Data clustering

Clustering is an inherently ill-defined problem since the correct clusters depend upon context and are in the eye of the beholder.

How many clusters are there?

Biological applications of clustering include grouping sequences/ESTs by similarity, genes by similar expression patterns, and samples by tissue type/disease.

Biological clustering is often associated with *dendograms* or phylogenic trees, although there is no reason there *need* be a tree associated with clusters.

Distance Measures

Certain mathematical properties are expected of any distance measure, or *metric*:

1. $d(x, y) \geq 0$ for all x, y.
2. $d(x, y) = 0$ iff $x = y$.
3. $d(x, y) = d(y, x)$ (symmetry)
4. $d(x, y) \leq d(x, z) + d(z, y)$ for all x, y, and z. (triangle inequality)

Distance Measures

Euclidean distance $d(x,y) = \sqrt{\sum_{i=1}^{d} |x_i - y_i|^2}$ is probably the most commonly used metric. Note that it weights all features/dimensions "equally".

Variations on Euclidean distance include L_k norms for $k \neq 2$:

$$d(x,y) = (\sum_{i=1}^{d} |x_i - y_i|^k)^{(1/k)}$$

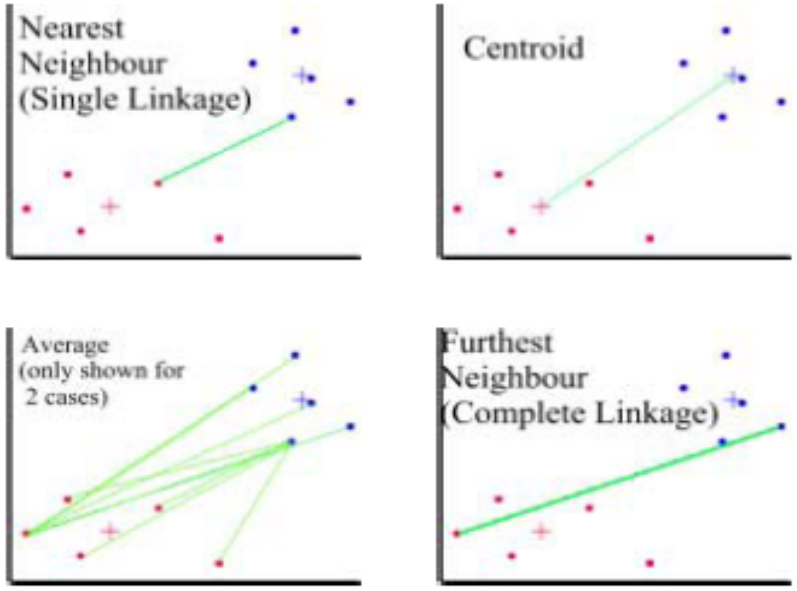
We get the "Manhattan" distance metric for $k = 1$ and the "maximum" component for $k = \infty$.

The *correlation coefficient* of sequences X and Y, yield a number from -1 to 1 but is *not* a metric.

The loss of the triangle inequality in non-metrics can cause strange clustering behavior, particularly in single linkage methods.

Distance measures between clusters

There are several ways to measure the distance between two clusters:



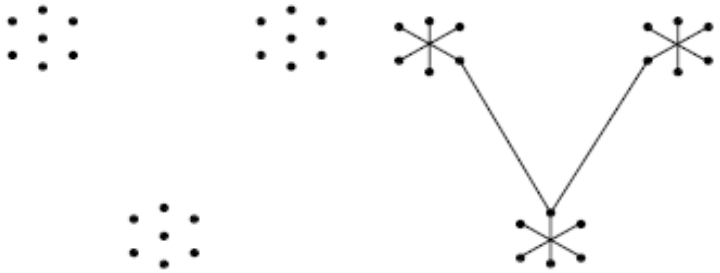
The choice of measure implies a tradeoff between computational efficiency and robustness.

