

Data Mining

Lab 4: Gruparea datelor

Sumar:

- Scopul grupării datelor
- Algoritmi partiționali
- Algoritmi ierarhici
- Algoritmi bazați pe densitatea datelor
- Algoritmi bazați pe modele probabiliste

1. Scopul grupării datelor

Gruparea (clusterizarea) datelor are ca scop identificarea de subseturi de date similare (într-un anumit sens), numite grupuri, clustere sau clase. Elementul specific unui proces de grupare este faptul că nu se cunosc apriori grupurile (uneori nici măcar numărul acestora), scopul algoritmilor de grupare fiind de a identifica aceste grupuri folosind doar relațiile/similaritățile dintre date.

2. Algoritmi partiționali

Produc o partiționare a setului de date în clustere, fiecare cluster având asociat un reprezentant (denumit prototip, centroid sau centru). Cel mai simplu algoritm partițional este kMeans care se caracterizează prin faptul că pentru un set de date și un număr de clustere (K, specificat de către utilizator) produce un set de K centroizi care determină o alocare univocă a datelor la clustere (fiecare dată este alocată clusterului corespunzător celui mai apropiat centroid).

Structura generală a algoritmului kMeans este:

Etapa 1. Inițializare: se aleg aleator din setul de date K centroizi

Etapa 2. **Repeat**

- Asignează fiecare dată la clusterul corespunzător celui mai apropiat centroid
- Recalculează centroizii (ca medii aritmetice ale datelor din fiecare cluster)

until partiția s-a stabilizat (nu se mai schimbă conținutul clusterelor)

Observatii:

- Rezultatul grupării depinde de pozițiile inițiale ale centroizilor
- Procesul iterativ de la kMeans are ca scop minimizarea varianței intra-cluster (suma pătratelor distanțelor dintre date și centroidul corespunzător unui cluster)
- kMeans este adecvat pentru identificarea clusterelor “sferice” (ce corespund unor date generate folosind repartiții normale) însă nu este adecvat în cazul clusterelor cu formă arbitrară

Implementări Rattle/R:

- Rattle: **Cluster -> KMeans** (necesită specificarea numărului de clustere)
- R: funcția
`kmeans(x, centers, iter.max = 10, nstart = 1, algorithm = c("Hartigan-Wong", "Lloyd", "Forgy", "MacQueen"), trace=FALSE)`

Exercițiul 1/Rattle, R:

- a) Deschideți în Rattle fișierul “iris.2D.arff” și eliminați atributul de clasă. Identificați 3 grupuri în date aplicând algoritmul [KMeans](#). Vizualizați rezultatele folosind [Plots \(Data, Discriminant\)](#).
- b) Încărcați setul de date iris în R ([data\(iris\)](#)), aplicați funcția [kmeans](#) și analizați rezultatele. Punct de pornire: [ClusteringKMeans.R](#)

3. Algoritmi ierarhici

Algoritmii ierarhici permit construirea unei ierarhii de partiții (dendrogramă) obținută în una dintre variantele:

- **Aglomerativă (bottom-up):** se pornește de la partiția în care fiecare cluster conține o singură dată și succesiv se grupează clusterelor folosind un criteriu de similaritate ([single-link](#), [complete-link](#), [average-link](#)) – la fiecare etapă se pot reuni două sau mai multe clusterelor. Procesul continuă până când se ajunge la nodul rădăcină al structurii care corespunde întregului set de date.
- **Divizivă (top-down):** se pornește de la un cluster unic ce conține toate datele și succesiv se aplică o strategie de partiționare (poate fi bazată pe un algoritm partițional de tip [kMeans](#))

Obs: doar varianta aglomerativă este implementată în Weka

Exercițiul 2/Rattle, R:

- a) Deschideți în Rattle fișierul “iris.2D.arff” și eliminați atributul de clasă. Aplicați Hierarchical Clustering pentru diferite variante de distanțe ([Distance: Euclidean, Manhattan, Correlation](#)) și variante de strategii de aglomerare ([Agglomerate: Ward, Complete, Single, Average](#)). Vizualizați dendrograma folosind [Dendrogram](#).
- b) Încărcați setul de date iris în R ([data\(iris\)](#)), aplicați funcția [hclust](#) și analizați rezultatele. Punct de pornire: [HierarchicalClustering.R](#)

4. Algoritmi bazați pe estimarea densității

Această familie de algoritmi se bazează pe ideea că grupurile (clusterelor) se caracterizează prin densitate mare a datelor și sunt separate unele de altele prin “regiuni” cu densitate mică a datelor. Elementul cheie este reprezentat de identificarea unei măsuri adecvate a densității. Pentru a decide dacă o dată face parte dintr-un cluster sau este izolată (zgomot) se estimează densitatea datelor în vecinătatea ei. Două dintre cele mai cunoscute metode (care diferă prin modul în care este estimată densitatea) sunt:

