terminologii.

- Každý reprezentant – v terminologii SOMu **neuron** – je opět reprezentován jeho souřadnicemi v prostoru.
- Souřadnice každého neuronu (reprezentanta) se označují jako váhy.
- Když si zkusím takovou síť vizualizovat, dostaneme například:

SOM - Pozice neuronů (II)



- WTF? Ještě před chvílí byla ta síť přece pravidelná!
- To ano, ale to byla idealizovaná projekce, aby bylo názorně vidět vztahy!

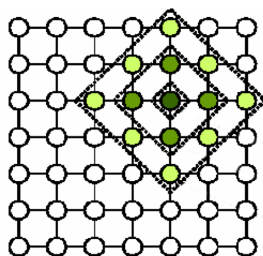
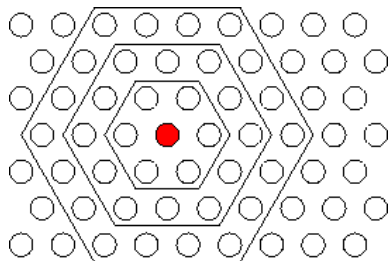
- Celý SOM algoritmus vypadá pak takto:
 - 1 Inicializuj váhy všech neuronů (souřadnice všech reprezentantů).
 - 2 Vyber *nějakou* vstupní instanci x_j .
 - 3 Spočítej vzdálenost mezi x_j a všemi neurony $w_i \forall i$.
 - 4 Urči nejbližší neuron – BMU (Best Matching Unit).
 - 5 Uprav váhy (pozici) BMU a jeho okolí.
 - 6 Pokud ještě chceš pokračovat, pokačuj bodem 2.

- Inicializace vah:
 - ▶ Rovnoměrné rozprostření pro prostoru.
 - ▶ Náhodně.
- Výběr instancí:
 - ▶ Opět můžeme vybírat instance úplně náhodně.
 - ▶ Ale mnohem častější je vybrat všechny instance jednou, pak všechny podruhé (v jiném pořadí), atd... Prochází se permutace vstupní množiny.

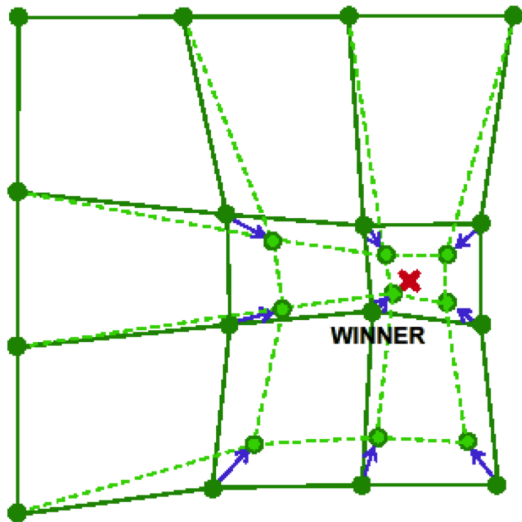
- Inicializace vah:
 - ▶ Rovnoměrné rozprostření pro prostoru.
 - ▶ Náhodně.
- Výběr instancí:
 - ▶ Opět můžeme vybírat instance úplně náhodně.
 - ▶ Ale mnohem častější je vybrat všechny instance jednou, pak všechny podruhé (v jiném pořadí), atd... Prochází se permutace vstupní množiny.

Detaily algoritmu (II)

- Výpočet vzdáleností a určení BMU je celkem jednoduchá záležitost.
- Určím si metriku, kterou budu využívat a tu aplikuji.
- Mnohem zajímavější je úprava pozice BMU a jeho okolí :).
- Jak vlastně určím neurony v okolí?



Změna vah graficky



Detaily algoritmu (III)

- Novou pozici neuronu w_i v kroku $t + 1$ (po předložení vzoru x_j) určím jako:
- $w_i^{t+1} = w_i^t + \mu(t)(x_j - w_i^t)$
- Kde $\mu(t)$ je sdružený učicí koeficient, který v sobě sdružuje jak vzdálenost neuronu od BMU tak i maximální možnou změnu vah (pozice).
- $\mu(t)$ s postupujícím časem klesá k nule.

