An Overview of Clustering Methods

What is Clustering?

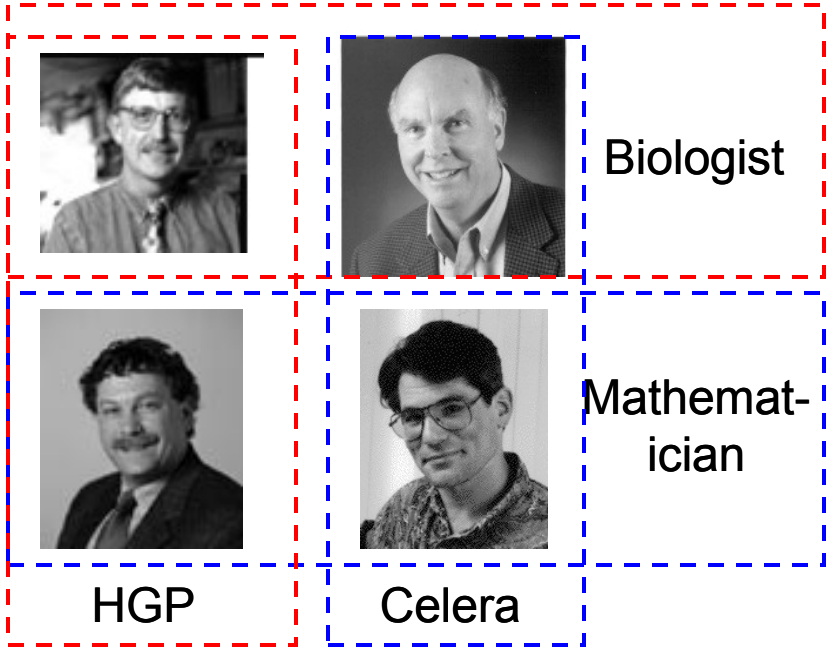
• Given a collection of objects, put objects into groups based on similarity.

•Used for “discovery-based” science, to find unexpected patterns in data.

•Also called “unsupervised learning” or “data mining”

•Inherently an ill-defined problem

•Do we put Collins with Venter because they’re both biologists, or do we put Collins with Lander because they both work for the HGP?



Data Representations for Clustering

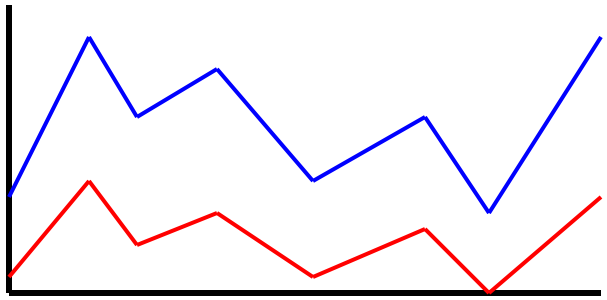
- Input data to algorithm is usually a vector (also called a "tuple" or "record")
- Types of data
  - Numerical
  - Categorical
  - Boolean
- Example: Clinical Sample Data
  - Age (numerical)
  - Weight (numerical)
  - Gender (categorical)
  - Diseased? (boolean)
- Must also include a method for computing similarity of or distance between vectors

Calculating Distance

- Distance is the most natural method for numerical data
- Lower values indicate more similarity
- Distance metrics
  - Euclidean distance
  - Manhattan distance
  - Etc.
- Does not generalize well to non-numerical data
  - What is the distance between “male” and “female”?

Calculating Numerical Similarity

- Traditionally over the range [0.0, 1.0]
  - 0.0 = no similarity, 1.0 = identity
- Converting distance to similarity
  - Distance and similarity are two sides of the same coin
  - To obtain similarity from distance, take the maximum pairwise distance and subtract from 1.0
- Pearson correlation
  - Removes magnitude effects
  - In range [-1.0, 1.0]
    - -1.0 = anti-correlated, 0.0 = no correlation, 1.0 = perfectly correlated
  - In the example below, the red and blue lines have high correlation, even though the distance between the lines is significant



Calculating Boolean Similarity

Boolean Similarity

- Given two boolean vectors X and Y, let A be the number of places where both are 1, etc. as shown below.
- Two standard methods for similarity given at right
- Can be generalized to handle categorical data as well.

