Data Mining

Lab 4: Data Clustering

Summary:

- Aim of data clustering
- Partitional clustering algorithms
- Hierarchical clustering algorithms
- Density based algorithms
- Clustering based on probabilistic models

Learning outcomes:

- Understand when and how to use clustering methods
- Learn to use implementations of traditional clustering algorithms and how their parameters influence the result of the clustering process
- Learn to evaluate the quality of a clustering result

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, i.e. data belonging to the same cluster are more similar than data belonging to different clusters.

2. Partitional clustering algorithms

These algorithms provide 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 assigns each data to the closest centroid.

The general structure of the KMeans algorithm (Lloyd variant) :

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 (a variant which reduces the risk of selected initial centroids which are close one to each other is kMeans++ variant)

- 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) and it does not provide a good clustering in the case of arbitrary shaped clusters.

Other resources:
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

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, Ward 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.

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

4. 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). The clusters are identified by using the concept of connectivity (two connected data should belong to the same cluster) which relies on the existence of a high density path between the data.

- 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.

Other resources:
https://scikit-learn.org/stable/modules/clustering.html#dbscan
https://scikit-learn.org/stable/modules/clustering.html#optics

5. 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 or covariance matrix).
- Maximization: compute the maximum likelihood estimates of the mixing probabilities using the current responsibility values as weights.

Other resources:

https://machinelearningmastery.com/expectation-maximization-em-algorithm/
https://scikit-learn.org/stable/modules/mixture.html#mixture **(2.1.1)**