Detaily algoritmu (IV)

$$\mu(t) = \alpha(t)e^{-\frac{\text{dist}(w_i, BMU)}{2\sigma^2(t)}}$$

- $\alpha(t)$ představuje učící krok (tedy jak moc se maximálně mohou změnit váhy neuronu).
- e^{blabla} určuje, že okolí neuron má tvar gausovky.
- $\sigma^2(t)$ určuje velikost okolí a postupně s časem klesá.

Příklad

Máme 3 neurony $w_1 = (0, 0)$ $w_2 = (2, 1)$ $w_3 = (0, 3)$ a ty jsou na lince. w_1 je sousedem w_2 , w_2 je sousedem w_3 a w_1, w_2 je sousedem w_3 .

A instanci $x = (1, 1)$

Který neuron je BMU? (Použijeme eukleidovskou metriku)

$$d(w_1, x) = \sqrt{(0 - 1)^2 + (0 - 1)^2} = \sqrt{2} = 1.41\dots$$

$$d(w_2, x) = \sqrt{(2 - 1)^2 + (1 - 1)^2} = \sqrt{1} = 1$$

$$d(w_3, x) = \sqrt{(0 - 1)^2 + (3 - 1)^2} = \sqrt{5} = 2.23\dots$$

Příklad

Máme 3 neurony $w_1 = (0, 0)$ $w_2 = (2, 1)$ $w_3 = (0, 3)$ a ty jsou na lince. w_1 je sousedem w_2 , w_2 je sousedem w_3 a w_1, w_2 je sousedem w_3 .

A instanci $x = (1, 1)$

Který neuron je BMU? (Použijeme eukleidovskou metriku)

$$d(w_1, x) = \sqrt{(0 - 1)^2 + (0 - 1)^2} = \sqrt{2} = 1.41\dots$$

$$d(w_2, x) = \sqrt{(2 - 1)^2 + (1 - 1)^2} = \sqrt{1} = 1$$

$$d(w_3, x) = \sqrt{(0 - 1)^2 + (3 - 1)^2} = \sqrt{5} = 2.23\dots$$

Příklad (II)

BMU je tedy w_2 . Řekněme, že $\sigma(t) = 1$ a $\alpha(t) = 0.25$

A zkusme vypočítat novou pozici BMU (w_2).

$$\mu(t) = \alpha(t)e^{-\frac{\text{dist}(w_i, \text{BMU})}{2\sigma^2(t)}} = 0.25 * e^{-\frac{\text{dist}(w_2, w_2)}{2*1^2}} = 0.25 * e^0 = 0.25$$

$$\begin{aligned}w_2^{t+1} &= w_2^t + \mu(t)(x - w_2^t) = (2, 1) + 0.25 * ((1, 1) - (2, 1)) = \\ &= (2, 1) + 0.25(-1, 0) = (1.75, 1)\end{aligned}$$

Pro w_1 se posune do pozice:

$$\mu(t) = \alpha(t)e^{-\frac{\text{dist}(w_1, w_2)}{2\sigma^2(t)}} = 0.25 * e^{-\frac{1}{2}} = 0.25 * 0.606 = 0.151$$

$$\begin{aligned}w_1^{t+1} &= w_1^t + \mu(t)(x - w_1^t) = (0, 0) + 0.151 * ((1, 1) - (0, 0)) = \\ &= (0, 0) + (0.151, 0.151) = (0.151, 0.151)\end{aligned}$$

Příklad (II)

BMU je tedy w_2 . Řekněme, že $\sigma(t) = 1$ a $\alpha(t) = 0.25$
A zkusme vypočítat novou pozici BMU (w_2).