		Y[j] 1	Y[j] 0
X[j]	1	A	B
	0	C	D

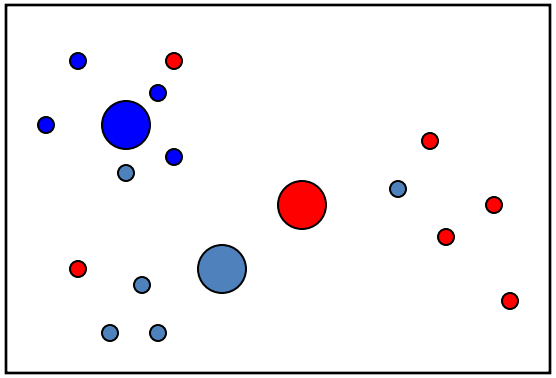
- Correlation = (A + D) / (A+B+C+D)
- Jaccard Coef. = A / (A+B+C+D)
  - Used when absence of a true value does not imply similarity
  - **Example:**
    - Suppose we are doing structural phylogenetics, and X[j] is true if an organism has wings.
    - Two organisms are not more inherently similar if both lack wings.
    - Hence, the Jaccard coefficient is more natural than the standard correlation coefficient in this case.

K-means: The Algorithm

- Given a set of numeric points in *d* dimensional space, and integer *k*
- Algorithm generates *k* (or fewer) clusters as follows:

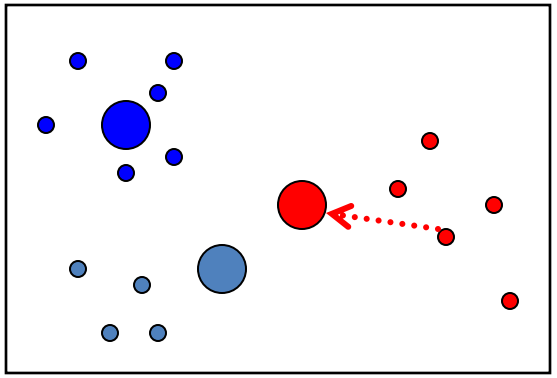
```
Assign all points to a cluster at random
Repeat until stable:
    Compute centroid for each cluster
    Reassign each point to nearest centroid
```

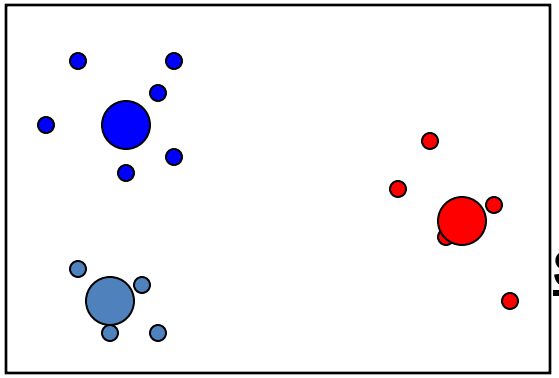
K-means: Example, k = 3



Step 1: Make random assignments and compute centroids (big dots)

Step 2: Assign points to nearest centroids

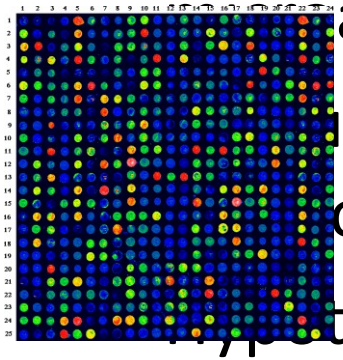




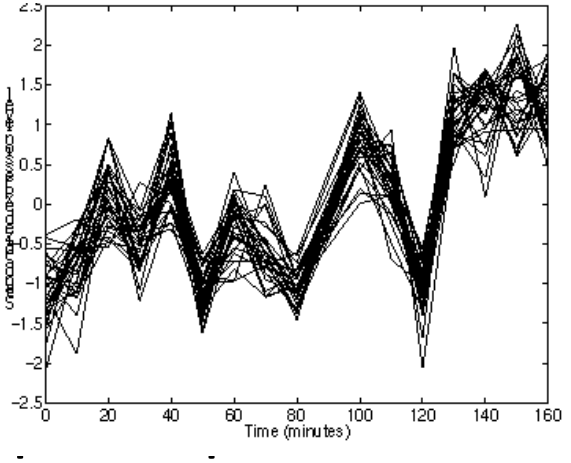
Step 3: Re-compute centroids (in this example, solution is now stable)

K-means: Sample Application

- Gene clustering
  - Given a series of microarray experiments measuring the expression of a set of genes at regular time intervals in a common cell line
  - Normalization allows comparisons across [illegible] arrays.
  - [illegible]ce clusters of genes [illegible]over time
  - [illegible]thesis: genes which [illegible] may be co-regulated and/[illegible] same pathway



Sample Array. Rows are genes and columns are time points.



A cluster of co-regulated genes.

