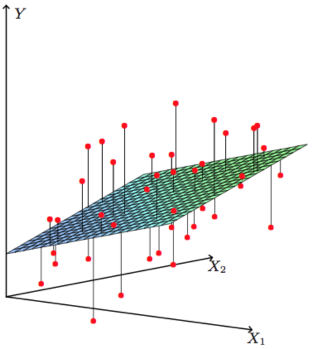
Statistical Methods for Data Mining

Kuangnan Fang

Xiamen University *Email: xmufkn@xmu.edu.cn*



Unsupervised Learning

Unsupervised vs Supervised Learning:

- Most of this course focuses on *supervised learning* methods such as regression and classification.
- In that setting we observe both a set of features $X_1, X_2, \ldots, X_p$ for each object, as well as a response or outcome variable Y. The goal is then to predict Y using $X_1, X_2, \ldots, X_p$.
- Here we instead focus on *unsupervised learning*, we where observe only the features $X_1, X_2, \ldots, X_p$. We are not interested in prediction, because we do not have an associated response variable Y.

The Goals of Unsupervised Learning

- The goal is to discover interesting things about the measurements: is there an informative way to visualize the data? Can we discover subgroups among the variables or among the observations?
- We discuss two methods:
  - *principal components analysis*, a tool used for data visualization or data pre-processing before supervised techniques are applied, and
  - *clustering*, a broad class of methods for discovering unknown subgroups in data.

The Challenge of Unsupervised Learning

- Unsupervised learning is more subjective than supervised learning, as there is no simple goal for the analysis, such as prediction of a response.
- But techniques for unsupervised learning are of growing importance in a number of fields:
  - subgroups of breast cancer patients grouped by their gene expression measurements,
  - groups of shoppers characterized by their browsing and purchase histories,
  - movies grouped by the ratings assigned by movie viewers.

Another advantage

- It is often easier to obtain *unlabeled data* — from a lab instrument or a computer — than *labeled data*, which can require human intervention.
- For example it is difficult to automatically assess the overall sentiment of a movie review: is it favorable or not?

Principal Components Analysis

- PCA produces a low-dimensional representation of a dataset. It finds a sequence of linear combinations of the variables that have maximal variance, and are mutually uncorrelated.
- Apart from producing derived variables for use in supervised learning problems, PCA also serves as a tool for data visualization.

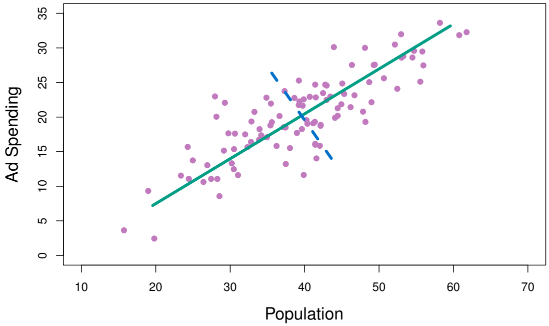
Principal Components Analysis: details

- The *first principal component* of a set of features $X_1, X_2, \ldots, X_p$ is the normalized linear combination of the features

$$Z_1 = \phi_{11}X_1 + \phi_{21}X_2 + \ldots + \phi_{p1}X_p$$

  that has the largest variance. By *normalized*, we mean that $\sum_{j=1}^{p} \phi_{j1}^2 = 1$.
- We refer to the elements $\phi_{11}, \ldots, \phi_{p1}$ as the loadings of the first principal component; together, the loadings make up the principal component loading vector, $\phi_1 = (\phi_{11}\ \phi_{21}\ \ldots\ \phi_{p1})^T$.
- We constrain the loadings so that their sum of squares is equal to one, since otherwise setting these elements to be arbitrarily large in absolute value could result in an arbitrarily large variance.

PCA: example



The population size (`pop`) and ad spending (`ad`) for 100 different cities are shown as purple circles. The green solid line indicates the first principal component direction, and the blue dashed line indicates the second principal component direction.

Computation of Principal Components

- Suppose we have a $n \times p$ data set $\mathbf{X}$. Since we are only interested in variance, we assume that each of the variables in $\mathbf{X}$ has been centered to have mean zero (that is, the column means of $\mathbf{X}$ are zero).
- We then look for the linear combination of the sample feature values of the form

$$z_{i1} = \phi_{11}x_{i1} + \phi_{21}x_{i2} + \ldots + \phi_{p1}x_{ip} \tag{1}$$

  for $i = 1, \ldots, n$ that has largest sample variance, subject to the constraint that $\sum_{j=1}^{p} \phi_{j1}^2 = 1$.
- Since each of the x_{ij} has mean zero, then so does z_{i1} (for any values of ϕ_{j1}). Hence the sample variance of the z_{i1} can be written as $\frac{1}{n}\sum_{i=1}^{n} z_{i1}^2$.

Computation: continued

- Plugging in (1) the first principal component loading vector solves the optimization problem

$$\underset{\phi_{11},\ldots,\phi_{p1}}{\text{maximize}} \frac{1}{n}\sum_{i=1}^{n}\left(\sum_{j=1}^{p}\phi_{j1}x_{ij}\right)^2 \text{ subject to } \sum_{j=1}^{p}\phi_{j1}^2 = 1.$$

- This problem can be solved via a singular-value decomposition of the matrix $\mathbf{X}$, a standard technique in linear algebra.
- We refer to Z_1 as the first principal component, with realized values $z_{11}, \ldots, z_{n1}$

Geometry of PCA

- The loading vector ϕ_1 with elements $\phi_{11}, \phi_{21}, \ldots, \phi_{p1}$ defines a direction in feature space along which the data vary the most.
- If we project the n data points $x_1, \ldots, x_n$ onto this direction, the projected values are the principal component scores $z_{11}, \ldots, z_{n1}$ themselves.

Further principal components

- The second principal component is the linear combination of $X_1, \dots, X_p$ that has maximal variance among all linear combinations that are *uncorrelated* with Z_1.
- The second principal component scores $z_{12}, z_{22}, \dots, z_{n2}$ take the form

  $$z_{i2} = \phi_{12}x_{i1} + \phi_{22}x_{i2} + \ldots + \phi_{p2}x_{ip},$$

  where ϕ_2 is the second principal component loading vector, with elements $\phi_{12}, \phi_{22}, \dots, \phi_{p2}$.

Further principal components: continued

- It turns out that constraining Z_2 to be uncorrelated with Z_1 is equivalent to constraining the direction ϕ_2 to be orthogonal (perpendicular) to the direction ϕ_1. And so on.
- The principal component directions ϕ_1, ϕ_2, $\phi_3, \ldots$ are the ordered sequence of right singular vectors of the matrix $\mathbf{X}$, and the variances of the components are $\frac{1}{n}$ times the squares of the singular values. There are at most $\min(n-1, p)$ principal components.

Illustration

- `USAarrests` data: For each of the fifty states in the United States, the data set contains the number of arrests per $100,000$ residents for each of three crimes: `Assault`, `Murder`, and `Rape`. We also record `UrbanPop` (the percent of the population in each state living in urban areas).
- The principal component score vectors have length $n = 50$, and the principal component loading vectors have length $p = 4$.
- PCA was performed after standardizing each variable to have mean zero and standard deviation one.

