Clustering

Intuitiv, *Clustering* - ul poate fi definit ca un proces de organizare a obiectelor, care sunt asemănătoare (similare) dintr-un anumit punct de vedere.
Astfel, putem defini noţiunea de *cluster* ca fiind o mulţime de obiecte asemănătoare între ele şi diferite de obiectele aparţinând altui cluster.

Clustering - ul este o tehnică de învăţare nesupervizată, care determină o structură intrinsecă într-o mulţime de date.

Avem următorul criteriu de similaritate:

Două sau mai multe obiecte aparţin aceluiaşi cluster dacă sunt foarte apropiate, relativ la distanţa considerată.

Folosind această tehnică, putem rezolva următoarele probleme:

- reducerea datelor (*data reduction*), găsind reprezentanţii unui grup omogen;
- determinarea unor clase convenabile („*useful data classes*”);
- determinarea unor clustere naturale şi descrierea proprietăţilor lor necunoscute („*natural data types*”);
- găsirea unor obiecte mai rar întâlnite (*outlier detection*)

Nu există un criteriu absolut care să decidă dacă un proces de clustering este reuşit sau nu, totul depinzând de scopul urmărit.

Algoritmii de clustering sunt aplicaţi în:

- *marketing*, pentru găsirea grupurilor de clienţi cu profil asemănător, pe baza unei baze de date a acestor clienţi, cu atributele lor şi cumpărăturile lor anterioare;
- *biologie*, pentru clasificarea plantelor şi animalelor;
- *asigurări*, pentru identificarea clienţilor ce prezintă riscuri mari şi pentru identificarea fraudelor;
- *biblioteci*, pentru ordonarea cărţilor după anumite criterii;
- *internet,* pentru împărţirea documentelor în funcţie de tematică.

Măsura de similaritate

Măsura de similaritate poate fi considerată ca fiind o metrică definită pe o anumită mulţime *M*.
Este aleasă în concordanţă cu tipul de date cu care se lucrează şi cu scopul propus.

Vom prezenta câteva măsuri de similaritate, uzual întâlnite.
Pentru a măsura similaritatea a două obiecte, le vom considera ca vectori:

$$\mathbf{x} = (x_1, x_2, ..., x_n) \text{ şi } \mathbf{y} = (y_1, y_2, ..., y_n).$$

distanţa *Minkowski*

distanţa *Minkowski* este definită de formula:

$$d(\mathbf{x}, \mathbf{y}) = \left(\sum_{i=1}^{n} | x_i - y_i |^p \right)^{\frac{1}{p}}, \quad p \in \mathbf{N}^*$$

Se observă că:

- dacă $p = 1$ obţinem distanţa *Manhattan* sau *cityblock*; distanţa Manhattan pentru vectori binari devine distanţa *Hamming*, distanţă între două coduri ce măsoară numărul de biţi diferiţi;
- dacă $p = 2$ avem distanţa *euclidiană*;
- dacă $p \to \infty$ obţinem distanţa *Cebâşev*.

y = pdist(X)

y = pdist(X) calculează distanţa euclidiană între perechi de obiecte dintr-o matrice cu n linii şi p coloane.
Liniile matricei corespund observaţiilor (obiectelor) în timp ce coloanele corespund variabilelor (atributelor). y este un vector de dimensiune $C_n^2 = \frac{n \cdot (n-1)}{2}$, corespunzător numărului de perechi de observaţii din X.

Distanţele sunt calculate în următoarea ordine:

(2,1), (3,1), …,(n,1), (3,2),…,(n,2),…,(n,n-1).

Din motive de economie de spaţiu şi de timp de calcul y este prezentat ca un vector. Dacă dorim să-l prezentăm ca o matrice pătratică folosim funcţia squareform şi astfel elementul (i,j) al matricei, unde $i<j$ corespunde distanţei dintre elementele i şi j din mulţimea iniţială

1. Exemplu

```
>>X=[162 255 74    258;210 324 93      220;259 208 115 266;460 1053 145 221;680 381 280 275]
>>  Y = pdist(X)
Y =

  Columns 1 through 9
   94.1807  115.5984  855.5805  571.7736  135.8565  772.4286  511.9990  870.2592  484.2272
  Column 10
  721.8899

>> squareform(Y)
ans =

       0       94.1807  115.5984  855.5805  571.7736
   94.1807       0       135.8565  772.4286  511.9990
  115.5984  135.8565       0       870.2592  484.2272
  855.5805  772.4286  870.2592       0       721.8899
  571.7736  511.9990  484.2272  721.8899       0
```

Pentru ierarhizarea fiecărui atribut în funcţie de scopul propus, pot fi incorporate ponderi şi astfel măsura *Minkowski ponderată* este de forma:

- $$d_{\alpha}(\mathbf{x}, \mathbf{y}) = \left(\sum_{i=1}^{n} \alpha_i \cdot | x_i - y_i |^p \right)^{\frac{1}{p}}, \ \alpha_i > 0, 1 \le i \le n,$$

$\sum_{i=1}^{n} \alpha_i = 1$, unde α_i reprezintă ponderea atributului *i*.

y = pdist(X,metric) calculează distanţa dintre obiectele din matricea X, utilizând metoda specificată de **metric**, care poate fi:

'cityblock'
'hamming'
'chebychev'
'jaccard'
'correlation'
'mahalanobis'

măsura *Jaccard*

- măsura *Jaccard* este definită de formula

$$d_J(\mathbf{x}, \mathbf{y}) = \frac{\sum_{i=1}^{n} x_i y_i}{\sum_{i=1}^{n} x_i^2 + \sum_{i=1}^{n} y_i^2 - \sum_{i=1}^{n} x_i y_i},$$

unde $\mathbf{x} = (x_1, \ldots, x_n)$ și $\mathbf{y} = (y_1, \ldots, y_n)$

măsura *Pearson's r*

- măsura *Pearson's r* (măsura coeficientului de corelaţie – correlation) este definită de formula:

$$r(\mathbf{x},\mathbf{y}) = \frac{\sum_{i=1}^{n}(x_i - \bar{x})\cdot(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2 \cdot \sum_{i=1}^{n}(y_i - \bar{y})^2}},$$

unde $\mathbf{x} = (x_1,\ldots,x_n)$ şi $\mathbf{y} = (y_1,\ldots,y_n)$, iar $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$

Matricea de covarianţă

Considerăm o mulţime de n vectori, priviţi ca un eşantion de dimensiune n a n variabile aleatoare independente . *Matricea de covarianţă* este o matrice pătratică ce defineşte interacţiunile (liniare) dintre aceste variabile

Pentru fiecare pereche (X_k, X_l), covarianţa lor este definită de formula:

$$\operatorname{cov}(X_k, X_l) = \frac{1}{n} \cdot \sum_{i=1}^{n} x_i^k x_i^l - \overline{X}_k \cdot \overline{X}_l$$

Reţinem că: $\operatorname{cov}(X_k, X_k) = \operatorname{var}(X_k)$

Astfel $$\operatorname{cov}\begin{pmatrix} X_1 \\ \vdots \\ X_n \end{pmatrix} = \begin{pmatrix} \operatorname{var}(X_1) & \operatorname{cov}(X_1, X_2) & \dots & \operatorname{cov}(X_1, X_n) \\ \operatorname{cov}(X_2, X_1) & \operatorname{var}(X_2) & \dots & \operatorname{cov}(X_2, X_n) \\ \vdots & \vdots & \vdots & \vdots \\ \operatorname{cov}(X_n, X_1) & \operatorname{cov}(X_n, X_2) & \dots & \operatorname{var}(X_n) \end{pmatrix}$$

cov(x) returnează dispersia, dacă x este vector.

In cazul matricelor, în care fiecare linie reprezintă o observaţie şi fiecare coloană o variabilă cov(X) returnează matricea covarianţei

diag(cov(X)) returnează vectorul ale cărui componente sunt dispersiile fiecărei coloane.
sqrt(diag(cov(X))) returnează vectorul deviaţiilor standard ale fiecărei coloane.

