Data Mining

Lab 5: Gruparea datelor

Sumar:

- Scopul grupării datelor
- Algoritmi partiționali
- Algoritmi ierarhici
- Algoritmi bazați pe densitate

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 dintre date.

2. Algoritmi partiționali

Produc o partiționare a setului de date în clustere, fiecare cluster având asociat un reprezentant (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 partitia s-a stabilizat

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

Implementare Weka:

- SimpleKMeans: varianta clasică a algoritmului; se poate opta între distanța euclidiană și distanța Manhattan precum și pentru diferite variante de inițializare a centrilor:
  - Random: centrii se aleg aleatory din date
  - kMeans++: cei k centri sunt inițializați parcurgând următoarele etape:
    - Pas 1: primul centru se alege aleator

        - Pas 2: se determină distanţa D(x) de la fiecare dată x din set la cel mai apropiat dintre centri si se construieşte o distribuţie de probabilitate (probabilitatea de a selecta elemental x este invers proporţională cu D(x)).
        - Pas 3: se selectează un nou centru folosind distribuţia de probabilitate construită la Pas 2
        - Pas 4: se repeat Pas 2 şi Pas 3 până când se selectează k centri.
    - Canopy: ideea principală este de a asigna succesiv datele la "pre-clustere" folosind două praguri de distanţe: T1 şi T2 (T1>T2) şi un proces iterativ ce constă în următoarele etape:
        - Pas 1: se selectează un element din setul de date şi se iniţiază un nou pre-cluster
        - Pas 2: toate datele din set care au distanţa mai mică decât T1 faţă de elemental ce a iniţiat noul pre-cluster sunt asignate la acesta. In plus, datele care au distanţa mai mica decât T2 sunt eliminate din setul initial
        - Pas 3: dacă mai există elemente în set neasignate unui pre-cluster se reia de la Pas 1
        - Pas 4: valorile iniţiale ale centrilor se stabilesc folosind pre-clusterele
    - Farthest First: primul centru se selectează aleator; următorii centri se aleg astfel încât să fie cât mai îndepărtaţi de centrii deja selectaţi.
- XMeans: este o variantă (doar în versiunile mai mici de 3.8) care estimează numărul de clustere (utilizatorul specifică o valoare minimă și o valoare maximă pentru numărul de clustere iar XMeans aplică KMeans pentru fiecare dintre valori și o alege pe cea care conduce la modelul de partiționare

Exercițiul 1:

a) Deschideti fișierul "iris.2D.arff" și eliminați atributul de clasă
b) Identificați 3 grupuri în date aplicând algoritmul kMeans (Cluster->SimpleKMeans). Vizualizati clusterele identificate prin click dreapta pe rezultat (din Result list) și Visualize Cluster Assignments
c) Comparați valorile obtinute pentru varianța intra-cluster (SSE = within cluster squared sum of errors) pentru diferite valori ale numărului de clustere (parametrul –N de la SimpleKMeans): 2,3,4,5
d) Comparați valorile obtinute pentru varianța intra-cluster (SSE = within cluster squared sum of errors) pentru diferite variante de inițializare (initializationMethod: Random, kMeans++, Canopy, Farthest First)
e) Aplicați pentru același set de date algoritmul EM (Expectation-Maximization) cu valorile implicite ale parametrilor

3. Algoritmi ierarhici

Algoritmii ierarhici permit construirea unei ierarhii de partiții (dendrogramă) obtinută in una dintre variantele:

- Aglomerativă (bottom-up): se pornește de la partiția in care fiecare cluster conține o singură data și succesiv se grupează clustere folosind un criteriu de similaritate (single-link, complete-link, average-link) – la fiecare etapă se pot reuni două sau mai multe clustere. 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:

a) Deschideti fișierul "data.csv"
b) Construiți și comparati dendrogramele corespunzătoare cazurilor în care se utilizează single-link, complete-link, average-link. Indicație: se selectează Cluster->Hierarchical. Pentru vizualizarea arborelui prin click dreapta pe numele rezultatului (in Result List) se selectează VisualizeTree

4. Algoritmi de grupare bazați pe densitate

Ideea algoritmilor bazaţi pe densitate este de a clasifica datele ca facând parte dintr-un cluster (core points, border points) sau ca fiind zgomot pe baza densităţii datelor. Pentru fiecare punct se estimează densitatea numarând câte date sunt într-o vecinătate de rază dată.

Obs:

- sunt utilizaţi în special pentru gruparea datelor spaţiale si permit identificarea de clustere de forme arbitrare.
- Începând cu versiunea 3.8 Weka nu mai include implementarea algoritmului DBSCAN

Exercițiul 3 (doar pt versiuni de Weka anterioare versiunii 3.8):

a) Deschideti fișierul "iris.2D.arff"
b) Aplicați algoritmul DBSCAN pentru diferite valori ale parametrilor Eps (raza vecinătății) și MinPoints (numărul minim de date în vecinătate) și comparați rezultatelor. Valori de test: Eps in {0.2,0.4,0.6,0.8}, MinPoints in {5,10,15,20,30}

5. Exemple din 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

DBSCAN: http://scikit-learn.org/stable/auto_examples/cluster/plot_dbscan.html#sphx-glr-auto-examples-cluster-plot-dbscan-py