LAPI – Laboratorul de Analiza şi Prelucrarea Imaginilor

Facultatea de Electronică, Telecomunicaţii şi Tehnologia Informaţiei

Universitatea POLITEHNICA din Bucureşti

Tehnici de analiză și clasificare automată a informației

Conf. dr. ing. Bogdan IONESCU
http://imag.pub.ro/~bionescu

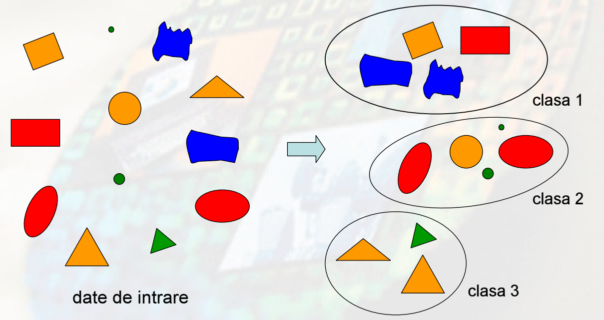
Bucureşti, 2015

Plan Curs

> M3. Tehnici de clasificare ne-supervizată

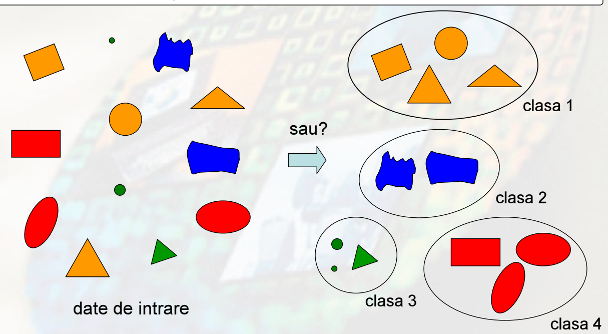
Clasificare ne-supervizată (clustering) - principiu

clustering = partiționarea datelor de intrare în mulțimi similare fără a dispune de informații a priori despre acestea (date de antrenare);



Clasificare ne-supervizată (clustering) - principiu (cont.)

clustering = partiționarea datelor de intrare în clase fără a dispune de exemple de partiționări (cont. exemplu);



Clasificare ne-supervizată (clustering) - principiu (cont.)

întreg procesul depinde de modul de definire al conceptului de **similaritate** între date;

- similaritate = un concept foarte subiectiv;

- la nivel uman, este greu de definit dar il recunoaștem atunci când il vedem;

Analiza similarității datelor

1. **Similaritatea descriptorilor**

determinarea gradului de asemănare dintre doi descriptori;

$$\left.\begin{aligned} X_1 &= [x_{1,1}, x_{1,2}, \ldots, x_{1,n}] \\ &\ldots \\ X_m &= [x_{m,1}, x_{m,2}, \ldots, x_{m,n}] \end{aligned}\right\} d(X_i, X_j) = ?$$

instanțe de intrare

Dacă *d(.)* este metrică presupune:

- simetrie: $d(X_i, X_j) = d(X_j, X_i)$
- valoare minimă (0): $d(X_i, X_i)$
- respectă: $d(X_i, X_k) \le d(X_i, X_j) + d(X_j, X_k) \quad \forall i, j, k$

Analiza similarității datelor (cont.)

1. **Similaritatea descriptorilor (cont.)**

↳ distanța Minkovski

$$d_{Mink}(X_i, X_j) = \sqrt[r]{\sum_{k=1}^{n} |x_{i,k} - x_{j,k}|^r}$$

unde X_i și X_j sunt două instanțe de intrare, $x_{i,k}$ cu *k=1,...,n* reprezintă valorile atributelor pentru instanța X_i iar |.| reprezintă modulul.

↳ distanța Manhattan (r=1)

$$d_{Manh}(X_i, X_j) = \sum_{k=1}^{n} |x_{i,k} - x_{j,k}|$$

Analiza similarității datelor (cont.)

1. **Similaritatea descriptorilor (cont.)**

↳ distanța Euclidiană (r=2)

$$d_{Euclid}(X_i, X_j) = \sqrt{\sum_{k=1}^{n} |x_{i,k} - x_{j,k}|^2}$$

> pentru a evita dependența de unitatea de măsură, datele pot fi standardizate = fiecare atribut să aibă pondere ~egală:

$$d_{wEuclid}(X_i, X_j) = \sqrt{\sum_{k=1}^{n} w_k \cdot |x_{i,k} - x_{j,k}|^2}$$

unde w_k sunt o serie de ponderi.

Analiza similarității datelor (cont.)

1. **Similaritatea descriptorilor (cont.)**

↳ distanța Canberra

$$d_{Canb}(X_i, X_j) = \sum_{k=1}^{n} \frac{|x_{i,k} - x_{j,k}|}{|x_{i,k}| + |x_{j,k}|}$$

↳ distanța Bray-Curtis

$$d_{B-C}(X_i, X_j) = \frac{\sum_{k=1}^{n} |x_{i,k} - x_{j,k}|}{\sum_{k=1}^{n} |x_{i,k} + x_{j,k}|}$$

Analiza similarității datelor (cont.)

1. **Similaritatea descriptorilor (cont.)**

↳ distanța între date binare (valori 0 sau 1)

$$d_{bin}(X_i, X_j) = \frac{r + s}{q + r + s + t}$$

unde:
- *q* este numărul de atribute ce au valoarea 1 pentru ambele instanțe,
- *t* este numărul de atribute cu valoare 0 pentru ambele instanțe,
- *s + r* reprezintă numărul de atribute de valori diferite pentru cele două instanțe (0 vs. 1 și respectiv 1 vs. 0).

Analiza similarității datelor (cont.)

1. **Similaritatea descriptorilor (cont.)**

↳ distanța între histograme de valori

$$d_{\text{inter}}(X_i, X_j) = \sum_{k=1}^{n} \min\{x_{i,k}, x_{j,k}\}$$

unde $x_{i,k}$ cu *k=1,…,n* (bini) reprezintă valorile histogramei iar *min{.}* returnează valoarea minimă a unei mulțimi.

$$d_{\text{hist}}(X_i, X_j) = \sqrt{(X_i - X_j)^T \cdot A \cdot (X_i - X_j)}$$

unde *X* reprezintă o histogramă, ^{T} este operația de transpusă iar $A=[a_{k,l}]$ cu *k,l=1,…,n* este o matrice pătratică ce indică corelația dintre binii *k* și *l*.

Analiza similarității datelor (cont.)

1. **Similaritatea descriptorilor (cont.)**

↳ distanța Bhattacharyya (între distribuții de probabilitate)

$$d_{\text{Bhatta}}(X_i, X_j) = \frac{1}{8}(\mu_{X_i} - \mu_{X_j})^T \cdot (\Sigma_{X_i,X_j})^{-1} \cdot (\mu_{X_i} - \mu_{X_j}) +$$

$$+\frac{1}{2} \cdot \ln\left(\frac{\det(\Sigma_{X_i,X_j})}{\sqrt{\det(\Sigma_{X_i}) \cdot \det(\Sigma_{X_j})}}\right)$$

unde μ_X este vectorul medie al distribuției de probabilitate a instanței X, Σ_X este matricea de covarianță a distribuției lui X, $\Sigma_{Xi,Xj}$ este media aritmetică a matricelor de covarianță pentru distribuțiile lui X_i și X_j, ^{T} este transpusa unei matrice iar *det(.)* returnează determinantul unei matrice.

Analiza similarității datelor (cont.)

1. **Similaritatea descriptorilor (cont.)**

↳ distanța Hausdorff (între mulțimi de valori)

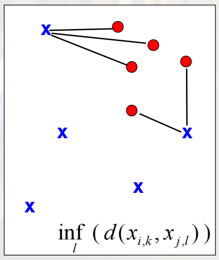
● - X_i

x - X_j

$$d_{\text{Haus}}(X_i, X_j) = \max\{\sup_k(\inf_l(d(x_{i,k}, x_{j,l}))),$$

$$\sup_l(\inf_k(d(x_{i,k}, x_{j,l})))\}$$

unde:
- *k,l=1,…,n*;
- *inf(.)* și *sup(.)* sunt infimum și respectiv supremum al unei mulțimi;
- *d(.)* este o metrică;
- *max{.}* returnează valoarea maximă a unei mulțimi.



Analiza similarității datelor (cont.)

1. **Similaritatea descriptorilor (cont.)**

↳ distanța Hausdorff (între mulțimi de valori; cont.)

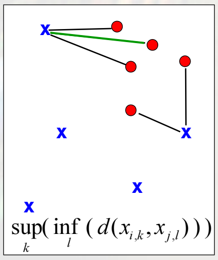
● - X_i

x - X_j

$$d_{\text{Haus}}(X_i, X_j) = \max\{\sup_k(\inf_l(d(x_{i,k}, x_{j,l}))),$$

$$\sup_l(\inf_k(d(x_{i,k}, x_{j,l})))\}$$

unde:
- *k,l=1,…,n*;
- *inf(.)* și *sup(.)* sunt infimum și respectiv supremum al unei mulțimi;
- *d(.)* este o metrică;
- *max{.}* returnează valoarea maximă a unei mulțimi.