USAarrests data: PCA plot

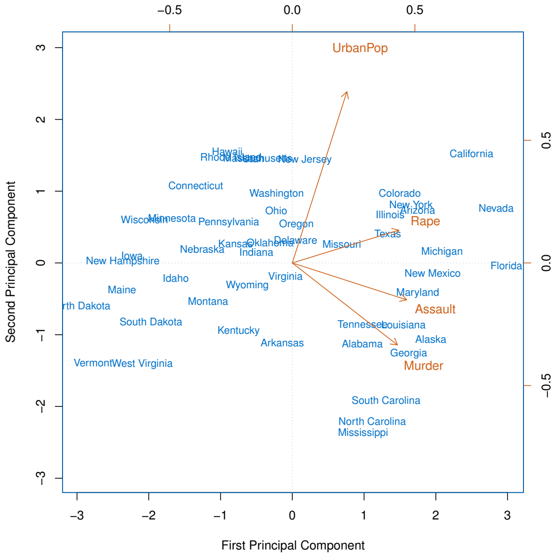


Figure details

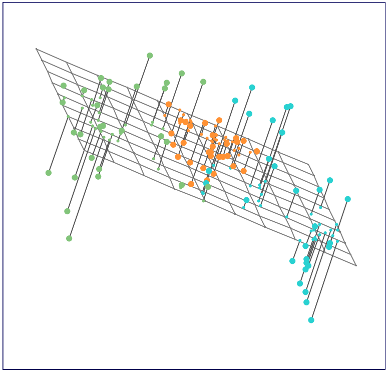
The first two principal components for the USArrests data.

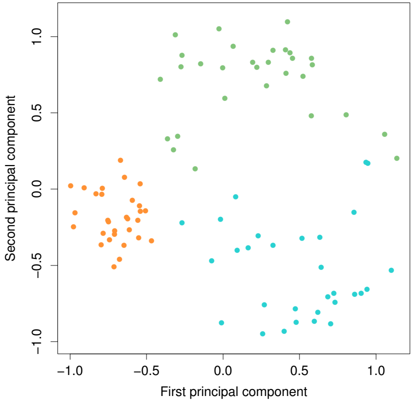
- The blue state names represent the scores for the first two principal components.
- The orange arrows indicate the first two principal component loading vectors (with axes on the top and right). For example, the loading for `Rape` on the first component is 0.54, and its loading on the second principal component 0.17 [the word `Rape` is centered at the point $(0.54, 0.17)$].
- This figure is known as a *biplot*, because it displays both the principal component scores and the principal component loadings.

PCA loadings

	PC1	PC2
Murder	0.5358995	-0.4181809
Assault	0.5831836	-0.1879856
UrbanPop	0.2781909	0.8728062
Rape	0.5434321	0.1673186

Another Interpretation of Principal Components



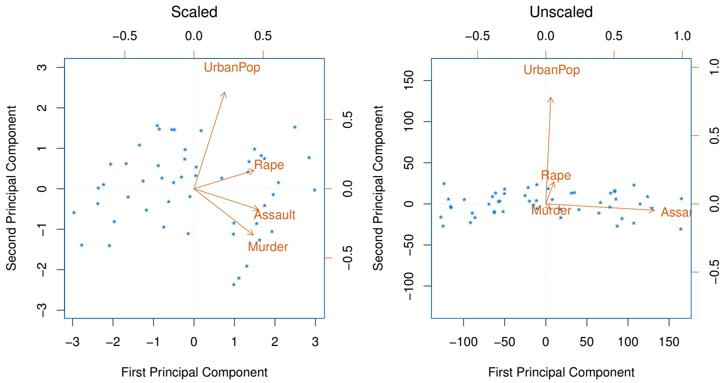


PCA find the hyperplane closest to the observations

- The first principal component loading vector has a very special property: it defines the line in p-dimensional space that is *closest* to the n observations (using average squared Euclidean distance as a measure of closeness)
- The notion of principal components as the dimensions that are closest to the n observations extends beyond just the first principal component.
- For instance, the first two principal components of a data set span the plane that is closest to the n observations, in terms of average squared Euclidean distance.

Scaling of the variables matters

- If the variables are in different units, scaling each to have standard deviation equal to one is recommended.
- If they are in the same units, you might or might not scale the variables.



Proportion Variance Explained

- To understand the strength of each component, we are interested in knowing the proportion of variance explained (PVE) by each one.
- The *total variance* present in a data set (assuming that the variables have been centered to have mean zero) is defined as

$$\sum_{j=1}^{p} \mathrm{Var}(X_j) = \sum_{j=1}^{p} \frac{1}{n} \sum_{i=1}^{n} x_{ij}^2,$$

  and the variance explained by the mth principal component is

$$\mathrm{Var}(Z_m) = \frac{1}{n} \sum_{i=1}^{n} z_{im}^2.$$

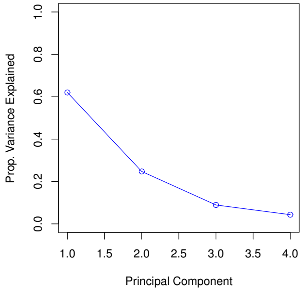
- It can be shown that $\sum_{j=1}^{p} \mathrm{Var}(X_j) = \sum_{m=1}^{M} \mathrm{Var}(Z_m)$, with $M = \min(n-1, p)$.

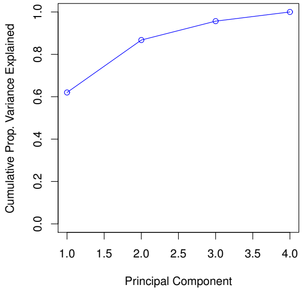
Proportion Variance Explained: continued

- Therefore, the PVE of the mth principal component is given by the positive quantity between 0 and 1

$$\frac{\sum_{i=1}^{n} z_{im}^2}{\sum_{j=1}^{p} \sum_{i=1}^{n} x_{ij}^2}.$$

- The PVEs sum to one. We sometimes display the cumulative PVEs.





How many principal components should we use?

If we use principal components as a summary of our data, how many components are sufficient?

- No simple answer to this question, as cross-validation is not available for this purpose.
    - *Why not?*

How many principal components should we use?

If we use principal components as a summary of our data, how many components are sufficient?

- No simple answer to this question, as cross-validation is not available for this purpose.
    - *Why not?*
    - When could we use cross-validation to select the number of components?

How many principal components should we use?

If we use principal components as a summary of our data, how many components are sufficient?

- No simple answer to this question, as cross-validation is not available for this purpose.
    - *Why not?*
    - When could we use cross-validation to select the number of components?
- the "scree plot" on the previous slide can be used as a guide: we look for an "elbow".

Clustering

- *Clustering* refers to a very broad set of techniques for finding *subgroups*, or *clusters*, in a data set.
- We seek a partition of the data into distinct groups so that the observations within each group are quite similar to each other,
- It make this concrete, we must define what it means for two or more observations to be *similar* or *different*.
- Indeed, this is often a domain-specific consideration that must be made based on knowledge of the data being studied.