Agglomerative clustering

Such methods merge smaller clusters into bigger clusters.

The simplest merging criteria is to join the two *nearest* clusters.

Minimum spanning tree methods define *single-link* clustering.



But does nearest mean the closest pair of examples? The closest mean/median examples?

In *complete link* clustering, we merge the pair which minimizes the maximum distance between elements. This is more expensive ($O(n^3)$ vs. $O(n^2)$), but presumably more robust.

K-means clustering

k-*Means clustering* and self-organizing maps (SOMs) are iterative clustering techniques where you specify the number of clusters k in advance.

Pick k points, and assign each example to the closest of the k points.

Pick the 'average' or 'center' point from each cluster, and repeat until sufficiently stable.

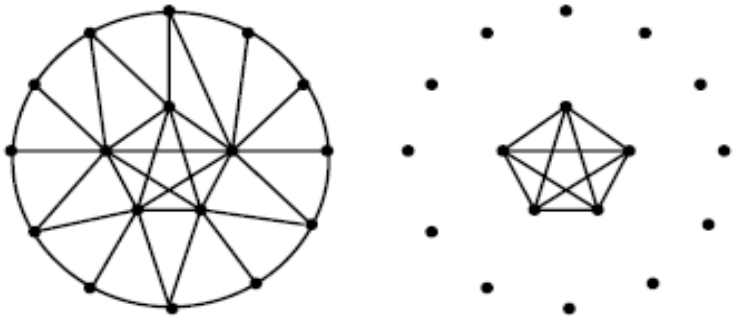
The question of how many clusters you have (or expect) is inherently fuzzy in most applications.

Such techniques represent a *partitioning*-based strategy, which is dual to the notion of agglomerative clustering.

Graph-theoretic clustering methods

An alternate approach to clustering constructs an underlying *similarity* graph, where elements i and j are connected by an edge iff i and j are similar enough that they should/can be in the same cluster.

If the similarity measure is totally correct and consistent, the graph will consist of disjoint cliques, one per cluster.



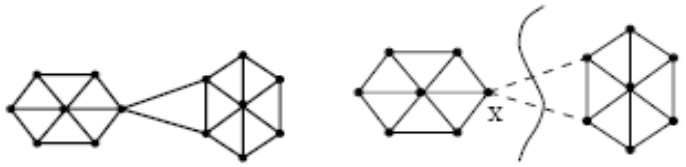
Overcoming spurious similarities reduces to finding maximum-sized cliques, an NP-complete problem.

Overcoming occasional missing edges means we really seek to find sets of vertices inducing dense graphs.

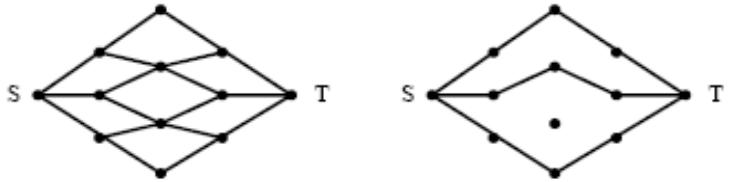
Repeatedly deleting low-degree vertices gives an efficient algorithm for finding an induced subgraph with minimum degree k, if one exists.

Cut-based algorithms

A real cluster in such a graph should be easy to disconnect by deleting a small number of edges. The *minimum edge cut* problem asks for the smallest number of edges whose deletion will disconnect the graph.



The *max flow / min cut theorem* states that the maximum possible flow between vertices i and j in a weighted graph G is the same as the minimum total edge weight needed to separate i from j. Thus network flow can be used to find the minimum cut efficiently.