Analiza similarității datelor (cont.)

1. **Similaritatea descriptorilor (cont.)**

↳ distanța Hausdorff (între mulțimi de valori; cont.)

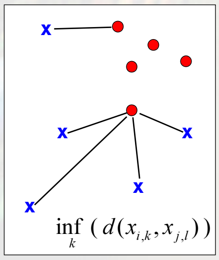
● - X_i

x - X_j

$$d_{\text{Haus}}(X_i, X_j) = \max\{\sup_k(\inf_l(d(x_{i,k}, x_{j,l}))),$$

$$\sup_l(\inf_k(d(x_{i,k}, x_{j,l})))\}$$

unde:
- *k,l=1,…,n*;
- *inf(.)* și *sup(.)* sunt infimum și respectiv supremum al unei mulțimi;
- *d(.)* este o metrică;
- *max{.}* returnează valoarea maximă a unei mulțimi.



Analiza similarității datelor (cont.)

1. **Similaritatea descriptorilor (cont.)**

↳ distanța Hausdorff (între mulțimi de valori; cont.)

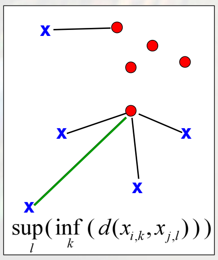
● - X_i

x - X_j

$$d_{\text{Haus}}(X_i, X_j) = \max\{\sup_k(\inf_l(d(x_{i,k}, x_{j,l}))),$$

$$\sup_l(\inf_k(d(x_{i,k}, x_{j,l})))\}$$

unde:
- *k,l=1,…,n*;
- *inf(.)* și *sup(.)* sunt infimum și respectiv supremum al unei mulțimi;
- *d(.)* este o metrică;
- *max{.}* returnează valoarea maximă a unei mulțimi.



Analiza similarității datelor (cont.)

1. **Similaritatea descriptorilor (cont.)**

↳ distanța Hausdorff (între mulțimi de valori; cont.)

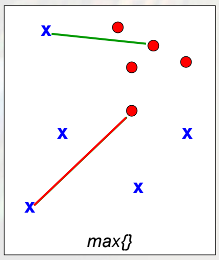
● - X_i

x - X_j

$$d_{\text{Haus}}(X_i, X_j) = \max\{\sup_k(\inf_l(d(x_{i,k}, x_{j,l}))),$$

$$\sup_l(\inf_k(d(x_{i,k}, x_{j,l})))\}$$

unde:
- *k,l=1,…,n*;
- *inf(.)* și *sup(.)* sunt infimum și respectiv supremum al unei mulțimi;
- *d(.)* este o metrică;
- *max{.}* returnează valoarea maximă a unei mulțimi.



Analiza similarității datelor (cont.)

1. **Similaritatea descriptorilor (cont.)**

↳ distanța cosinus

$$d_{\cos}(X_i,X_j)=\frac{X_i \cdot X_j}{\| X_i \| \cdot \| X_j \|}$$

unde ∘ reprezintă produsul scalar iar ||.|| reprezintă norma unui vector, astfel:

$$\| X \|^2=\sum_{k=1}^{n} x_k^2$$

unde $X=[x_1,x_2,\ldots,x_n]$.

> distanța este practic cosinusul unghiului celor doi vectori normalizați.

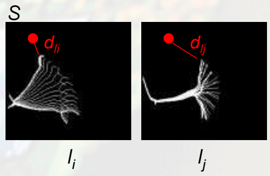
Analiza similarității datelor (cont.)

1. **Similaritatea descriptorilor (cont.)**

↳ distanța Baddeley (între obiecte)

$$d_{\text{Badd}}(I_i,I_j)=\left[\frac{1}{M \cdot N}\sum_{p\in S}| d_{I_i}(p)-d_{I_j}(p)|^q\right]^{\frac{1}{q}}$$

unde:
- *I* este o imagine binară,
- *S* reprezintă setul de puncte din imagine (*M*x*N* puncte),
- $d_I(p)$ reprezintă o anumită metrică de la punctul *p* la cel mai apropiat punct al obiectului din imaginea *I*,
- *q* este exponentul (ex. *q=2*).



Analiza similarității datelor (cont.)

1. **Similaritatea descriptorilor (cont.)**

↳ distanța Earth Mover's Distance (între date de dimensiuni diferite)

$$d_{\text{EMD}}(X_i,X_j)=\frac{\sum_{k=1}^{m}\sum_{l=1}^{n} d_{k,l}\cdot f_{k,l}}{\sum_{k=1}^{m}\sum_{l=1}^{n} f_{k,l}}$$

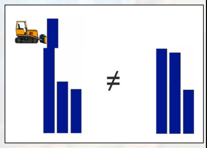
unde X_i și X_j au dimensiuni diferite (*m*, *n*), $d_{k,l}$ reprezintă distanța dintre valorile $x_{i,k}$ și $x_{j,l}$ iar $f_{k,l}$ este o funcție de cost ce cuantizează deplasarea între $x_{i,k}$ și $x_{j,l}$ determinată ca minimizând valoarea costului total:

$$\sum_{k=1}^{m}\sum_{l=1}^{n} d_{k,l}\cdot f_{k,l}$$

Analiza similarității datelor (cont.)

1. **Similaritatea descriptorilor (cont.)**

↳ distanța Earth Mover's Distance (cont.)



> reprezintă practic “volumul de muncă” necesar transformării unei instanțe în cealaltă;

> exemplu, fie:

$$X=[(x_1,w_1),(x_2,w_2),\ldots,(x_n,w_n)]$$

$$Y=[(y_1,u_1),(y_2,u_2),\ldots,(y_n,u_n)]$$

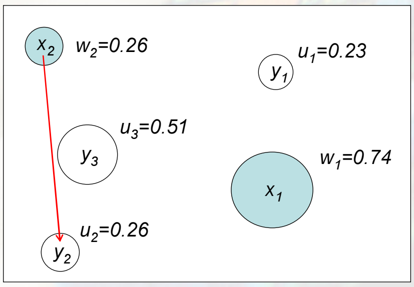
unde *X* și *Y* sunt două instanțe de comparat iar *w* și *u* reprezintă ponderile atributelor (~masă);

Analiza similarității datelor (cont.)

1. **Similaritatea descriptorilor (cont.)**

↳ distanța Earth Mover's Distance (cont.)

> exemplu (cont.) - presupunem ponderi egale, Σ*w*= Σ*u*;



- calculăm necesarul de muncă ca să transformăm *X* în *Y* (mutăm masa de la *X* la *Y*);

$work_{2,2} = f_{2,2} * d_{2,2} = 0.26 * 316$

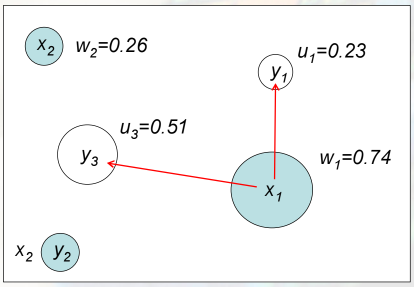
[S. Cohen, 1999]

Analiza similarității datelor (cont.)

1. **Similaritatea descriptorilor (cont.)**

↳ distanța Earth Mover's Distance (cont.)

> exemplu (cont.) - presupunem ponderi egale, Σ*w*= Σ*u*;



$work_{1,1} = f_{1,1} * d_{1,1} = 0.23 * 155$

$work_{1,3} = f_{1,3} * d_{1,3} = 0.51 * 252$

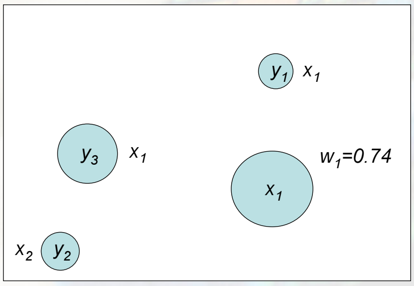
[S. Cohen, 1999]

Analiza similarității datelor (cont.)

1. **Similaritatea descriptorilor (cont.)**

└→ distanța Earth Mover's Distance (cont.)

> exemplu (cont.) - presupunem ponderi egale, $\Sigma w = \Sigma u$;



*tw*ork = 0.23 * 155 + 0.51 * 252 + 0.26 * 316 = 246

Este corect ca măsură de distanță ?

Nu au fost alese costurile optimale!