PCA vs Clustering

- PCA looks for a low-dimensional representation of the observations that explains a good fraction of the variance.
- Clustering looks for homogeneous subgroups among the observations.

Clustering for Market Segmentation

- Suppose we have access to a large number of measurements (e.g. median household income, occupation, distance from nearest urban area, and so forth) for a large number of people.
- Our goal is to perform *market segmentation* by identifying subgroups of people who might be more receptive to a particular form of advertising, or more likely to purchase a particular product.
- The task of performing market segmentation amounts to clustering the people in the data set.

Two clustering methods

- In *K-means clustering*, we seek to partition the observations into a pre-specified number of clusters.
- In *hierarchical clustering*, we do not know in advance how many clusters we want; in fact, we end up with a tree-like visual representation of the observations, called a *dendrogram*, that allows us to view at once the clusterings obtained for each possible number of clusters, from 1 to n.

Dissimilarities Based on Attributes

Most often we have measurements x_{ij} for $i = 1, 2, ..., N$, on variables $j = 1, 2, ..., p$(also called attributes). Since most of the popular clustering algorithms take a dissimilarity matrix as their input, we must first construct pairwise dissimilarities between the observations. In the most common case, we define a dissimilarity $d_j(x_{ij}, x_{ij})$ between values of the jth attribute, and then define

$$D(x_i, x_{i'}) = \sum_{j=1}^{p} d_j(x_{ij}, x_{i'j})$$

as the dissimilarity between objects i and i. By far the most common choice is squared distance

$$d_j(x_{ij}, xi'j) = (x_{ij} - x_{i'j})^2$$

However, other choices are possible, and can lead to potentially different results. For nonquantitative attributes, squared distance may not be appropriate. In addition, it is sometimes desirable to weight attributes differently.

Dissimilarities Based on Attributes

Quantitative variables. Measurements of this type of variable or attribute are represented by continuous real-valued numbers. It is natural to define the "error" between them as a monotone-increasing function of their absolute difference

$$d(x_i, x_{i'}) = l(|x_i - x_{i'}|).$$

Besides squared-error loss $(x_i - x_{i'})^2$, a common choice is the identity (absolute error). The former places more emphasis on larger differences than smaller ones. Alternatively, clustering can be based on the correlation

$$\rho(x_i, x_{i'}) = \frac{\sum_j (x_{ij} - \bar{x}_i)(x_{i'j} - \bar{x}_{i'})}{\sqrt{\sum_j (x_{ij} - \bar{x}_i)^2 \sum_j (x_{i'j} - \bar{x}_{i'})^2}}, \qquad (14.22)$$

with $\bar{x}_i = \sum_j x_{ij}/p$. Note that this is averaged over *variables*, not observations. If the *observations* are first standardized, then $\sum_j (x_{ij} - x_{i'j})^2 \propto 2(1 - \rho(x_i, x_{i'}))$. Hence clustering based on correlation (similarity) is equivalent to that based on squared distance (dissimilarity).

Dissimilarities Based on Attributes

Ordinal variables. The values of this type of variable are often represented as contiguous integers, and the realizable values are considered to be an ordered set. Examples are academic grades (A, B, C, D, F), degree of preference (can't stand, dislike, OK, like, terrific). Rank data are a special kind of ordinal data. Error measures for ordinal variables are generally defined by replacing their M original values with

$$\frac{i - 1/2}{M}, \; i = 1, \ldots, M \tag{14.23}$$

in the prescribed order of their original values. They are then treated as quantitative variables on this scale.

Object Dissimilarity

Next we define a procedure for combining the p-individual attribute dissimilarities $d_j(x_{ij}, x_{i'j})$, $j = 1, 2, \ldots, p$ into a single overall measure of dissimilarity $D(x_i, x_{i'})$ between two objects or observations $(x_i, x_{i'})$ possessing the respective attribute values. This is nearly always done by means of a weighted average (convex combination)

$$D(x_i, x_{i'}) = \sum_{j=1}^{p} w_j \cdot d_j(x_{ij}, x_{i'j}); \quad \sum_{j=1}^{p} w_j = 1. \tag{14.24}$$

Here w_j is a weight assigned to the jth attribute regulating the relative influence of that variable in determining the overall dissimilarity between objects. This choice should be based on subject matter considerations.

Object Dissimilarity

It is important to realize that setting the weight w_j to the same value for each variable (say, $w_j = 1 \; \forall \; j$) does *not* necessarily give all attributes equal influence. The influence of the jth attribute X_j on object dissimilarity $D(x_i, x_{i'})$ (14.24) depends upon its relative contribution to the average object dissimilarity measure over all pairs of observations in the data set

$$\bar{D} = \frac{1}{N^2} \sum_{i=1}^{N} \sum_{i'=1}^{N} D(x_i, x_{i'}) = \sum_{j=1}^{p} w_j \cdot \bar{d}_j,$$

with

$$\bar{d}_j = \frac{1}{N^2} \sum_{i=1}^{N} \sum_{i'=1}^{N} d_j(x_{ij}, x_{i'j}) \tag{14.25}$$

being the average dissimilarity on the jth attribute. Thus, the relative influence of the jth variable is $w_j \cdot \bar{d}_j$, and setting $w_j \sim 1/\bar{d}_j$ would give all attributes equal influence in characterizing overall dissimilarity between objects. For example, with p quantitative variables and squared-error distance used for each coordinate, then (14.24) becomes the (weighted) squared Euclidean distance

Object Dissimilarity

$$D_I(x_i, x_{i'}) = \sum_{j=1}^{p} w_j \cdot (x_{ij} - x_{i'j})^2 \tag{14.26}$$

between pairs of points in an $\mathbb{R}^p$, with the quantitative variables as axes. In this case (14.25) becomes

$$\bar{d}_j = \frac{1}{N^2} \sum_{i=1}^{N} \sum_{i'=1}^{N} (x_{ij} - x_{i'j})^2 = 2 \cdot \text{var}_j, \tag{14.27}$$

where var_j is the sample estimate of $\text{Var}(X_j)$. Thus, the relative importance of each such variable is proportional to its variance over the data

Object Dissimilarity

set. In general, setting $w_j = 1/d_j$ for all attributes, irrespective of type, will cause each one of them to equally influence the overall dissimilarity between pairs of objects $(x_i, x_{i'})$. Although this may seem reasonable, and is often recommended, it can be highly counterproductive. If the goal is to segment the data into groups of similar objects, all attributes may not contribute equally to the (problem-dependent) notion of dissimilarity between objects. Some attribute value differences may reflect greater actual object dissimilarity in the context of the problem domain.

If the goal is to discover natural groupings in the data, some attributes may exhibit more of a grouping tendency than others. Variables that are more relevant in separating the groups should be assigned a higher influence in defining object dissimilarity. Giving all attributes equal influence in this case will tend to obscure the groups to the point where a clustering algorithm cannot uncover them. Figure 14.5 shows an example.