- **DBSCAN** – se bazează pe estimarea densității numărând câte date se află în vecinătatea datei analizate (dimensiunea vecinătății este controlată printr-un parametru – [eps](#)). Datele care au un număr de vecini apropiați mai mare decât o valoare prag ([MinPts](#)) sunt considerate date care fac parte din nucleul ([core](#)) unui cluster. Celelalte date sunt considerate fie la frontiera unui cluster ([border](#)) fie izolate ([noise](#)). Datele de pe frontieră

se caracterizează prin faptul că nu sunt în regiune densă dar în vecinătatea lor se află cel puțin o data de tip nucleu.

- DENCLUE – se bazează pe utilizarea unor funcții de influență (similar funcțiilor nucleu – asociate tuturor datelor din set) și prin estimarea densității calculând suma valorilor funcțiilor de influență corespunzătoare tuturor datelor din set. Clusterelor sunt definite de maximele locale ale acestei funcții de densitate. Pentru a determina clusterul de care aparține o data se aplică un algoritm de tip gradient (pentru determinarea punctului de maxim local) folosind data respectivă ca aproximare inițială. Dacă valoarea densității este mai mică decât o valoare prag atunci se consideră data ca fiind izolată.

Exercițiul 3/R:

- a) Analizați influența parametrilor algoritmului DBSCAN asupra performanței acestuia. Set de date de test: DS3 din pachetul DBSCAN. Punct de pornire: [DBSCANexample.r](#)
- b) Comparați rezultatele obținute folosind DBSCAN cu cele obținute folosind varianta [OPTICS](#) (din pachetul DBSCAN)

Indicație: detalii privind pachetul DBSCAN la:

<https://cran.r-project.org/web/packages/dbscan/vignettes/dbscan.pdf>

5. Clustering folosind modele probabiliste (Expectation Maximization)

Ideea pe care se bazează acești algoritmi este că datele sunt generate de o mixtură de distribuții de probabilitate. La fel ca și în cazul algoritmului kMeans, numărul de clustere trebuie specificat ca parametru de intrare (corespunde numărului de modele incluse în mixtură) iar rolul algoritmului este să estimeze probabilitatea fiecărei componente a mixturii precum și parametrii tuturor acestor componente (de exemplu, în cazul în care se presupune că datele sunt generate folosind distribuții normale, trebuie estimate media și varianța/ matricea de covarianță pentru fiecare dintre componente). Acest proces de estimare are caracter iterativ: se pornește de la valori inițiale ale parametrilor și se iterează următoarele două etape:

- **Expectation:** pentru fiecare dată se estimează probabilitatea să fi fost generată de către fiecare dintre modelele componente (calculând valoarea funcției de densitate a fiecărui model pe baza valorilor curente ale parametrilor); aceste valori sunt denumite “responsibility values”.
- **Maximization:** pentru fiecare dintre modele se estimează
 - probabilitatea de mixare (ca medie pe tot setul de date a responsabilităților determinate la pasul anterior)
 - parametrii asociați (folosind metoda verosimilității maxime) – în cazul distribuției normale se calculează media și (co)varianța de selecție folosind valorile responsabilităților ca ponderi

Exercițiul 4/R:

- a) Utilizați implementarea din [EM.r](#) pentru a analiza funcționarea algoritmului EM în cazul prelucrării unor date scalare
- b) Analizați facilitățile oferite de pachetul [EMcluster](#) și utilizați-l pentru a prelucra datele din setul Iris (date [myiris](#) din pachetul EMcluster).

6. Exemple adiționale (scikit-learn)

kMeans:

http://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_assumptions.html#sphx-glr-auto-examples-cluster-plot-kmeans-assumptions-py

http://scikit-learn.org/stable/auto_examples/cluster/plot_cluster_iris.html#sphx-glr-auto-examples-cluster-plot-cluster-iris-py

Agglomerative clustering:

http://scikit-learn.org/stable/auto_examples/cluster/plot_agglomerative_clustering.html#sphx-glr-auto-examples-cluster-plot-agglomerative-clustering-py

Bioinfo: biclustering (pachet biclust din R) – analiza datelor de tip microarray (expresie genică).
Punct de pornire: [BioinfoBiclustering.r](#)

Temă. Implementați algoritmul kMeans în R (fără a utiliza funcția kmeans sau alte pachete).