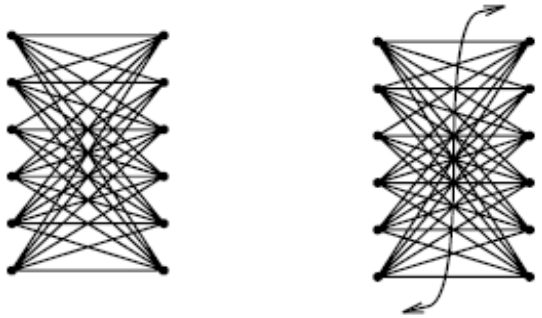
However the globally minimum cut is likely to just slice off a single vertex as opposed to an entire cluster.

Graph partitioning approaches

Graph partitioning problems which seek to ensure large components on either side of the cut tend to be NP-complete.

By *complementing* graph G, we get a graph G' with an edge (i, j) in G' iff the edge did not exist in G.

By finding the *maximum cut* in G' and recurring on each side, we can have a top-down (i.e. divisive) clustering algorithm.



Finding the maximum cut is NP-complete, but an embarrassingly simple algorithm gives us a cut which is on average half as big as the optimal cut:

For each vertex, flip a coin to decide which side of the cut (left or right) it goes. This means that edge (i, j) has a 50% chance of being cut (yes if coins i and j both came up either heads or tails).

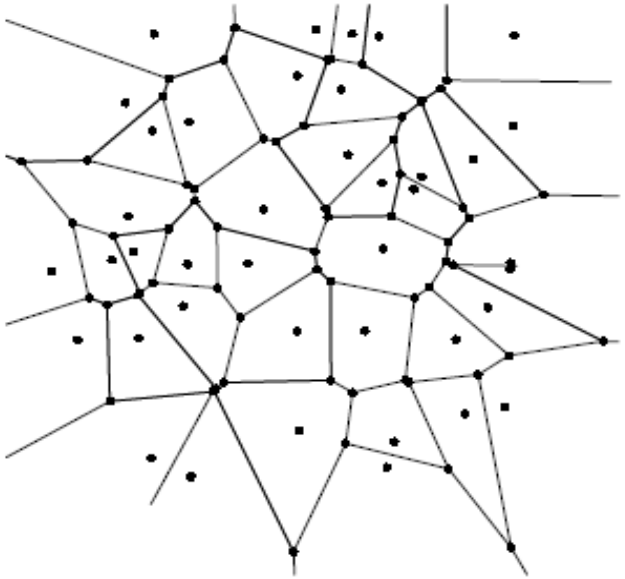
Local improvement heuristics can be used to refine this meaningless cut.

Nearest neighbor clustering

Assign each point to its nearest neighbor who has been clustered if the distance is sufficiently small.

Repeat until there are sufficient number of clusters.

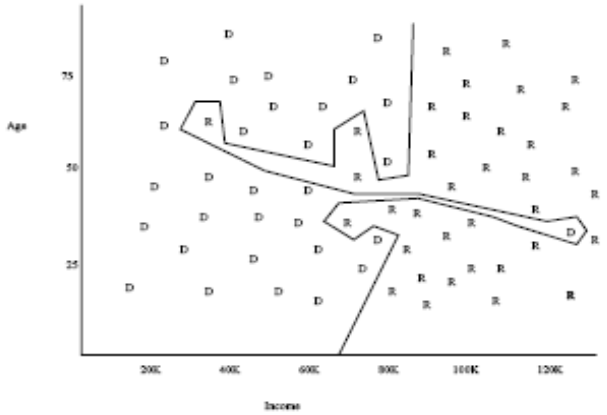
Nearest neighbor *classifiers* assign an unknown sample the classification associated with the closest point.



Conceptually, all classifiers carve up space into labeled regions. *Voronoi diagrams* partition space into cells defining the boundaries of the nearest neighbor regions.

Nearest neighbor clustering

Nearest neighbor classifiers are simple and reasonable, but not robust. A natural extension assigns the majority classification of the k closest points.



An important task in building classifiers is avoiding *over-fitting*, training your classifier to replicate your data too faithfully.

Other classifier techniques include *decision trees* and *support-vector machines*.