Biostatistică și Bioinformatică.

Lab 6: Algoritmi de grupare a datelor

Gruparea datelor ("data clustering") este una dintre cele mai cunoscute și frecvent utilizate prelucrări din domeniul analizei datelor. Scopul grupării este identificarea unor grupuri de date ("clustere") caracterizate prin faptul ca datele din fiecare grup sunt suficient de similare între ele iar datele din grupuri diferite sunt suficient de disimilare. Gruparea datelor este cunoscută și sub numele de clasificare nesupervizată a datelor întrucât nu se pornește de la exemple de asignare a datelor la grupuri iar în cazul cel mai general nu este cunoscut apriori nici măcar numărul de grupuri care trebuie identificat. Fiecărui cluster i se asociază de regulă un reprezentant (numit *centru* sau *centroid* sau *prototip*) care are rolul de a reprezenta clusterul.

Gruparea datelor are aplicații în analiza imaginilor (identificarea regiunilor omogene din imagine), în identificarea profilurilor utilizatorilor unei resurse web (gruparea în aceeași categorie a utilizatorilor cu comportament similar), în analiza expresiei genice (cu scopul de a identifica gene cu funcții similare) etc.

Elementul cheie în dezvoltarea unui algoritm de grupare este stabilirea unei măsuri care să permită evaluarea similarității dintre date. Rezultatul aplicării unei metode de grupare poate fi o partiționare a datelor în grupuri (în cazul algoritmilor partiționali, cum este *kMeans*) sau o structură arborescentă care permite identificarea mai multor partiționări (în cazul *algoritmilor ierarhici*).

1. Algoritmul kMeans

Este un algoritm iterativ care permite partiționarea unui set de date într-un număr prestabilit de clustere, fiecare caracterizat printr-un reprezentant, astfel încât suma distanțelor dintre date și reprezentanții clusterilor corespunzători este minimă. Varianta cea mai simplă a algoritmului de partiționare constă în:

- Se inițializează reprezentanții clusterilor cu date selectate aleator din setul de date
- Se repetă:
    i. Se asignează datele la clustere pe baza criteriului distanței minime (fiecare dată din set este asignată clusterului având cel mai apropiat centru);
    ii. Se recalculează centrii clusterelor ca medii aritmetice ale datelor pe care le conțin;

  Procesul continuă până când centrii nu se mai modifică.

In R algoritmul este implementat prin funcția kmeans care se apelează, în varianta cea mai simplă, prin:

```
rez=kmeans(date,nrclustere)
```

Functia returnează o structură care conține mai multe câmpuri cu informații privind gruparea identificată:

- cluster: vector cu etichete care indică cărui cluster îi aparține fiecare dată
- centers**:** coordonatele centrilor corespunzători clusterelor identificate

- size: vector cu numărul de date din fiecare cluster
- totss, withinss, betweenss: mărimi statistice corespunzătoare grupării (suma pătratelor distanțelor dintre punctele din set (totss=total sum of squares), dintre cele aparținând unui cluster (withinss=within cluster sum of squares), dintre cele aparținând unor clustere diferite (betweenss=between clusters sum of squares)).

Exercitiul 1.

a) să se genereze un set de date bidimensionale
b) să se aplice tehnica kMeans pentru diferite valori ale numărului de clustere
c) să se vizualizeze clusterii obținuți prin reprezentarea cu culori diferite a datelor aparținând unor clusteri diferiți

Indicatie. Datele pot fi generate aleator folosind distribuții normale având medii generate aleator (de exemplu centrii sunt aleși uniform aleator în [0,1]x[0,1]). Exemple de funcții care realizează generarea datelor și construirea clusterelor sunt prezentate în continuare.