[S. Cohen, 1999]

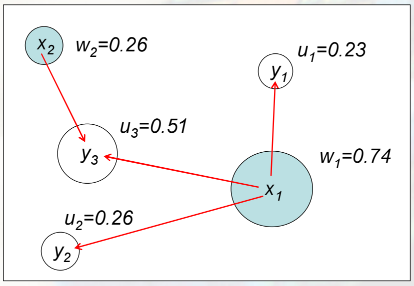
Analiza similarității datelor (cont.)

1. **Similaritatea descriptorilor (cont.)**

twork = 222

└→ distanța Earth Mover's Distance (cont.)

> exemplu (cont.) - presupunem ponderi egale, $\Sigma w = \Sigma u$;



- optimal:

$work_{2,3} = f_{2,3} * d_{2,1} = 0.26 * 198$

$work_{1,1} = f_{1,1} * d_{1,1} = 0.23 * 155$

$work_{1,2} = f_{1,2} * d_{1,2} = 0.26 * 277$

$work_{1,3} = f_{1,3} * d_{1,3} = 0.25 * 252$

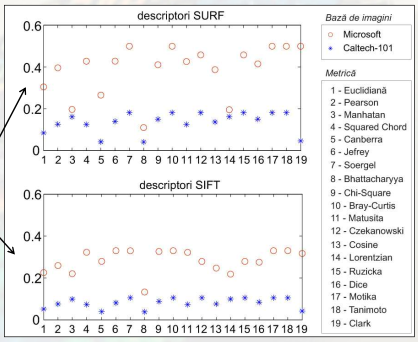
[S. Cohen, 1999]

Analiza similarității datelor (cont.)

1. **Similaritatea descriptorilor (cont.)**

> cât de mult contează alegerea adecvată a măsurii de distanță adaptată datelor?

măsură de performanță (valoare > performanță >; maxim 1 – 100%)



[I. Mironică et al., EUSIPCO 2012]

Analiza similarității datelor (cont.)

2. **Similaritatea la nivel de structură**

determinarea gradului de asemănare a două obiecte la nivel structural (ex. aranjare spațială, structurare text, etc);

> exemplu, compararea a două documente video;

> idee, reprezentare textuală a structurii temporale (vezi M2):

- "s" – plan video;
- "c" – tranziție de tip cut;
- "w" – tranziție de tip wipe;
- "d" – tranziție de tip dissolves;

> documentul video este reprezentat ca o secvență de litere:

"**scswsdcs**"

Analiza similarității datelor (cont.)

2. **Similaritatea la nivel de structură (cont.)**

> exemplu, compararea a două documente video (cont.);

└→ distanța de editare

costul minim de transformare a instanței X_i în instanța X_j, unde X_i și X_j au *n* și respectiv *m* caractere ce pot lua valori într-un alfabet Σ iar *E* definește setul de operații de editare și costurile acestora.

X_i="scswsdcs"
X_j="sdswscscs"

$d(X_i,X_j)$= 2 înlocuiri + 1 inserare = 1 + 1 + 1 = 3

Σ={c,w,d,s}

E={"inserare","ștergere","înlocuire"} (costuri egale, 1)

Analiza similarității datelor (cont.)

3. **Similaritatea semantică**

determinarea gradului de asemănare la nivel de concepte (reprezentare semantică a informației);

> ontologii de informații:

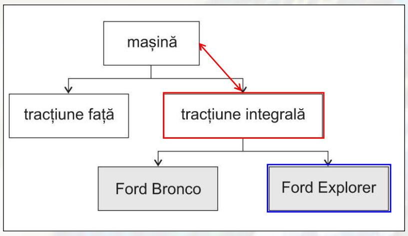
- mod formal de reprezentare a cunoașterii sub formă de concepte și a relațiilor dintre acestea;

- folosesc următoarele componente:
  - obiecte/instanțe de date;
  - clase (mulțimi, colecții, concepte);
  - atribute (proprietăți);
  - relații (între clase și instanțe);
  - restricții;
  - reguli (de tip "if-then");
  - evenimente (modul de schimbare al atributelor).

Analiza similarității datelor (cont.)

3. **Similaritatea semantică (cont.)**

> ontologii de informații (cont.):



ontologie clasă "mașină" (simplificat)

- structură irarhică de reprezentare a informației;

- clase subordonate moștenesc proprietățile claselor superioare;

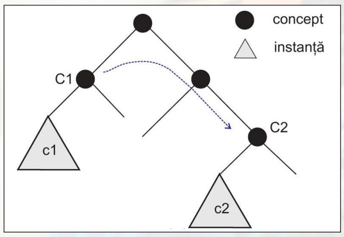
- obiectele sunt descrise de atribute, exemplu:
  -nume "Ford Explorer";
  -ușă (4);
  -motor (4l);
  -transmisie (6v).

Analiza similarității datelor (cont.)

3. **Similaritatea semantică (cont.)**

> ontologii de informații (cont.):

↳ distanța între concepte



ontologie

- datele sunt reprezentate pe bază de ontologii semantice;

- exemplu: distanța dintre instanța c_1 și respectiv c_2 (descriptori) = numărul de pași din arbore necesari pentru a ajunge de la conceptul C_1 (definește c_1) la conceptul C_2 (definește c_2);

- număr de pași = număr de laturi (3 în exemplu).

Clasificarea ierarhică (hierarchical clustering)

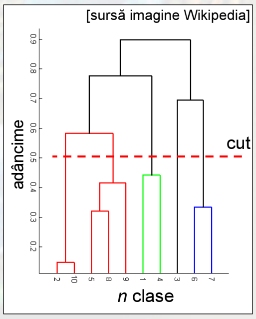
datele de intrare sunt regrupate în funcție de similaritatea acestora într-un număr variabil de clase (1-n), sub formă arborescentă;

- complexitate de calcul redusă;

- numărul de clase rezultate poate fi adaptat în funcție de scenariu;

- aglomerativ sau "**bottom up**" – de jos în sus (în figura de alături);

- diviziv sau "**top down**" – de sus în jos (în figura de alături).



Clasificarea ierarhică (hierarchical clustering; cont.)

1. **Aglomerativă**

> date de intrare, $X_i=[x_{i,1},\ldots,x_{i,n}]$, $i=1,\ldots,m$;

> algoritm:

p1. fiecare dintre instanțe este asociată unei clase -> X_1 є clasa$_1$, …, X_m є clasa$_m$;

p2. se calculează pentru fiecare pereche de clase o măsură de similaritate între acestea;

$$\begin{array}{c|cccc} & \text{clasa}_1 & \text{clasa}_2 & \ldots & \text{clasa}_m \\ \hline \text{clasa}_1 & 0 & d(1,2) & \ldots & d(1,m) \\ & d(2,1) & 0 & \ldots & d(2,m) \\ \vdots & \ldots & \ldots & \ldots & \ldots \\ \text{clasa}_m & d(m,1) & \ldots & d(m,m-1) & 0 \end{array}$$

Clasificarea ierarhică (hierarchical clustering; cont.)

1. **Aglomerativă (cont.)**

> algoritm (cont.):

p3. cele mai similare două clase sunt fuzionate într-una singură;

$$\begin{array}{c|cccc} & \text{clasa}_{1\&2} & \text{clasa}_3 & \ldots & \text{clasa}_m \\ \hline \text{clasa}_{1\&2} & 0 & d(1\&2,3) & \ldots & d(1\&2,m) \\ & d(3,1\&2) & 0 & \ldots & d(3,m) \\ \vdots & \ldots & \ldots & \ldots & \ldots \\ \text{clasa}_m & d(m,1\&2) & \ldots & d(m,m-1) & 0 \end{array}$$

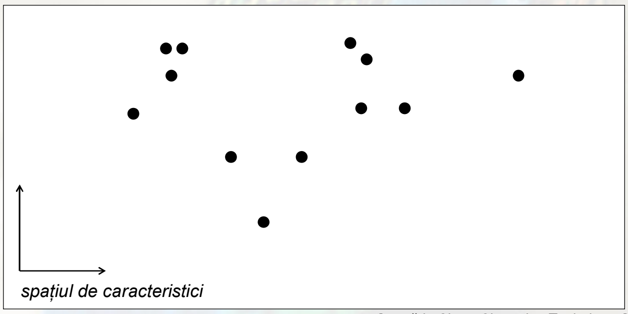
p4. dacă numărul de clase obținute <> 1, mergi la pasul 2 (se re-calculează similaritatea între noile clase și se fuzionează în continuare);

p5. STOP -> dendrograma claselor.

Clasificarea ierarhică (hierarchical clustering; cont.)

1. **Aglomerativă (cont.)**

> exemplu:

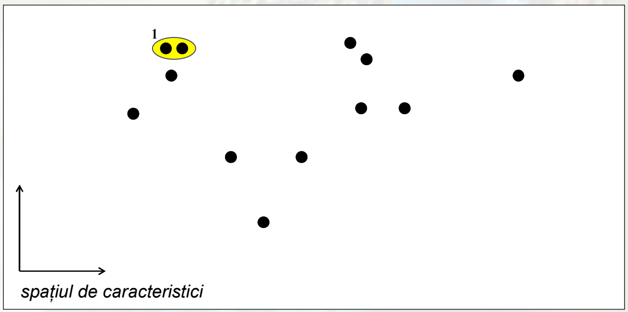


[sursă L. Shan, Clustering Techniques]

Clasificarea ierarhică (hierarchical clustering; cont.)

