18NES1 Neuronové sítě 1 - Samoorganizace

18NES1 - 21. hodina, LS 2024/25

Zuzana Petříčková

5. května 2025

Co jsme probírali minule

Lehký úvod do dalších modelů hlubokého učení

Učení bez učitele (samoorganizace, unsupervised learning)

- Úvod, základní pojmy
- Klasifikace a Algoritmus k nejbližších sousedů
- Shlukování a Algoritmus k středů (k-means clustering)
- Ukázky - úvod

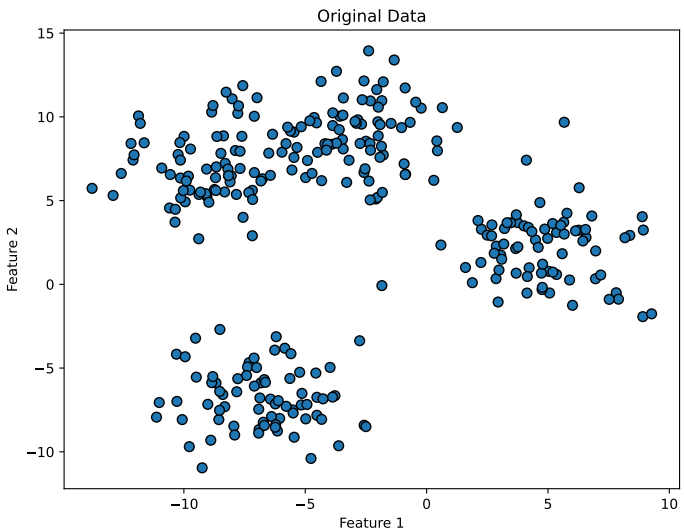
Dnešní hodina

Shlukování

- Algoritmus k středů (k-means clustering) - pokračování
- Metody pro zhodnocení kvality shlukování a pro určení optimálního počtu shluků
- Další metody pro shlukování: hierarchické shlukování

Učení bez učitele, unsupervised learning, samoorganizace

- trénovací množina T tvaru $T = \{\vec{x}_1, ..., \vec{x}_N\}$ (pouze vstupy)
- $\vec{x}_i \in R^n$ 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)

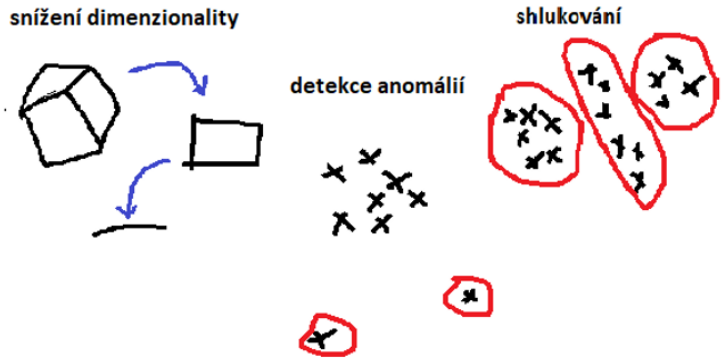


- máme data a neznáme jejich strukturu
- snažíme se strukturu a vlastnosti dat odhalit, najít v nich vzory, popř. strukturu

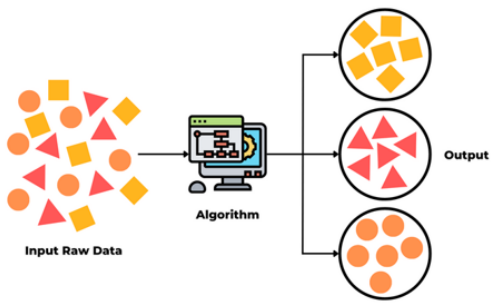
Učení bez učitele, unsupervised learning, samoorganizace

- **Cíl učení:** najít strukturu nebo vzory v datech
- **Aplikace: snížení dimenzionality** (komprese dat, vizualizace),
  detekce anomálií (např. v bankovních transakcích),
  shlukování (např. zákazníků podle chování, detekce pagiátů)
  e-komerce (doporučovací systémy)

Typy úloh:



Inspirováno: https://towardsdatascience.com/unsupervised-learning-algorithms-cheat-sheet-d391a39de44a



https://eastgate-software.com/what-is-unsupervised-learning/#ftoc-heading-7

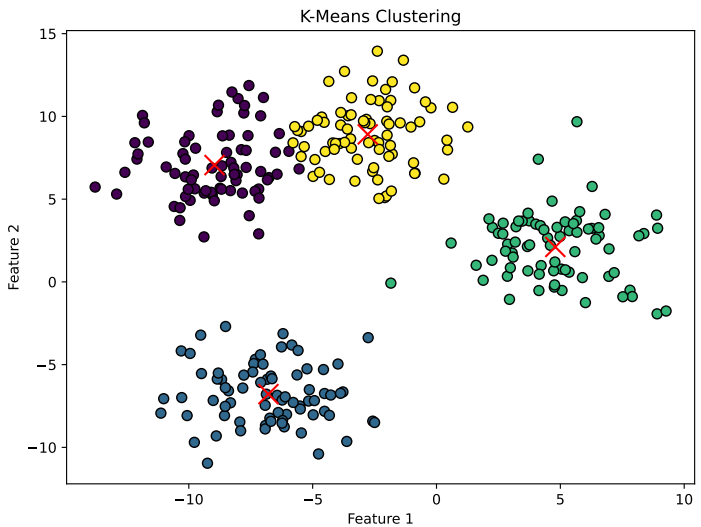
Učení bez učitele, unsupervised learning, samoorganizace

Shluk (klastr)

- Skupina vzorů s **velkou podobností mezi sebou** a **malou podobností se vzory s ostatních shluků**
- Zjednodušeně: **podobnost = vzdálenost**

Shlukování (klastrování)

