CPS260/BGT204.1 Algorithms in Computational Biology **October 30, 2003**

Lecture 18: Clustering & classification

Lecturer: Pankaj K. Agarwal *Scribe: Dajun Hou*

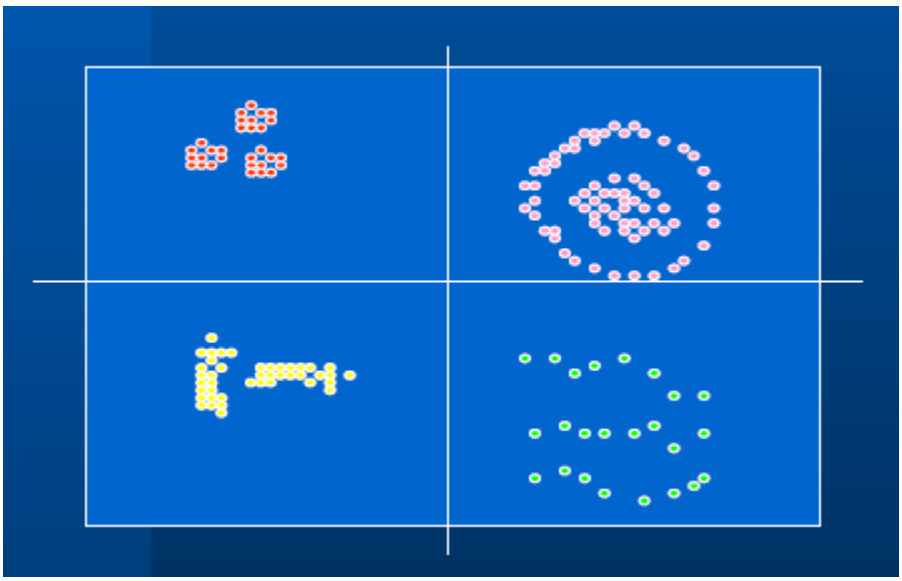
Open Problem

In HomeWork 2, problem 5 has an open problem which may be easy or may be hard. You can publish a paper if you can find the solution. The problem is:

How to improve the space complexity of the algorithm to O(k) or O(k+logn)

1. What is Clustering?

A loose definition of clustering could be "the process of organizing objects into groups whose members are similar in some way". A cluster is therefore a collection of objects which are "similar to each other and are "dissimilar" to the objects belonging to other clusters. Cluster analysis is also used to form descriptive statistics to ascertain whether or not the data consists of a set distinct subgroups, each group representing objects with substantially different properties. The latter goal requires an assessment of the degree of difference between the objects assigned to the respective clusters [5].



Example of Clustering

Central to clustering is to decide what constitutes a good clustering. This can only come from subject matter considerations and there is no absolute "best" criterion which would be independent of the final aim of the clustering. For example, we could be interested in

finding representatives for homogeneous groups (data reduction), in finding "natural clusters" and describe their unknown properties *(*"natural" data types), in finding useful and suitable groupings (*"*useful" data classes) or in finding unusual data objects (outlier detection).

Two important components of cluster analysis are the similarity (distance) measure between two data samples and the clustering algorithm.

2. Distance Measure

Different formula in defining the distance between two data points can lead to different classification results. Domain knowledge must be used to guide the formulation of a suitable distance measure for each particular application. For high dimensional data, a popular measure is the ***Minkowski Metric***:

$$d(x_i, x_j) = \left(\sum_{k=1}^{d} \left| x_{i,k} - x_{j,k} \right|^p \right)^{\frac{1}{p}}$$

where *d* is the dimensionality of the data.

Special Cases:

- *p*=2: *Euclidean* distance
- *p*=1: *Manhattan* distance
- *p*->∞ : *Super* distance

However, there are no general theoretical guidelines for selecting a measure for any given application.

In the case that the components of the data feature vectors are not immediately comparable, such as the days of the week, domain knowledge must be used to formulate an appropriate measure.

3. Clustering algorithms

Clustering algorithms may be classified as listed below:

- Exclusive Clustering

  In exclusive clustering data are grouped in an exclusive way, so that a certain datum belongs to only one definite cluster. K-means clustering is one example of the exclusive clustering algorithms.

- Overlapping Clustering

  The overlapping clustering uses fuzzy sets to cluster data, so that each point may belong to two or more clusters with different degrees of membership.

