CSE 527 Lecture Notes - 10/13/03
prepared by: Martha Mercaldi (mercaldi@cs)

Clustering Expression Data

Why Cluster?

- Tissue classification
- Identify biologically related genes (sporulation study)
- 1st step in understanding regulatory networks (so that we might intervene in some way in the pathway)
- Identify common promoter element
- Hypothesis generation (as opposed to hypothesis validation/testing)
- Tool for expression analysis (both widely used and widely criticized in this capacity, because you can only *support a hypothesis not* confirm it.

What has been done?

- Heirarchical average-link (Eisen et. al '98)
- Self Organizing Maps (Tamayo et. al '99)
- CAST (Ben-Dor et. al '99)
- Support Vector Machines (Grundy et. al '00)

Why so many methods?

Clustering is NP-hard, even with simple objectives and data. In practice it is even harder due to dimensionality of the data and noise. Therefore many heuristics have been applied (local search, approximation algorithms) among which there is no clear winner. The good news, however, is that the data is good enough that even the results of imperfect algorithms look good.

Clustering Algorithms

Agglomerative Heirarchical Algorithms

(From Ka-Yee's Clustering 101 Slides)

Algorithm Inputs

The algorithms are run on the raw data matrix (genes X experiments) which can also be transformed into a similarity matrix (genes X genes). Additionally some algorithms might be parametrized with the number of expected clusters or other information.

Algorithm Classification

- heirarchical v. partitional (bottom up v. top down)
- heuristic v. model-based
- soft (fuzzy, probabilistic membership) v. hard (each data point is in a unique cluster)

Single Link

A bottom up algorithm in which the two closest clusters (as measured by the distance between their nearest points) are merged. Running this algorithm is analogous to finding the minimum spanning tree of the data.

Complete Link

Same as the single link, except that the measure of two clusters' closeness is the distance between their most distant points. This is the dual of the single link. Runtime cubic in the number of data points, $O(n^3)$.

Average Link

Here the clusters are formed hierarchically, as above, but the average similarity between all pairs of points in the two clusters is used to choose which two should be merged. This algorithm tends to produce tight clusters, though not as tight as complete link. It is also somewhat slow (presumably due to the large number of link calculations to be run).

Centroid Link

Faster than Average Link where in each round merge the two clusters with the closest centroids (the average position of all points in that cluster).

TreeView

A software program that takes some set of heirarchical clusters and produces a red/green picture of gene expression annotated with cluster information. Tree diagrams, such as this, are called *dendrograms*. These trees can be pivoted at each node without changing the information, causing there to be an exponential number of configurations, but a polynomial time algorithm exists to order the leaves (it is based on dynamic programming).

Divisive Hierarchical AKA. Partitional Algorithms

K-Means

To create k clusters:

- Pick k random points (call them flags) and divide data, assigning each data point to the nearest flag
- Calculate the centroid of each cluster and move the flag to that point.
- Reassign data points
- Iterate until there is no change in the centroids/flags' positions

This partitions the data into k groups, minimizing the value of the following objective function:
$\sum_{i=1}^{k} \sum_{x \in C_i} (x - Centroid(C_i))^2$
However it does not guarantee a globally optimal solution (it is just a local search algorithm). Also, it can be helpful to initialize the K-Means algorithm with the results of some other simple algorithm instead of just random points. This algorithm tends to produce spherical clusters of roughly equal size.

Agglomerative v. Partitional Algorithms

Agglomerative algorithms take an irreversible first decision whereas K-Means allows you to "back up" and correct it. You also are looking locally at the data at the start, which makes the algorithm particularly bad for noisy data.

Self-Organizing Maps

The basic idea is that you map high dimensional data into a two-, or perhaps three dimensional, grid. Neighboring nodes are considered more similar that those farther away. Points are mapped to the nearest node, and the nodes are then moved to reflect the centroid of each cluster.