2. Exemplu

Fie matricea $A = \begin{pmatrix} -1 & 1 & 2 \\ -2 & 3 & 1 \\ 4 & 0 & 3 \end{pmatrix}$. Să calculăm vectorul ce are drept componente dispersiile fiecărei coloane din A şi matricea covarianţelor lui A.

```
>> A = [-1 1 2 ; -2 3 1 ; 4 0 3]
>> v = diag(cov(A))
v =
   10.3333
    2.3333
    1.0000

>> C= cov(A)
ans =
   10.3333   -4.1667    3.0000
   -4.1667    2.3333   -1.5000
    3.0000   -1.5000    1.0000
```

Observăm că elementele $C(i,i)$ de pe diagonala principală reprezintă dispersiile coloanelor matricei A. Restul elementelor $C(i,j), i \neq j$ reprezintă covarianţele coloanelor i si j.

măsura *Mahalanobis*

Notând cu cov(A) matricea de covarianţă corespunzătoare eşantionului, măsura *Mahalanobis* este dată de formula:

$$d_M(\mathbf{x}, \mathbf{y}) = d_M(\mathrm{x}, \mathrm{y}) = \sqrt{(\mathrm{x} - \mathrm{y}) \cdot \mathrm{cov}(A)^{-1} \cdot (\mathrm{x} - \mathrm{y})^T}$$

Dacă variabilele eşantionului nu sunt corelate, distanţa Mahalanobis se rezumă la distanţa euclidiană.

3. Exemplu

Pentru a decide dacă un pacient are sau nu cancer hepatic, studiile clinice arată că se iau în considerare anumite enzime serice cele mai importante din punct de vedere clinic: alkaline phosphatase (FA), g-glutamyl transferase (gGT), leucine amino peptidase (LAP) şi colesterol (C).

Prezentăm rezultatelor investigaţiilor pentru 3 pacienţi:

	FA	gCT	LAP	C
P1	430	289	302	148
P2	162	255	74	258
P3	422	284	154	250

Calculăm distanţele menţionate anterior în acest caz:

```
>> A=[430 289 302 148; 162 255 74 258;422 284 154 250];
>> y1=pdist(A)
y1 =
  370.2216  179.9917  273.6878
```

```
>> y2=pdist(A, 'cityblock')
y2 =
  640   263   377

>> y3=pdist(A, 'chebychev')
y3 =
  268   148   260

>> y4=pdist(A, 'jaccard')
y4 =
    1     1     1

>> y5=pdist(A, 'correlation')
y5 =
   1.4949    0.4100    0.6973

>> y6=pdist(A, 'mahalanobis')
Warning: Matrix is singular to working precision.
> In pdist at 161
y6 =
  NaN   NaN   NaN
```

măsuri de similaritate *fuzzy*

Măsurile *fuzzy* sunt utilizate pentru compararea de vectori ale căror componente iau valori în [0,1].
Pentru vectorul $\mathbf{x} = (x_1, x_2, ..., x_n), x_i \in [0,1]$, valoarea lui x_j reprezintă gradul în care vectorul **x** posedă al j – lea atribut (caracteristică).

În scopul definirii măsurilor de similaritate *fuzzy* definim:

$$s(x_i, y_i) = \max\{\min\{x_i, y_i\}, \min\{1 - x_i, 1 - y_i\}\}$$

și astfel, plecând de la cazul clasic, avem de exemplu măsura *fuzzy Minkowski* :

$$d_F(\mathbf{x}, \mathbf{y}) = \left(\sum_{i=1}^{n} s(x_i, y_i)^p \right)^{\frac{1}{p}}.$$

măsură de similaritate *mixtă şi ponderată*

În cele mai multe cazuri, obiectele au ca reprezentare un vector
în care componentele au semnificaţii diferite, având naturi diferite:
numerice, nominale, fuzzy etc.
În această situaţie vom defini o măsura de similaritate *mixtă şi ponderată*,
pentru cuantificarea tipului de date, respectiv semnificaţia atributelor.

De exemplu avem de comparat două obiecte:

$$\mathbf{x} = ((x_1^1,...,x_{k_1}^1),.....,(x_1^p,...,x_{k_p}^p))$$

$$\mathbf{y} = ((y_1^1,...,y_{k_1}^1),.....,(y_1^p,...,y_{k_p}^p))$$

care au p tipuri de date de dimensiuni $k_1,...,k_p$.

Vom pondera cele p secvenţe cu valorile $\alpha_i > 0, \sum_{i=1}^{p} \alpha_i = 1$, în funcţie de importanţa lor în context şi apoi vom aplica fiecărui cuplu de secvenţe $x_j = (x_1^j, ..., x_{k_j}^j)$ şi $y_j = (y_1^j, ..., y_{k_j}^j)$ măsura de similaritate specifică $d_j, 1 \le j \le p$. Astfel:

$$d(\mathbf{x}, \mathbf{y}) = \sum_{j=1}^{p} \alpha_j \cdot d_j(x_j, y_j)$$

Standardizarea caracteristicilor

Standardizarea caracteristicilor este o problemă ce o avem de rezolvat înainte de a calcula măsura de similaritate între obiecte.

Considerăm obiectele $\mathbf{x}_1,\ldots,\mathbf{x}_n$, unde $\mathbf{x}_k = (x_1^k,\ldots,x_p^k)$, care pot fi reprezentate sub forma unei matrice $\mathbf{X}$, cu p linii şi n coloane:

$$\mathbf{X} = \begin{pmatrix} x_1^1 & \ldots & x_i^1 & \ldots & x_p^1 \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ x_1^k & \ldots & x_i^k & \ldots & x_p^k \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ x_1^n & \ldots & x_i^n & \ldots & x_p^n \end{pmatrix}$$

Pentru standardizare putem folosi transformările:

- $x_i^k \to \dfrac{x_i^k - \overline{\mathbf{x}^i}}{SD(\mathbf{x}^i)}$, unde $\mathbf{x}^i = \begin{pmatrix} x_i^1 \\ .. \\ x_i^n \end{pmatrix}$, $1 \le k \le n, 1 \le i \le p$

- $x_i^k \to \dfrac{x_i^k - \overline{\mathbf{x}_k}}{SD(\mathbf{x}_k)}$, unde $\mathbf{x}_k = (x_1^k, ..., x_p^k)$, $1 \le k \le n, 1 \le i \le p$.

Astfel, pentru fiecare vector, media este zero şi dispersia egală cu unitatea.

Algoritmul *k-means*

Vom studia algoritmul *k-means* (Mac Queen, 1967), care clasifică o mulţime de *n* obiecte într-un număr *k* dat *apriori* de clustere.

Problema care se pune este aceea de a crea un algoritm pe baza căruia să se poată forma exact atâtea clustere cât s-a decis iniţial, cât mai distincte cu putinţă.

Rămâne deschisă problema estimării numărului optim de clustere care să conducă la separarea cea mai bună a obiectelor.
Vom menţiona doar că metoda cel mai des utilizată pentru rezolvarea acestei chestiuni este algoritmul *v-fold cross-validation* pentru determinarea automată a numărului de clustere în mulţimea de date.

Algoritm

1. Alegem aleator *k* puncte în spaţiul definit de obiectele din baza de date, ce urmează a fi clusterizate.
Acestea sunt *centroizii* iniţiali.
De obicei încercăm să îi plasăm cât se poate de departe unul de altul.
2. Asociem fiecare obiect din mulţimea de date celui mai apropiat centroid (în sensul distanţei considerate), obţinând astfel un prim grupaj.
3. Vom calcula centrele de greutate ale clusterelor obţinute, definind astfel noii centroizi.
4. Reluăm procedeul şi astfel cei *k* centroizi îşi vor schimba locul pas cu pas, până ajung în poziţia dorită.

Există o problemă majoră în utilizarea metodei *k-means*, şi anume necesitatea definirii mediei, ceea ce implică faptul că este neaplicabilă la date nenumerice (e.g. categoriale).

Algoritmul urmăreşte minimizarea *funcţiei obiectiv*:

$$J = \sum_{j=1}^{k} \sum_{i=1}^{n} \left\| x_i^{(j)} - c_j \right\|^2$$

unde $\left\| x_i^{(j)} - c_j \right\|^2$ este distanţa aleasă între punctul $x_i^{(j)}$ din baza de date şi centroidul c_j al clusterului C_j.

Funcţia obiectiv, care este în acest caz funcţia erorii pătratice, este un indicator al distanţei dintre cele *n* puncte din baza de date şi centrele clusterelor corespunzătoare.

Deoarece algoritmul nu găseşte întotdeauna configuraţia optimă, corespunzătoare minimului global al funcţiei obiectiv şi este sensibil la alegerea iniţială a centroizilor, se rulează de mai multe ori algoritmul, pentru reducerea acestor efecte.

