Clustering: K-means and Hierarchical Clustering

Examples
- Expression profiles for genes via microarrays or sequencing
- Sequences from organisms sampled in the environment ("metagenome")
- Morphological features of tumor cells

Consider a set of data points, say corresponding to points in some δ dimensional space. Is the data clustered? How many clusters are there & which points are in which cluster? Finding clusters can be very useful but there is no universally best way of clustering - one has to think about where the data comes from & what clusters might mean when choosing a clustering algorithm.

Two popular, but very different, clustering algorithms are K-means & Hierarchical Clustering.

- K-means

Idea: Try to find K clusters each of which is as compact as possible (algorithm doesn't tell us best K to use).

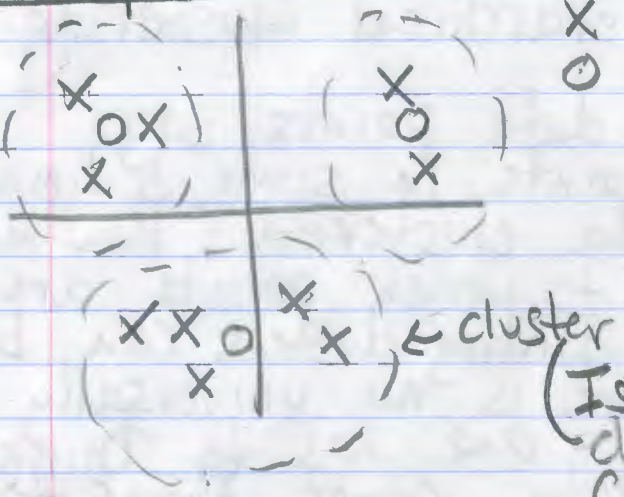
Definition: Define clusters to minimize

$$\text{Error} = \sum_{i=1}^{K} \sum_{j \in S_i} \| \vec{x}_j - \vec{c}_i \|^2$$

where $\vec{x}_j$ are data points divided among K clusters S_j, where $\vec{c}_i$ is the centroid of each cluster.

[In practice, often start with PCA to reduce dimension, then visualize and do clustering.]

Example K=3



x - data point
o - centroid of cluster

For this data set K=3 makes sense. What would happen for K=2? for K=4?

(Is there a delineration between clusters found w/ K & those found w/ K+1 or K-1? Note method is not hierarchical.)

Algorithm: To try to minimize Error, start by putting the K "centroids" somewhere (different options here, e.g. pick a random data point as the first centroid, then choose a different data point for the next centroid to maximize total distance between all centroids).

Next, assign each data point to the nearest centroid. This gives a set of K clusters. For each cluster find the new centroid. Repeat until convergence.

(Generalization - "Fuzzy" K-means assigns points to clusters in a weighted manner.)

Caveats: Will this iterative algorithm always find the best solution? No, can get stuck in local minimum. What to do? Try starting from many different initial centroid choices to find the minimum Error solution. (If one always gets a different solution, maybe we're using the wrong K, or the data is not clustered...)

Aside:

What is the best value of K?
Higher K will always reduce error, but w/ making K bigger leads to overfitting. (When K = # data points, Error = 0, but we learn nothing.)

For aficionados: Instead of minimizing Error, minimize

$$\text{Error} + \lambda D K \log K$$

(D: dimension; K: centroids; K: # data points)

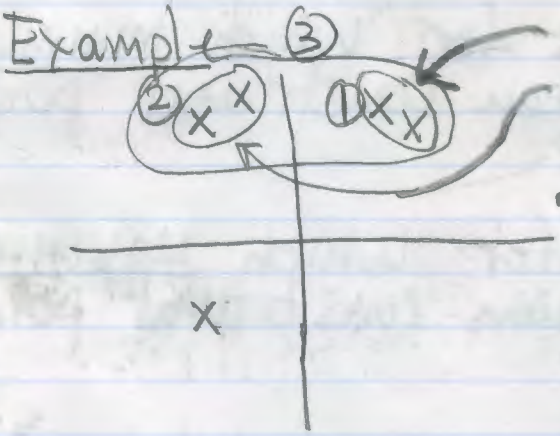
"Schwarz' Criterion".

- Agglomerative Hierarchical Clustering

Idea: Build clusters progressively - works well if the data comes from a tree, think phylogeny of species...

[Compare "Divisive HC" - start with one cluster and divide.]

Example



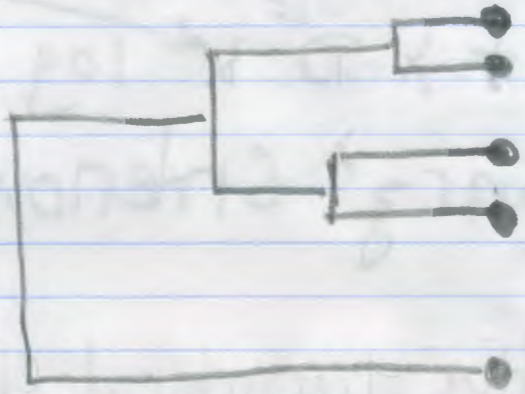
- 1st cluster closest points
- next cluster remaining closest points
- Treat existing clusters as "points", e.g. using centroids of each cluster, so next cluster two pre-existing clusters.
- Etc.

Notice that at each stage the clusters depend on the clusters present at the previous stage, i.e. the methods "hierarchical".

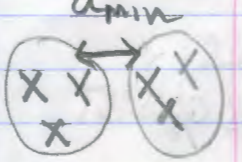
Example (continued)

- Keep clustering until there is only one cluster left.

Notice that unlike K-means, one is left with a tree that connects all the data points, e.g.:

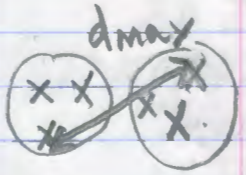


Algorithm: Different versions of HC use different metrics (e.g. Euclidean, "Manhattan") + criteria for when clusters to join:



- Join clusters for which <u>minimum</u> distance between constituent points is smallest

or



- Join clusters for which <u>maximum</u> distance between constituent points is smallest

or

- Join clusters so that total within-cluster variance is minimized ("Ward's Method")

Caveats: Different versions of HC lead to different trees, e.g. minimum distance rule will connect strings of points, whereas maximum distance rule won't.

Take-home message -

The type of data + where the data come from are important in choosing a good clustering algorithm. It's up to you to make sure that your clusters make sense!