- Hierarchical Clustering

  Hierarchical clustering algorithm has two versions: agglomerative clustering and divisive clustering

  Agglomerative clustering is based on the union between the two nearest clusters. The beginning condition is realized by setting every datum as a cluster. After a few iterations it reaches the final clusters wanted. Basically, this is a bottom-up version

  Divisive clustering starts from one cluster containing all data items. At each step, clusters are successively split into smaller clusters according to some dissimilarity. Basically this is a top-down version.

- Probabilistic Clustering

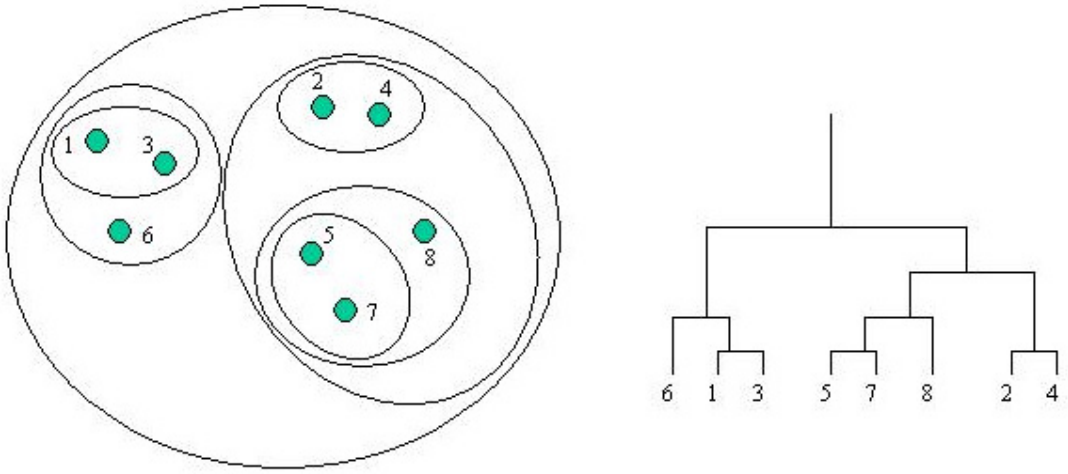
  Probabilistic clustering, e.g. Mixture of Gaussian, uses a completely probabilistic approach.

4. Hierarchical Algorithm

Given a set of N objects $S=\{s_1,s_2,...s_N\}$ to be clustered and a function of distance $D(c_i,c_j)$ between two clusters c_i and c_j, build a hierarchy tree on S : $c_i,c_j \subset S$, $c_i \cap c_j = \varnothing$. The basic process of hierarchical clustering (S.C. Johnson in 1967) is as follows:

1. Start by assigning each object to a cluster $c_i = s_i$ $(i=1,...N)$, so that if you have N objects, you have N clusters $\ell = \{c_1,c_2...c_N\}$, each containing just one item.
2. Find the pair of clusters (c_i,c_j) such that $D(c_i,c_j) \le D(c_{i'},c_{j'}) \forall c_{i'} \ne c_{j'} \in \ell$ and merge them into a single cluster $c_k = c_i \cup c_j$. Delete c_i,c_j from ℓ and insert c_k into ℓ so that now you have one cluster less.
3. Compute distances (similarities) between the new cluster and each of the old clusters.
4. Repeat steps 2 and 3 until all items are clustered into a single cluster of size N.

An example of how the hierarchical algorithm leads to long clusters is shown in the following figure:



Example of Hierarchical Clustering

Step 3 in the hierarchical algorithm can be done in different ways, which is what distinguishes *single-linkage* from *complete-linkage* and *average-linkage* clustering.

- In *single-linkage* clustering, the distance between one cluster and another cluster is equal to the shortest distance from any member of one cluster to any member of the other cluster: $D(c_i, c_j) = \min d(a,b)\ a \in c_i, b \in c_j$. It is obvious that:

$$D(c_k, c_l) = \min\left\{D(c_i, c_l), D(c_j, c_l)\right\} \quad \text{for} \quad c_k = c_i \cup c_j$$