1. **Aglomerativă (cont.)**

   > exemplu (cont.): iterația 1

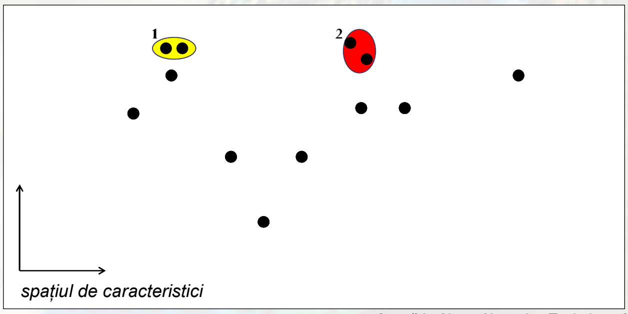


[sursă L. Shan, Clustering Techniques]

Clasificarea ierarhică (hierarchical clustering; cont.)

1. **Aglomerativă (cont.)**

   > exemplu (cont.): iterația 2

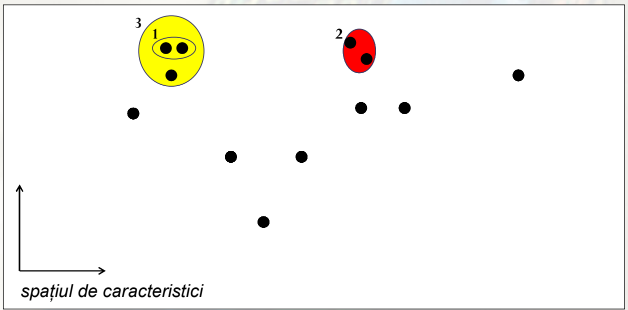


[sursă L. Shan, Clustering Techniques]

Clasificarea ierarhică (hierarchical clustering; cont.)

1. **Aglomerativă (cont.)**

   > exemplu (cont.): iterația 3

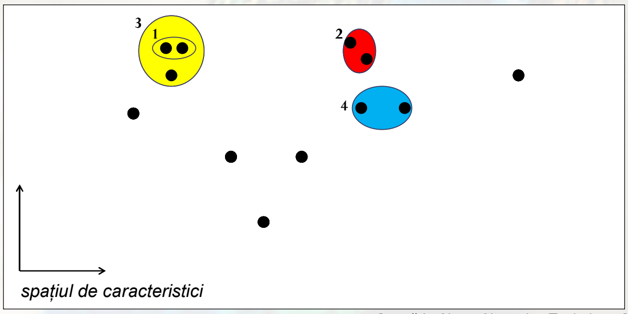


[sursă L. Shan, Clustering Techniques]

Clasificarea ierarhică (hierarchical clustering; cont.)

1. **Aglomerativă (cont.)**

   > exemplu (cont.): iterația 4

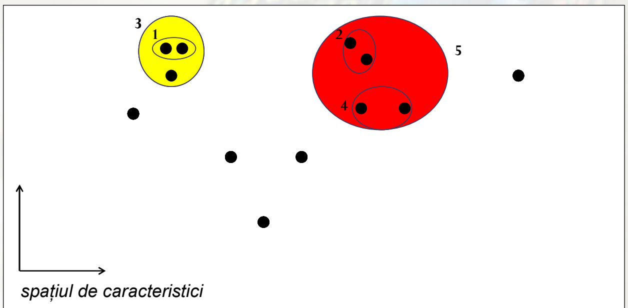


[sursă L. Shan, Clustering Techniques]

Clasificarea ierarhică (hierarchical clustering; cont.)

1. **Aglomerativă (cont.)**

   > exemplu (cont.): iterația 5

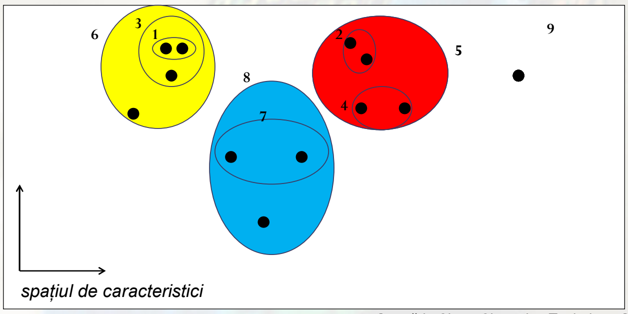


[sursă L. Shan, Clustering Techniques]

Clasificarea ierarhică (hierarchical clustering; cont.)

1. **Aglomerativă (cont.)**

   > exemplu (cont.): iterația *k*

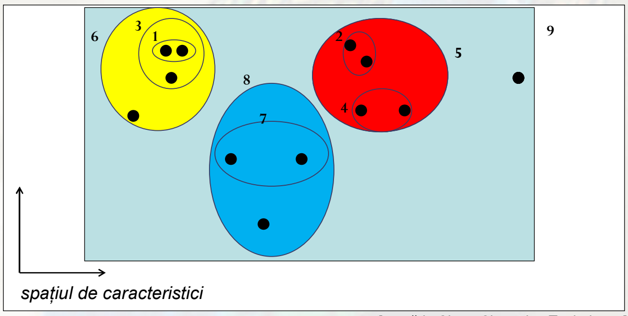


[sursă L. Shan, Clustering Techniques]

Clasificarea ierarhică (hierarchical clustering; cont.)

1. **Aglomerativă (cont.)**

> exemplu (cont.): iterația 11

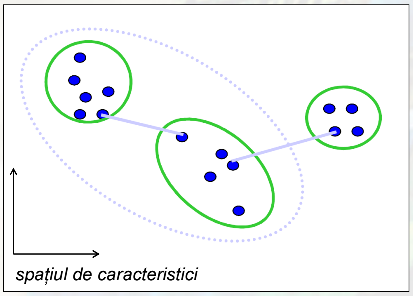


[sursă L. Shan, Clustering Techniques]

Clasificarea ierarhică (hierarchical clustering; cont.)

1. **Aglomerativă (cont.)**

> cum evaluăm similaritatea între clase?



> *single link* = distanța dintre cele mai apropiate două instanțe ale claselor;

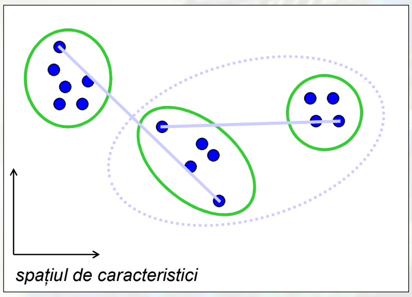
-> clasele rezultate tind să fie subțiri și lungi.

[sursă H. Lin, 15-381 Artificial Intelligence]

Clasificarea ierarhică (hierarchical clustering; cont.)

1. **Aglomerativă (cont.)**

> cum evaluăm similaritatea între clase? (cont.)



> *complete link* = distanța dintre cele mai depărtate două instanțe ale claselor;

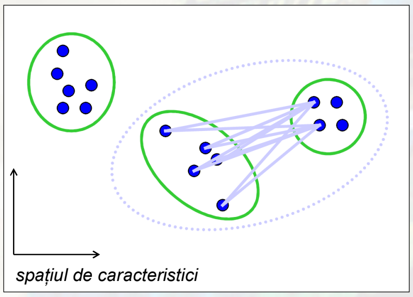
-> clasele rezultate tind să fie foarte apropiate.

[sursă H. Lin, 15-381 Artificial Intelligence]

Clasificarea ierarhică (hierarchical clustering; cont.)

1. **Aglomerativă (cont.)**

> cum evaluăm similaritatea între clase? (cont.)



> *average link* = distanța medie dintre toate instanțele celor două clase;

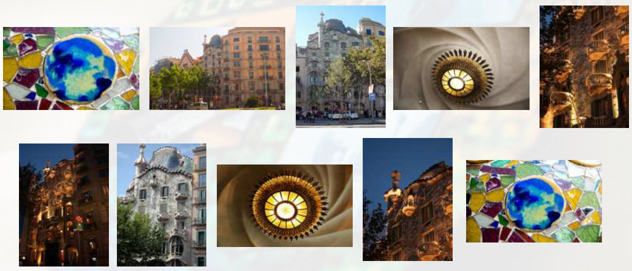
-> robustețe la zgomot.

[sursă H. Lin, 15-381 Artificial Intelligence]

Clasificarea ierarhică (hierarchical clustering; cont.)