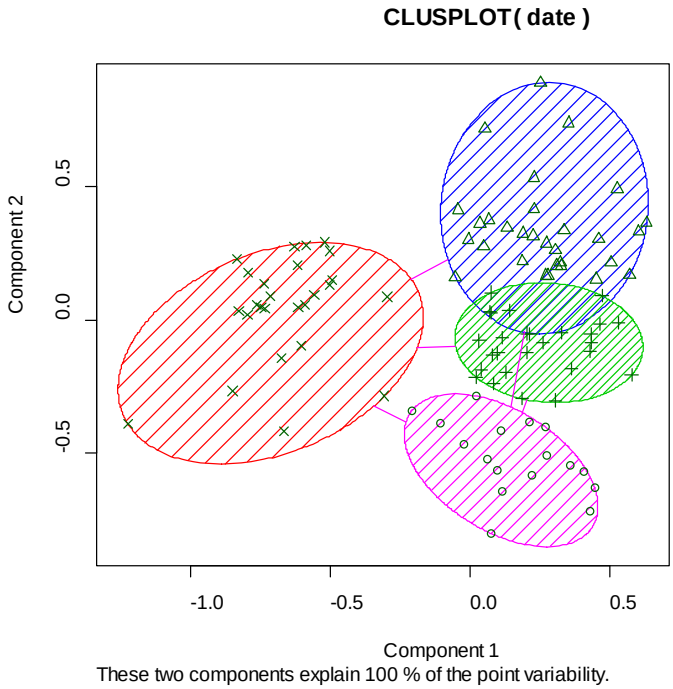
```
generareDate=function(nrclustere,nrdate)
{
  # generare centre
  centre=array(runif(2*nrclustere,min=0,max=1),c(nrclustere,2))
  # generare date
  dimcluster=nrdate/nrclustere
date=cbind(rnorm(dimcluster,mean=centre[1,1],sd=0.2),rnorm(dimcluster,m
ean=centre[1,2],sd=0.2))
  for (i in 2:nrclustere)
  {grup=cbind(rnorm(dimcluster,mean=centre[i,1],sd=0.2),rnorm(dimcluste
  r,mean=centre[i,2],sd=0.2))
   date=rbind(date,grup)
    }
  return(date)
}

# grupare partitionala (algoritm kmeans) + vizualizare
grupareKmeans=function(date, nrclustere)
{
  # aplicare kmeans
  rez=kmeans(date,nrclustere)

  # vizualizare date - punctele din clustere diferite sunt colorate
diferit
  plot(date,col=rez$cluster)
  # marcare centri cu * si culori diferite
  points(rez$centers,col=1:nrclustere,pch=8)
  return(rez)
}
```

Obs.

1. Pachetul cluster (necesită încărcare prin library(cluster)) conține funcții care permit vizualizarea formei clusterilor. De exemplu prin: clusplot(date,rez$cluster,color=TRUE, shade=TRUE) se pot vizualiza formele clusterilor (vezi figura de mai jos).



2. Funcția kmeans permite specificarea variantei de algoritm prin parametrul algorithm. Varianta extinsă de apel este:

```
kmeans(x, centers, iter.max = 10, nstart = 1, algorithm =
c("Hartigan-Wong", "Lloyd", "Forgy", "MacQueen"), trace=FALSE)
```

2. Algoritmi ierarhici aglomerativi

Etapele principale în aplicarea unui algoritm ierarhic aglomerativ sunt:

1. Se construiește matricea de distanțe pornind de la datele de prelucrat. Pentru un set cu n date matricea are dimensiunea nxn iar pe linia i, coloana j se află distanța dintre elementul i și elementul j.

2. Construirea unei structuri arborescente binare folosind un algoritm aglomerativ:
   - la primul pas se identifică cele mai apropiate două date (determinând minimul din matricea de distanțe construită la pasul anterior) și se grupează în același cluster;
   - o dată construit un cluster se calculează distanțele dintre el și celelalte date sau clusteri și se actualizează matricea de distanțe;
   - la fiecare dintre pașii următori se procedează similar: se determină distanța minimă din matrice și se grupează entitățile corespunzătoare (pot fi date individuale sau clustere).

3. Se determină partiția prin secționarea arborelui la un anumit nivel.

2.1. Construirea matricii de distanțe.

In R pentru a construi matricea de distanțe corespunzătoare unui set de date se poate folosi funcția dist apelată prin dist(date,method="tip distanta"). Tipurile de distanțe care pot fi folosite sunt: "euclidean", "maximum", "manhattan", "canberra", "binary" sau "minkowski".

2.2. Crearea unei dendrograme.

Pornind de la o matrice de distanțe se poate construe dendrograma folosind funcția R hclust care în varianta cea mai simplă se apelează prin: hclust(matriceDistante,method="varianta")

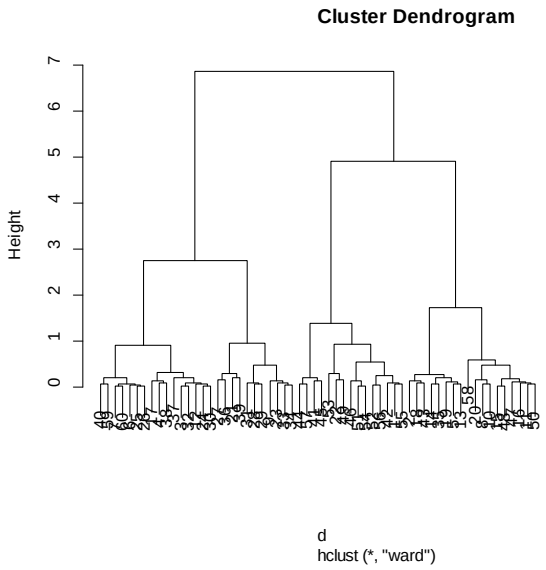
Variantele de clustering ce pot fi specificate în funcția hclust sunt: "ward", "single", "complete", "average", "mcquitty", "median" sau "centroid".