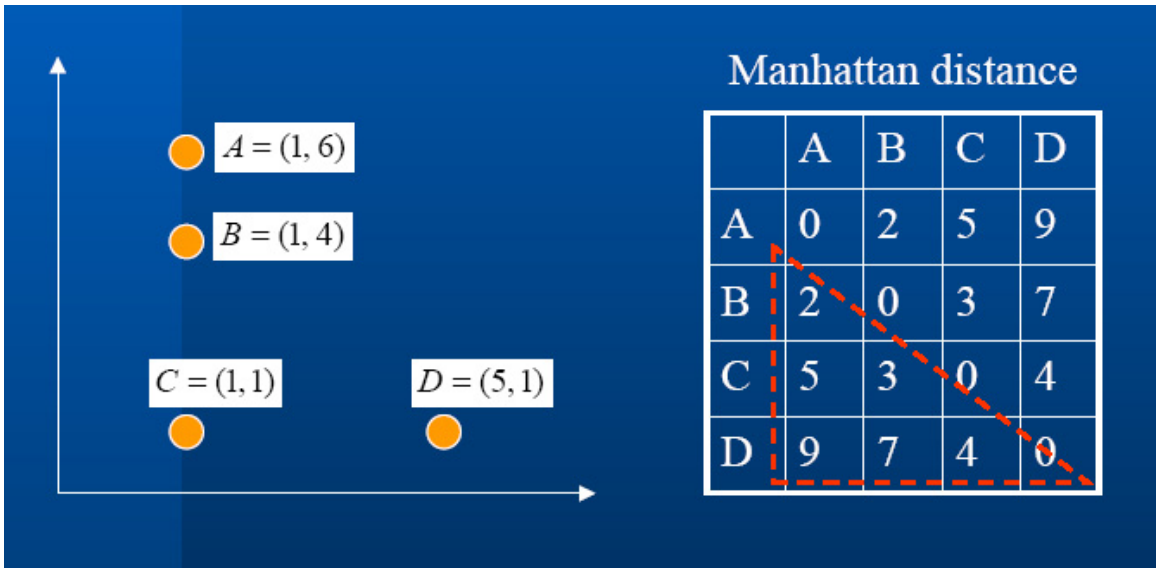
- In *complete-linkage* clustering, the distance between one cluster and another cluster is equal to the greatest distance from any member of one cluster to any member of the other cluster: $D(c_i, c_j) = \max d(a,b)\ a \in c_i, b \in c_j$.

- In *average-linkage* clustering, the distance between one cluster and another cluster is equal to the average distance from any member of one cluster to any member of the other cluster: $D(c_i, c_j) = \frac{1}{|c_i||c_j|} \sum_{a \in c_i, b \in c_j} d(a,b)$. It is obvious that

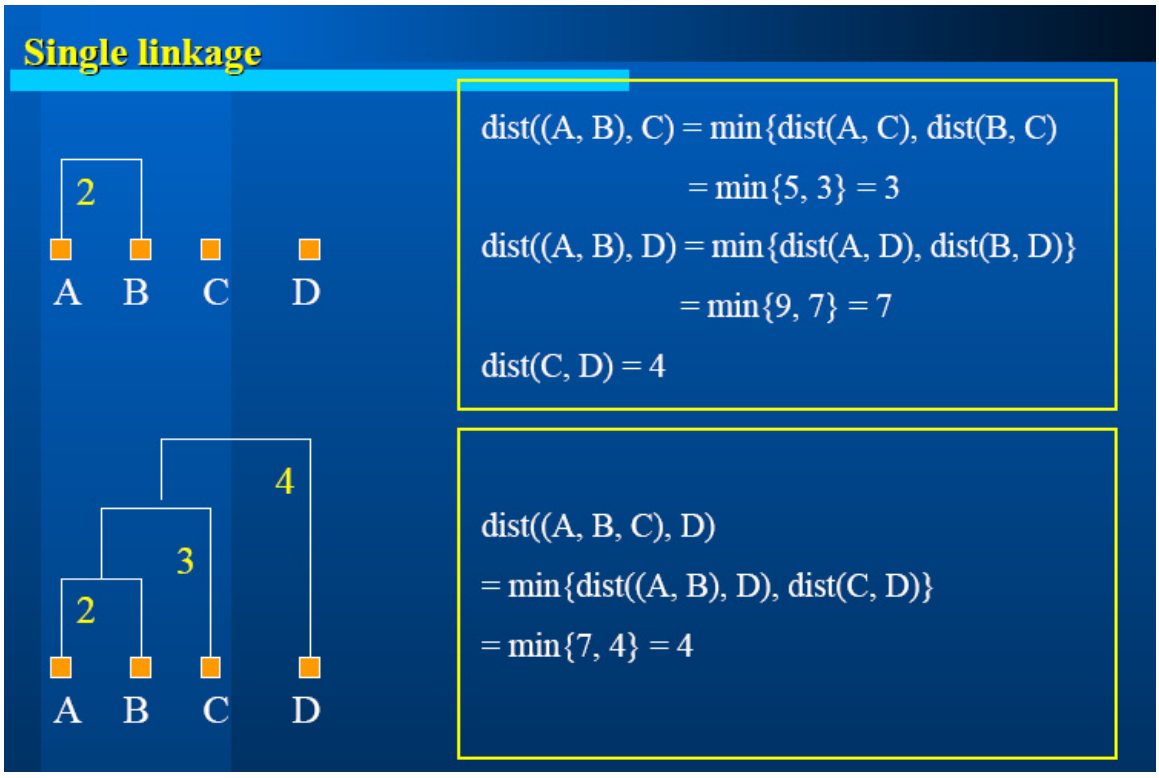
$$D(c_k, c_l) = \frac{|c_i|}{|c_k|} D(c_i, c_l) + \frac{|c_j|}{|c_k|} D(c_j, c_l) \quad \text{for } c_k = c_i \cup c_j$$

