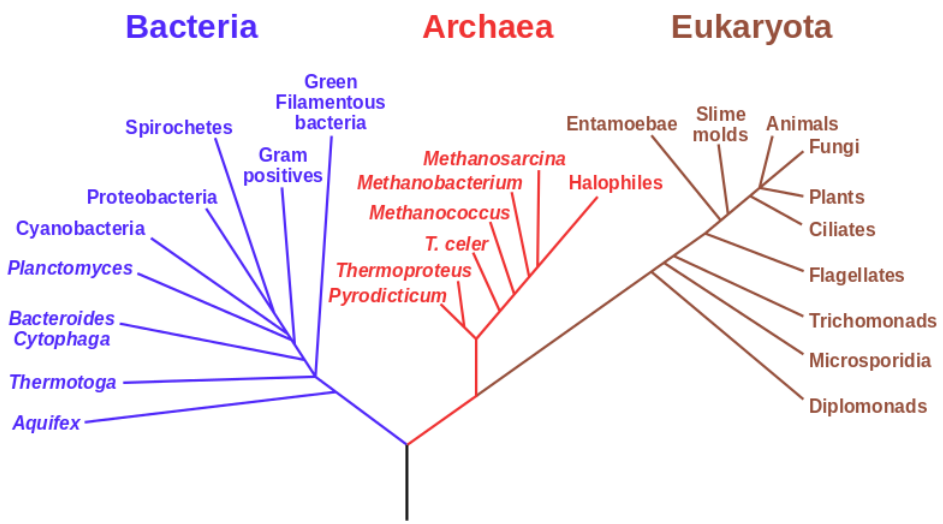
Hierarchical Unsupervised Methods

Dec, 2017

Introduction

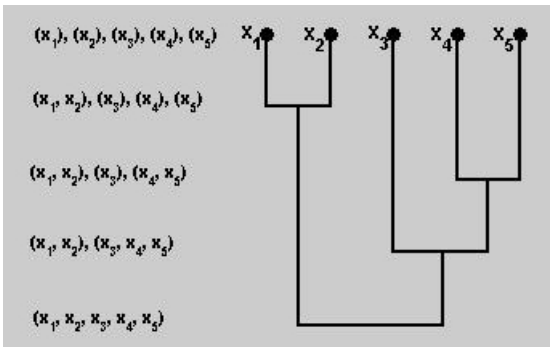
- **Hierarchical methods can be used for clustering data**
  - Clustering is a subset of unsupervised techniques (Auto-encoders and Word2vec are other unsupervised algorithms for different purposes)
- First, we construct a decision tree on the data, and there are two tree construction methods: leaf-to-root or **Agglomerative** vs. root-to-leaf or **Divisive**)
- And then another algorithm that operates on the tree **depth** determines the cluster memberships (extraction step)
- The following figure is called a **Dendrogram**



Agglomerative clustering

- Agglomerative clustering is the simplest method of the two (in terms of computation):
  1. Initially, assign each observation to its own cluster.
  2. Compute the similarity (or generalized distance) between each of the clusters and join the two most similar clusters.
  3. Repeat step 2 until there is only a single cluster left.

$t = 0$
Choose $R_0 = [C_i = x_i, i = 1, ..., N]$ as initial clustering
Repeat
  t=t+1
  Find the closest clusters C_i, C_j in the existing clustering R_{t-1} such that
    $g(C_i, C_j) = \max_{r,s}(C_r, C_s)$ if g is similarity function
    $g(C_i, C_j) = \min_{r,s}(C_r, C_s)$ if g is dissimilarity function
  Define $C_q = C_i \cup C_j$ and produce the new clustering $R_t = [R_{t-1} - C_i - C_j] \cup C_q$
Until only one cluster is left



Divisive clustering

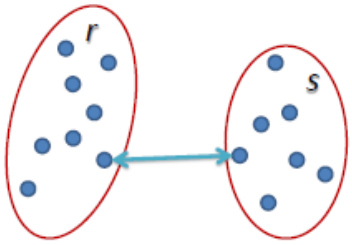
- Divisive clustering requires more computation than agglomerative
  1. Initially, all sample units are in a single cluster of size n.
  2. Clusters are partitioned into a pair of daughter clusters, selected to maximize the distance between each daughter.
  3. Repeat step 2 until all sample units are partitioned into n clusters of size 1.