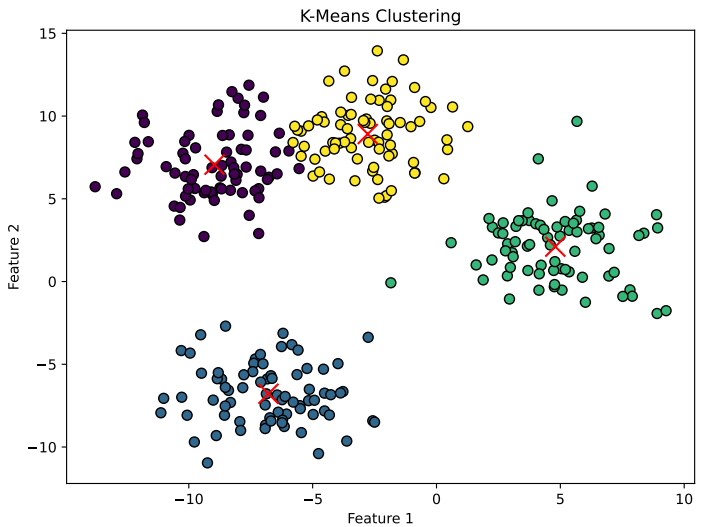
- Disjunktní rozdělení dat na shluky



Shlukování

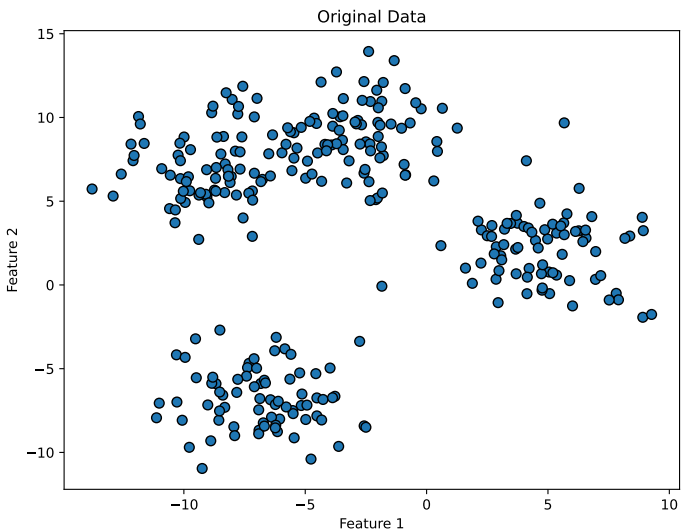
Problémy:

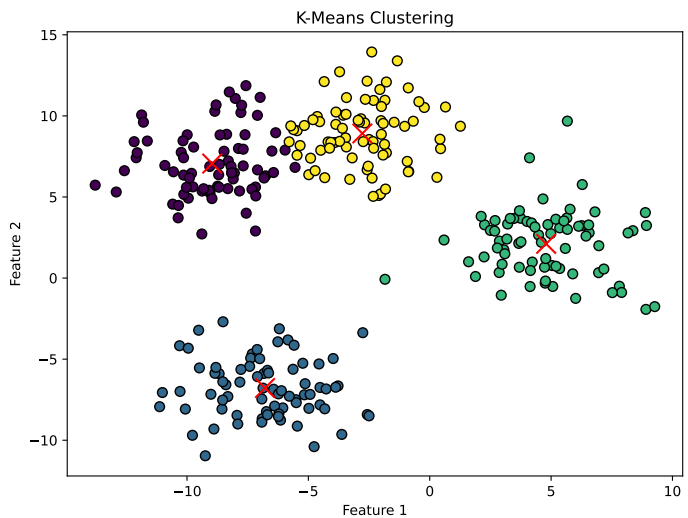
- Jak určit počet a rozložení shluků v příznakovém prostoru?
- Jak vybrat reprezentanta/y shluku?
  - Vhodně vybrané / všechny trénovací vzory patřící do shluku
  - např. střed shluku (*těžiště*, *centroid*)



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





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

Parametry algoritmu k-means

- Počet shluků k
- Metrika vzdálenosti - jak počítat vzdálenost (podobnost) vektorů?
- Inicializační metoda - jak inicializovat centroidy?
- Kritéria ukončení - kdy algoritmus ukončit?

Parametry algoritmu k-means

Jak počítat vzdálenost (podobnost) číselných vektorů?

- **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** $d(\vec{p}, \vec{q}) = \max_i |p_i - q_i|$
  ... **„Co je největší problém?"**
- **Minkewského metrika:** $d(\vec{p}, \vec{q}) = (\sum_{i=1}^{n} |p_i - q_i|^r)^{\frac{1}{r}}$
  ... zobecnění předchozích metrik $(r = 2, 1, \to \infty)$
- **Kosínová podobnost:** $\cos(\vec{p}, \vec{q}) = \frac{\vec{p} \cdot \vec{q}}{\|\vec{p}\| \|\vec{q}\|}$
  ... nezajímá nás velikost vektorů, ale jejich směr (např. zpracování textu)
- (a další)

Parametry algoritmu k-means

Inicializace centroidů

- Výsledek k-means závisí na počáteční volbě centroidů.
- Špatná inicializace $\rightarrow$ špatné shluky, pomalá konvergence, uvíznutí v lokálním minimu.
- Možnosti inicializace:
  - Náhodné vektory v prostoru R^n nebo z rozsahu vzorů
  - Náhodný výběr bodů z trénovací množiny T
  - **k-means++**:
    - první centroid se volí náhodně,
    - další centroidy se vybírají s pravděpodobností úměrnou čtverci vzdálenosti od nejbližšího již zvoleného centroidu.
  - Vícenásobné spuštění algoritmu a výběr nejlepšího řešení (nejnižší suma čtverců vzdáleností)

Parametry algoritmu k-means - shrnutí

- Počet shluků k (obvykle zadán uživatelem)
- Metrika vzdálenosti (standardně Euklidovská)
- Inicializační metoda (random, k-means++, vlastní volba)
- Kritéria ukončení:
  - dokud se mění příslušnost vzorů
  - dokud se mění centroidy
  - dosažení maximálního počtu iterací
- Další vylepšení: Pokud k nějakému centroidu není přiřazen žádný vzor $\rightarrow$ inicializujeme ho znova

Příklad (ukázka)

kmeans_clustering.ipynb,

- Ukázka vlastní implementace k-means algoritmu včetně vizualizace průběhu učení
- Několik datových sad, různé možnosti inicializace vah

Otázky:

- Jaký má na učení vliv inicializace centroidů?
- Jak dlouho trvá učení pro větší data?
- Jak najít optimální počet shluků?
- Jak zhodnotit, zda jsou vytvořené shluky kvalitní?