1. **Aglomerativă (cont.)**

> exemplu real (imagini, descriptor - culoare, metrică - Euclidiană, similaritate - average link): 10 clase

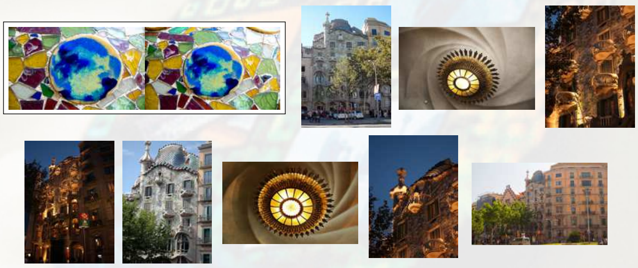


[credit B. Boteanu, 2015; sursă imagini Flickr, “Casa Batllo”]

Clasificarea ierarhică (hierarchical clustering; cont.)

1. **Aglomerativă (cont.)**

> exemplu real (imagini, descriptor - culoare, metrică - Euclidiană, similaritate - average link; cont.) - 9 clase



[credit B. Boteanu, 2015; sursă imagini Flickr, “Casa Batllo”]

Clasificarea ierarhică (hierarchical clustering; cont.)

1. **Aglomerativă (cont.)**

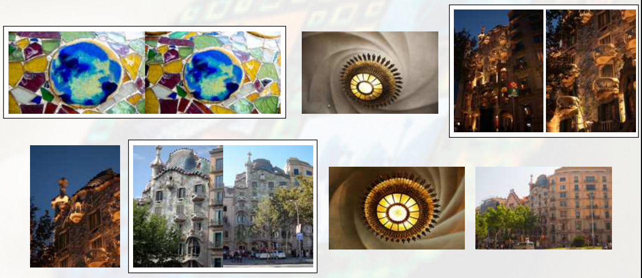
> exemplu real (imagini, descriptor - culoare, metrică - Euclidiană, similaritate - average link; cont.) - 8 clase

[credit B. Boteanu, 2015; sursă imagini Flickr, “Casa Batllo”]

Clasificarea ierarhică (hierarchical clustering; cont.)

1. **Aglomerativă (cont.)**

> exemplu real (imagini, descriptor - culoare, metrică - Euclidiană, similaritate - average link; cont.) - 7 clase



[credit B. Boteanu, 2015; sursă imagini Flickr, “Casa Batllo”]

Clasificarea ierarhică (hierarchical clustering; cont.)

1. **Aglomerativă (cont.)**

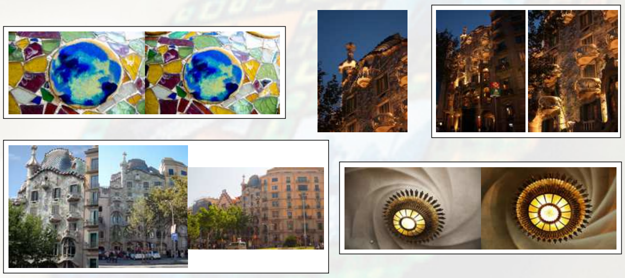
> exemplu real (imagini, descriptor - culoare, metrică - Euclidiană, similaritate - average link; cont.) - 6 clase

[credit B. Boteanu, 2015; sursă imagini Flickr, “Casa Batllo”]

Clasificarea ierarhică (hierarchical clustering; cont.)

1. **Aglomerativă (cont.)**

> exemplu real (imagini, descriptor - culoare, metrică - Euclidiană, similaritate - average link; cont.) - 5 clase

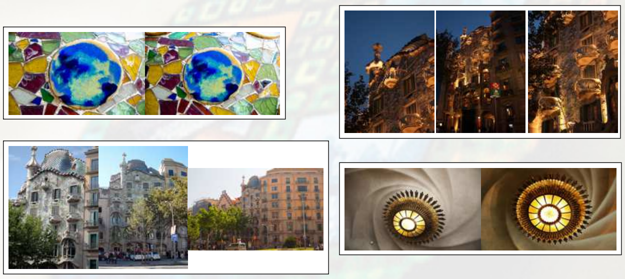


[credit B. Boteanu, 2015; sursă imagini Flickr, “Casa Batllo”]

Clasificarea ierarhică (hierarchical clustering; cont.)

1. **Aglomerativă (cont.)**

> exemplu real (imagini, descriptor - culoare, metrică - Euclidiană, similaritate - average link; cont.) - 4 clase

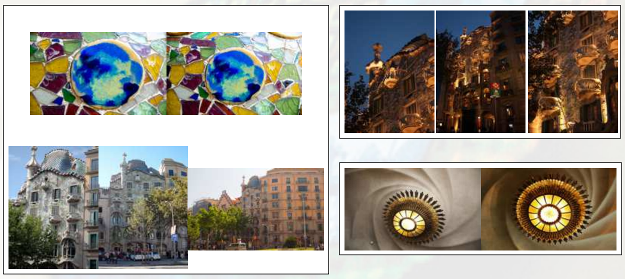


[credit B. Boteanu, 2015; sursă imagini Flickr, “Casa Batllo”]

Clasificarea ierarhică (hierarchical clustering; cont.)

1. **Aglomerativă (cont.)**

> exemplu real (imagini, descriptor - culoare, metrică - Euclidiană, similaritate - average link; cont.) - 3 clase



[credit B. Boteanu, 2015; sursă imagini Flickr, “Casa Batllo”]

Clasificarea ierarhică (hierarchical clustering; cont.)

1. **Aglomerativă (cont.)**

> exemplu real (imagini, descriptor - culoare, metrică - Euclidiană, similaritate - average link; cont.) - 2 clase

[credit B. Boteanu, 2015; sursă imagini Flickr, "Casa Batllo"]

Clasificarea ierarhică (hierarchical clustering; cont.)

2. **Divizivă**

> date de intrare, $X_i=[x_{i,1},\ldots,x_{i,n}]$, $i=1,\ldots,m$;

> algoritm:

p1. toate instanțele sunt asociate unei singure clase -> $X_1,\ldots, X_m$ є clasa$_1$;

p2. clasele curente sunt divizate în două subclase folosind orice algoritm de partiționare;

p3. dacă numărul de clase <> *m* se repetă pasul 2;

p4. STOP -> dendrograma claselor.

Clasificarea ierarhică (hierarchical clustering; cont.)

> care dintre cele două abordări ("top down" vs. "bottom up") este mai complexă?

= top down pentru că necesită o altă metodă de clustering;

> care dintre cele două abordări tinde să fie mai eficientă?

= top down, complexitate liniară funcție de numărul de clase (folosind k-means pentru partiționare);

vs. bottom up, cel puțin pătratică.

> care dintre cele două abordări tinde să fie mai precisă?

- bottom up – deciziile de agreagare sunt luate local fără a ține cont de distribuția globală (deciziile inițiale nu mai pot fi schimbate ulterior);

- top down – țin cont de distribuția globală.

k-means

partiționarea iterativă a datelor în *k* clase în funcție de proximitatea acestora față de reprezentanții claselor (centroizi);

> date de intrare:

- instanțe de clasificat în *k* clase:

$$X = \{X_1, X_2, \ldots, X_m\} \rightarrow c_1, \ldots, c_k;$$

- un dicționar de *k* instanțe:

$$V = \{V_1, V_2, \ldots, V_k\}$$

- o matrice de partiționare:

$$\Gamma = [\gamma_{l,i}],\ \gamma_{l,i} = \begin{cases} 1 & X_i \in c_l \\ 0 & altfel \end{cases}$$

k-means (cont.)

$$optimizare\ \ E(\Gamma,V) = \sum_{l=1}^{k}\sum_{i=1}^{m}\gamma_{li}\,\|X_i - V_l\|^2$$

> algoritm:

p1. se alege o valoare pentru *k* (numărul de clase);

p2. se inițializează vocabularul *V* cu *k* instanțe din datele de intrare *X*. Acestea definesc o partiție inițială a claselor (centroizi);

p3. fiecare instanță este atribuită clasei celei mai apropiate (ca distanță față de centroidul clasei);

p4. se calculează matricea Γ de partiționare în clase;

p5. se re-calculează vocabularul, fiecare vector fiind înlocuit de centroidul (= media) clasei respective;

p6. se reia pasul 3 până când nici o instanță nu-și mai schimbă apartenența la clase (Γ nu se modifică).

k-means (cont.)