The main weaknesses of hierarchical clustering methods include that they do not scale well: the time complexity is at least $O(N^2)$, where N is the number of total objects, and that they can never undo what was done previously. It is basically a greedy approach.

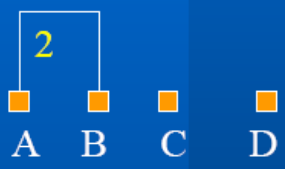
An example [4]: clustering 4 data items in 2-dimensional space



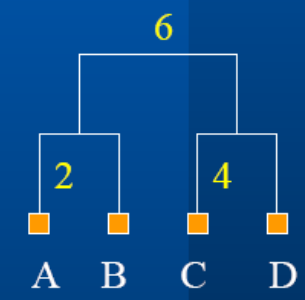
	A	B	C	D
A	0	2	5	9
B	2	0	3	7
C	5	3	0	4
D	9	7	4	0



Average linkage

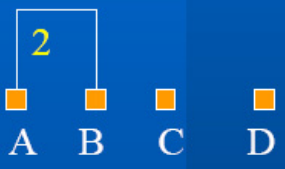


dist((A, B), C) = avg{dist(A, C), dist(B, C)
= (5+3)/2 = 4
dist((A, B), D) = avg{dist(A, D), dist(B, D)}
= (9+7)/2 = 8
dist(C, D) = 4

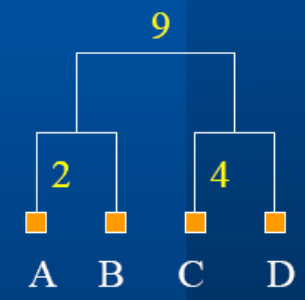


dist((C, D), (A, B))
= avg{dist(C, (A, B)), dist(D, (A, B))}
= (4+8)/2 = 6

Complete linkage



dist((A, B), C) = max{dist(A, C), dist(B, C)
= max{5, 3} = 5
dist((A, B), D) = max{dist(A, D), dist(B, D)}
= max{9, 7} = 9
dist(C, D) = 4



dist((C, D), (A, B))
= max{dist(C, (A, B)), dist(D, (A, B))}
= 9

5. K-Means Clustering

K-means (MacQueen, 1967) is one of the simplest unsupervised learning algorithms that solves the clustering problem. The objective is to classify a given data set $S = \{s_1, s_2, ...s_N\}$ into a certain number of clusters (assume initial clusters) fixed a priori. The idea is to define initial centroids, one for each cluster $c_i (i = 1, ...k)$. The procedure is:

1. Initial clusters: $\ell^o = \{c_1^{\,0}, c_2^{\,0} ... c_k^{\,0}\}$; the initial centroids should be placed as far as possible from each other.

2: Calculate the centroids of the clusters: $u_j^i = \frac{1}{|c_j^{\,i}|} \sum_{x \in c_j^{\,i}} x$ where $j = 1, ...k$ and i denotes the i^{th} iteration.

3. Take each point belonging to a given data set and associate it to the nearest centroid:

$$c_j^{\,i+1} = \{x \mid d(x, u_j^i) \le d(x, u_{j'}^i)\ 1 \le j' \le k\}$$

$$\ell^{i+1} = \{c_j^{\,i+1} \mid 1 \le j \le k\}$$

4. Repeat steps 2 and 3 until no more changes can be made to the clusters, that is, $\ell^{i+1} = \ell^i$. In other words centroids do not move any more.

Each iteration takes *O(Nk)* time, but we don't know in how many iterations it will take to converge. Finally, this algorithm aims at minimizing an objective function, in this case a squared error function:

$$J = \sum_{j=1}^{k} \sum_{x \in c_j} \|x - u_j\|^2$$

Although it can be proved that the k-means algorithm will always terminate, the algorithm does not necessarily find the most optimal configuration, corresponding to the global objective function minimum. It might get stuck in a local minimum. The algorithm is also significantly sensitive to the initial randomly selected cluster centers. To get out of local minimum, the k-means algorithm can be run multiple times from different initial clustering or simulated annealing technique could be used.