$$\mu(t) = \alpha(t)e^{-\frac{\text{dist}(w_i, \text{BMU})}{2\sigma^2(t)}} = 0.25 * e^{-\frac{\text{dist}(w_2, w_2)}{2*1^2}} = 0.25 * e^0 = 0.25$$

$$\begin{aligned}w_2^{t+1} &= w_2^t + \mu(t)(x - w_2^t) = (2, 1) + 0.25 * ((1, 1) - (2, 1)) = \\ &= (2, 1) + 0.25(-1, 0) = (1.75, 1)\end{aligned}$$

Pro w_1 se posune do pozice:

$$\mu(t) = \alpha(t)e^{-\frac{\text{dist}(w_1, w_2)}{2\sigma^2(t)}} = 0.25 * e^{-\frac{1}{2}} = 0.25 * 0.606 = 0.151$$

$$\begin{aligned}w_1^{t+1} &= w_1^t + \mu(t)(x - w_1^t) = (0, 0) + 0.151 * ((1, 1) - (0, 0)) = \\ &= (0, 0) + (0.151, 0.151) = (0.151, 0.151)\end{aligned}$$

- Stejně jako v Hierarchickém shlukování a K-Means potřebujeme nějakou míru "dobrého shluknutí".
- Kvantizační chyba
 - ▶ Ale tu už známe! To je přece chyba, o které jsme mluvili na začátku přednášky!
 - ▶ Průměrná vzdálenost mezi instancemi a nejbližšími neurony.
- Topografická chyba
 - ▶ Popisuje kvalitu "natažení" mřížky sítě na vstupní data.
 - ▶ Procento instancí, pro které platí, že jejich BMU a druhý nejbližší neuron nejsou sousedy v mřížce sítě.

$$err_{topo} = \frac{1}{n} \sum_{i=1}^n u(x_i)$$

$u(x_i) = 1 \iff$ BMU a druhý nejbližší neuron pro x_i nejsou sousedé v mřížce.

- Stejně jako v Hierarchickém shlukování a K-Means potřebujeme nějakou míru "dobrého shluknutí".
- Kvantizační chyba
 - ▶ Ale tu už známe! To je přece chyba, o které jsme mluvili na začátku přednášky!
 - ▶ Průměrná vzdálenost mezi instancemi a nejbližšími neurony.
- Topografická chyba
 - ▶ Popisuje kvalitu "natažení" mřížky sítě na vstupní data.
 - ▶ Procento instancí, pro které platí, že jejich BMU a druhý nejbližší neuron nejsou sousedy v mřížce sítě.

$$err_{topo} = \frac{1}{n} \sum_{i=1}^n u(x_i)$$

$u(x_i) = 1 \iff$ BMU a druhý nejbližší neuron pro x_i nejsou sousedé v mřížce.

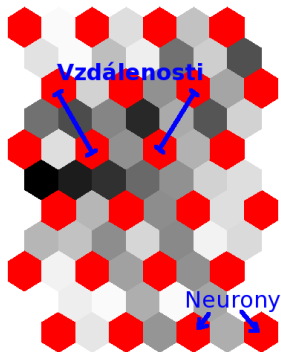
- Stejně jako v Hierarchickém shlukování a K-Means potřebujeme nějakou míru "dobrého shluknutí".
- Kvantizační chyba
 - ▶ Ale tu už známe! To je přece chyba, o které jsme mluvili na začátku přednášky!
 - ▶ Průměrná vzdálenost mezi instancemi a nejbližšími neurony.
- Topografická chyba
 - ▶ Popisuje kvalitu "natažení" mřížky sítě na vstupní data.
 - ▶ Procento instancí, pro které platí, že jejich BMU a druhý nejbližší neuron nejsou sousedy v mřížce sítě.

$$err_{topo} = \frac{1}{n} \sum_{i=1}^n u(x_i)$$

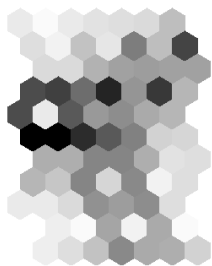
$u(x_i) = 1 \iff$ BMU a druhý nejbližší neuron pro x_i nejsou sousedé v mřížce.