2.3. Vizualizarea unei dendrograme

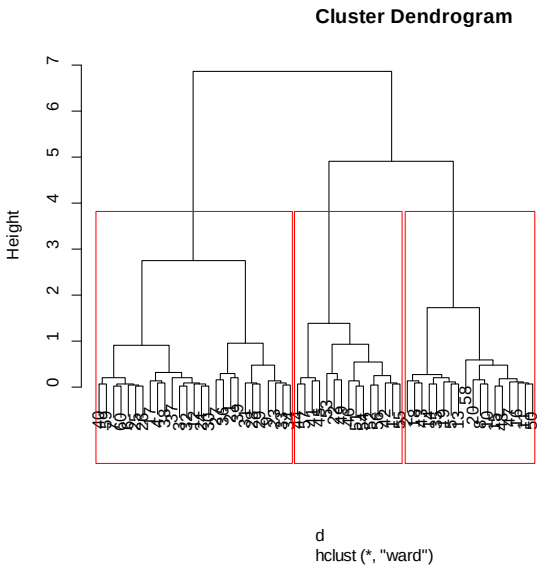
Dendrogramele construite cu hclust pot fi vizualizate direct prin plot.

Dacă se dorește secționarea dendrogramei astfel încât să fie identificați un număr de clusteri atunci se poate folosi cutree(dendrograma, k=nr clusteri)

Pentru vizualizarea pe dendrogramă a clusterilor identificați se poate utiliza rect.hclust(dendrograma, k=nr clusteri, border="culoare")



(a) vizualizare dendrograma



(b) vizualizare clustere marcate pe dendrograma

Exercițiul 2.

Pentru datele generate la exercițiul 1:

a) să se construiască matricea de distanțe (pentru distanța euclidiană)
b) să se construiască și să se vizualizeze dendrograma corespunzătoare următoarelor tipuri de distanțe: "average", "complete" și "single" precum și variantei "ward" – să se compare dendrogramele între ele
c) pe baza dendrogramelor construite la etapa anterioară să se determine partițiile corespunzătoare unui număr precizat de clusteri (de exemplu 3)

Indicație.

a. d=dist(date,method="euclidean").
b. dendrograma=hclust(d,method="single")
   plot(dendrograma)
c. partitie=cutree(dendrograma,k=3)

3. Biclustering

Există date (printre care și cele care conțin valori ale nivelului de exprimare a genelor) care nu sunt similar în raport cu toate atributele ci doar cu o parte dintre ele. In aceste situații se urmărește nu doar identificarea unor subseturi de date similare ci și a unor subseturi de atribute suficient de similare. Acesta este scopul algoritmilor de biclustering care realizează grupare atât a instanțelor cât și a atributelor. Principiul algoritmilor de biclustering este de a identifica subseturi de instanțe și atribute astfel încât variația datelor în aceste subseturi să fie cât mai mica. Unul dintre algoritmii incrementali de biclustering este cel propus de Cheng și Church (descrierea este in slide-urile de la cursul 10).

In R există diferite pachete destinate biclustering-ului. Unul dintre acestea este biclust care conține implementări corespunzătoare mai multor variante de algoritmi precum și funcții pentru analiza și vizualizarea rezultatelor.

Exercițiul 3. Testați funcțiile descrise mai jos:

```
library(biclust)  # incarcarea pachetului – necesită instalare prealabila

# generare date
set.seed(1)
# generarea a 30 de instante cu 30 de atribute – valori cu repartitia normala
x=matrix(rnorm(900),30,30)

# modificare date din 3 submatrici (pentru a crea biclustere)
x[1:5,1:5]=rnorm(25,3,0.3)
x[11:15,11:15]=rnorm(25,-3,0.3)
x[21:25,21:25]=rnorm(25,6,0.3)

# adaugare etichete pentru coloane (atribute)
colnames(x)=paste("Var.",1:30)

# aplicare algoritm de biclustering (Cheng-Church)
bicRez = biclust(x,BCCC(),delta=0.5,alpha=1.5,number=100)
```

```
# afisare continut biclustere
bicluster(x,bicRez,number=1:bicRez@Number)

# vizualizare grafica rezultat biclustering
biclustbarchart(x,bicRez)

# varianta de vizualizare in care coloanele sunt ordonate
ord=bicorder(bicRez, cols=TRUE, rev=TRUE)
biclustbarchart(x,bicRez,which=ord)

# vizualizare distribuire atribute la clustere
biclustmember(bicRez,x)

# vizualizare grafica biclustere folosind "harta de culori"
heatmapBC(x,bicRez,number = 0)

# reprezentarea grafica a valorilor corespunzatoare unui bicluster (liniile gri corespund datelor din
#celelalte clustere)
parallelCoordinates(x,bicRez,number = 1)

# prin parametrul number se specifica nr biclusterului de reprezentat
parallelCoordinates(x,bicRez,number = 1, plotBoth=TRUE )

# compararea valorilor din fiecare bicluster cu valoarea medie/mediana corespunzatoare setului
#de date
plotclust(bicRez,x,bicluster=TRUE)
```

Temă. Să se implementeze explicit algoritmul kMeans (variant descrisă în slide-urile de la curs 10).