Object Dissimilarity

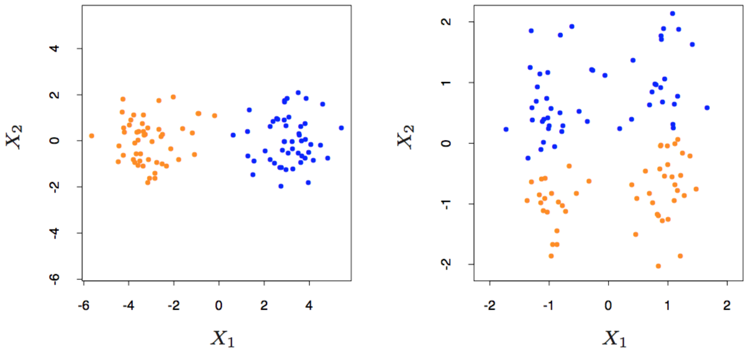
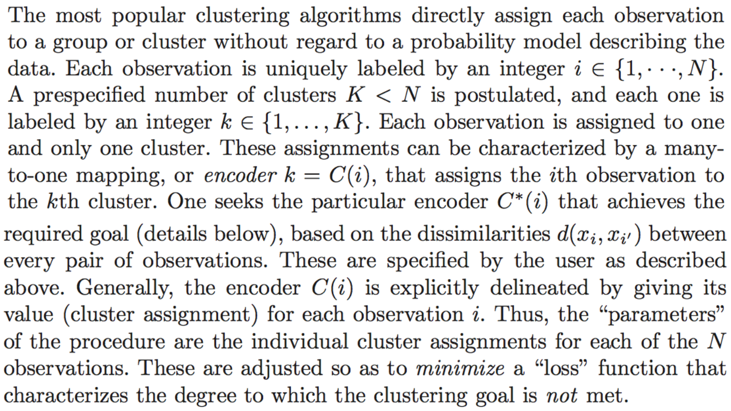


FIGURE 14.5. *Simulated data: on the left, K-means clustering (with $K=2$) has been applied to the raw data. The two colors indicate the cluster memberships. On the right, the features were first standardized before clustering. This is equivalent to using feature weights $1/[2 \cdot \text{var}(X_j)]$. The standardization has obscured the two well-separated groups. Note that each plot uses the same units in the horizontal and vertical axes.*

Combinatorial Algorithms



The most popular clustering algorithms directly assign each observation to a group or cluster without regard to a probability model describing the data. Each observation is uniquely labeled by an integer $i \in \{1, \cdots, N\}$. A prespecified number of clusters $K < N$ is postulated, and each one is labeled by an integer $k \in \{1, \ldots, K\}$. Each observation is assigned to one and only one cluster. These assignments can be characterized by a many-to-one mapping, or *encoder* $k = C(i)$, that assigns the ith observation to the kth cluster. One seeks the particular encoder $C^*(i)$ that achieves the required goal (details below), based on the dissimilarities $d(x_i, x_{i'})$ between every pair of observations. These are specified by the user as described above. Generally, the encoder $C(i)$ is explicitly delineated by giving its value (cluster assignment) for each observation i. Thus, the "parameters" of the procedure are the individual cluster assignments for each of the N observations. These are adjusted so as to *minimize* a "loss" function that characterizes the degree to which the clustering goal is *not* met.

Combinatorial Algorithms

One approach is to directly specify a mathematical loss function and attempt to minimize it through some combinatorial optimization algorithm. Since the goal is to assign close points to the same cluster, a natural loss (or "energy") function would be

$$W(C) = \frac{1}{2} \sum_{k=1}^{K} \sum_{C(i)=k} \sum_{C(i')=k} d(x_i, x_{i'}). \tag{14.28}$$

This criterion characterizes the extent to which observations assigned to the same cluster tend to be close to one another. It is sometimes referred to as the "within cluster" point scatter since

$$T = \frac{1}{2} \sum_{i=1}^{N} \sum_{i'=1}^{N} d_{ii'} = \frac{1}{2} \sum_{k=1}^{K} \sum_{C(i)=k} \left(\sum_{C(i')=k} d_{ii'} + \sum_{C(i')\neq k} d_{ii'} \right),$$

or

$$T = W(C) + B(C),$$

Combinatorial Algorithms

where $d_{ii'} = d(x_i, x_{i'})$. Here T is the *total* point scatter, which is a constant given the data, independent of cluster assignment. The quantity

$$B(C) = \frac{1}{2} \sum_{k=1}^{K} \sum_{C(i)=k} \sum_{C(i')\neq k} d_{ii'} \qquad (14.29)$$

is the *between-cluster* point scatter. This will tend to be large when observations assigned to different clusters are far apart. Thus one has

$$W(C) = T - B(C)$$

and minimizing $W(C)$ is equivalent to *maximizing* $B(C)$.

Cluster analysis by combinatorial optimization is straightforward in principle. One simply minimizes W or equivalently maximizes B over all possible assignments of the N data points to K clusters. Unfortunately, such optimization by complete enumeration is feasible only for very small data sets. The number of distinct assignments is (Jain and Dubes, 1988)

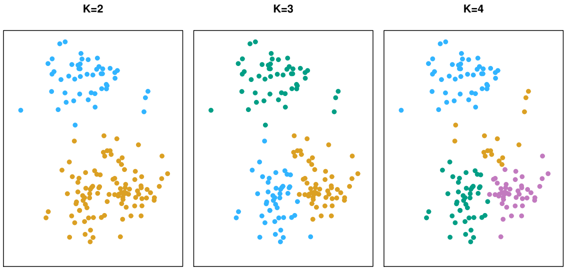
$$S(N,K) = \frac{1}{K!} \sum_{k=1}^{K} (-1)^{K-k} \binom{K}{k} k^N. \qquad (14.30)$$

Combinatorial Algorithms

For example, $S(10,4) = 34,105$ which is quite feasible. But, $S(N,K)$ grows very rapidly with increasing values of its arguments. Already $S(19,4) \simeq$

10^{10}, and most clustering problems involve much larger data sets than $N = 19$. For this reason, practical clustering algorithms are able to examine only a very small fraction of all possible encoders $k = C(i)$. The goal is to identify a small subset that is likely to contain the optimal one, or at least a good suboptimal partition.

K-means clustering



A simulated data set with 150 observations in 2-dimensional space. Panels show the results of applying K-means clustering with different values of K, the number of clusters. The color of each observation indicates the cluster to which it was assigned using the K-means clustering algorithm. Note that there is no ordering of the clusters, so the cluster coloring is arbitrary. These cluster labels were not used in clustering; instead, they are the outputs of the clustering procedure.

Details of K-means clustering

Let $C_1, \ldots, C_K$ denote sets containing the indices of the observations in each cluster. These sets satisfy two properties:

1. $C_1 \cup C_2 \cup \ldots \cup C_K = \{1, \ldots, n\}$. In other words, each observation belongs to at least one of the K clusters.
2. $C_k \cap C_{k'} = \emptyset$ for all $k \neq k'$. In other words, the clusters are non-overlapping: no observation belongs to more than one cluster.

For instance, if the ith observation is in the kth cluster, then $i \in C_k$.

Details of K-means clustering: continued

- The idea behind K-means clustering is that a *good* clustering is one for which the *within-cluster variation* is as small as possible.
- The within-cluster variation for cluster C_k is a measure $\mathrm{WCV}(C_k)$ of the amount by which the observations within a cluster differ from each other.
- Hence we want to solve the problem

$$\underset{C_1,\ldots,C_K}{\text{minimize}} \left\{ \sum_{k=1}^{K} \mathrm{WCV}(C_k) \right\}. \tag{2}$$

- In words, this formula says that we want to partition the observations into K clusters such that the total within-cluster variation, summed over all K clusters, is as small as possible.