- Dokud máme jen 2D data, tak s vizualizací není problém. Ale co když máme více dimenzí?
 - ▶ U-Matice
 - ▶ Analýza hlavních komponent
 - ▶ Sammonova projekce

- Matice vzdáleností mezi váhovými vektory jednotlivých neuronů, typicky se vizualizuje, vzdáleností vyjádřeny barvou – světlá barva = malá vzdálenost.
- Zobrazuje strukturu vzdáleností v prostoru dat.

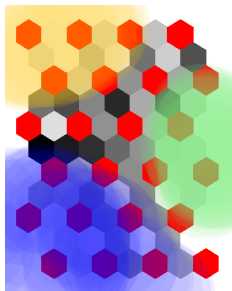


- Barva neuronu je vzdálenost je váhového vektoru od všech ostatních váhových vektorů.
- Tmavé váhové vektory jsou vzdáleny od ostatních datových vektorů ve vstupním prostoru.
- Světlé váhové vektory jsou obklopeny cizími vektory ve vstupním prostoru.



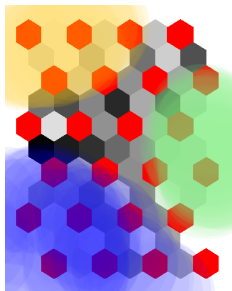
U-Matice (III)

- Jak z U-Matice poznám shluky?
 - Ze vzdáleností mezi neurony.
 - Kopce oddělují clustery (údolí).



U-Matice (III)

- Jak z U-Matice poznám shluky?
- Ze vzdáleností mezi neurony.
- Kopce oddělují clustery (údolí).

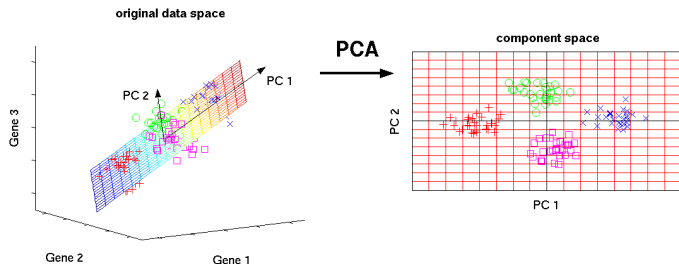


Analýza hlavních komponent

- Jde o statistickou metodu pro redukci dimenzionality.
- Označuje se jako PCA z anglického Principal Component Analysis.
- Snaží se najít nové osy, které lépe popisují data s minimální ztrátou informace.
- První osa vede směrem, který má největší rozptyl hodnot, druhá osa směrem, kde je druhý největší rozptyl, atd...
- Vždy mi vrátí stejný počet nových os, jako mají původní data dimenzí, ale já se mohu rozhodnout některé nepoužít.

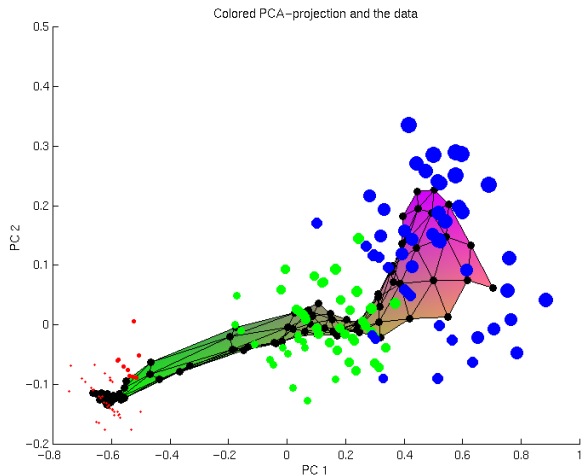
Analýza hlavních komponent (II)

- Výpočet nových souřadnic pomocí
 - ▶ kovariance, vlastních čísel a vlastních vektorů.
- http://www.cs.otago.ac.nz/cosc453/student_tutorials/principal_components.pdf



Využití PCA v SOM

- Nyní můžeme provést PCA projekci SOM sítě do 2D a zobrazit si ji.