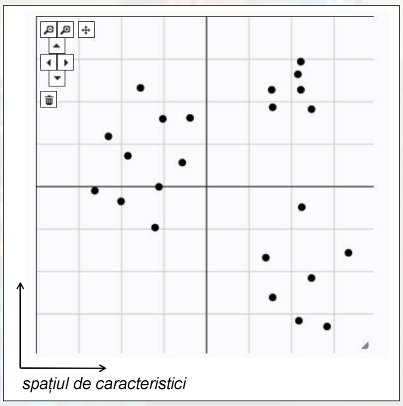
[sursă http://util.io/k-means]

> exemplu:

$$X = \{X_1, \ldots, X_{23}\}$$

$$k = 3$$

$$c_1, c_2, c_3$$



[sursă http://util.io/k-means]

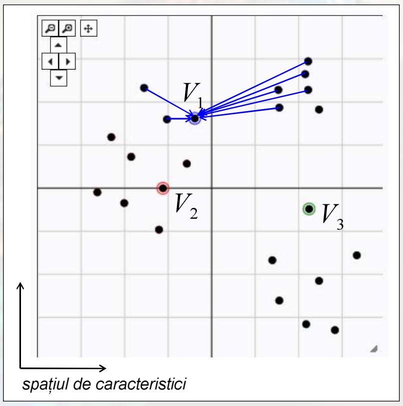
k-means (cont.)

> exemplu (cont.):

- se alege vocabularul din instanțele de intrare;

- acesta definește cele 3 clase;

- instanțele sunt asociate claselor cele mai apropiate.

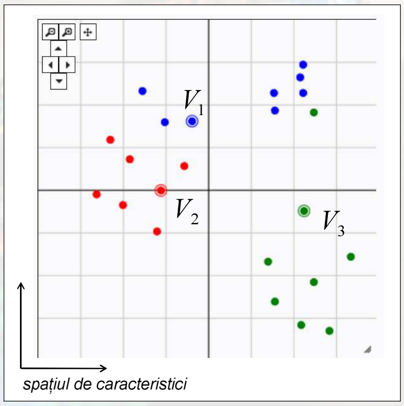


[sursă http://util.io/k-means]

k-means (cont.)

> exemplu (cont.):

- se recalculează vectorii *V* pentru fiecare clasă ca fiind centroizii claselor curente (medie);

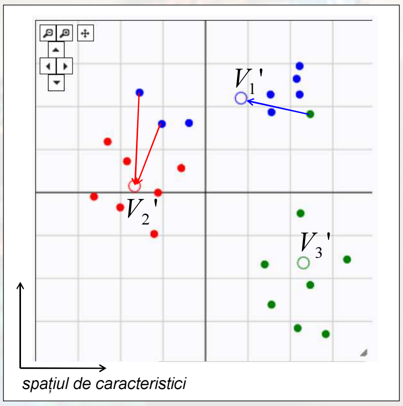


[sursă http://util.io/k-means]

k-means (cont.)

> exemplu (cont.):

- instanțele sunt asociate claselor celor mai apropiate pe baza noilor centroizi;

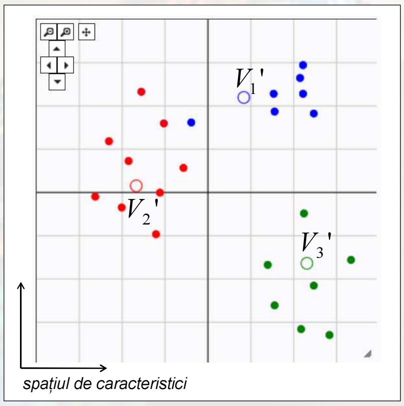


[sursă http://util.io/k-means]

k-means (cont.)

> exemplu (cont.):

- se repetă pașii anteriori ...;

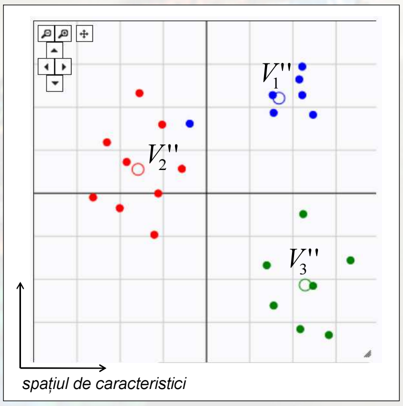


[sursă http://util.io/k-means]

k-means (cont.)

> exemplu (cont.):

- etc;

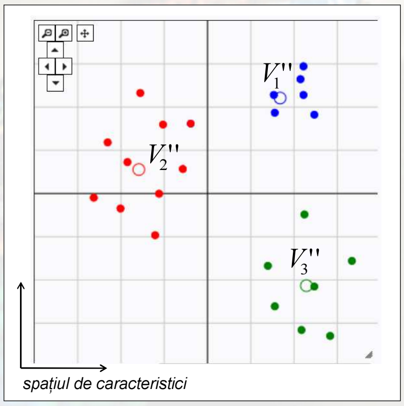


[sursă http://util.io/k-means]

k-means (cont.)

> exemplu (cont.):

- etc;

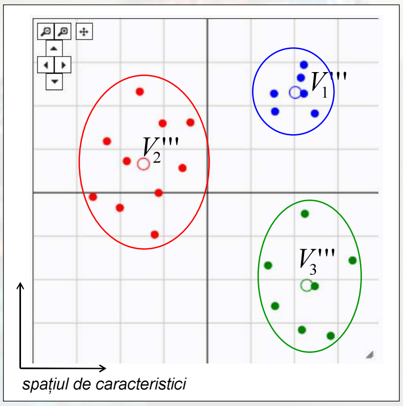


k-means (cont.)

[sursă http://util.io/k-means]

> exemplu (cont.):

- în acest moment nu se mai schimbă repartiția în clase a instanțelor;



k-means (cont.)

> avantaje:

- simplu de implementat;
- optimizează în mod intuitiv similaritatea intra-clasă;
- relativ eficient, complexitate O(*m* x *k* x *nr.iterații*).

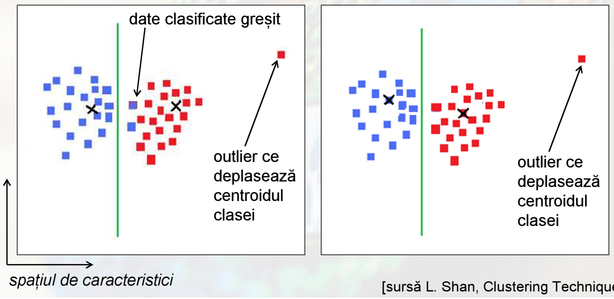
> dezavantaje:

- necesită definirea noțiunii de centroid ca medie instanțe;
- optimizare locală – depinde practic de alegerea (bună) a vocabularului inițial pentru clase;
- numărul de clase trebuie anticipat;
- sensibil la date atipice;
- nu este eficient pentru clustere cu forme non-convexe.

k-means (cont.)

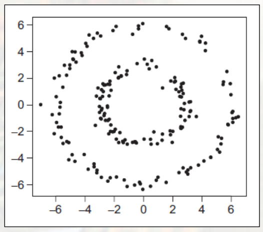
> date atipice (*outliers*):

- potențială soluție: *k-medoids*, centrele claselor sunt alese ca fiind chiar unele dintre instanțe și nu mediile;



[sursă L. Shan, Clustering Techniques]

k-means (cont.)



> clustere cu forme non-convexe?

- potențială soluție: *kernel trick*;
- funcția de cost standard de minimizat este:

$$E(\Gamma,V)=\sum_{l=1}^{k}\sum_{i=1}^{m}\gamma_{li}\,\|X_i-V_l\|^2$$

- idee: transformăm *X* printr-o funcție (nucleu – kernel):

$$E(\Gamma,V)=\sum_{l=1}^{k}\sum_{i=1}^{m}\gamma_{li}\,\|\varphi(X_i)-\varphi(V_l)\|^2$$

$\varphi(V_l)=\frac{1}{m_l}\sum_{i=1}^{m}\gamma_{li}\varphi(X_i)$, m_l este numărul de instanțe din clasa *l*.

k-means (cont.)

> clustere cu forme non-convexe (cont.)

- potențială soluție: *kernel trick* (cont.);

$$E(\Gamma,V)=\sum_{l=1}^{k}\sum_{i=1}^{m}\gamma_{li}\,\|\varphi(X_i)-\varphi(V_l)\|^2$$

$$\|\varphi(X_i)-\varphi(V_l)\|^2=\varphi(X_i)^T\cdot\varphi(X_i)-\varphi(X_i)^T\cdot\varphi(V_l)-$$
$$-\varphi(V_l)^T\cdot\varphi(X_i)+\varphi(V_l)^T\cdot\varphi(V_l)$$

- funcția nucleu este dată de: $\varphi(X_i)^T\cdot\varphi(X_j)=K(X_i,X_j)$

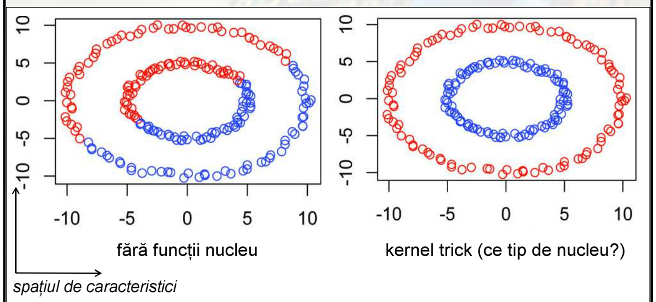
$$\|\varphi(X_i)-\varphi(V_l)\|^2=K(X_i,X_i)-2K(X_i,V_l)+K(V_l,V_l)$$

- exemple de nuclee: $K(X_i,X_j)=e^{-q\|X_i-X_j\|}$ (Gaussian); $K(X_i,X_j)=(c+X_i^T\cdot X_j)^d$ (polinomial).

k-means (cont.)