4. Exemplu

Prezentăm o aplicaţie referitoare la cancerul hepatic.
Considerăm un lot de 17 pacienţi dintre care 6 au cancer hepatic (HCC).
Pentru a decide dacă un pacient are sau nu cancer hepatic, se iau în considerare anumite enzime serice dintre care cele mai importante din punct de vedere clinic sunt: alkaline phosphatase (FA), g-glutamyl transferase (gGT), leucine amino peptidase (LAP) şi colesterol (C)

	FA	gCT	LAP	C	diagnostic
P1	162	255	74	258	non HCC
P2	210	324	93	220	non HCC
P3	259	208	115	266	non HCC
P4	120	114	89	171	non HCC
P5	246	173	98	210	non HCC
P6	138	189	76	165	non HCC
P7	132	177	48	138	non HCC
P8	152	183	105	178	non HCC
P9	186	220	140	148	non HCC
P10	180	119	114	171	non HCC
P11	236	270	88	150	non HCC
P12	422	488	183	292	HCC
P13	607	259	65	275	HCC
P14	446	283	176	309	HCC
P15	460	1053	145	221	HCC
P16	680	381	280	275	HCC
P17	561	450	180	164	HCC

Alegem aleator doi centroizi, să zicem:

$$C1(150,100,100,100) \text{ şi } C2(400,400,200,200)$$

şi calculăm distanţele dintre aceştia şi punctele Pi (pacienţii din baza de date), împărţindu-le astfel în două submulţimi după regula:

„dacă $d(C1,Pi) < d(C2,Pi)$ pacientul Pi aparţine mulţimii $M1$;
dacă $d(C2,Pi) < d(C1,Pi)$ pacientul Pi aparţine mulţimii $M2$."

	$d(C1, Pi)$	$d(C2, Pi)$	mulțimea
P1	231.3	153.9	M2
P2	276.9	165.3	M2
P3	254.4	122.9	M2
P4	75.8	164.1	M1
P5	196.8	115.5	M2
P6	119	143.4	M1
P7	105.3	179.1	M1
P8	125.3	110	M2
P9	160.3	83	M2
P10	109.5	123.2	M1
P11	223.6	145.7	M2
P12	545.8	375.4	M2
P13	560.5	439.2	M2
P14	450.1	282.5	M2
P15	1026.9	893.6	M2
P16	691.6	524.5	M2
P17	587.8	441	M2

Am obţinut mulţimile

$$M1 = \{P4, P6, P7, P10\}$$

şi

$$M2 = \{P1, P2, P3, P5, P8, P9, P11, P12, P13, P14, P15, P16, P17\}$$

Calculăm centroizii acestor mulţimi, care au drept componente media aritmetică a componentelor vectorilor din mulţimea respectivă:

$$\frac{1}{4}(P4 + P6 + P7 + P10),$$

$$\frac{1}{13}(P1 + P2 + P3 + P5 + P8 + P9 + P11 + P12 + P13 + P14 +$$

$$+ P15 + P16 + P17)$$

şi anume:

$$C_1(142.5, 149.7, 81.7, 161.2)$$

$$C_2(355.9, 349.7, 134, 228.1)$$

şi distanţele dintre vectori (pacienţi) şi aceştia obţinând astfel un nou grupaj:

	$d(C_1, Pi)$	$d(C_2, Pi)$	mulţimea
P1	144.5	225.9	M_1'
P2	196.2	153.9	M_2'
P3	170.4	176.8	M_1'
P4	43.9	341.3	M_1'
P5	117.9	211.9	M_1'
P6	40.1	283.9	M_1'
P7	50.2	308.9	M_1'
P8	44.9	296.6	M_1'
P9	102	228	M_1'
P10	59	296.3	M_1'
P11	152.9	170..1	M_1'
P12	468.9	173.1	M_2'
P13	490.8	279.7	M_2'
P14	374.9	144.4	M_2'
P15	961.4	711.1	M_2'
P16	628.2	359.9	M_2'
P17	524.3	241.1	M_2'

Calculăm centroizii mulțimilor M_1' și M_2'

$C_1'(181.1, 190.8, 94.7, 185.5)$

$C_2'(485, 462.5, 160.2, 250.8)$

și astfel:

	$d(C_1', Pi)$	$d(C_2', Pi)$	mulţimea
P1	100.8	393.5	M_1''
P2	140.6	316.6	M_1''
P3	115.1	343.6	M_1''
P4	99.3	515.8	M_1''
P5	71.6	382.7	M_1''
P6	51.2	457.8	M_1''
P7	83.8	481	M_1''
P8	32.7	444.2	M_1''
P9	65.8	398.9	M_1''
P10	75.7	468.5	M_1''
P11	102.9	338.2	M_1''
P12	406.8	82.6	M_2''
P13	441.5	286.7	M_2''
P14	317	193.3	M_2''
P15	908.2	591.9	M_2''
P16	572.2	244.1	M_2''
P17	468.2	117.7	M_2''

Continuând procedeul, centroizii mulţimilor obţinute sunt:

$C_1''(183.7, 202.9, 94.5, 188.6)$
$C_2''(355.9, 349.7, 134, 228.1)$.

Prezentăm tabelul cu distanţele faţa de aceşti centroizi
şi cu mulţimile cărora le aparţin pacienţii.

	$d(C_1'', Pi)$	$d(C_2'', Pi)$	mulţimea
P1	91.7	445.7	M_1'''
P2	127.8	369.4	M_1'''
P3	110	392.7	M_1'''
P4	110.9	566.4	M_1'''
P5	72.4	431.6	M_1'''
P6	56.3	532.4	M_1'''
P7	89.9	532.4	M_1'''
P8	40.2	495.5	M_1'''
P9	63.3	449.5	M_1'''
P10	87.9	517.6	M_1'''
P11	93.6	389.3	M_1'''
P12	395.7	115.2	M_2'''
P13	436.6	282.4	M_2'''
P14	310.4	225.9	M_2'''
P15	895.8	573.4	M_2'''
P16	565.6	212.9	M_2'''
P17	459.7	103.5	M_2'''

După cum se observă, am obţinut aceleaşi mulţimi, ca în tabelul precedent, ceea ce ne indică faptul că procesul s-a încheiat.
Acest fapt constituie condiţia de stop a algoritmului.

Am împărţit mulţimea de obiecte în două clustere, şi testând cu mulţimea noastră de date, rezultatul este optim.

Algoritmul este sensibil la alegerea centroizilor iniţiali.

În exemplul anterior, alegem alţi centroizi:

$C1(150,100,100,100)$ şi $C2(400,400,200,200)$

obţinând următorul rezultat, asemănător cu rezultatul din cel de-al doilea tabel:

	$d(C1, Pi)$	$d(C2, Pi)$	mulțimea
P1	231.3	311.3	M1
P2	276.9	231.7	M2
P3	254.4	261.3	M1
P4	75.8	416.3	M1
P5	196.8	292.8	M1
P6	119	360.2	M1
P7	105.3	385.3	M1
P8	125.3	343.6	M1
P9	160.3	290.6	M1
P10	109.5	368.2	M1
P11	223.6	242.5	M1
P12	545.8	130.3	M2
P13	560.5	294.2	M2
P14	450.1	168.3	M2
P15	1026.9	658.3	M2
P16	691.6	301.3	M2
P17	587.8	173.5	M2

kmeans

Pentru a rezolva problema în Matlab vom apela **kmeans** , cu k numărul de clustere dorite, în cazul nostru 2. Algoritmul utilizează ca centroizi iniţiali vectori 4-dimensionali, aleator aleşi.

```
>>X=[162 255 74 258;210 324   93 220;259 208 115 266;120 114 89 171;
246 173 98 210;138 189 76 165;132 177 48 138;152 183 105 178;
186 220 140 148;180 119 114 171;236 270 88 150;422 488 183 292;
607 259 65 275; 446 283 176 309;460 1053 145 221;680 381 280 275;
561 450 180 164]
>> idx2 = kmeans(X,2);
>> idx2'
ans =
ans =
Columns 1 through 15
2    2    2    2    2    2    2    2    2    2    2    1    1    1    1
Columns 16 through 17
 1    1
```

Dacă dorim să folosim altă distanţă decât cea euclidiană vom specifica, de exemplu distanţa city:

```
>>idx2 = kmeans(X,2,'distance','city')
```

5. exemplu

Aplicăm algoritmul k-means bazei de date Iris:

```
>> load fisheriris
>> X=[meas(:,1), meas(:,2), meas(:,3), meas(:,4)];
>> idx3 = kmeans(X,3);
>> idx3'

ans =
  Columns 1 through 15
    3    3    3    3    3    3    3    3    3    3    3    3    3    3    3
  Columns 16 through 30
    3    3    3    3    3    3    3    3    3    3    3    3    3    3    3
  Columns 31 through 45
    3    3    3    3    3    3    3    3    3    3    3    3    3    3    3
  Columns 46 through 60
    3    3    3    3    3    1    1    2    1    1    1    1    1    1    1
  Columns 61 through 75
    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1
  Columns 76 through 90
    1    1    2    1    1    1    1    1    1    1    1    1    1    1    1
  Columns 91 through 105
    1    1    1    1    1    1    1    1    1    1    2    1    2    2    2
  Columns 106 through 120
    2    1    2    2    2    2    2    2    1    1    2    2    2    2    2
  Columns 121 through 135
    2    1    2    1    2    2    1    1    2    2    2    2    2    1    2
  Columns 136 through 150
    2    2    2    1    2    2    2    1    2    2    2    1    2    2    2
```

Se observă că al treilea cluster care ar fi etichetat cu Virginica, conţine destul de multe Setosa.