Algoritmus k středů (k-means clustering)

Výhody

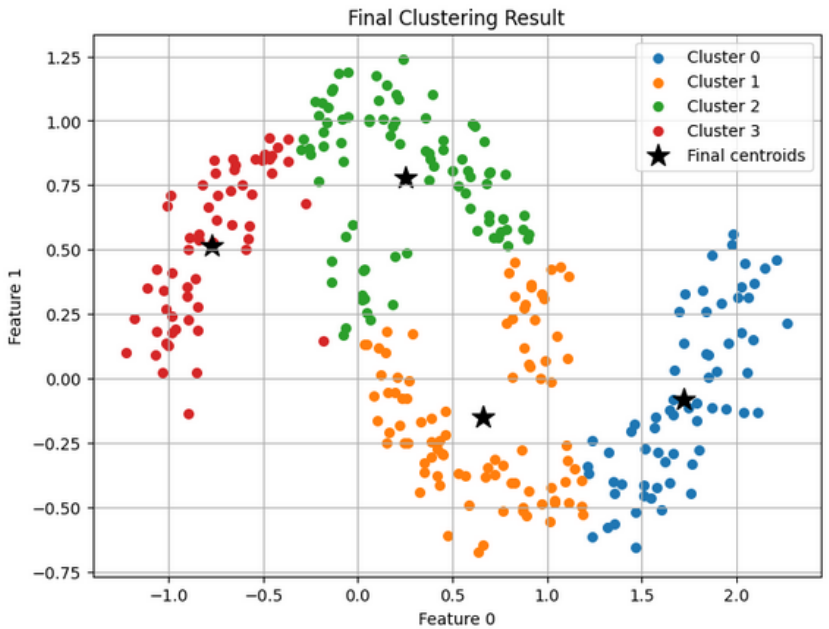
- Rychlý algoritmus, jednoduchý na implementaci
- Vhodný pro rychlý náhled na strukturu dat

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)

Ukázky složitějších úloh: kmeans_clustering.ipynb



Další ukázky na webu ScikitLearn

Algoritmus k středů (k-means clustering)

Nevýhody a jejich řešení

- Nutnost zvolit počet shluků k předem
  $\rightarrow$ zkusíme spustit algoritmus pro různé hodnoty k a určíme optimální hodnotu
- Dávkové zpracování (problém pro velká data nebo online učení)
  $\rightarrow$ minibatch k-means, online k-means
- Vysoká citlivost k počáteční volbě centroidů $\rightarrow$ vhodná počáteční volba centroidů (např. podle náhodně vybraných vzorů)
- Citlivost k odlehlým vzorům
  $\rightarrow$ normalizace dat:
  - zajistí také invarianci vůči změně měřítka a posunutí
  - ale ne vždy pomůže (např. přiblíží vzdálené shluky)

Algoritmus k středů (k-means clustering)

Nevýhody a jejich řešení

- Pro složitá data nemusí být úspěšný: vyhledává sférické shluky $\rightarrow$ volba jiné metriky pro vzdálenosti
- Problém, pokud je vysoká dimenze vstupních dat (**prokletí dimenzionality**), popř. navzájem silně korelované příznaky $\rightarrow$ použijeme PCA (Principal Component Analysis) pro předzpracování vstupních příznaků

PCA analýza

PCA (Principal component analysis)

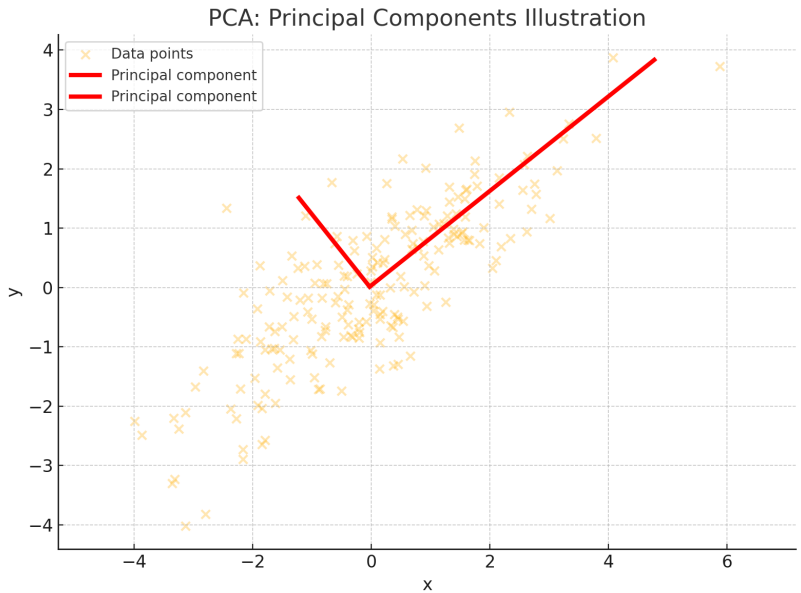
- redukce dimenzionality vstupních dat
  - použít méně příznaků bez ztráty podstatné informace
  - výběr nejdůležitějších příznaků (hlavní komponenty, principal components)
- Každá hlavní komponenta (PC) je ortogonální směr, který zachycuje co největší rozptyl v datech.
- Nejdůležitější příznak (první komponenta) je tedy takový jednotkový vektor $\vec{w}$, který maximalizuje rozptyl projekcí všech datových bodů:

$$\vec{w} = \arg \max_{\|\vec{w}\|=1} \frac{1}{N} \sum_{i=1}^{N} (\vec{w}^\top \vec{x}_i)^2$$

- Hledáme směr, do kterého jsou data nejvíce „roztažená“.

PCA analýza

PCA (Principal component analysis)



- Každá hlavní komponenta (PC) je ortogonální směr, který zachycuje co největší rozptyl v datech.
- Obvyklá strategie: ponechat tolik komponent, aby vysvětlily např. 90–95 % celkového rozptylu.

Metriky pro hodnocení kvality shlukování

Jak poznat, že jsou shluky vytvořené algoritmem opravdu kvalitní?