- PCA není limitována jen na použití v SOM, ale můžu ji použít například pro průzkum dat.
- Stejně tak, některé metody vytěžování dat nemají rády příliš mnoho dimenzí a PCA jim můžete pomoci k lepším výsledkům.
- Nevýhodou je umělost nových os, která znesnadňuje interpretaci získaných výsledků.
- $0.125 * \text{petal_length} + 0.578 * \text{petal_width} + 0.934 * \text{sepal_length} - .0346 * \text{sepal_width}$

Sammonova projekce

- Jinou možností je Sammonova projekce.
- Ta netransformuje osy, ale znovu umísťuje objekty v novém (méně dimenzionálním) prostoru.
- Při umísťování se snaží zachovat vztahy v datech (data, která byla blízko v původním prostoru, budou blízko i v novém prostoru).

Sammonova projekce (2)

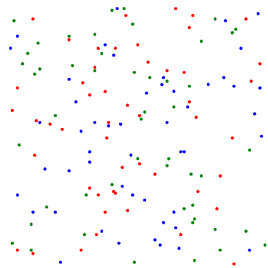
- Sammonova projekce se snaží minimalizovat následující funkci:

$$E = \frac{1}{\sum_{i < j} dist^*(x_i, x_j)} \sum_{i < j} \frac{(dist^*(x_i, x_j) - dist(x_i, x_j))^2}{dist^*(x_i, x_j)}$$

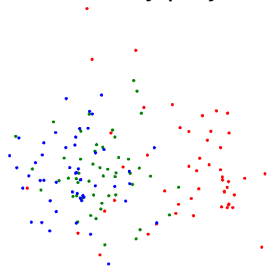
- $dist^*(x_i, x_j)$ je vzdálenost x_i a x_j v původním prostoru.
- $dist(x_i, x_j)$ je vzdálenost x_i a x_j v novém prostoru (v projekci).
- Pro minimalizaci se používají standardní optimalizační metody – pro tuto úlohu typicky iterační metody.
- Při minimalizaci se pohybuje body v novém prostoru (v projekci). Tím ovlivníte $dist(x_i, x_j)$ a můžete dosáhnout zmenšení E .

Sammonova projekce - ukázka

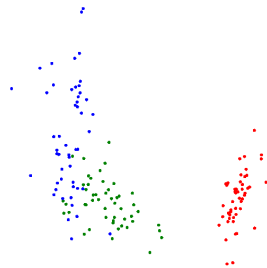
Ukázka několika iterací Sammonovy projekce na Iris datech.



Počáteční stav



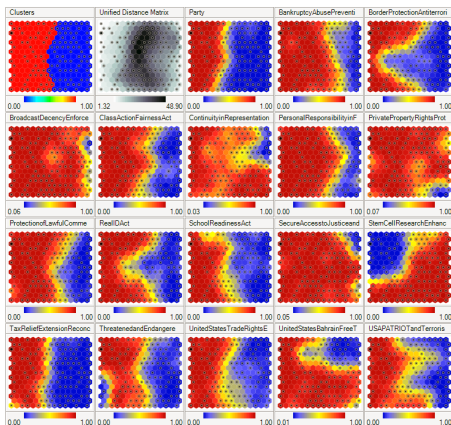
1. iterace



10. iterace

Další vizualizace – Příznakové grafy

- Vychází z U-Maticy, ale místo vzdálenosti jednotlivých vektorů se do šestiúhelníků kreslí hodnoty vybrané proměnné.



- <http://www.cis.hut.fi/somtoolbox/theory/somalgorithm.shtml>
- <http://www.cis.hut.fi/research/som-research/>
- <http://www.ai-junkie.com/ann/som/som1.html>
- www.cs.bham.ac.uk/~jxb/NN/l16.pdf