6. Principal Component Analysis

The principal components $v_i\{i=1,2...d\}$ of a data set $S \subseteq \Re^d$ consisting of N d-dimensional random vectors $S=\{s_1,s_2,...s_N\}$ (d>>log$N$) provides a sequence of best linear approximation to that data in terms of minimum mean-square error. The original vector $s_j\{j=1,2...N\}$ hence can be represented as a linear combination of the principal components:

$$s_j = \sum_{i=1}^{d} a_i{}^j v_i$$

This representation has several attractive properties:

1. The effectiveness of each principal components, $v_i\{i=1,2...d\}$, in terms of representing S, is determined by its corresponding eigenvalue. Therefore, the principal component with the largest eigenvalue has the greatest importance in effectively approximating the original data set S, and vice versa.

2. The principal components, $v_i\{i=1,2...d\}$ are mutually uncorrelated, that is, $v_i \cdot v_j = 0$ $(1 \le i < j \le d)$. Furthermore, in the special case where S is normally distributed, the principal components are mutually independent.

3. The set of k principal components $v_i\{i=1,2...k\}$, which corresponds to the k largest eigenvalues, minimizes the mean-square error over all choices of k orthogonal vectors.

The main application of Principal Component Analysis is for feature space reduction: the d-dimensional data can be projected onto a k-dimensional subspace using the first k principal components. In classification, the data then might be clustered around the k-dimensional hyperplane after the transformation projection. To calculate the principal components $v_i\{i=1,2...d\}$, the covariance matrix X of the data set S is first calculated:

$$X = \frac{1}{N}\sum_{j=1}^{N}(s_j-u)(s_j-u)^T \text{ where } u = \frac{1}{N}\sum_{j=1}^{N} s_j$$

The average u of all vectors s_j in the data set is subtracted so that the eigenvector with the largest eigenvalue represents the subspace dimension in which the variance of the data set is maximized in the correlation sense. The principal components $v_i\{i=1,2...d\}$ are the eigenvectors of the covariance matrix X and can be determined by solving the well-known eigenvalue decomposition problem:

$$\lambda_i v_i = X v_i$$

Here $\lambda_1 \ge \lambda_2 \ge ... \ge \lambda_k \ge ... \ge \lambda_d$. One approach in selecting k such that the first k eigenvectors of X capture important variations in the data set S is:

$$\frac{\sum_{i=1}^{k}\lambda_i}{\sum_{i=1}^{d}\lambda_i} \geq T$$

where the threshold T is close to but less than unity.

7. Supervised Learning

All the clustering analysis methods introduced above are examples of unsupervised learning algorithms. A learning method is considered unsupervised if it learns in the absence of a teacher signal that provides prior knowledge of the correct answer. Supervised learning has a substantial advantage over unsupervised learning. In particular, supervised learning allows us to take advantage of our own knowledge about the classification problem we are trying to solve. Instead of just letting the algorithm work out for itself what the classes should be, we can tell it what we know about the classes: how many there are and what examples of each one look like. The supervised learning algorithm's job is then to find the features in the examples that are most useful in predicting the classes. Neural networks and support vector machines are widely used in supervised learning. The following is an example of supervised learning:

For feature vectors $S = \{s_1, s_2, ...s_N\}$ and classifier $\chi : s \to \{-1, +1\}$, the objective is to find a hyperplane H passing through the origin so that the objective function is maximized: $\sum_{s_i} sign(H(s_i))\chi(s_i)$. Basically you want to maximize the number of times $sign(H(s_i)) = \chi(s_i)$. Sometimes a hyperplane dose not work because the points are not linearly separable. In that case you could look at different hyper-surfaces to separate the data points.

References:

[1] S. C. Johnson (1967): "Hierarchical Clustering Schemes" *Psychometrika*, 2:241-254.
[2] J.B.MacQueen (1967): "Some Methods for classification and Analysis of Multivariate Observations, *Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability"*, Berkeley, University of California Press, 1:281-297.
[3] A Tutorial on Clustering Algorithms
http://www.elet.polimi.it/upload/matteucc/Clustering/tutorial_html/

[4] Jeong-Ho Chang http://cbit.snu.ac.kr/tutorial-2002/ppt/ClusterAnalysis.pdf

[5] T. Hastie, R. Tibshirani, and J. Friedman, The Elements of Statistical Learning