How to define within-cluster variation?

- Typically we use Euclidean distance

$$\mathrm{WCV}(C_k) = \frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2, \tag{3}$$

  where $|C_k|$ denotes the number of observations in the kth cluster.

- Combining (2) and (3) gives the optimization problem that defines K-means clustering,

$$\underset{C_1,\ldots,C_K}{\text{minimize}} \left\{ \sum_{k=1}^{K} \frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2 \right\}. \tag{4}$$

K-Means Clustering Algorithm

1. Randomly assign a number, from 1 to K, to each of the observations. These serve as initial cluster assignments for the observations.
2. Iterate until the cluster assignments stop changing:
    2.1 For each of the K clusters, compute the cluster *centroid*. The kth cluster centroid is the vector of the p feature means for the observations in the kth cluster.
    2.2 Assign each observation to the cluster whose centroid is closest (where *closest* is defined using Euclidean distance).

Properties of the Algorithm

- This algorithm is guaranteed to decrease the value of the objective (4) at each step. *Why?*

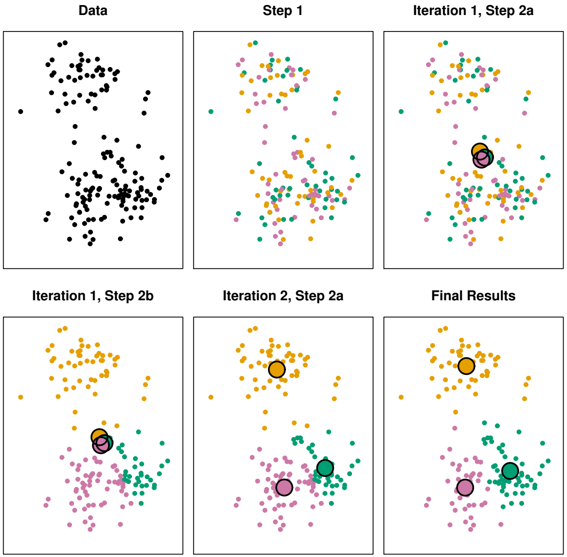
Properties of the Algorithm

- This algorithm is guaranteed to decrease the value of the objective (4) at each step. *Why?* Note that

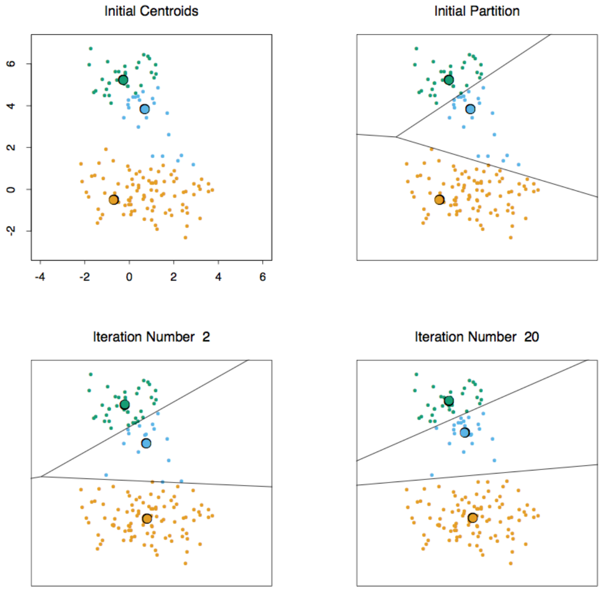
$$\frac{1}{|C_k|}\sum_{i,i'\in C_k}\sum_{j=1}^{p}(x_{ij}-x_{i'j})^2 = 2\sum_{i\in C_k}\sum_{j=1}^{p}(x_{ij}-\bar{x}_{kj})^2,$$

  where $\bar{x}_{kj} = \frac{1}{|C_k|}\sum_{i\in C_k} x_{ij}$ is the mean for feature j in cluster C_k.
- however it is not guaranteed to give the global minimum. *Why not?*

Example


Data
Step 1
Iteration 1, Step 2a
Iteration 1, Step 2b
Iteration 2, Step 2a
Final Results

K-means



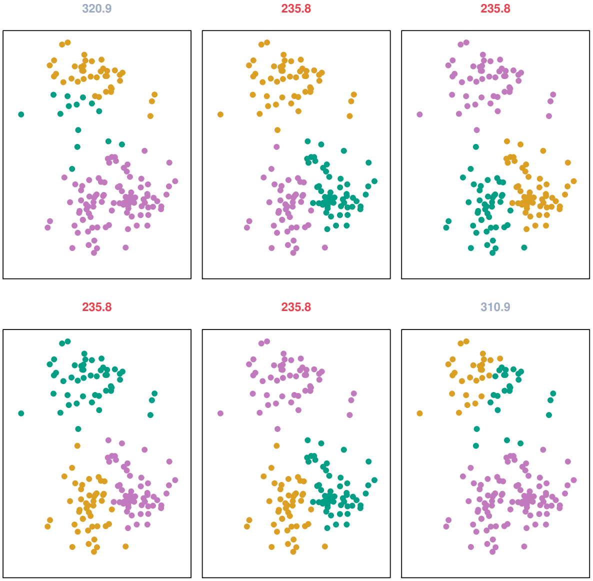
Initial Centroids
Initial Partition
Iteration Number 2
Iteration Number 20
-4
-2
0
2
4
6
-2
0
2
4
6

Details of Previous Figure

The progress of the K-means algorithm with $K=3$.

- *Top left:* The observations are shown.
- *Top center:* In Step 1 of the algorithm, each observation is randomly assigned to a cluster.
- *Top right:* In Step 2(a), the cluster centroids are computed. These are shown as large colored disks. Initially the centroids are almost completely overlapping because the initial cluster assignments were chosen at random.
- *Bottom left:* In Step 2(b), each observation is assigned to the nearest centroid.
- *Bottom center:* Step 2(a) is once again performed, leading to new cluster centroids.
- *Bottom right:* The results obtained after 10 iterations.

Example: different starting values



Details of Previous Figure

K-means clustering performed six times on the data from previous figure with $K = 3$, each time with a different random assignment of the observations in Step 1 of the K-means algorithm.
Above each plot is the value of the objective (4).
Three different local optima were obtained, one of which resulted in a smaller value of the objective and provides better separation between the clusters.
Those labeled in red all achieved the same best solution, with an objective value of 235.8

K-means

$$C^* = \min_C \sum_{k=1}^{K} N_k \sum_{C(i)=k} ||x_i - \bar{x}_k||^2$$

can be obtained by noting that for any set of observations S

$$\bar{x}_S = \underset{m}{\operatorname{argmin}} \sum_{i \in S} ||x_i - m||^2. \tag{14.32}$$

Hence we can obtain C^* by solving the enlarged optimization problem

$$\min_{C, \{m_k\}_1^K} \sum_{k=1}^{K} N_k \sum_{C(i)=k} ||x_i - m_k||^2. \tag{14.33}$$

This can be minimized by an alternating optimization procedure given in Algorithm 14.1.