> clustere cu forme non-convexe (cont.)

- potențială soluție: *kernel trick* (cont.);

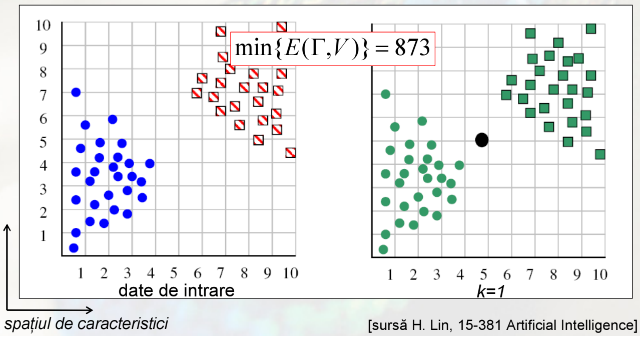


[sursă R. Chitta, Kernel K-Means]

k-means (cont.)

> cum putem decide în mod "automat" numărul de clase?

> idee: pentru setul de date, încercăm mai multe valori pentru *k*:

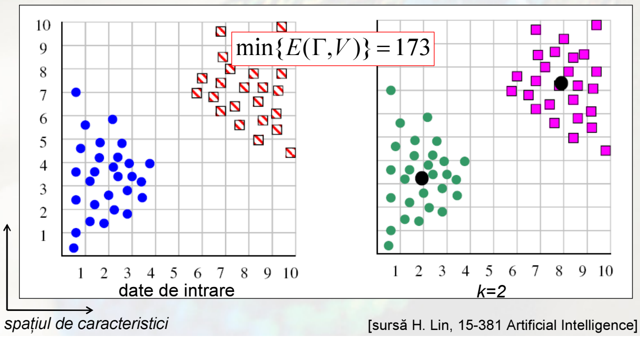


[sursă H. Lin, 15-381 Artificial Intelligence]

k-means (cont.)

> cum putem decide în mod "automat" numărul de clase (cont.)

> idee: pentru setul de date, încercăm mai multe valori pentru *k*:

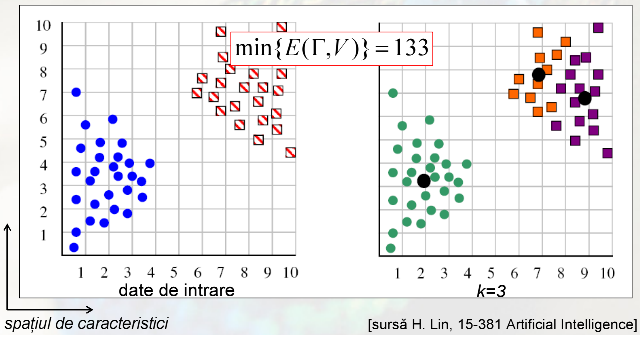


[sursă H. Lin, 15-381 Artificial Intelligence]

k-means (cont.)

> cum putem decide în mod "automat" numărul de clase (cont.)

> idee: pentru setul de date, încercăm mai multe valori pentru *k*:

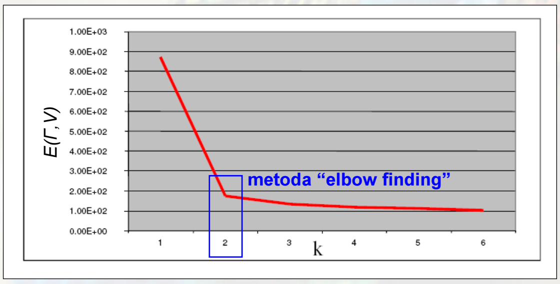


[sursă H. Lin, 15-381 Artificial Intelligence]

k-means (cont.)

> cum putem decide în mod "automat" numărul de clase (cont.)

> idee: pentru setul de date, încercăm mai multe valori pentru *k*:



[sursă H. Lin, 15-381 Artificial Intelligence]

Gaussian Mixture Models

abordare bazată pe modele; clasele sunt considerate a avea distribuții Gaussiene ale căror parametri sunt optimizați astfel încât să se potrivească cel mai bine datelor;

• *funcția de repartiție:*

$$F_X(x) = P\{X \le x\}$$

unde *X* este o variabilă aleatoare, *x* reprezintă o valoare iar *P{}* reprezintă probabilitatea în sensul statistic.

> reprezintă probabilitatea ca realizarea particulară a variabilei aleatoare *X* să fie mai mică sau egală decât *x*.

$$0 \le F_X(x) \le 1$$

Gaussian Mixture Models (cont.)

• *funcția de densitate de probabilitate:*

$$f_X(x) = \frac{dF_X(x)}{dx},\ f_X(x) \ge 0$$

unde *d/dx* reprezintă derivata de ordinul 1.

$$P\{X \le x\} = F_X(x) = \int_{-\infty}^{x} f_X(t)dt$$

> aria de sub graficul format de densitatea de probabilitate.

$$P\{x_1 < X \le x_2\} = \int_{x_1}^{x_2} f_X(t)dt$$

$$P\{X \approx x\} = f_X(x)dx = P\{x < X \le x + dx\}$$

Gaussian Mixture Models (cont.)

[sursă imagine Wikipedia]

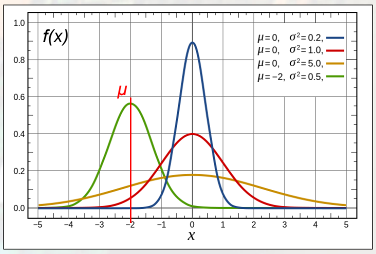
• *densitate de probabilitate normală, Gaussiană (1D):*

$$N(X;\mu,\sigma^2): f(x)=\frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

unde μ reprezintă media valorilor și σ este abaterea pătratică medie.

> 68% din valori sunt în intervalul [μ-σ; μ+σ];

> 99% din valori sunt în intervalul [μ-3σ; μ+3σ].



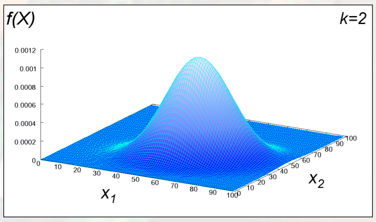
Gaussian Mixture Models (cont.)

[sursă imagine Wikipedia]

• *densitate de probabilitate normală, Gaussiană (nD):*

$$N(X;\mu,\Sigma): f(x_1,\ldots,x_k)=\frac{1}{\sqrt{(2\pi)^k\det(\Sigma)}}e^{-\frac{1}{2}(X-\mu)^T\Sigma^{-1}(X-\mu)}$$

unde X=[x_1,…,x_k] reprezintă o variabilă aleatoare k dimensională, μ=[μ_1,…, μ_k] reprezintă vectorul medie (μ_i este media lui x_i), Σ este matricea de covarianță (dimensiune k x k), *^{T} reprezintă transpusa, $^{-1}$ reprezintă inversa iar det() returnează determinantul.*



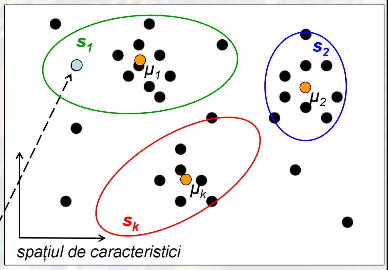
Gaussian Mixture Models (cont.)

> ipoteza GMM:

- se presupune faptul că avem la îndemână k surse de date;

- fiecare sursă i generează date de medie μ_i și matrice de covarianță Σ_i (distribuție Gaussiană);

- astfel, pentru o sursă i, de probabilitate p_i, datele generate de aceasta au distribuție *~N(μ_i,Σ_i).*



[Andrew W. Moore]

Gaussian Mixture Models (cont.)

> *clasificator*: determinarea optimală a acestor distribuții ce se potrivesc cel mai bine repartiției datelor de intrare în spațiul de caracteristici (amestec de Gaussiene - GMM);

> *optimizare* = algoritm Expectation-Maximization (EM);

> date de intrare:

- instanțele de clasificat în k clase:

$$X=\{X_1,X_2,\ldots,X_m\}\rightarrow c_1,\ldots,c_k;$$

- probabilitățile celor k surse:

$$p_1,\ldots,p_k$$

- valorile medii și matricele de covarianță:

$$\mu_1,\ldots,\mu_k,\Sigma_1,\ldots,\Sigma_k$$

Gaussian Mixture Models (cont.)