- posouzení ,,pouhým okem" funguje jen u dat s nízkou dimenzí (např. 2D, 3D)
- pro obecná data potřebujeme definovat metriky, které umožní automatické vyhodnocení:
  - **kompaktnost:** jsou body ve shluku blízko sebe?
  - **separabilita:** jsou různé shluky od sebe dobře oddělené?
- Takové metriky lze využít i k určení optimálního počtu shluků
- *Žádná metrika však není univerzálně nejlepší pro všechny typy dat a situací*

Metriky pro hodnocení kvality shlukování

Silueta (Silhouette)

- měří, jak jsou si navzájem vzdálené body uvnitř svého shluku ve srovnání s ostatními shluky.

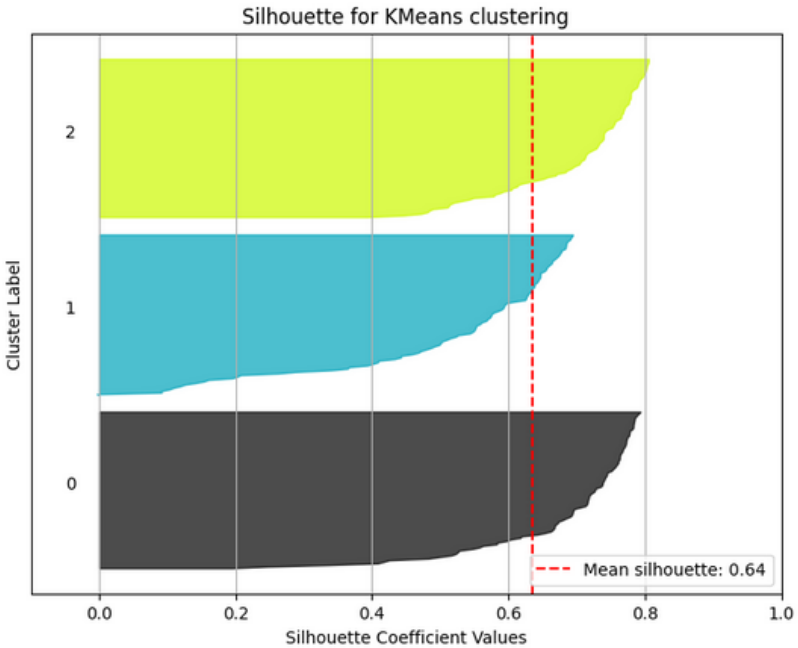
$$S(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$

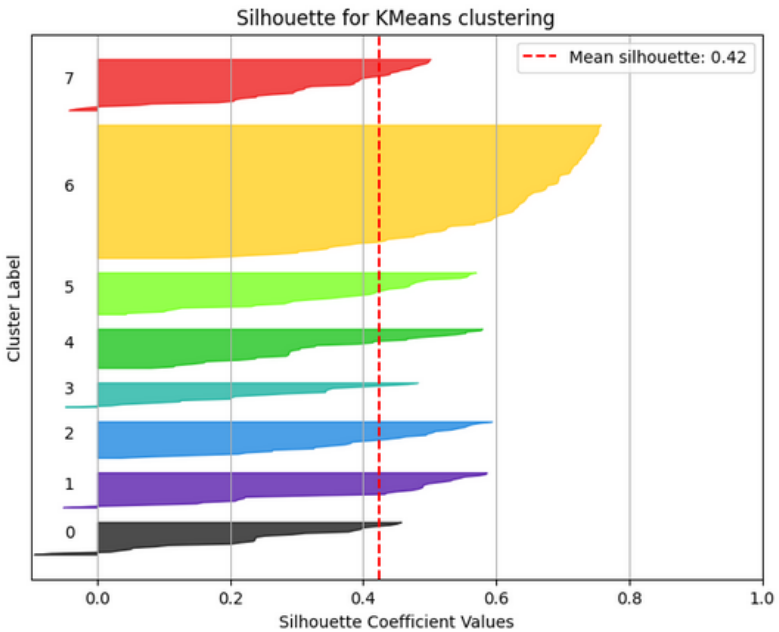
  - $S(i)$ je silueta pro i-tý bod
  - $a(i)$ je průměrná vzdálenost bodu i k bodům ve stejném shluku
  - $b(i)$ je průměrná vzdálenost bodu i k bodům v nejbližším sousedním shluku.
- čím větší hodnota (≈ 1), tím lépe je bod přiřazen; hodnota ≈ 0 značí hraniční bod; záporná hodnota značí špatné přiřazení
- oblíbená míra pro určení optimálního počtu shluků (maximalizuje se)

Metriky pro hodnocení kvality shlukování

Silueta (Silhouette)

- výsledky lze znázornit pomocí **silhouette plotu**:
  - každý vzor je zobrazen jako vodorovný pruh (jeho délka odpovídá hodnotě $S(i)$)
  - ideálně mají všechny pruhy kladnou délku a jsou dlouhé
  - červená svislá čára značí průměrné silhouette skóre všech bodů





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 průměrná vzdálenost bodů ve shluku i od jeho středu c_i (kompaktnost)
- $d(c_i, c_j)$ je vzdálenost mezi středy i-tého a j-tého shluku (separabilita)

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 vzdáleností pro shluk
  - n je počet bodů

Metriky pro hodnocení kvality shlukování

Vnitřní suma čtverců (WCSS)

- Součet čtverců vzdáleností mezi jednotlivými body a středy jejich shluků.
- Nižší WCSS znamená kompaktnější shluky.
- Typicky klesá s rostoucím počtem shluků k.