K-means: Weaknesses

- Must choose parameter *k* in advance, or try many values.
- Data must be numerical and must be compared via Euclidean distance (there is a variant called the *k*-medians algorithm to address these concerns)
- The algorithm works best on data which contains spherical clusters; clusters with other geometry may not be found.
- The algorithm is sensitive to *outliers*---points which do not belong in any cluster. These can distort the centroid positions and ruin the clustering.

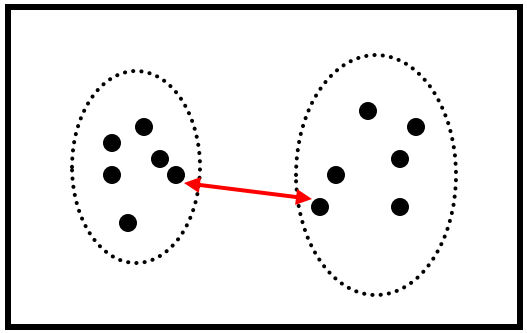
Hierarchical Clustering: The Algorithm

- Hierarchical clustering takes as input a set of points
- It creates a tree in which the points are leaves and the internal nodes reveal the similarity structure of the points.
  - The tree is often called a "dendogram."
- The method is summarized below:

```
Place all points into their own clusters
While there is more than one cluster, do
    Merge the closest pair of clusters
```

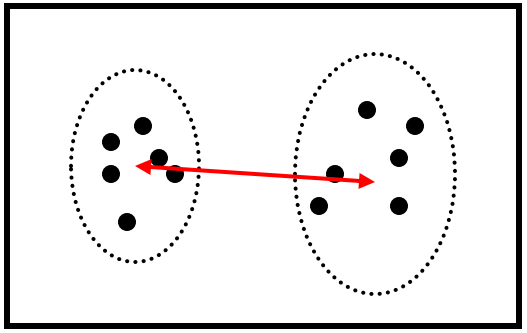
- The behavior of the algorithm depends on how "closest pair of clusters" is defined

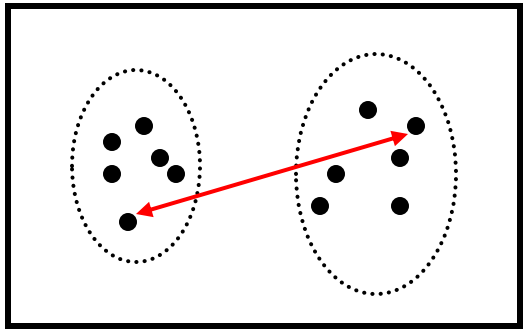
Hierarchical Clustering: Merging Clusters



Single Link: Distance between two clusters is the distance between the closest points. Also called "neighbor joining."

Average Link: Distance between clusters is distance between the cluster centroids.