> algoritm GMM + EM:

p1. se alege numărul de surse k (= numărul de clase);

p2. se inițializează parametrii de intrare, p_i, μ_i, Σ_i cu i=1,…,k (ex. valori aleatorii);

$$\lambda=\{\mu_1,\ldots,\mu_k,\Sigma_1,\ldots,\Sigma_k,p_1,\ldots,p_k\}$$

p3. sunt calculate clasele estimate (Expectation-step):

$$P\{c_i\mid X_j,\lambda\}=\frac{\boxed{P\{X_j\mid c_i,\lambda\}\cdot P\{c_i\mid\lambda\}}}{P\{X_j\mid\lambda\}}$$

se eval. N(X_j;μ_i,Σ_i)

$$P\{X_j\mid c_i,\lambda\}\cdot P\{c_i\mid\lambda\}=P\{X_j\mid c_i,\mu_i,\Sigma_i\}\cdot p_i$$

Gaussian Mixture Models (cont.)

> algoritm GMM + EM (cont.):

p3. sunt calculate clasele estimate (Expectation-step; cont):

$$P\{c_i\mid X_j,\lambda\}=\frac{P\{X_j\mid c_i,\lambda\}\cdot P\{c_i\mid\lambda\}}{\boxed{P\{X_j\mid\lambda\}}}$$

$$P\{X_j\mid c_i,\lambda\}\cdot P\{c_i\mid\lambda\}=P\{X_j\mid c_i,\mu_i,\Sigma_i\}\cdot p_i$$

$$P\{X_j\mid\lambda\}=\sum_{i=1}^{k}P\{X_j\mid c_i,\mu_i,\Sigma_i\}\cdot p_i$$

se eval. N(X_j;μ_i,Σ_i)

Gaussian Mixture Models (cont.)

> algoritm GMM + EM (cont.):

p4. sunt maximizate mediile și sunt recalculați parametrii (Maximization-step):

$$\mu_i = \frac{\sum_{j=1}^{m} P\{c_i \mid X_j, \lambda\} \cdot X_j}{\sum_{j=1}^{m} P\{c_i \mid X_j, \lambda\}}$$

$$\Sigma_i = \frac{\sum_{j=1}^{m} P\{c_i \mid X_j, \lambda\}[X_j - \mu_i][X_j - \mu_i]^T}{\sum_{j=1}^{m} P\{c_i \mid X_j, \lambda\}}$$

Gaussian Mixture Models (cont.)

> algoritm GMM + EM (cont.):

p4. sunt maximizate mediile și sunt recalculați parametrii (Maximization-step; cont.):

$$p_i = \frac{\sum_{j=1}^{m} P\{c_i \mid X_j, \lambda\}}{m}$$

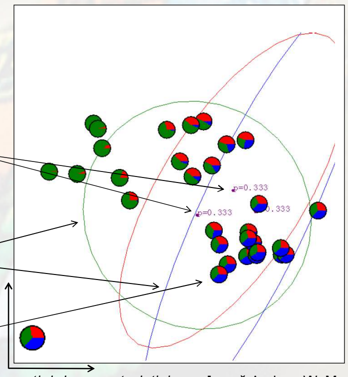
p5. dacă parametrii de intrare, în urma actualizării, se schimbă foarte puțin -> STOP;

p6. altfel se repetă procesul cu pasul 3.

Gaussian Mixture Models (cont.)

> exemplu: iterația 0

- inițializare:
  - *k*=3;
  - probabilități surse egale (0.33);
  - medii;
  - matrice de covarianță.
- calcul probabilități de aparteneță la distribuții;

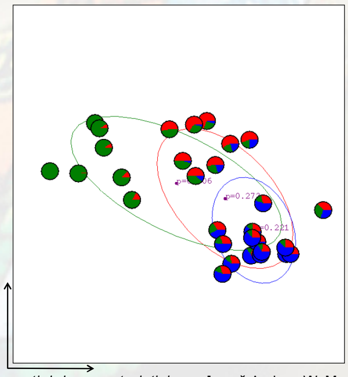


spațiul de caracteristici [sursă Andrew W. Moore]

Gaussian Mixture Models (cont.)

> exemplu: iterația 1

- în urma actualizării se redistribuie probabilitățile claselor, medii și matricea de covarianță ...

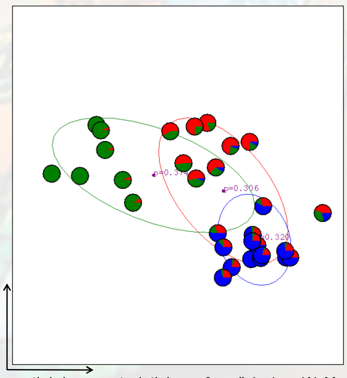


spațiul de caracteristici [sursă Andrew W. Moore]

Gaussian Mixture Models (cont.)

> exemplu: iterația 2

- în urma actualizării se redistribuie probabilitățile claselor, medii și matricea de covarianță ...

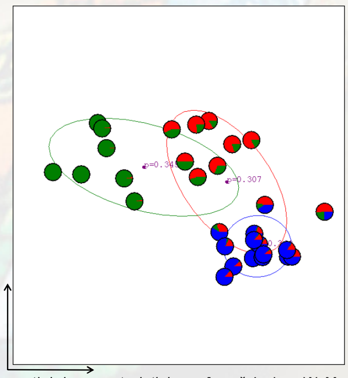


spațiul de caracteristici [sursă Andrew W. Moore]

Gaussian Mixture Models (cont.)

> exemplu: iterația 3

- în urma actualizării se redistribuie probabilitățile claselor, medii și matricea de covarianță ...

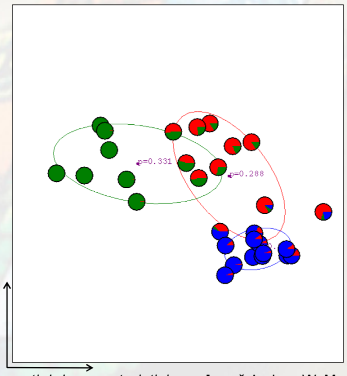


spațiul de caracteristici [sursă Andrew W. Moore]

Gaussian Mixture Models (cont.)

> exemplu: iterația 4

- în urma actualizării se redistribuie probabilitățile claselor, medii și matricea de covarianță ...

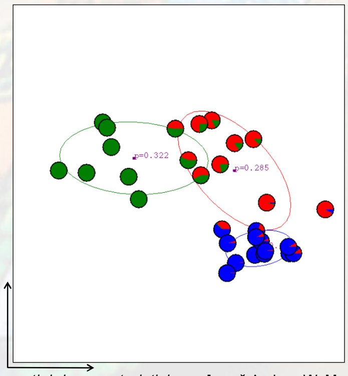


[sursă Andrew W. Moore]

Gaussian Mixture Models (cont.)

> exemplu: iterația 5

- în urma actualizării se redistribuie probabilitățile claselor, medii și matricea de covarianță ...

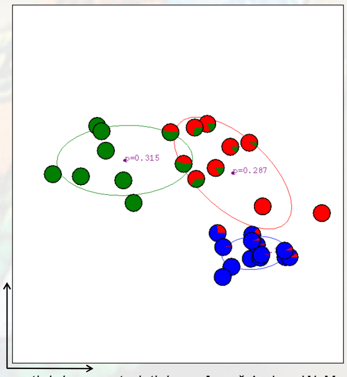


[sursă Andrew W. Moore]

Gaussian Mixture Models (cont.)

> exemplu: iterația 6

- în urma actualizării se redistribuie probabilitățile claselor, medii și matricea de covarianță ...



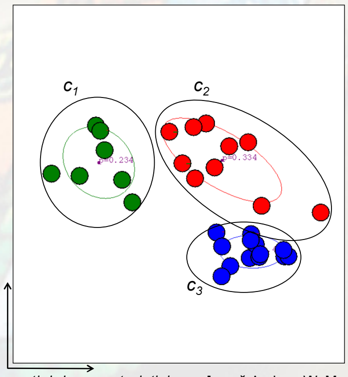
[sursă Andrew W. Moore]

Gaussian Mixture Models (cont.)

> exemplu: iterația 20

- în urma actualizării se redistribuie probabilitățile claselor, medii și matricea de covarianță ...

- rezultă repartiția optimală în clase de distribuție normală.



[sursă Andrew W. Moore]

Gaussian Mixture Models (cont.)

> avantaje:

- interpretabilitate: determină un model de generare a datelor (se pot genera date noi);
- relativ eficient, complexitate O(m x k x *nr.iterații*);
- extensibil la alt tip de distribuții de date.

> dezavantaje:

- EM conduce de regulă la un minim local – depinde de inițializare;
- numărul de clase trebuie determinat a priori;
- mai puțin eficient pentru clase de formă ne convexă.

> Sfârşit M3