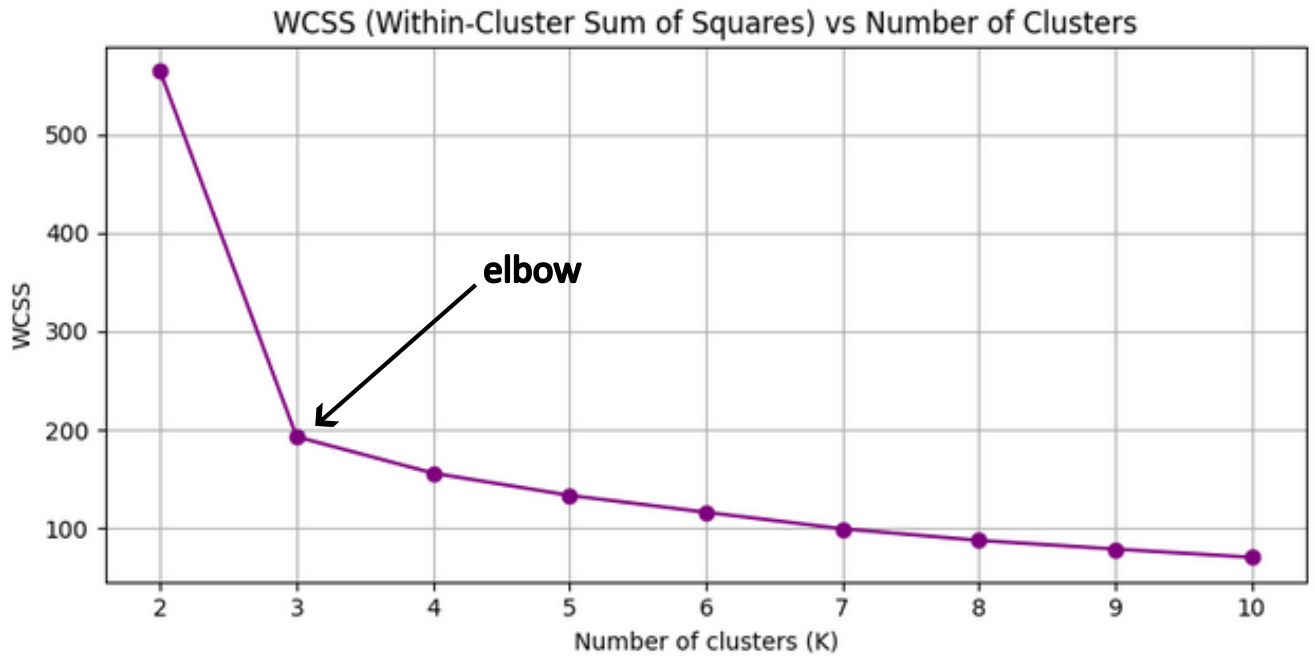
$$WCSS = \sum_{i=1}^{k} \sum_{x \in C_i} ||x - c_i||^2$$

  - C_i je množina bodů přiřazených do i-tého shluku,
  - c_i je centroid i-tého shluku.
- Používá se pro určení optimálního počtu shluků pomocí tzv. **Elbow** metody (metoda lokte)

Jak využít metriky k určení optimálního počtu shluků?

Vnitřní suma čtverců (WCSS) - Elbow metoda (metoda lokte)

- Sledujeme pokles vnitřní sumy čtverců.
- Zvolíme k v bodě zlomu ("loket"), kde další zvýšení k už výrazně nesnižuje WCSS.



Jak využít metriky k určení optimálního počtu shluků?

- **WCSS (Elbow Method):**
  - Sledujeme pokles vnitřní sumy čtverců.
  - Zvolíme k v bodě zlomu ("loket"), kde další zvýšení k už výrazně nesnižuje WCSS.
- **Silhouette Score, Calinski-Harabasz Index:**
  - Vybereme k s nejvyšší hodnotou indexu.
- **Davies-Bouldin Index:**
  - Vybereme k s nejnižší hodnotou indexu (nižší = lepší oddělení shluků).

Poznámka: Různé metriky mohou navrhovat různá k — při rozhodování je vhodné porovnat více hledisek.

kmeans_clustering.ipynb,

Algoritmus k středů (k-means clustering)

Aplikace: 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

vector_quantization.ipynb,

- Podívejte se, jak lze využít shlukování k vektorové kvantizaci.
  - Zkuste různé počty centroidů a porovnejte výsledky
  - Vyzkoušejte různé obrázky - fotografie, klidně i jiné než z repozitáře.

Další přístupy ke shlukování

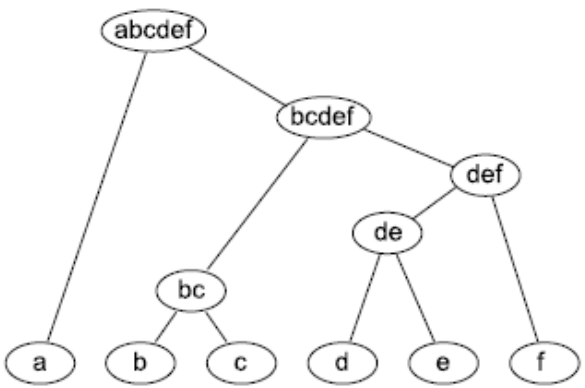
K-means není jediný způsob, jak rozdělit data do shluků.

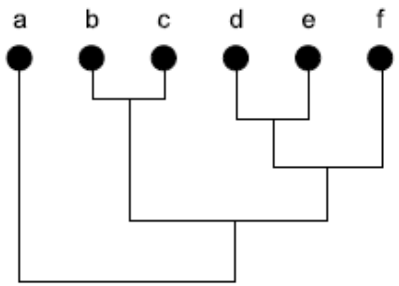
- Různé typy algoritmů lépe fungují na různých typech dat
- Některé metody nevyžadují předem daný počet shluků
- Jiné dokáží odhalit shluky složitějších tvarů nebo pracovat s kategoriálními daty
- Mezi známé alternativy patří:
  - hierarchické shlukování
  - shlukování založené na hustotě dat (např. DBSCAN)
  - spektrální shlukování - využívá vlastní vektory grafu podobnosti
  - modelově založené metody (např. Gaussian Mixture Models)

Hezké srovnání: Scikit Learn

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 **,**$C_j \subset R^n$ **?**