Rulând algoritmul, considerând ca atribute doar dimensiunile petalelor, clasificare este net mai bună.

```
>> load fisheriris
>> X=[ meas(:,3), meas(:,4)];
>> idx3 = kmeans(X,3);
>> idx3'
```

Clustering ierarhic

Metoda de *clustering ierarhic* constă în gruparea obiectelor în mod iterativ, utilizând o anumită articulare - *linkage*. Există două tipuri de metode:

- *aglomerativă*, care constă într-o serie de fuziuni a celor *n* obiecte în grupuri (cea mai utilizată)
- *de divizare*, care separă succesiv cele *n* obiecte în grupuri mai fine.

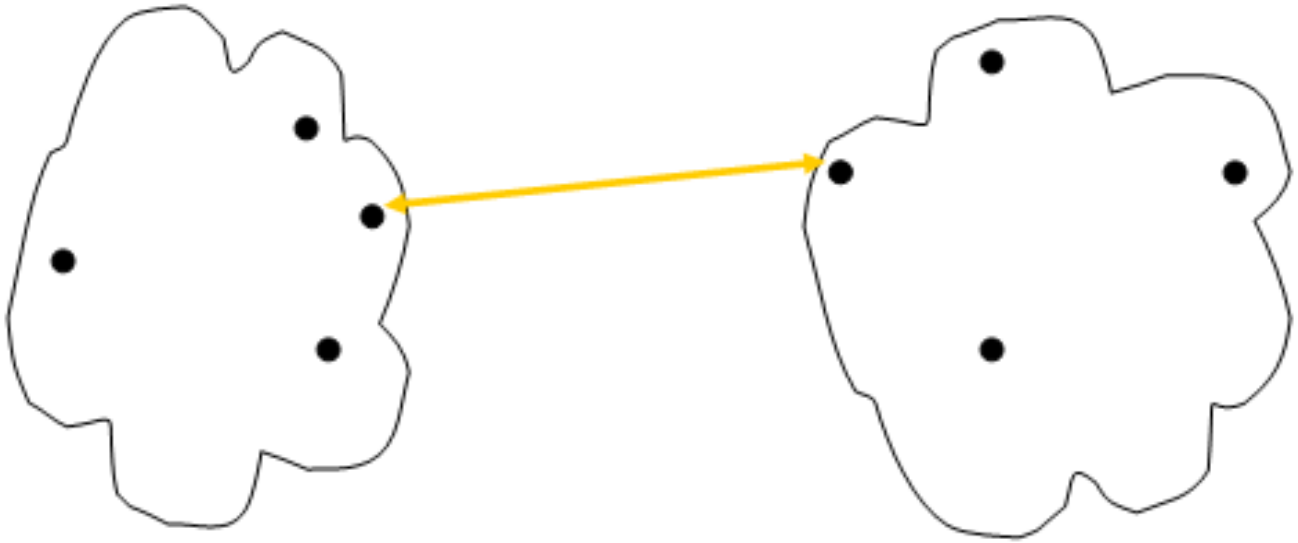
Putem defini distanţa dintre două clustere în mai multe moduri:

Single linkage

- *Single linkage*, caz în care distanța dintre clustere este determinată de distanța între cele mai apropiate obiecte:

$$d(A,B) = \min\{d(\mathbf{x}_k, \mathbf{y}_j), \mathbf{x}_k \in A, \mathbf{y}_j \in B\},$$

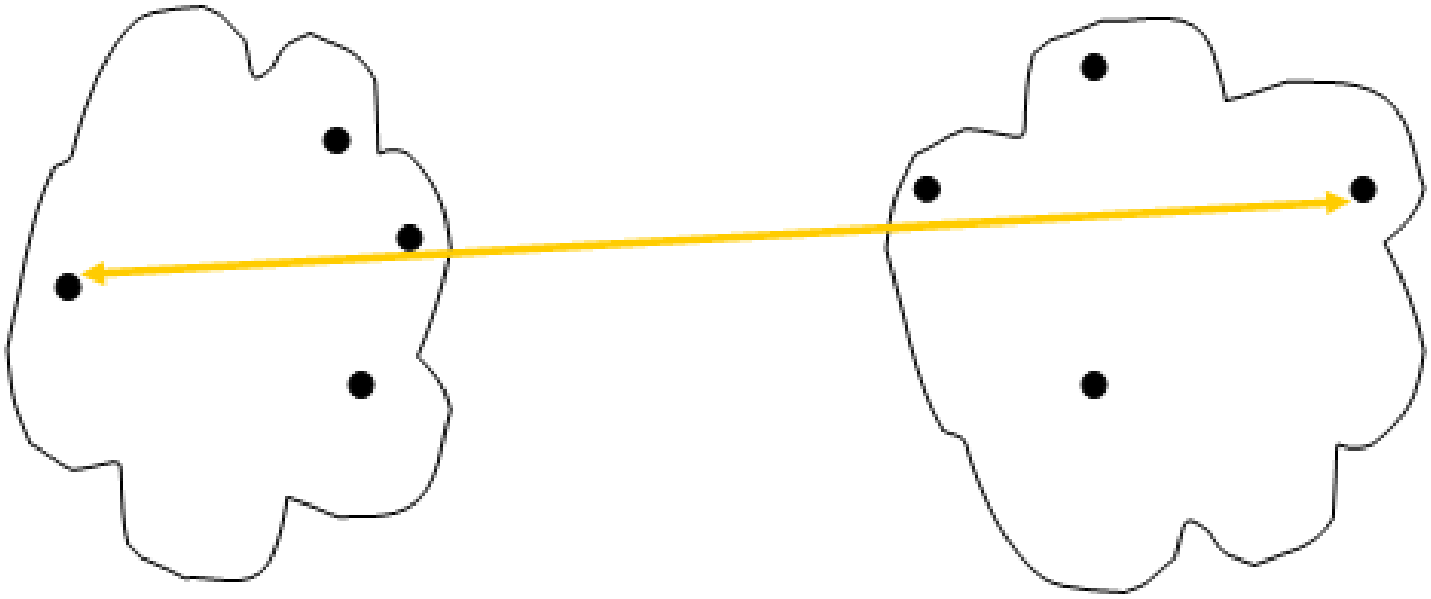
unde am notat cu *A* și *B* cele două clustere



complete linkage

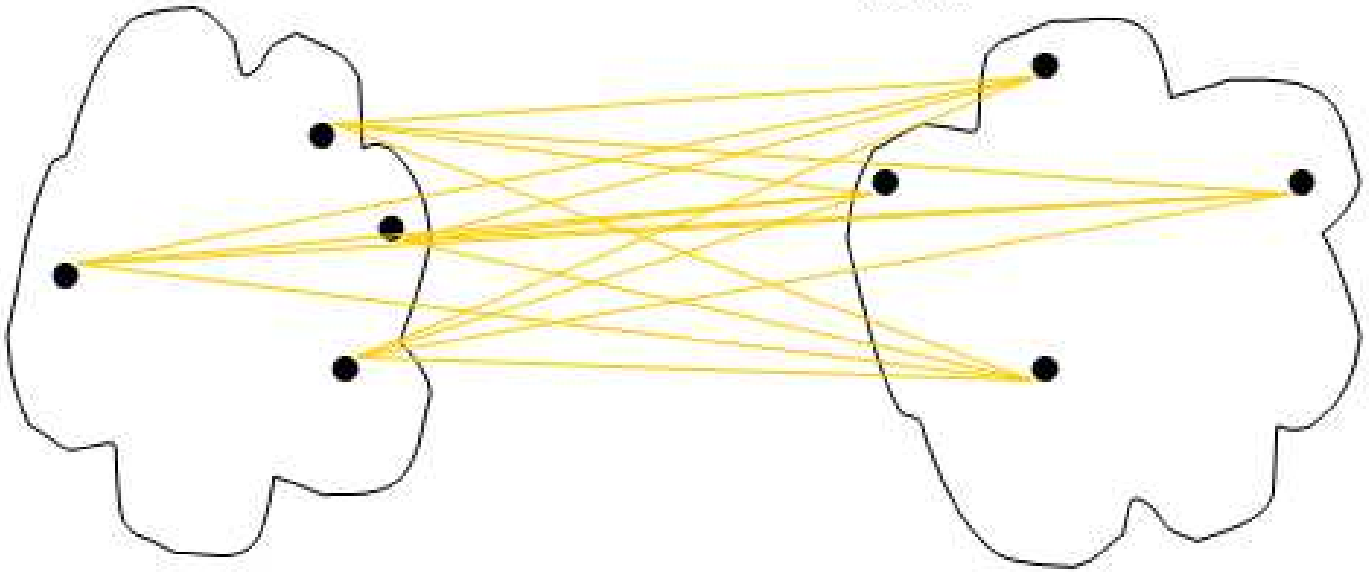
- *Complete linkage*, caz în care distanța dintre clustere este determinată de distanța între cele mai îndepărtate obiecte:

$$d(A, B) = \max\{d(\mathrm{x}_k, \mathrm{y}_j), \mathrm{x}_k \in A, \mathrm{y}_j \in B\}$$



average linkage

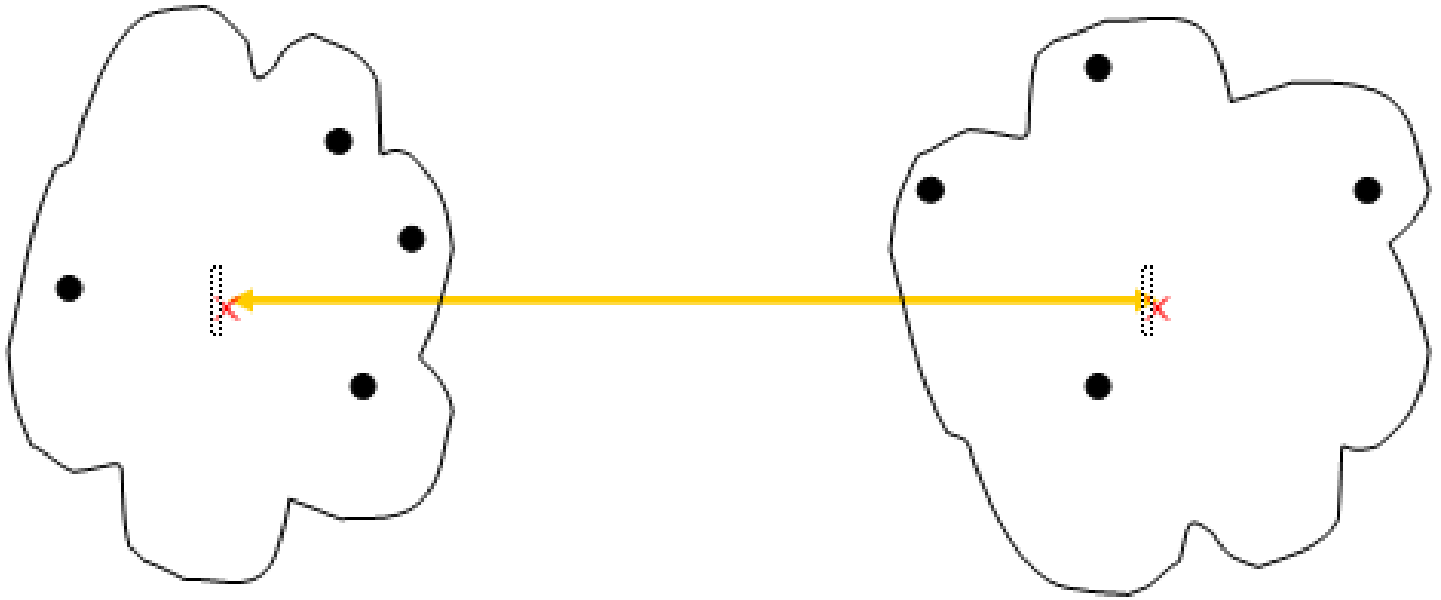
- *average linkage*, caz în care distanța este calculată conform formulei: $d(A,B) = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m} d(\mathbf{x}_i, \mathbf{y}_j)}{m \cdot n}$,



centroid linkage

- *centroid linkage* este distanța dintre centroizi. Este de mentionat faptul că aceasta poate fi folosita doar dacă se utilizează distanța euclidiană.

$d_{cen}(A,B) = d(\bar{x}_A, \bar{x}_B)$ unde $\bar{x}_A = \frac{1}{n}\sum_{i=1}^{n} x_A^i$.



ward linkage

Suma pătratelor distanțelor din interiorul unui cluster este definită ca fiind suma pătratelor distanțelor dintre obiectele existente în cluster şi centroidul acestuia.
Ward linkage se defineşte prin formula:

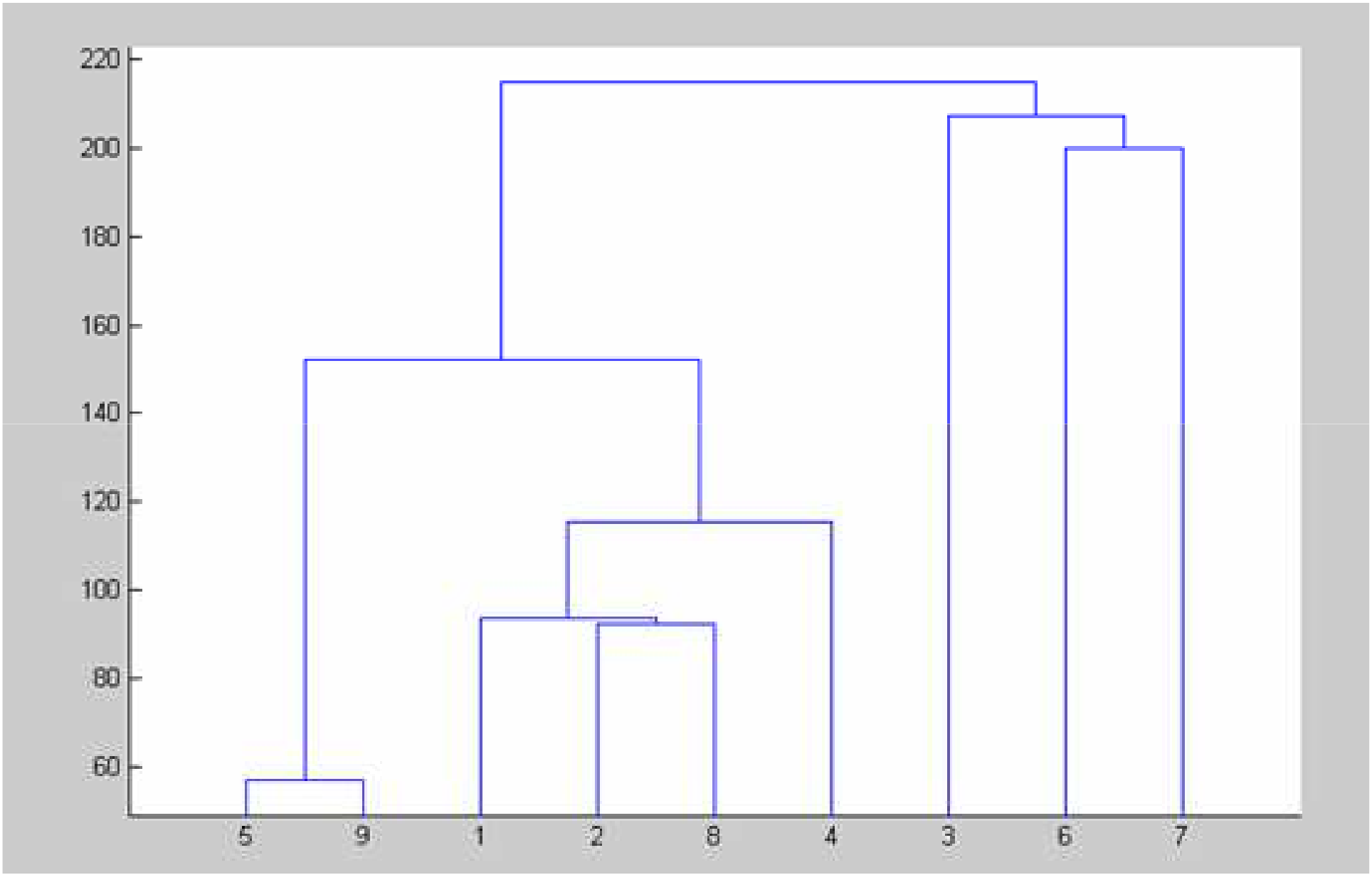
$$d_W(A,B) = \frac{n \cdot m d_{cen}(A,B)}{n+m}$$

unde clusterul A conține n obiecte şi clusterul B conține m obiecte

dendrograma

Clustering-ul ierarhic permite o reprezentare printr-o diagramă bidimensională, numită *dendogramă*, care ilustrează fuziunile sau divizările din fiecare pas al procesului efectuat.

