Data Mining

Lab 4: Data Clustering

Summary:

- Aim of data clustering
- Partitional clustering algorithms
- Hierarchical clustering algorithms

1. Aim of data clustering

Data clustering aims to identify natural groups in data, i.e. subsets of similar data (according to a specific similarity measure) called clusters or groups or classes. The particularity of a clustering task is the fact that the class labels (or even their number) are not known apriori. The goal of a clustering algorithm is to identify the clusters by using the similarity between the data.

2. Partitional clustering algorithms

These algorithms provides a partition of the initial dataset in several clusters (usually the clusters are disjoint but there are also algorithms which lead to overlapping clusters, e.g. fuzzy clustering algorithms). In the case of crisp algorithms (which assign each data to only one cluster) each cluster has a cluster *representative* or cluster *prototype*. In the case when the cluster prototype is computed such that it is the average of the data from the cluster then it is usually called *centroid*.

The simplest and most popular partitional clustering algorithm is KMeans which generates a set of K centroids and assign each data to the closest centroid.

The general structure of the KMeans algorithm :

```
Step 1. Initialization: random selection of K centroids from the dataset
Step 2. Repeat
            • Assign each data to the cluster represented by the nearest centroid
            • Recompute the centroids (as averages of data in each cluster)
        until the partition has not been changed during the last iteration
```

Remarks:

- The clustering result is sensitive with respect to the initial values of the centroids
- The KMeans iterative process aims to minimize the intra-cluster variance (the average sum of the distances between data and the centroid of their corresponding cluster)
- KMeans is appropriate for "spherical" clusters (e.g. data generated by normal distributions) but do not provide a good clustering in the case of arbitrary shaped clusters.

Rattle/R implementations:

- Rattle: Cluster -> KMeans (it requires to specify the number of clusters)
- R: function

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

Weka implementations:

- SimpleKMeans: standard variant of the algorithm; the user can choose between the Euclidean and Manhattan distances. Also there are several variants for the initial selection of the centroids:
  - Random: the centroids are randomly selected from the data
  - kMeans++: a pre-clustering is used:
    - Step 1: first centroid is randomly selected
    - Step 2: the distances D(x) between each data x and the closest centroid are computed; based on these distances is computed a probability distribution (the probability to select x is proportional with D(x), i.e. the probability is higher if the distance is larger).
    - Step 3: select the next centroid using the distribution probability constructed at Step 2
    - Step 4: repeat Step 2 and Step 3 until all k centers have been selected
  - Canopy: the main idea is to sequentially assign the data to "pre-clusters" by using two thresholds: T1 and T2 (T1>T2) and an iterative process consisting of:
    - Step 1: select randomly a data used to initialize a new pre-cluster
    - Step 2: all data at a distance smaller than T1 wrt the seed of the new pre-cluster are assigned to this pre-cluster. Those for which the distance is smaller even than T2 are removed from the initial set.
    - Step 3: if there are data un-assigned to a pre-cluster go to Step 1
    - Step 4: the initial centroids for KMens are set based on the pre-clusters
  - Farthest First: the first centroid is randomly selected; the next ones are selected such that they are as far as possible from the already selected centroids.

- XMeans: is a variant which estimates the number of clusters (the user provides a minimal and a maximal value for the number of clusters and XMeans applies KMeans for each of these values and selects the variant leading to the best quality clustering – e.g. smallest intra-variance and largest inter-variance). Remark: only for versions of Weka less than 3.8.

Exercise 1/Rattle, R:

a) Open in Rattle the file "iris.2D.arff" and remove the class attribute. Identfy 3 clusters in data by using KMeans. Visualize the results using Plots (Data, Discriminant).
b) Open the complete iris dataset in R (data(iris)), apply the function kmeans and analyze the results. Starting point: ClusteringKMeans.R

Exercise 1/Weka:

c) Open the file "iris.2D.arff" and remove the class attribute
d) Identify 3 clusters in data by applying KMeans (Cluster->SimpleKMeans). Visualize the identified clusters by right clicking on the result (from Result list) and select Visualize Cluster Assignments
e) Analayze the values obtained for the intra-cluster variance (SSE = within cluster squared sum of errors) for several values of the number of clusters (the parameter –N from SimpleKMeans): 2,3,4,5

f) Compare the results obtained by using different methods for centroids initialization (initializationMethod: Random, kMeans++, Canopy, Farthest First)

g) Apply EM (Expectation-Maximization) to the same set of data using the default values for the parameters.