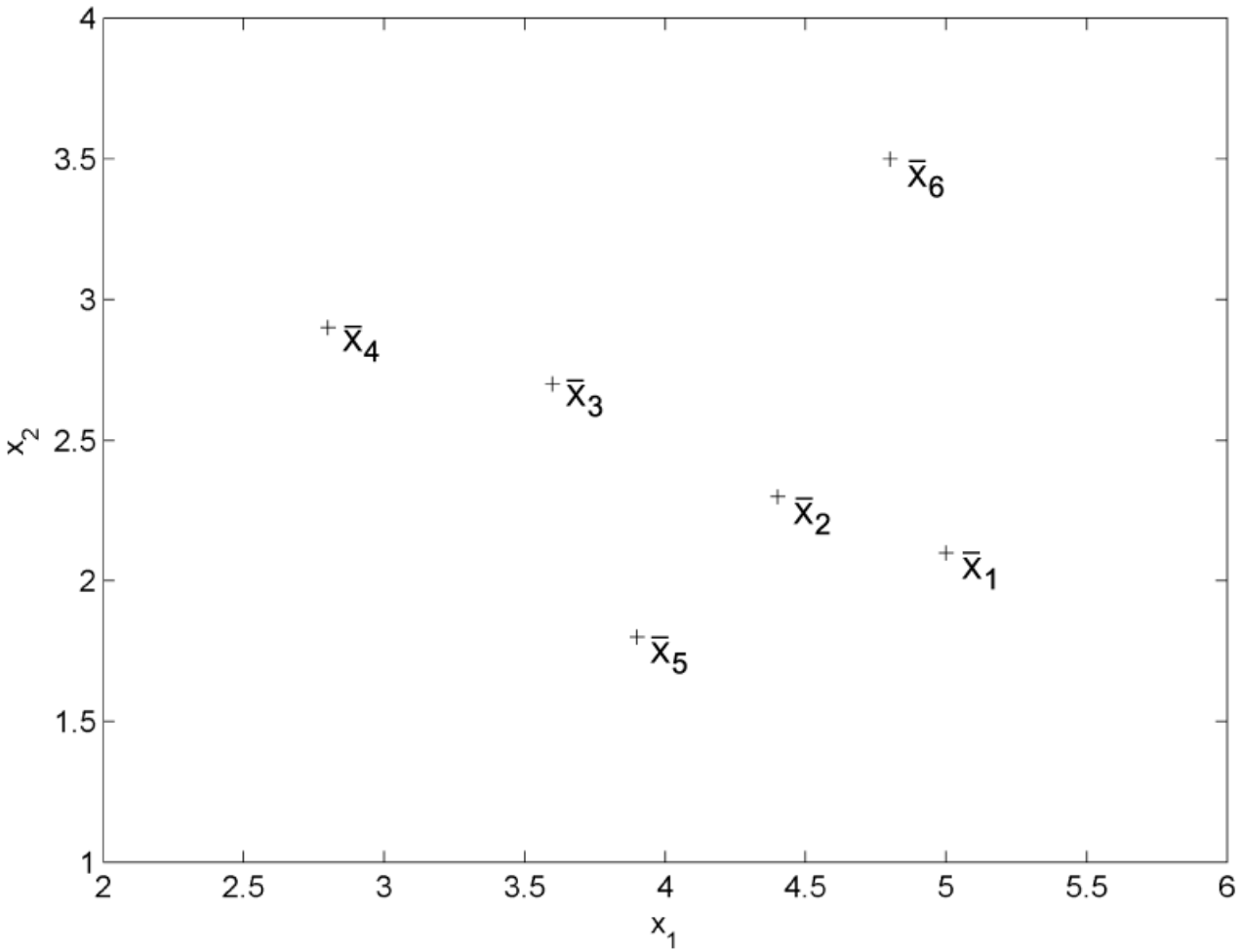
- **Metoda nejbližšího souseda (single linkage, min)**
  $d(C_i, C_j) = \min\{d(\vec{x}, \vec{y}) | \vec{x} \in C_i, \vec{y} \in C_j, i \neq j\}$
- **Metoda nejvzdálenějšího souseda (complete linkage, max)**
  $d(C_i, C_j) = \max\{d(\vec{x}, \vec{y}) | \vec{x} \in C_i, \vec{y} \in C_j, i \neq j\}$
- **Průměrná vzdálenost shluků (average linkage, mean)**
  $d(C_i, C_j) = \frac{1}{m_i m_j} \sum_{\vec{x} \in C_i} \sum_{\vec{x'} \in C_j} d(\vec{x}, \vec{y})$
- **Vzdálenost středů shluků (centroid linkage)**
  $d(C_i, C_j) = d(\vec{\mu}_i, \vec{\mu}_j)$
- **Wardova metoda minimálního rozptylu** uvnitř shluku

$$d(C_i, C_j) = \frac{m_i m_j}{m_i + m_j} d(\vec{\mu}_i, \vec{\mu}_j)$$

Hierarchické shlukování

- **Metoda nejbližšího souseda (single linkage, min)**
  - vhodná pro podlouhlé shluky
  - je citlivá na šum a na změny v poloze bodů
- **Metoda nejvzdálenějšího souseda (complete linkage, max)**
  - podporuje vznik kompaktních, kulatých shluků
  - omezuje vznik podélných shluků, citlivá na šum
- **Průměrná vzdálenost shluků (average linkage, mean)**
  - kompromis mezi předchozími dvěma metodami
  - méně citlivá na šum
- **Vzdálenost středů shluků (centroid linkage)**
  - málo citlivá na šum a přitom méně náročná na výpočet
  - má tendenci chybně spojit vzdálenější klastry
- **Wardova metoda minimálního rozptylu**
  - minimalizuje celkový rozptyl uvnitř klastru
  - upřednostňuje slučování malých shluků s velkými oproti slučování stejně velkých shluků
  - vytváří klastry podobné velikosti