Prezentăm o asemenea dendogramă:


220
200
180
160
140
120
100
80
60
5
9
1
2
8
4
3
6
7

6. exemplu

Considerăm o mulţime de 10 clienţi ai unei reprezentanţe auto. Fiecare client are 4 caracteristici: venit lunar, vârstă, starea civilă şi studiile. Există 3 tipuri de maşini A, B şi C.

	venit	vârstă	stare civilă	studii	Tip maşină
C1	1600	28	necăsătorit	medii	A
C2	1800	24	necăsătorit	superioare	A
C3	2000	36	căsătorit	superioare	A
C4	1700	40	căsătorit	medii	B
C5	2100	30	căsătorit	superioare	B
C6	2000	48	căsătorit	medii	B
C7	3000	52	căsătorit	medii	B
C8	3200	35	căsătorit	medii	C
C9	4000	30	necăsătorit	superioare	C
C10	5000	46	căsătorit	superioare	C

Vom standardiza valorile atributelor venit, respectiv vârstă,
folosind formulele:

$$ven_s(i) = \frac{ven(i) - 26400}{1.1374 \cdot 10^4} \; ; \; v_S(i) = \frac{v(i) - 36.9}{9.36}$$

Convenim ca pentru atributul *căsătorit* să atribuim valoarea 1,
pentru *necăsătorit* 0, pentru atributul *studii superioare* atribuim 1,
în timp ce pentru *studii medii* 0.

```
>> V=[1600 1800 2000 1700 2100 2000 3000 3200 4000 5000];
>> mu=mean(V);s=std(V);
>> for i=1:10 Vs(i)=(V(i)-mu)/s;end
>> Vs
Vs =
  Columns 1 through 9
 -0.9143   -0.7385   -0.5627   -0.8264   -0.4747   -0.5627    0.3165    0.4923    1.1957
  Column 10
    2.0748
>> v=[28 24 36 40 30 48 52 35 30 46];
>> mu1=mean(v); s=std(v);
>> for i=1:10 vs(i)=(v(i)-mu1)/s;end
>> vs
vs =
  Columns 1 through 9
-0.9506   -1.3778   -0.0961    0.3311   -0.7370    1.1856    1.6128   -0.2029   -0.7370
  Column 10
    0.9720
>> sc=[0 0 1 1 1 1 1 1 0 1];
>> st=[0 1 1 0 1 0 0 0 1 1];
>> X=[Vs' vs' sc' st'];
```

Avem 10 clustere pentru început.
Calculăm distanţele între vectorii standardizaţi şi vom alcătui o matrice, matricea distanţelor, unde elementul $a_{ij} = d(Ci, Cj)$.

Vom determina astfel vectorii cei mai apropiaţi, în sensul distanţei, pe care îi vom grupa în clustere:

```
>> D=pdist(X)
>> D1=squareform(D)
```

ans =

Columns 1 through 10

0	**1.1016**	1.6893	1.6280	1.4963	2.3847	3.0143	1.8809	2.3447	3.8251
1.1016	0	1.6351	2.2200	1.2167	2.9329	3.4723	2.2125	2.0376	3.7995
1.6893	1.6351	0	1.1190	**0.6469**	1.6257	2.1664	1.4575	2.1219	2.8456
1.6280	2.2200	1.1190	0	1.5048	**0.8943**	1.7173	1.4228	2.6888	3.1349
1.4963	1.2167	**0.6469**	1.5048	0	2.1689	2.6735	1.4901	1.9469	3.0693
2.3847	2.9329	1.6257	**0.8943**	2.1689	0	0.9775	1.7439	2.9645	2.8288
3.0143	3.4723	2.1664	1.7173	2.6735	0.9775	0	1.8243	2.8800	2.1219
1.8809	2.2125	1.4575	1.4228	1.4901	1.7439	1.8243	0	**1.6673**	2.2101
2.3447	2.0376	2.1219	2.6888	1.9469	2.9645	2.8800	**1.6673**	0	2.1664
3.8251	3.7995	2.8456	3.1349	3.0693	2.8288	2.1219	2.2101	2.1664	0

Alegem pe fiecare coloană j, cea mai mică distanţă, să zicem D_{ij} ;

dacă D_{ij} este ceamai mică distanţă pe linia i, obiectele i şi j vor forma un cluster.

Astfel avem 1&2, 3&5, 4&6, 8&9

In coloana a 7-a minimul este $D_{67} = 0.9775$, însă această distanţă nu o vom lua în considerare deoarece pe linia 6 $D_{67} > D_{64}$.

In prima etapă avem 6 clustere (deocamdată clienţii 7 şi 10 formează clustere separate).

Avem următorii vectori:

```
>> C1=X(1,:)+X(2,:)  % corespunde clusterului 1&2
C1 =
  -1.6528   -2.3284         0    1.0000
>> C2=X(3,:)+X(5,:) ,:)  % corespunde clusterului 3&5
C2 =
  -1.0374   -0.8331    2.0000    2.0000
>> C3=X(4,:)+X(6,:) ,:)  % corespunde clusterului 4&6
C3 =
  -1.3891    1.5167    2.0000         0
>> C4=X(8,:)+X(9,:) % corespunde clusterului 8&9
C4 =
   1.6880   -0.9399    1.0000    1.0000
>> X1=[C1;C2;C3;C4;X(7,:);X(10,:)];
```

```
>> D2=pdist(X1)
>> D3=squareform(D2)
D3 =
```

0	**2.7595**	4.4559	3.7535	4.6273	5.0782
2.7595	0	3.1057	3.0723	3.5799	3.8658
4.4559	3.1057	0	4.1837	**1.9795**	3.7809
3.7535	3.0723	4.1837	0	3.0655	**1.9506**
4.6273	3.5799	**1.9795**	3.0655	0	2.1219
5.0782	3.8658	3.7809	**1.9506**	2.1219	0

Obţinem următoarele clustere:

C1 &C2, adică 1&2&3&5

C3&7, adică 4&6&7

C4&10, adică 8&9&10

Clustere corespunzatoare celor 3 tipuri de maşini

Testând cu mulţimea de date, observăm că rezultatul este validat în proporţie de 90%.

În final, este necesară aprecierea validităţii modelului, lucru realizabil prin repetarea procesului folosind alte măsuri de similaritate şi eventual alte tehnici de clustering.

Folosind aceste tehnici de clusterizare descoperim clase „naturale” în care se plasează obiectele din baza de date şi nu realizăm o nouă ordine în structura datelor.

Clustering ierarhic in Matlab

Y=pdist(X,'metric') calculează distanţele dintre obiecte utilizând distanţa specificată în 'metric'

Grupăm obiectele intr-un *cluster tree* binar (arbore binar de cluster ierarhic), legând perechi de obiecte, care sunt apropiate, folosind funcţia **linkage**.

Funcţia **linkage** foloseşte informaţiile asupra distanţei pentru a determina apropierea obiectelor, unele de altele.
Cum obiectele sunt cuplate în clustere binare, noile clustere formate vor fi grupate în clustere mai mari, şi astfel vom construi arborele ierarhic.

Z = linkage(Y,'method') calculează clusterul ierarhic folosind algoritmul cerut prin „method”

- 'single'= *Single linkage*
- 'complete' = *Complete linkage*
- 'average' = *Average linkage*
- 'ward' = *Ward linkage.*
- 'centroid' = *Centroid linkage*

Să reținem faptul că *centroid linkage* poate fi folosită doar dacă se utilizează distanța euclidiană.

Outputul Z este o matrice de tip $(m-1)\times 3$ ce conţine informaţie despre cluster-tree.

Nodurile frunze in ierarhia clusterului sunt obiectele din baza de date numerotate de la 1 la m. Ele sunt clustere cu un singur element, din care vor fi construite următoarele clustere.

Fiecărui cluster nou format corespunzător liniei i în Z îi asociem indicele m+i, unde m este numărul frunzelor iniţiale.

Coloanele 1 şi 2 din Z conţin indicele obiectelor ce sunt grupate pentru a forma un nou cluster. În coloana 3, pe linia i din Z avem afişată distanţa dintre obiectele ce apar pe această linie.

Considerăm m=30; dacă al 10-lea cluster combină prin funcția linkage

obiectul 5 cu obiectul 7 și distanța dintre ele este 1.5, atunci cea de-a 10-a

linie a lui Z va fi

5 7 1.5

și noul cluster format va avea indicele 30+10=40

Dacă pe altă linie apare clusterul 40, înseamnă ca acest cluster a fost combinat

cu altul, într-un cluster mai mare.