$t = 0$
Choose $R_0 = X$ as initial clustering
Repeat
  t=t+1
  For i = 1 to t,
    Among all possible pairs of clusters (C_r, C_s) that form a clustering $C_{t-1,i}$
    find the pair $(C^1_{t-1,i}, C^2_{t-1,i})$ with the largest dissimilarity.
  End
  From the t pairs of clusters defined at the previous stage, choose the one
  with the largest dissimilarity
  Let this pair be $C^1_{t-1,j}, C^2_{t-1,j}$
  Define the new clustering $R_t = R_{t-1} - C_{t-1,j} \cup (C^1_{t-1,j}, C^2_{t-1,j})$
  Relabel the clusters
Until each vector forms its own cluster

Linkage functions

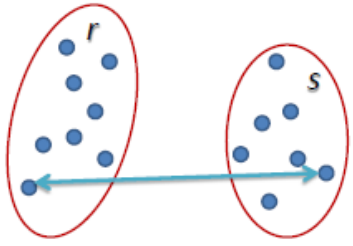
- Both tree construction methods rely on computing a linkage matrix between clusters. The aggregate distance function or **linkage** between two clusters can be the following:

Single



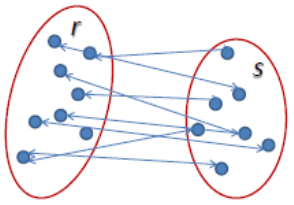
$$L(r,s) = \min(D(x_{ri}, x_{sj}))$$

Complete



$$L(r,s) = \max(D(x_{ri}, x_{sj}))$$

Average

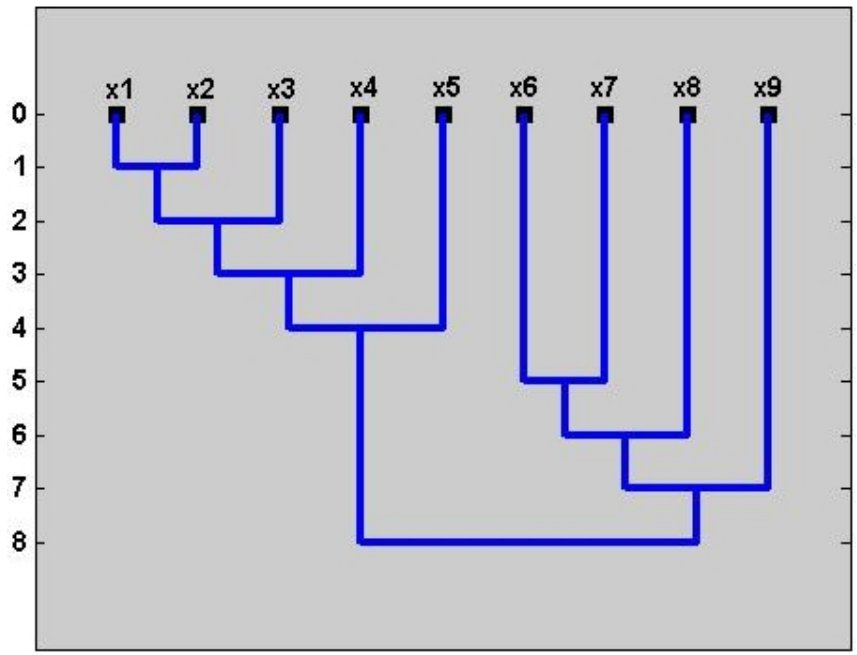


$$L(r,s) = \frac{1}{n_r n_s} \sum_{i=1}^{n_r} \sum_{j=1}^{n_s} D(x_{ri}, x_{sj})$$

- Other linkages such as centroid and minimum energy can be used.
- Notes on which linkage to use
- The distance function $D(x_{ri}, x_{sj})$ can be any general symmetric distance (e.g. Euclidean, Manhattan, Mahalanobis, Word Mover's Distance, Dynamic Time Warp Distance)

Choosing the number of clusters

- The 'lifetime' of a cluster is defined as the difference between the depth level at which the cluster was created and the depth level at which the cluster was combined with another cluster.



- There are several meta-algorithms to determine how to extract the the number of clusters from the tree.
- One method is the 'cut at depth' method which determines a depth level to cur the tree based on some linkage threshold. *What is the problem with this method?*