Each of steps 1 and 2 reduces the value of the criterion (14.33), so that convergence is assured. However, the result may represent a suboptimal local minimum. The algorithm of Hartigan and Wong (1979) goes further, and ensures that there is no single switch of an observation from one group to another group that will decrease the objective. In addition, one should start the algorithm with many different random choices for the starting means, and choose the solution having smallest value of the objective function.

K-means

Algorithm 14.1 *K-means Clustering.*

1. For a given cluster assignment C, the total cluster variance (14.33) is minimized with respect to $\{m_1, \dots, m_K\}$ yielding the means of the currently assigned clusters (14.32).

2. Given a current set of means $\{m_1, \dots, m_K\}$, (14.33) is minimized by assigning each observation to the closest (current) cluster mean. That is,
$$C(i) = \underset{1 \le k \le K}{\operatorname{argmin}} \, ||x_i - m_k||^2. \tag{14.34}$$

3. Steps 1 and 2 are iterated until the assignments do not change.

K-medoids

- the K-means algorithm is appropriate when the dissimilarity measure is taken to be squared Euclidean distance $D(x_i, x_i)$.
- This requires all of the variables to be of the quantitative type.
- In addition, using squared Euclidean distance places the highest influence on the largest distances. This causes the procedure to lack robustness against outliers that produce very large distances.

K-medoids

Algorithm 14.2 *K-medoids Clustering.*

1. For a given cluster assignment C find the observation in the cluster minimizing total distance to other points in that cluster:
$$i_k^* = \operatorname*{argmin}_{\{i:C(i)=k\}} \sum_{C(i')=k} D(x_i, x_{i'}). \tag{14.35}$$
Then $m_k = x_{i_k^*}$, $k = 1, 2, \ldots, K$ are the current estimates of the cluster centers.

2. Given a current set of cluster centers $\{m_1, \ldots, m_K\}$, minimize the total error by assigning each observation to the closest (current) cluster center:
$$C(i) = \operatorname*{argmin}_{1 \le k \le K} D(x_i, m_k). \tag{14.36}$$

3. Iterate steps 1 and 2 until the assignments do not change.

K-medoids

Solving (14.32) for each provisional cluster k requires an amount of computation proportional to the number of observations assigned to it, whereas for solving (14.35) the computation increases to $O(N_k^2)$. Given a set of cluster "centers," $\{i_1, \ldots, i_K\}$, obtaining the new assignments

$$C(i) = \underset{1 \leq k \leq K}{\operatorname{argmin}}\, d_{i i_k^*} \tag{14.37}$$

requires computation proportional to $K \cdot N$ as before. Thus, K-medoids is far more computationally intensive than K-means.

Alternating between (14.35) and (14.37) represents a particular heuristic search strategy for trying to solve

$$\min_{C,\, \{i_k\}_1^K} \sum_{k=1}^{K} \sum_{C(i)=k} d_{i i_k}. \tag{14.38}$$

K-medoids

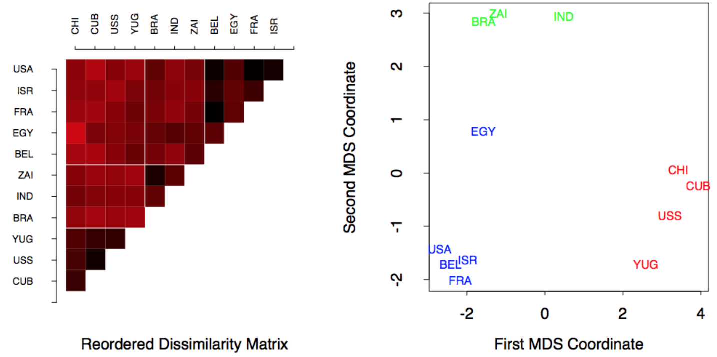


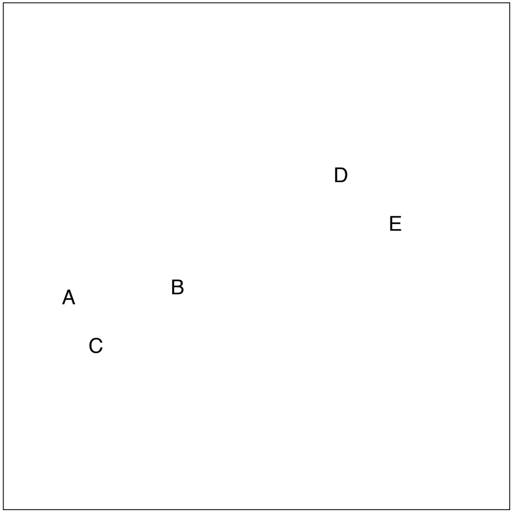
FIGURE 14.10. *Survey of country dissimilarities. (Left panel:) dissimilarities reordered and blocked according to 3-medoid clustering. Heat map is coded from most similar (dark red) to least similar (bright red). (Right panel:) two-dimensional multidimensional scaling plot, with 3-medoid clusters indicated by different colors.*

Hierarchical Clustering

- K-means clustering requires us to pre-specify the number of clusters K. This can be a disadvantage (later we discuss strategies for choosing K)
- *Hierarchical clustering* is an alternative approach which does not require that we commit to a particular choice of K.
- In this section, we describe *bottom-up* or *agglomerative* clustering. This is the most common type of hierarchical clustering, and refers to the fact that a dendrogram is built starting from the leaves and combining clusters up to the trunk.

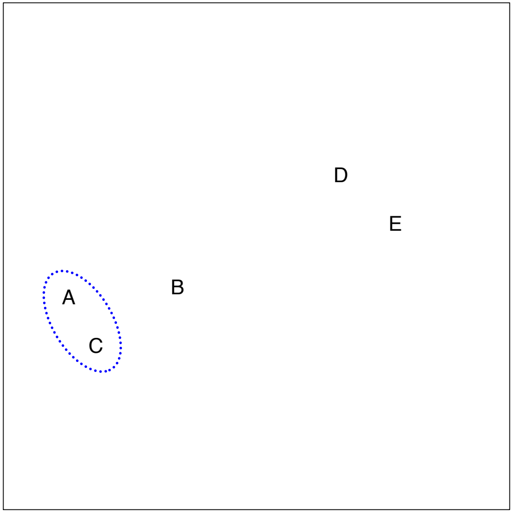
Hierarchical Clustering: the idea

Builds a hierarchy in a “bottom-up” fashion...



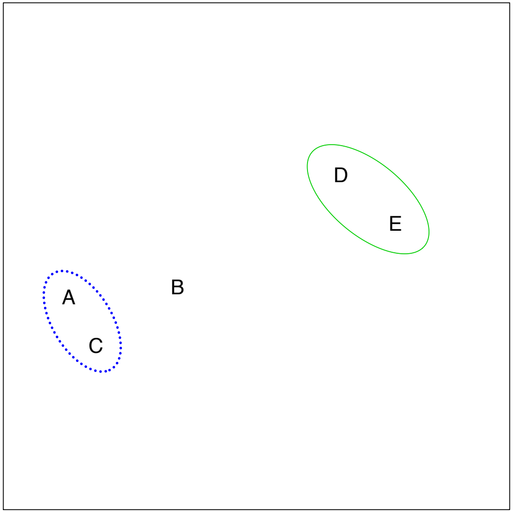
Hierarchical Clustering: the idea

Builds a hierarchy in a “bottom-up” fashion...