7. exemplu

Considerăm un lot de de 9 pacienți dintre care 3 au cancer hepatic (HCC). Pentru a decide dacă un pacient are sau nu cancer hepatic, studiile clinice arată că se iau în considerare anumite enzime serice, dintre care cele mai importante din punct de vedere clinic sunt: alkaline phosphatase (FA), g-glutamyl transferase (gGT), leucine amino peptidase (LAP) și colesterol (C).

	FA	gCT	LAP	C	diagnostic
P1	162	255	74	258	non HCC
P2	210	324	93	220	non HCC
P3	422	488	183	292	HCC
P4	259	208	115	266	non HCC
P5	129	114	89	171	non HCC
P6	607	259	65	275	HCC
P7	446	283	176	309	HCC
P8	236	270	88	150	non HCC
P9	180	119	114	171	non HCC

```
>>X=[162 255 74 257;210 324 93 220;422 488 183 292;259 208 115 266;
129 114  89 171; 607 259 65 275;446 283 176 309;236 270 88 150;
180 119 114 171];
>> Y=pdist(X);
>> Z=linkage(Y)
Z =
    5.0000    9.0000   57.0175
    2.0000    8.0000   92.2876
    1.0000   11.0000   93.7817
    4.0000   12.0000  115.6719
   10.0000   13.0000  152.2761
    6.0000    7.0000  199.9350
    3.0000   15.0000  207.2173
   14.0000   16.0000  214.8581
```

Pentru început se formează un cluster între pacientul 5 şi pacientul 9, între care distanţa euclidiană este de 57.0175, cluster ce va fi indexat cu numărul 10.

Obţinem apoi clusterul 11 format prin alăturarea pacientului 2 cu pacientul 8.

Clusterul 11 este unit cu pacientul 1 obţinând clusterul 12.

Din clusterul 12 unit cu pacientul 4 obţinem clusterul13.

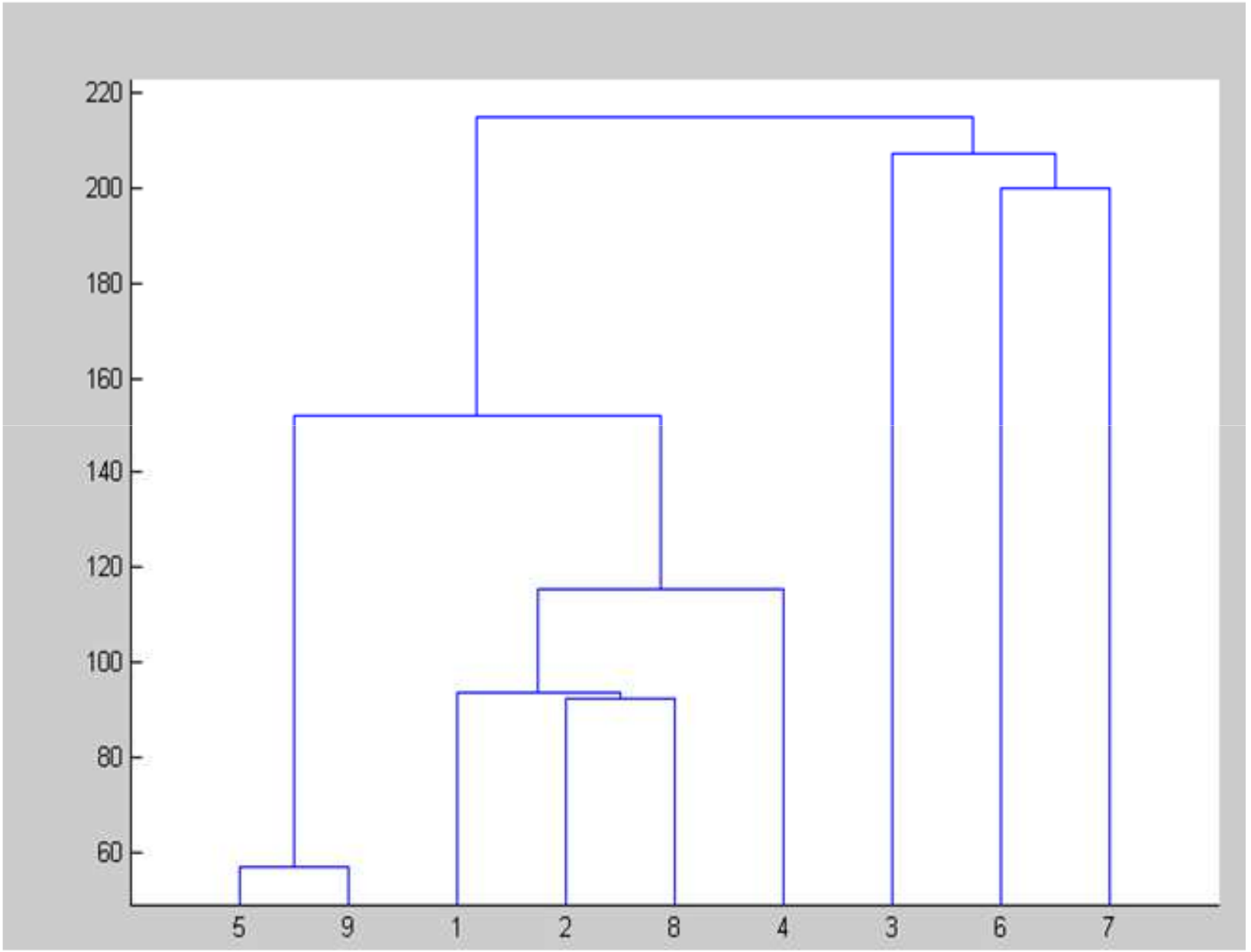
Clusterul 13 se uneşte cu clusterul 10, rezultând clusterul 14.

Pacientul 6 unit cu pacientul 7 formează clusterul 15.

Pacientul 3 împreună cu clusterul 15 formează clusterul 16.

În final se calculează distanţa între clusterul 14 şi clusterul 16.

```
>> dendrogram(Z)
```



După ce am grupat obiectele în clustere avem de verificat dacă clusterul ierarhic reprezintă grupări de obiecte similare.

Aceasta verificare se face măsurând valabilitatea informaţiei generată de funcţia **linkage** şi anume prin compararea cu datele generate de funcţia **pdist**.

Dacă clusterizarea este validă legăturile obiectelor în cluster trebuie să aibă o puternică corelaţie cu distanţele dintre obiecte, din vectorul distanţă.

Funcţia **cophenet** compară aceste două mulţimi de date şi calculează corelaţia lor, returnând o valoare numită *coeficient cophenetic de corelaţie.*

```
c= cophenet(Z,Y)
```

Cu cât acest coeficient este mai apropiat de 1, cu atât mai bună este soluţia de clusterizare.

Putem folosi coeficientul cophenetic pentru a compara rezultatele clusterizării aceloraşi date, folosind distanţe diferite sau algoritmi diferiţi de clusterizare.

În exemplul prezentat, folosind distanţa euclidiană şi *Single linkage* avem:

```
>> Y=pdist(X)
>> Z = linkage(Y)
>> c = cophenet(z,y)
        c =
                0.8321
```

Să folosim *Single linkage* şi diferite distanţe:

- ```
  »Y2=pdist(X,'mahalanobis');
  »Z= linkage(Y2);
  »c2= cophenet(Z,Y2)
  c2 =
          0.6859
  ```

- ```
  »Y3=pdist(X,'cityblock');
  »Z = linkage(Y3);
  »c3= cophenet(Z,Y3)
  c3 =
  0.8446
  ```

Vom continua cu restul variantelor.

Cu toate că teoretic ştim că metoda centroid linkage nu poate fi utilizată dacă în calculul distanţelor nu am folosit-o pe cea euclidiană, încercăm pentru a vedea răspunsul dat de soft.