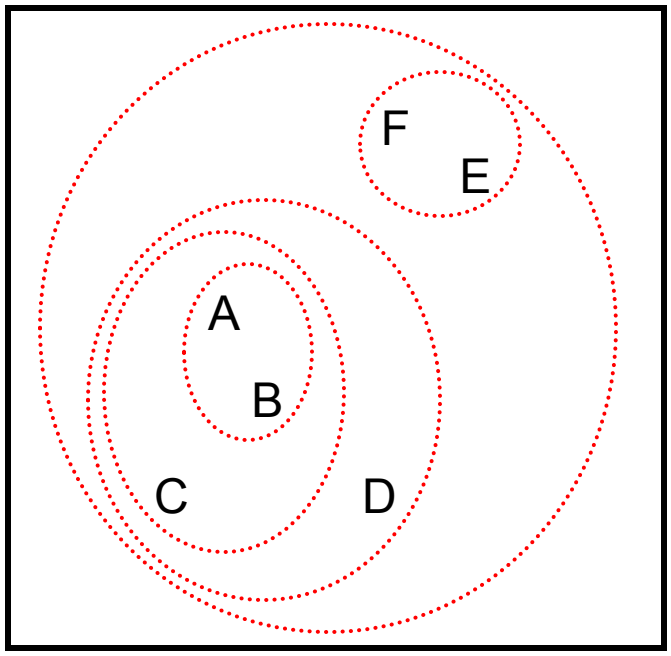


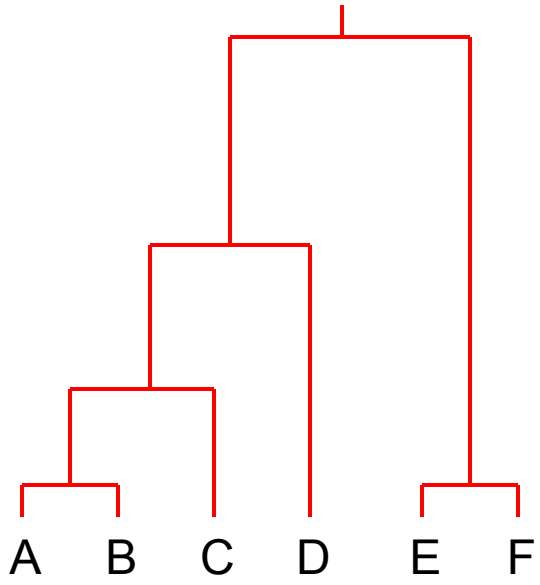


Complete Link: Distance between clusters is distance between farthest pair of points.

Hierarchical Clustering: Example

This example illustrates single-link clustering in Euclidean space on 6 points.





Hierarchical Clustering: Sample Application

- Multiple sequence alignment
  - Given a set of sequences, produce a global alignment of all sequences against the others
  - NP-hard
  - One popular heuristic is to use hierarchical clustering
- The hierarchical clustering approach
  - Each cluster is represented by its consensus sequence
  - When clusters are merged, their consensus sequences are aligned via optimal pairwise alignment
  - The heuristic uses hierarchical clustering to merge the most similar sequences first, the idea being to minimize potential errors in the alignment.
  - A slightly more sophisticated version of this method is implemented by the popular **clustalw** program

Hierarchical Clustering: Weaknesses

- The most commonly used type, single-link clustering, is particularly greedy.
  - If two points from disjoint clusters happen to be near each other, the distinction between the clusters will be lost.
  - On the other hand, average- and complete-link clustering methods are biased towards spherical clusters in the same way as *k*-means
- Does not really produce clusters; the user must decide where to split the tree into groups.
  - Some automated tools exist for this
- As with *k*-means, sensitive to noise and outliers

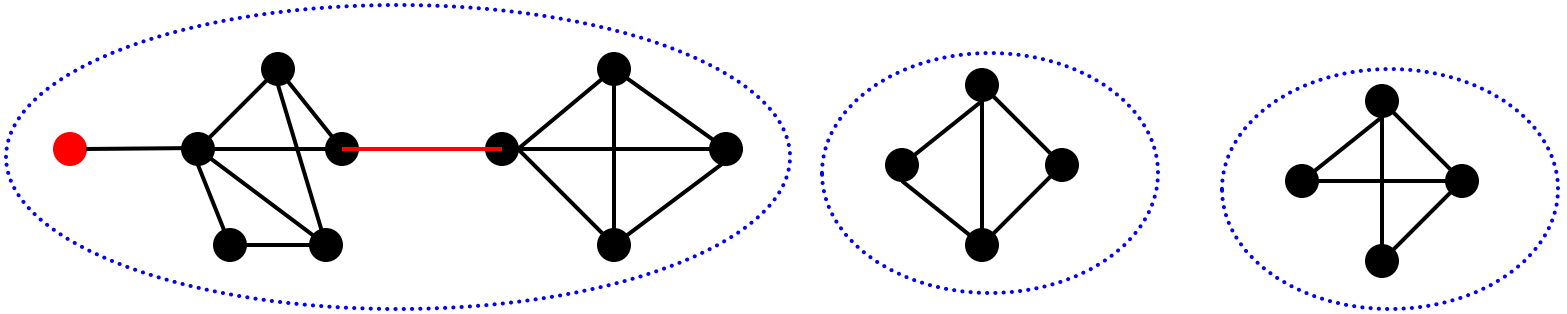
Graph Methods: Components and Cuts

- Define a similarity graph over a set of objects as follows:
  - Vertices are the objects themselves
  - Undirected edges join objects which are deemed “similar”
    - Edges may be weighted by degree of similarity

- A connected component is a maximal set of objects such that each object is path-reachable from the others

- A minimum-weight cut is a set of edges of minimum total weight that creates a new connected component in the graph.