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

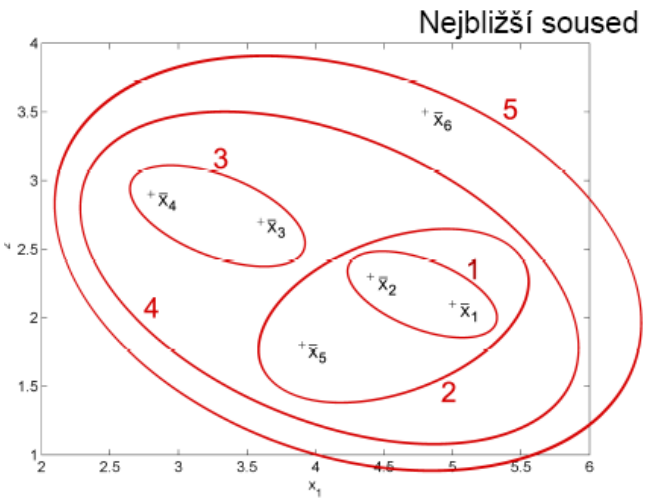
Hierarchické shlukování - příklad

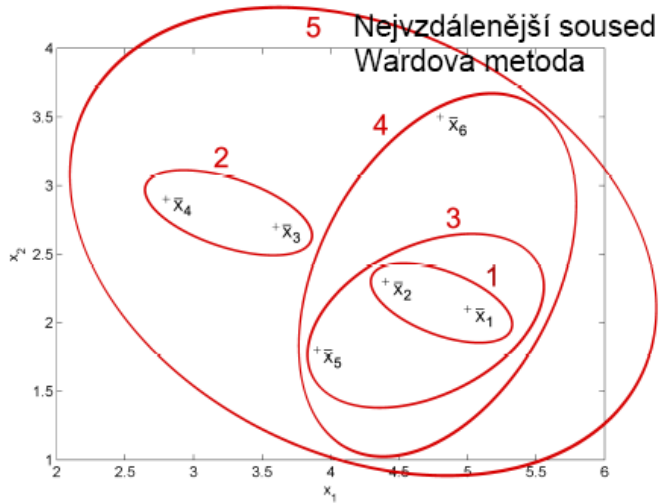
Matice vzdáleností mezi trénovacími vzory:

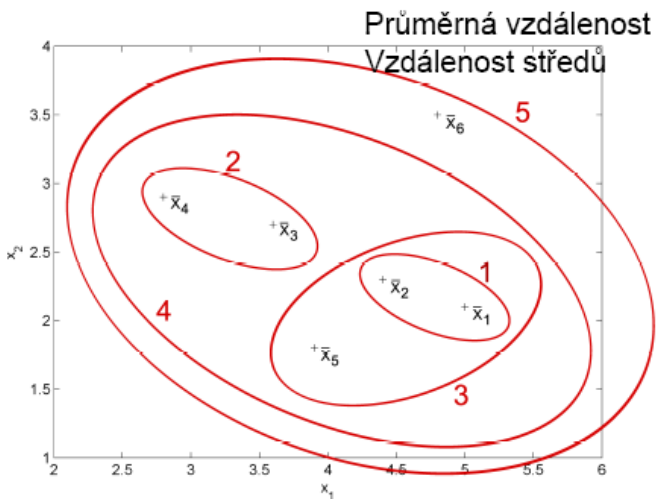
	$\overline{x}_1$	$\overline{x}_2$	$\overline{x}_3$	$\overline{x}_4$	$\overline{x}_5$	$\overline{x}_6$
$\overline{x}_1$	0,0000	0,6325	1,5232	2,3409	1,1402	1,4142
$\overline{x}_2$	0,6325	0,0000	0,8944	1,7088	0,7071	1,2649
$\overline{x}_3$	1,5232	0,8944	0,0000	0,8246	0,9487	1,4422
$\overline{x}_4$	2,3409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