- »Y1=pdist(X,'mahalanobis');
  »Z1=linkage(Y1,'centroid');
  Warning: Non-monotonic cluster tree -- the centroid method is probably not appropriate.
  In C:\MATLAB6p5\toolbox\stats\linkage.m at line 166

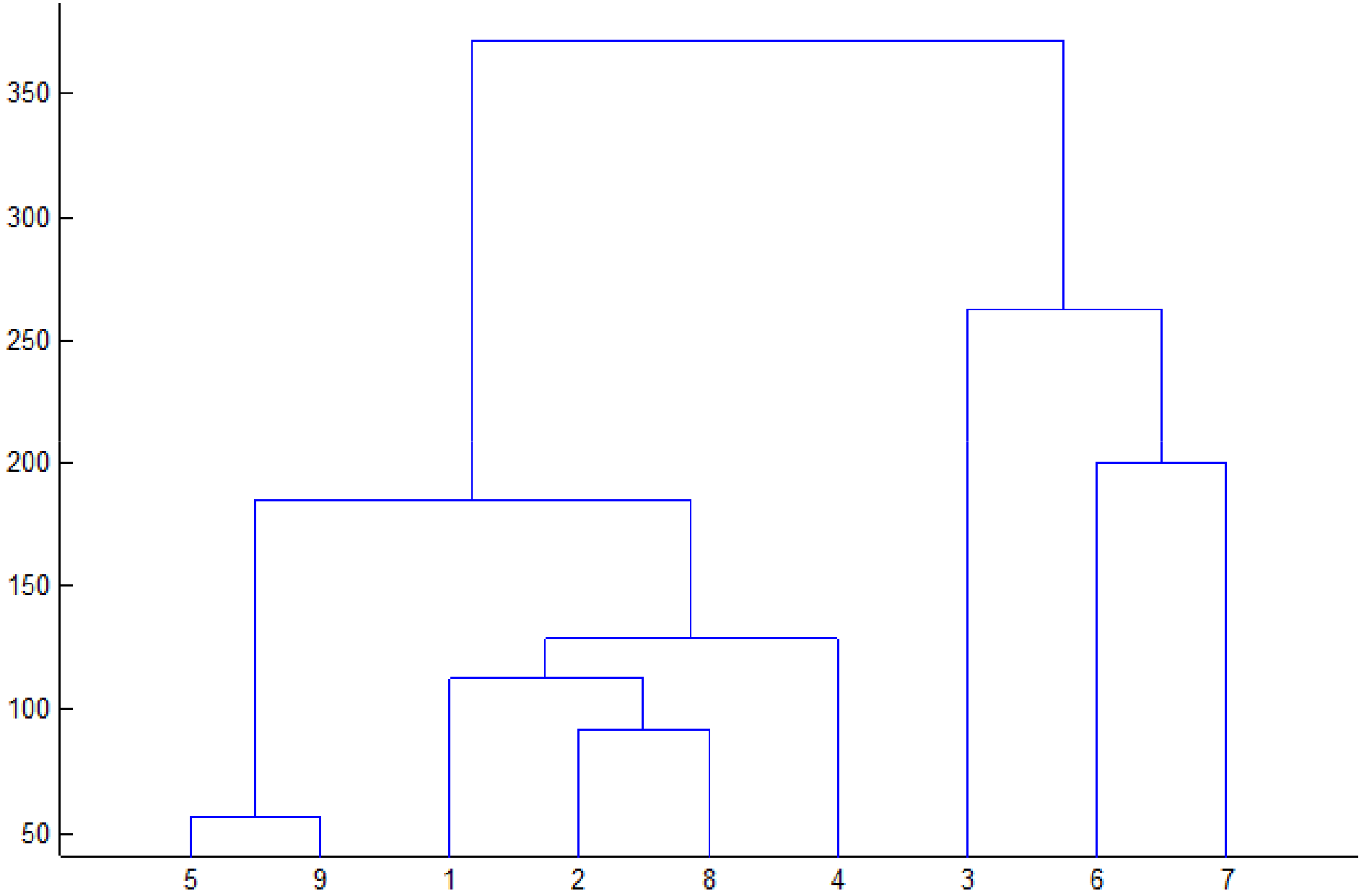
- Y1=pdist(X,'cityblock');
  Z1=linkage(Y1,'centroid');
  Warning: Centroid method specified with non-Euclidean dissimilarity matrix.
  In C:\MATLAB6p5\toolbox\stats\linkage.m at line 91

Prezentăm coeficienții cophenetici:

	single	complete	average	centroid	ward
euclidian	0.8321	0.8739	**0.8763**	0.8755	0.8631
Mahalanobis	0.6859	0.6804	0.7747	-	0.6377
cityblock	0.8446	0.8602	0.8635	-	0.8518
Minkowski	0.8321	0.8739	0.8763	0.8755	0.8631
correlation	0.7272	0.6518	0.6577	-	0.7365

Desenăm dendrograma corespunzătoare celui mai mare coeficient cophenetic:

```
»Y=pdist(X);
»Z1=linkage(Y,'average');
»dendrogram(Z1)
```


350
300
250
200
150
100
50
5
9
1
2
8
4
3
6
7

O posibilitate de a determina divizările naturale ale clusterului într-o mulţime de date constă în a compara înăălţimea fiecărui link din cluster tree cu înălţimile linkurilor vecine, care se află sub acesta în arbore.

Faptul că un link este aproximativ la aceeaşi înălţime cu linkurile vecine, indică existenţa similarităţilor între obiectele unite la acelaşi nivel ierarhic.
Se spune că aceste linkuri prezintă *un înalt nivel de consistenţă*.

Faptul că înălţimea unui link diferă de cele ale linkurilor din vecinătate indică existenţa disimilarităţilor între obiecte la acest nivel al cluster arborelui şi se spune că acest link este inconsistent cu linkurile din jurul lui.

Coeficientul de inconsistenţă compară lungimea unui link cu media lungimilor linkurilor de la acelaşi nivel ierarhic.
Cu cât aceasta valoare este mai mare cu atât mai puţine asemănări există între obiectele legate de acest link.

»U=inconsistent (Z) calculează coeficientul de inconsistenţă,

U este o matrice cu m-1 linii si 4 coloane:
Coloana 1 _ media lungimii linkurilor luate în calcul
Coloana 2 _deviaţia standard a linkurilor luate în calcul
Coloana 3 _numărul linkurilor luate în calcul
Coloana 4 _coeficientul de inconsistenţă

Tinând seama de descrierea matricei Z=linkage(Y), vom avea următoarea formula de calcul pentru coeficientul de inconsistență:

$$U(k,4) = \frac{Z(k,3) - U(k,1)}{U(k,2)}$$

În exemplul prezentat avem:

```
» U=inconsistent(Z)
U =
```

57.0175	0	1.0000	0
92.2876	0	1.0000	0
93.0346	1.0565	2.0000	0.7071
104.7268	15.4788	2.0000	0.7071
108.3219	48.0527	3.0000	0.9147
199.9350	0	1.0000	0
203.5761	5.1494	2.0000	0.7071
191.4505	34.1405	3.0000	0.6856

În analiza clusterelor, linkurile inconsistente indică frontiera unei împărţiri naturale în mulţimea datelor.

Funcţia **cluster** utilizează o măsura a inconsistenţei pentru a determina unde să împărţim mulţimea de obiecte în alte clustere.

Intr-un cluster tree ierarhic, mulţimea de date poate fi natural împărţită în clustere. Aceasta poate evident fi într-o dendogramă în care grupuri de obiecte se află într-o anumită arie.

Coeficientul de inconsistenţă a linkurilor poate identifica acele puncte in care se schimbă similaritatea obiectelor.

Dacă folosim funcția **cluster** pentru a grupa obiectele in clustere specificăm treshhold-ul pentru coeficientul de inconsistență:

```
»T=cluster(Z,0.9)
T =
     1
     1
     3
     1
     2
     3
     3
     1
     2
```

Ceea ce înseamnă că pacienții 1,2, 4 și 8 aparțin primului cluster, 5 și 9 celui de al doilea și pacienții 3, 6 și 7 aparțin celui de-al treilea cluster.

Aceste explicaţii detaliate le putem obţine şi folosind funcţia **find**

```
»find(T==1)
ans =
     1
     2
     4
     8
```

```
»find(T==2)
ans =
     5
     9
```

```
» find(T==3)
ans =
     3
     6
     7
```

```
» find(T==4)
ans =
Empty matrix: 0-by-1
```

În acest caz niciun link din clusterul ierarhic nu are coeficientul de inconsistenţă mai mare decât 0.9

În acest caz niciun link din clusterul ierarhic nu are coeficientul de

inconsistență mai mare decât 0.9

Dacă vom considera treshhold-ul pentru coeficientul de inconsistență 0.7, vom avea 6 clustere:
1; 2&8; 3; 4: 5; 6&7

sau când treshholdul este 0.915, caz în care avem un singur mare cluster.

Funcţia **T = cluster(Z,'maxclust',n)** specifică numărul maxim de clustere n, ce dorim să obţinem în cluster tree-ul ierarhic.

Revenind la exemplul comentat

```
T = cluster(Z,'maxclust',2)
»T =
     1
     1
     2
     1
     1
     2
     2
     1
     1
```

Este important de reținut faptul că folosind aceste tehnici de clusterizare descoperim clase *naturale* în care se plasează obiectele din baza de date și nu realizăm o nouă ordine în structura datelor.