3. Hierarchical algorithms

These algorithms provide not only one partition but a hierarchy of partitions organized as a tree (*dendrogram*). The hierarchy can be obtained by one of the following approaches:

- *Agglomerative (bottom-up)*: at the beginning each cluster contains only one data and then at each step the most similar clusters are joined. The similarity between clusters can be measured using different criteria (*single-link*, *complete-link*, *average-link*). The merging process continues until all data belong to one cluster (this corresponds to the root of the dendrogram).
- *Divisive (top-down)*: the process starts with a unique cluster containing all data in the set and apply iteratively a partitional clustering strategy (which can be KMeans).

A particular partition can be obtained by cutting the dendrogram at a specific level.

Exercise 2/Rattle, R:

a) Open in Rattle the file "iris.2D.arff" and remove the class attribute. Apply Hierarchical Clustering for different types of distances (Distance: Euclidean, Manhattan, Correlation) and grouping variants (Agglomerate: Ward, Complete, Single, Average). Visualize the dendrogram by using Dendrogram.

b) Open the complete iris dataset in R (data(iris)), apply the function hclust and analyze the results. Starting point: HierarchicalClustering.R

Exercise 2/Weka:

a) Open the file "data.arff"

b) Construct and compare the dendrograms corresponding to the cases when different cluster similarity measures are used: single-link, complete-link, average-link. Hint: select Cluster->Hierarchical. To visualize the tree: right click on the result (from Result List) and select VisualizeTree

4. Clustering based on probabilistic models (Expectation Maximization)

The main idea of clustering based on probabilistic models is that the data are generated by a mixture of some probability distributions. As in the case of kMeans the number of clusters should be specified as input parameter (it corresponds to the number of models included in the mixture). The algorithm iterates two main operations:

- Expectation: estimate the responsibility values (probability of each data to be generated by each model) based on the current values of the mixing probabilities and of the model parameters (in the case of Gaussian models it is the average and the standard deviation)
- Maximization: compute the maximum likelihood estimates of the mixing probabilities using the current responsibility values as weights.

Exercise 3/R:

a) Use the implementation in EM.r to analyze the structure of the EM algorithm and its behavior in the case of clustering scalar values.

b) Analyze the facilities offered by the EMcluster package and use it to cluster the iris dataset (data myiris from EMcluster package).

5. Density based algorithms

This family of algorithms relies on the idea that the clusters are characterized by high density of data separated by regions of low density. The key idea is to define an appropriate measure of density and use it to decide if a data belongs to a cluster or is just noise (outliers with respect to the clusters defined by the dense regions). Two of the most popular approaches are:

- DBSCAN – it relies on estimating the density corresponding to a data by counting the number of other data in its neighborhood (the neighborhood is defined by the eps parameter). The data are classified in core (the number of other data in their eps-neighborhood is at least MinPts), border (it is not a core point but its eps-neighborhood contains at least one core point) and noise (it is neither core nor border point).
- DENCLUE – it relies on using some influence functions (similar to kernels, e.g. Gaussian) and defining a density at a point as the sum of the influence functions corresponding to all the other points. The clusters are defined by the local maxima of the density function. To find the cluster to which a data belongs, a local maximization method (e.g. gradient-like method) should be applied by using that data as initial approximation.

Exercise 4/R:

a) Analyze the influence of the DBSCAN parameters on the performance of the algorithm using the dataset DS3 from the DBSCAN package. Starting point: DBSCANexample.r

b) Compare the results obtained by DBSCAN with those obtained by using the OPTICS variant (in DBSCAN package)

Hint: see https://cran.r-project.org/web/packages/dbscan/vignettes/dbscan.pdf for details on the DBSCAN package.

6. Additional: examples from 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

Homework:

1. Implement kMeans in R (without using the kmeans function or other packages).