Data Clustering

Jerónimo Arenas-García

Universidad Carlos III de Madrid

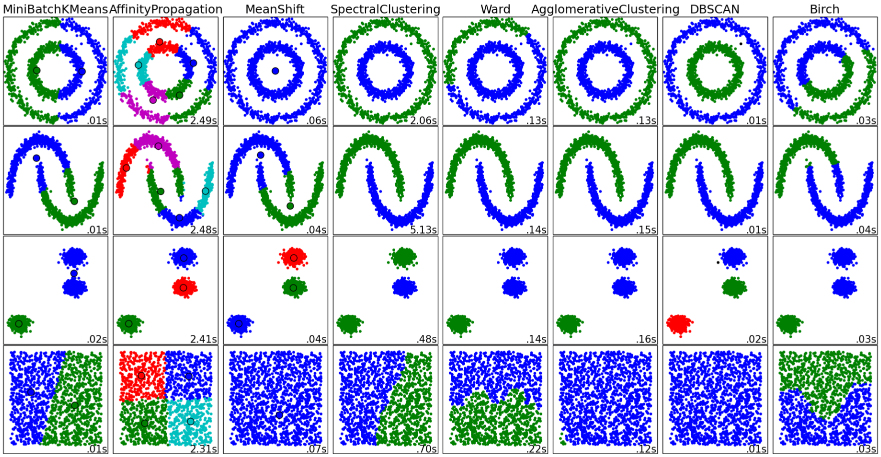
jarenas@tsc.uc3m.es

November 2, 2015

Data clustering

- Clustering algorithms try to split a set of data points into mutually exclusive clusters
- Some common assumptions:
    - Points in some cluster should lie close to each other
    - Points in different clusters should be far away
    - Clusters should be separated by regions of low density of points
    - Clustering should preserve "connectivity"
    - Clusters may get represented by a representative or centroid

Illustration of some algorithms



Taken from
http://scikit-learn.org/stable/modules/clustering.html

Scikit Learn implementations

We will study the following algorithms included in the Scikit-learn module:

- K-means
- Spectral clustering
- Agglomerative clustering

We will illustrate the performance of the algorithms with the following data

- Synthetic data
- Digit recognition dataset
- Images

You could also work with your text documents from the previous session, using the "topic representation" of each document.

K-means (1957)

K-means is a centroid-based method that tries to minimize the following distortion function

$$D = \sum_{k=1}^{K} \sum_{\mathbf{x} \in \mathcal{G}_k} \|\mathbf{x} - \mu_k\|_2^2$$

After initialization of centroids:

1. Assign each data point to closest centroid
2. Recalculate centroids positions
3. Go back to 1 until no further changes or max iterations achieved

K-means initializations

K-means convergence is guaranteed ... but just to a local minimum of D.

Different initialization possibilities:

1. k-means++: To maximize inter-centroid distance
2. Random among training points
3. User-selected

Typically, different runs are executed, and the best one is kept.

Check out `http://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html` for parameters, attributes, and methods.

Spectral clustering

The key idea of spectral clustering is to search for groups of connected data. I.e, rather than pursuing convex clusters, spectral clustering allows for arbitrary shape clusters.

Preliminary steps:

1. Calculate distances among each pair of training points
2. Obtain an affinity (i.e., similarity) matrix. In doing so, we can set to zero values below a threshold.

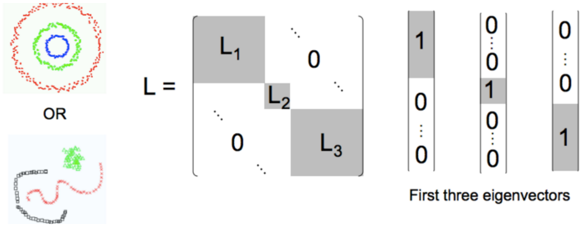
$$A_{i,j} = \exp\left(-\alpha \|\mathbf{x}_i - \mathbf{x}_j\|_2^2\right)$$

Spectral clustering (Graph cutting algorithms)

We split data into groups, so that links between points in the same group have small values, whereas "across-groups" links can be have large weight.

Main steps:

1. Obtain the Laplacian of the Graph ($L = D - A$), or a normalized version (variants)
2. Obtain K leading eigenvectors, and project onto them all training data
3. Form clusters in the new space, using, e.g., K-means



Taken from
`https://charlesmartin14.wordpress.com/2012/10/09/spectral-clustering/`

Spectral clustering (Graph cutting algorithms)

The algorithm in Scikit-learn requires the number of clusters to be specified. It works well for a small number of clusters but is not advised when using many clusters and/or data.

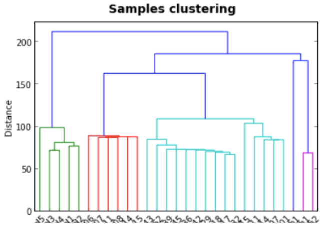
`http://scikit-learn.org/stable/modules/generated/sklearn.cluster.SpectralClustering.html`

Agglomerative/Hierarchical clustering

Bottom-up approach:

- At the beginning, each data point is a different cluster
- At each step of the algorithm two clusters are merged, according to certain performance criterion
- At the end of the algorithm, all points belong to the root node

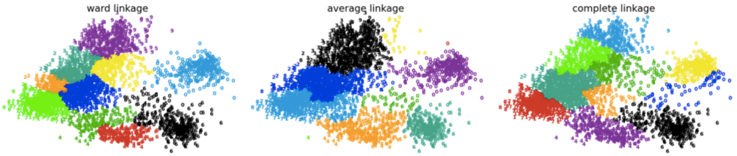
In practice, this creates a hierarchical tree, that can be visualized with a dendogram. We can cut the tree at different levels, in each case obtaining a different number of clusters.



Criteria for merging clusters

We merge the two closest clusters, where the distance between clusters is defined as:

- Single: Distance between clusters is the minimum of the distances between any two points in the clusters
- Complete: Maximal distance between any two points in each cluster
- Average: Average distance between points in both clusters
- Centroid: Distance between the (Euclidean) centroids of both clusters
- Ward: We merge centroids so that the overall increment of *within-cluster* variance is minimum.



Python implementations

Hierarchical clustering may lead to clusters of very different sizes. Complete linkage is the worst strategy, while Ward gives the most regular sizes. However, the affinity (or distance used in clustering) cannot be varied with Ward, thus for non Euclidean metrics, average linkage is a good alternative.

There are at least three different implementations of the algorithm:

1. Scikit-learn: Only implements ‘complete’, ‘ward’, and ‘average’ linkage methods. Allows for the definition of connectivity constraints
2. Scipy
3. fastcluster: Similar to Scipy, but more efficient with respect to computation and memory.