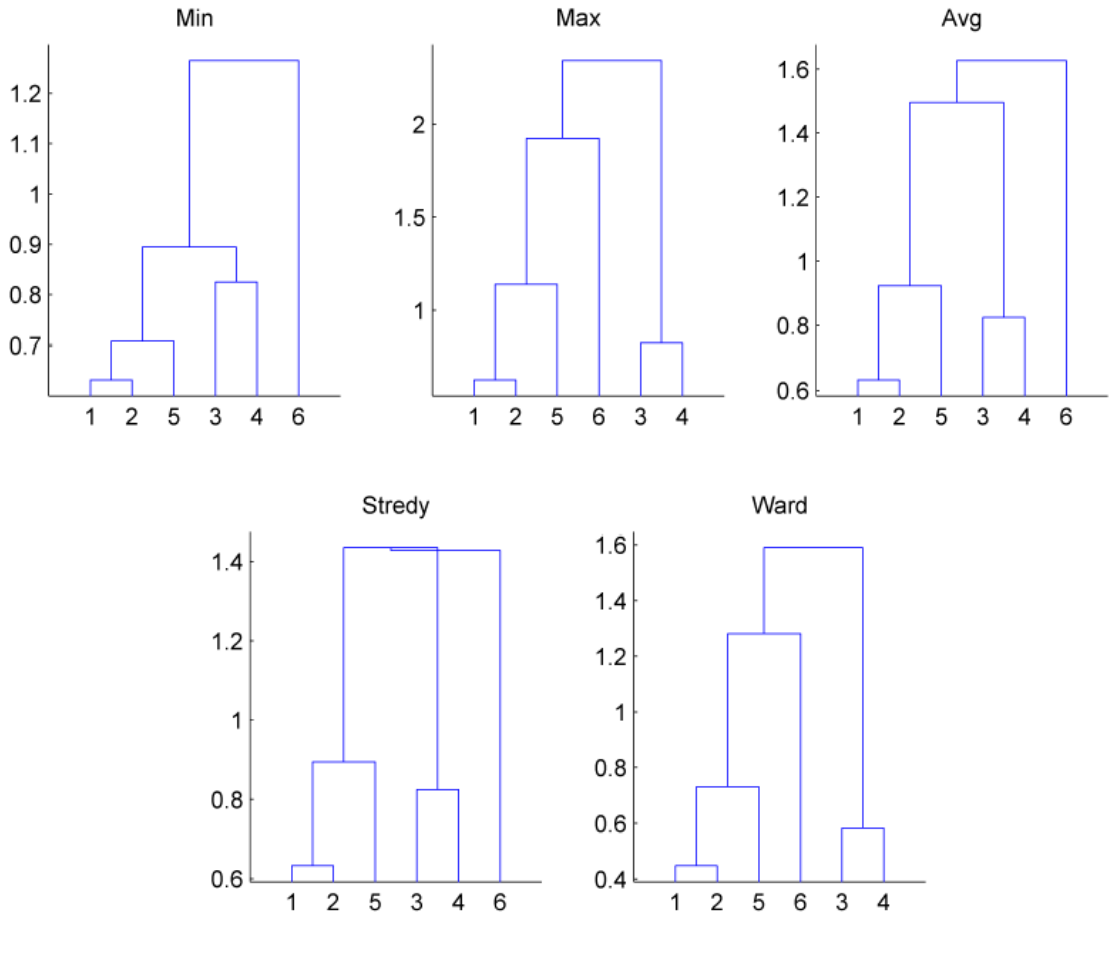






Zdroj: Kateřina Horaisová: slidy k předmětu Neuronové sítě 2, FJFI ČVUT Děčín

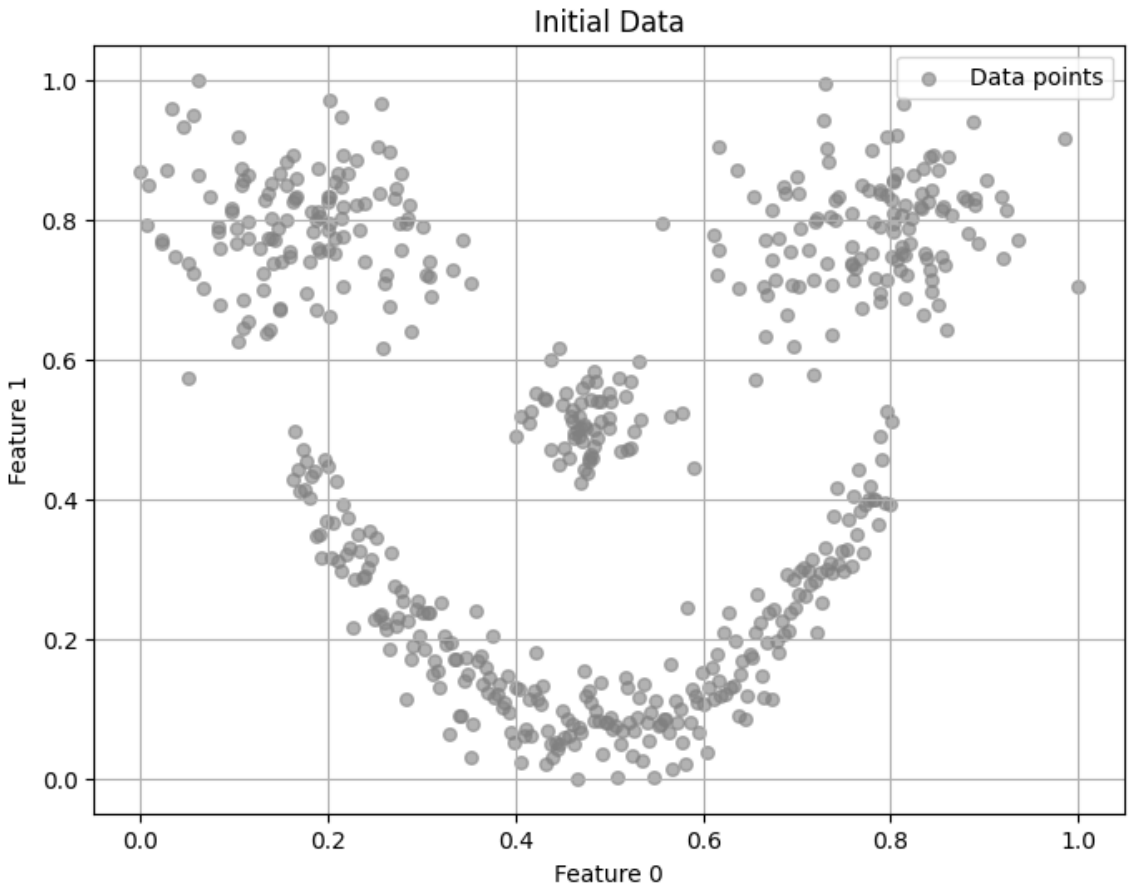
Hierarchické shlukování - příklad



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?



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

...

Příklad (ukázka)

hierarchical_clustering.ipynb,

- Ukázka hierarchického shlukování pomocí knihovny SciPy
- Dendrogramy pro různé metody (single, complete, ward,....)

Otázky:

- Jaký je rozdíl mezi dendrogramy pro různé metriky?
- Pokuste se nalézt optimální dvojice: metrika + počet shluků pro zvolená data.
- Jak na základě dendrogramu určíme počet shluků?

Vytvoření shluků na základě dendrogramu:

- **Pevně zadaný počet shluků**
- **Řez stromem podle výšky spojení** – rozdělí strom ve zvolené výšce (výška odpovídá vzdálenosti)
- **Adaptivní dělení pomocí koeficientu nekonzistence** – lokální rozhodování podle odchylek od předchozích spojení

Koeficient nekonzistence v hierarchickém shlukování

Koeficient nekonzistence pomáhá určit, kde řezat dendrogram na základě toho, jak moc se dané spojení liší od předchozích slučování.

$$\text{inconsistence}(i) = \frac{d_i - \mu_i}{\sigma_i}$$

- d_i – výška spojení (vzdálenost) ve i-tém kroku slučování
- μ_i – průměrná výška spojení v předchozích r úrovních
- σ_i – směrodatná odchylka těchto r výšek

Vyšší hodnota značí, že dané spojení může být mezi výrazně odlišnými shluky.

- řežeme větve, kde míra nekonzistence překročí daný práh (obecně v různých výškách)

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

Kompetitivní modely a kompetiční učení

- Vraťme se k modelu neuronové sítě s jednou vrstvou
- **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)*

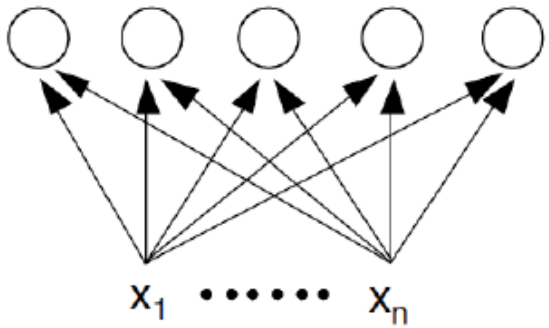
Cíl učení

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

Kompetitivní modely

Základní 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

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

   - 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

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 - 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 a nevýhody:

- Online učení („online verze algoritmu k-means")
- 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)