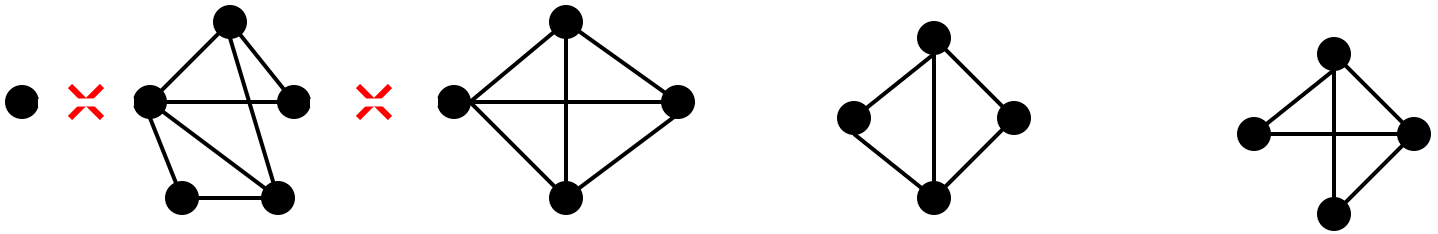
Connected Components for Clustering



- Above graph has three components (or clusters)
- Algorithm to find them is very fast and simple
- This approach has obvious weaknesses; for example,
  - The red node not similar to most objects in its cluster
  - The red edge connects two components that should probably be separate

Minimum Weight Cuts for Clustering

• Run minimum-weight cutset algorithm twice on graph from previous example to produce good clustering (assuming the weight of each edge is 1):



• If objects within a cluster are much more similar than objects between clusters, then method works well.

• Disadvantages

  • Maximum cut algorithm is very slow, and potentially must be run many times

  • Not necessarily clear when to stop running the algorithm

Graph Methods: Application

- EST Clustering
  - Given: a set of short DNA sequences which are derived from expressed genes in the genome
  - Produce: mapping of sequences to their original gene sequence
  - Define two sequences as “similar” if they overlap by a certain amount

- Each gene should have its own connected component in the similarity graph
- Sometimes fragments can be shared between genes, or nearby genes can share an edge.
- A minimum-weight cutset algorithm can be used to resolve these occasional discrepancies.

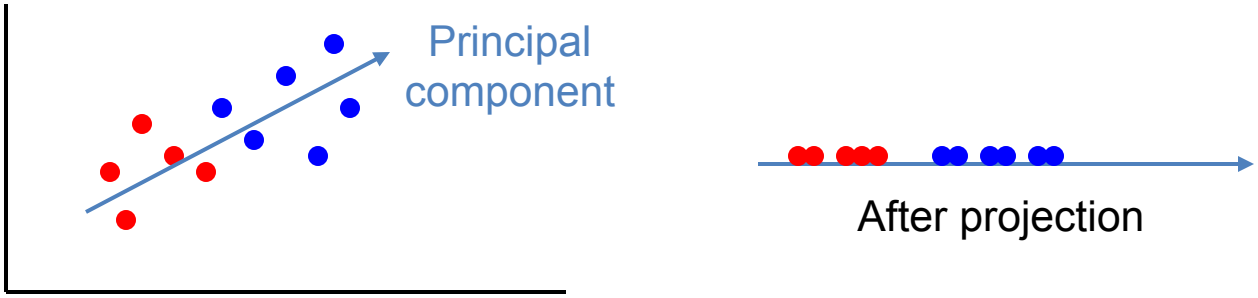
Principal Component Analysis

- Problem: many types of data have too many attributes to be visualized or manipulated conveniently.
  - For example, a single microarray experiment may have 6,000-8,000 genes

- PCA is a method for reducing the number of attributes (dimensions) of numerical data while attempting to preserve the cluster structure.
  - After PCA, we hopefully get the same clusters as we would if we clustered the data before PCA
  - After PCA, plots of the data should still have the clusters falling into obvious groups.
  - By using PCA to reduce the data to 2 or 3 dimensions, off-the-shelf geometry viewers can be used to visualize data

PCA: The Algorithm

- Consider the data as an *m* by *n* matrix in which the columns are the samples and the rows are the attributes.
- The eigenvectors corresponding to the *d* largest eigenvalues of this matrix are the "principal components"
- By projecting the data onto these vectors, one obtains *d*-dimensional points
- Consider the example below, projecting 2D data with 2 clusters (red and blue) into 1 dimension



Challenges in Clustering

- Similarity Calculation
  - Results of algorithms depend entirely on similarity used
  - Clustering systems provide little guidance on how to pick similarity
  - Computing similarity of mixed-type data is hard
  - Similarity is very dependent on data representation. Should one
    - Normalize?
    - Represent one's data numerically, categorically, etc.?
    - Cluster on only a subset of the data?
    - The computer should do more to help the user figure this out!
- Parameter selection
  - Current algorithms require too many arbitrary, user-specified parameters

Conclusion

- Clustering is a useful way of exploring data, but is still very *ad hoc*

- Good results are often dependent on choosing the right data representation and similarity metric
  - Data: categorical, numerical, boolean
  - Similarity: distance, correlation, etc.

- Many different choices of algorithms, each with different strengths and weaknesses
  - *k*-means, hierarchical, graph partitioning, etc.