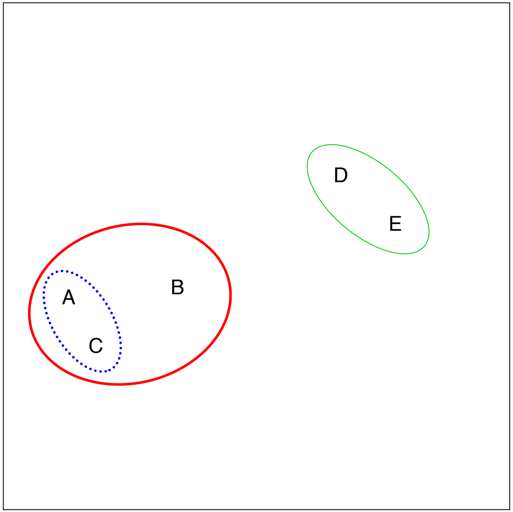
Hierarchical Clustering: the idea

Builds a hierarchy in a “bottom-up” fashion...



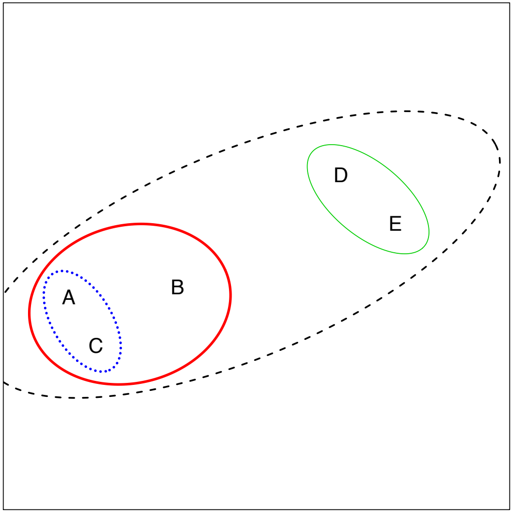
Hierarchical Clustering: the idea

Builds a hierarchy in a “bottom-up” fashion...



Hierarchical Clustering: the idea

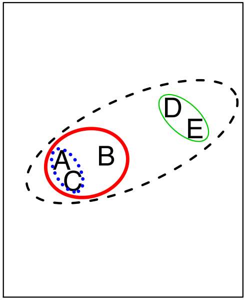
Builds a hierarchy in a “bottom-up” fashion...

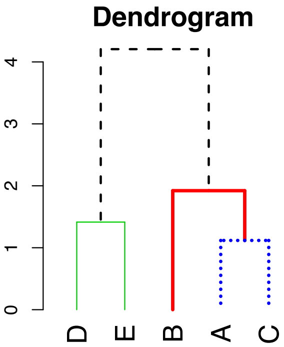


Hierarchical Clustering Algorithm

The approach in words:

- Start with each point in its own cluster.
- Identify the closest two clusters and merge them.
- Repeat.
- Ends when all points are in a single cluster.

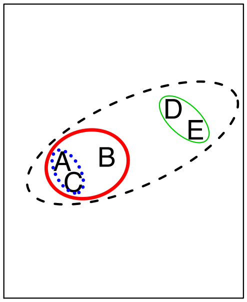


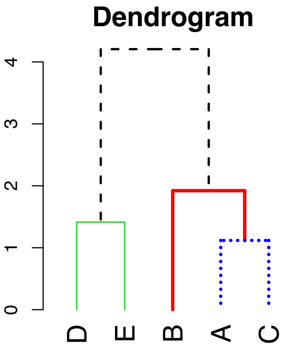


Hierarchical Clustering Algorithm

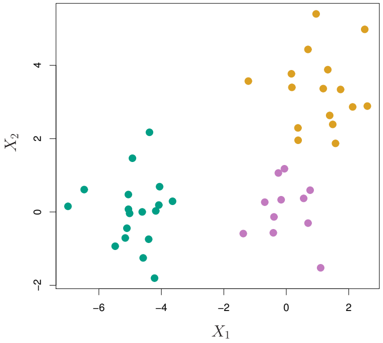
The approach in words:

- Start with each point in its own cluster.
- Identify the closest two clusters and merge them.
- Repeat.
- Ends when all points are in a single cluster.



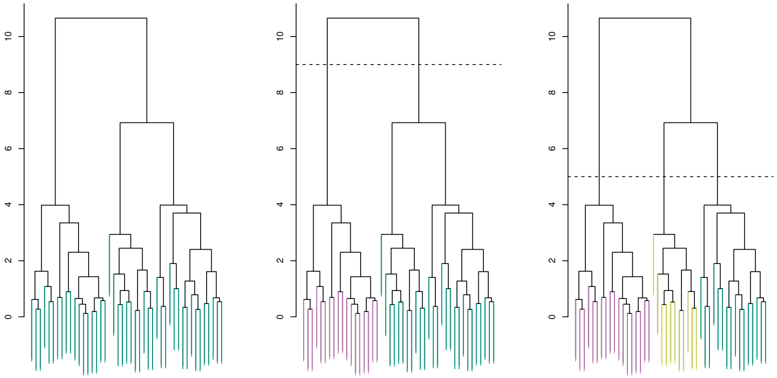


An Example



45 observations generated in 2-dimensional space. In reality there are three distinct classes, shown in separate colors. However, we will treat these class labels as unknown and will seek to cluster the observations in order to discover the classes from the data.

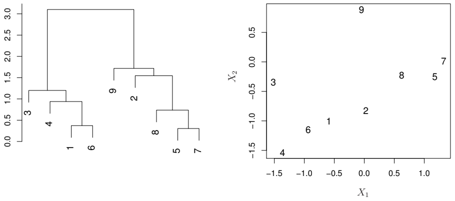
Application of hierarchical clustering



Details of previous figure

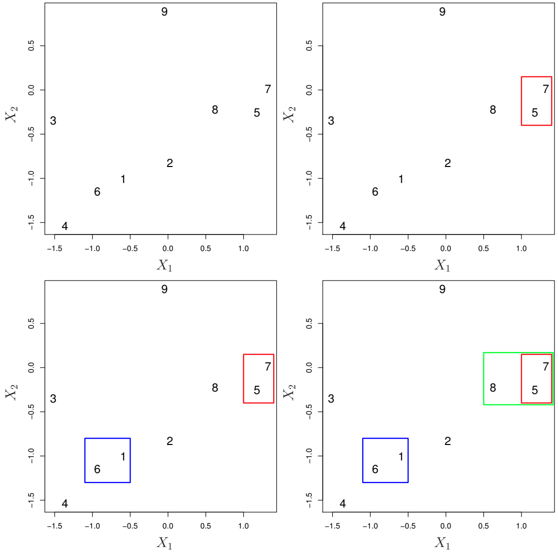
- *Left:* Dendrogram obtained from hierarchically clustering the data from previous slide, with complete linkage and Euclidean distance.
- *Center:* The dendrogram from the left-hand panel, cut at a height of 9 (indicated by the dashed line). This cut results in two distinct clusters, shown in different colors.
- *Right:* The dendrogram from the left-hand panel, now cut at a height of 5. This cut results in three distinct clusters, shown in different colors. Note that the colors were not used in clustering, but are simply used for display purposes in this figure

Another Example



- An illustration of how to properly interpret a dendrogram with nine observations in two-dimensional space. The raw data on the right was used to generate the dendrogram on the left.
- Observations 5 and 7 are quite similar to each other, as are observations 1 and 6.
- However, observation 9 is *no more similar to* observation 2 than it is to observations $8, 5$, and 7, even though observations 9 and 2 are close together in terms of horizontal distance.
- This is because observations $2, 8, 5$, and 7 all fuse with observation 9 at the same height, approximately 1.8.