Choosing the number of clusters

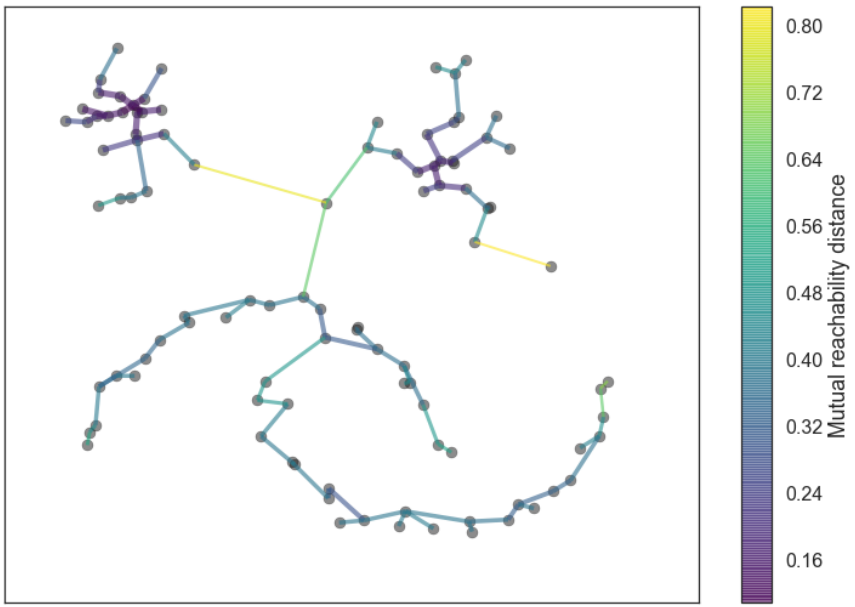
- Unfortunately, clustering algorithms cannot in general give you an 'optimal' cluster number k^*.
- In fact, different runs on the same dataset of the same algorithm can give you different results (different trees in this case).
- In general, you have to approach clustering in two steps:
  1. Approximate the optimal k using some bootstrapped method (such as k-means with Pseudo-F)
  2. Using the approximated optimal k to perform an ensemble clustering on bootstraps and average the cluster memberships
- The scikit-learn agglomerative clustering function is here: http://scikit-learn.org/stable/modules/generated/sklearn.cluster.AgglomerativeClustering.html

Summary of Hierarchical methods

- **Pros**
- Suitable when data has presumed hierarchical structure (e.g. the correlations in financial markets)
- Can handle categorical variables, whereas basic k-means cannot
- Can use general distance functions, k-means is restricted to Euclidean
- **Cons**
- Hierarchical clustering is $O(n^2)$ whereas k-means is $O(n)$: can be a lot slower than k-means
- Hierarchical clustering is at its core a greedy algorithm. Greedy algorithms can miss global extrema for different hyperparameters.

HDBSCAN

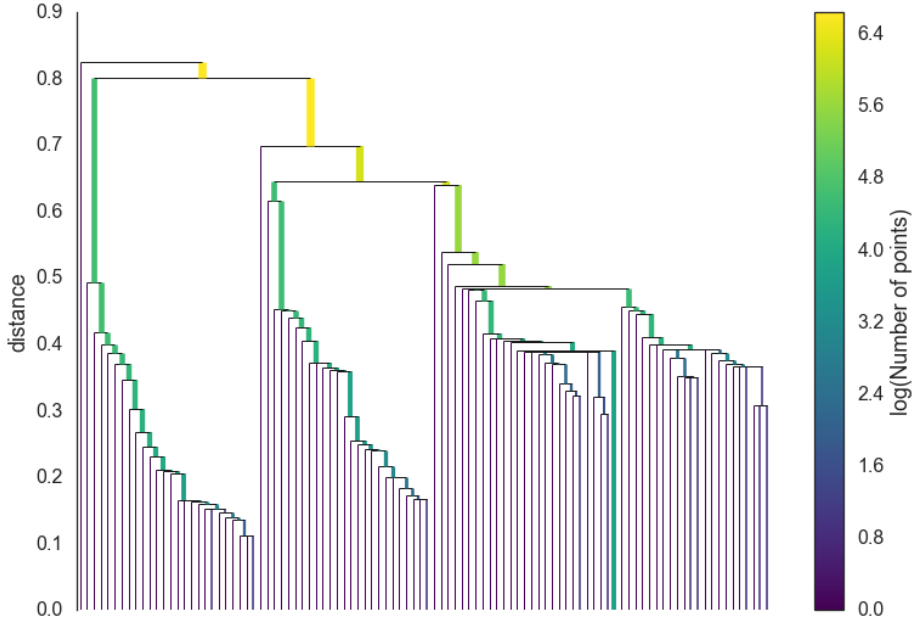
- Hierarchical DBSCAN or HDBSCAN is an extension of the DBSCAN algorithm that is more robust against parameter changes (i.e. should give you more similar clusterings across different runs of same dataset or if you change the hyperprameters)
- HDBSCAN constructs a minimally spanning tree of the data:



- The span distance is the 'mutual reachability' which is a measure of spread of points with low density

HDBSCAN

- Then we construct a hierarchical form of the minimally spanning tree



- Finally, we extract the clusters from this hierarchy.
- Please read this article to fully understand HDBSCAN: http://hdbscan.readthedocs.io/en/latest/how_hdbscan_works.html
- You can install the corresponding Python package from the above website and run it on some toy data.