Merges in previous example

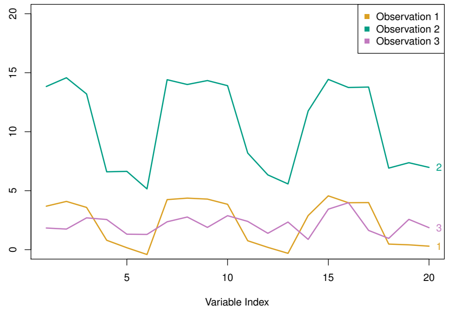


Types of Linkage

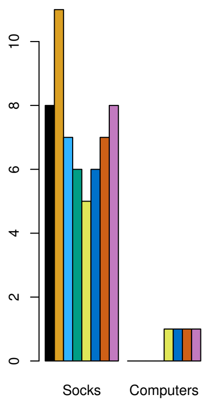
Linkage	*Description*
Complete	Maximal inter-cluster dissimilarity. Compute all pairwise dissimilarities between the observations in cluster A and the observations in cluster B, and record the *largest* of these dissimilarities.
Single	Minimal inter-cluster dissimilarity. Compute all pairwise dissimilarities between the observations in cluster A and the observations in cluster B, and record the *smallest* of these dissimilarities.
Average	Mean inter-cluster dissimilarity. Compute all pairwise dissimilarities between the observations in cluster A and the observations in cluster B, and record the *average* of these dissimilarities.
Centroid	Dissimilarity between the centroid for cluster A (a mean vector of length p) and the centroid for cluster B. Centroid linkage can result in undesirable *inversions*.

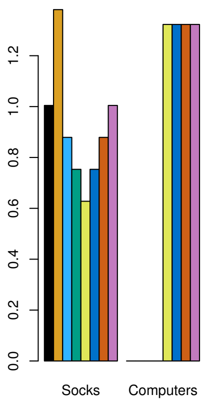
Choice of Dissimilarity Measure

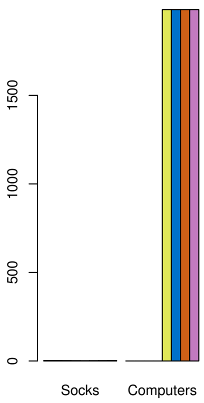
- So far have used Euclidean distance.
- An alternative is *correlation-based distance* which considers two observations to be similar if their features are highly correlated.
- This is an unusual use of correlation, which is normally computed between variables; here it is computed between the observation profiles for each pair of observations.



Scaling of the variables matters





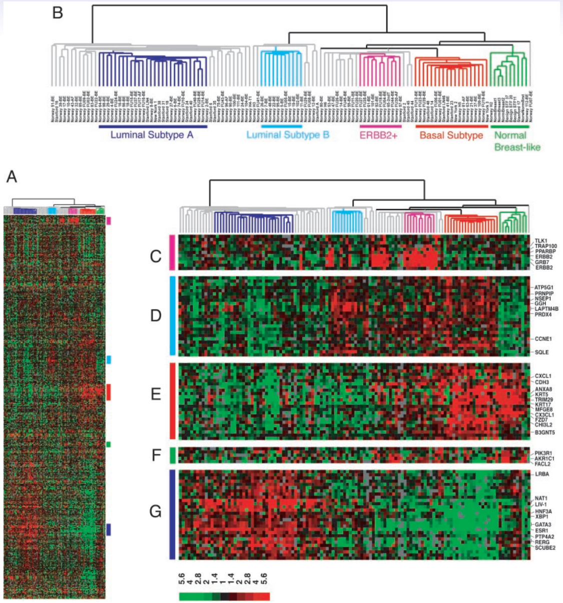


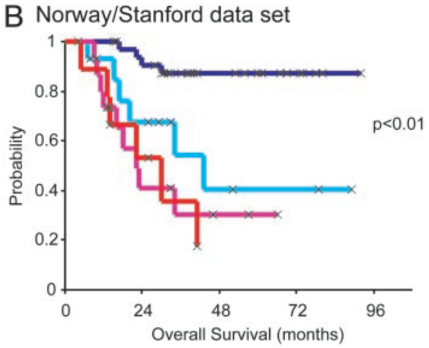
Practical issues

- Should the observations or features first be standardized in some way? For instance, maybe the variables should be centered to have mean zero and scaled to have standard deviation one.
- In the case of hierarchical clustering,
  - What dissimilarity measure should be used?
  - What type of linkage should be used?
- How many clusters to choose? (in both K-means or hierarchical clustering). Difficult problem. No agreed-upon method. See Elements of Statistical Learning, chapter 13 for more details.

Example: breast cancer microarray study

- "Repeated observation of breast tumor subtypes in independent gene expression data sets;" Sorlie at el, PNAS 2003
- Average linkage, correlation metric
- Clustered samples using 500 *intrinsic genes:* each woman was measured before and after chemotherapy. Intrinsic genes have smallest within/between variation.


B
Luminal Subtype A
Luminal Subtype B
ERBB2+
Basal Subtype
Normal
Breast-like
A
C
D
E
F
G
TLK1
TRAP100
PPARBP
ERBB2
GRB7
ERBB2
ATP5G1
PRNPIP
NSEP1
GGH
LAPTM4B
PRDX4
CCNE1
SQLE
CXCL1
CDH3
ANXA8
KRT5
TRIM29
KRT17
MFGE8
CX3CL1
FZD7
CHI3L2
B3GNT5
PIK3R1
AKR1C1
FACL2
LRBA
NAT1
LIV-1
HNF3A
XBP1
GATA3
ESR1
PTP4A2
RERG
SCUBE2
5.6
4
2.8
2
1.4
1
1.4
2
2.8
4
5.6


B Norway/Stanford data set
1
0.8
0.6
0.4
0.2
0
Probability
0
24
48
72
96
Overall Survival (months)
p<0.01

Conclusions

- *Unsupervised learning* is important for understanding the variation and grouping structure of a set of unlabeled data, and can be a useful pre-processor for supervised learning
- It is intrinsically more difficult than *supervised learning* because there is no gold standard (like an outcome variable) and no single objective (like test set accuracy)
- It is an active field of research, with many recently developed tools such as *self-organizing maps*, *independent components analysis* and *spectral clustering*.
  See *The Elements of Statistical Learning*, chapter 14.