Clustering and Classification

Dimitris Karlis

Department of Statistics
Athens University of Economics
karlis@aueb.gr

Athens, June, 2014

Aim

- To present state of the art methods for clustering and classification
- To investigate their differences but also their relationships
- To see how R implements (some) of the methods and see real data examples
- To discuss further issues not treated here due to time limitations

Learning

Both methods are widely used in other disciplines like machine learning and data mining.

- Cluster analysis is also known as unsupervised learning : we do not know the labelling of the observations and we try to identify it.
- Classification methods are also known as supervised learning

The goal

The goal is to find clusters/groups of observations with the property that

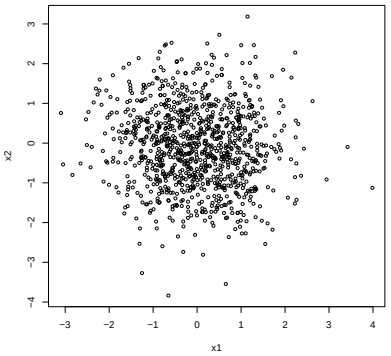
- within the cluster the observations "look similar"/"are close together" but
- observations from different clusters are "different"/"not close together"

Main approaches

- Distance based methods
- Partition methods
- Models based methods

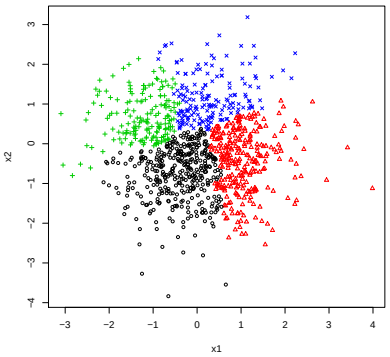
Warnings

Packages will always find some clusters which can be very bad and meaningless



Warnings

Packages will always find some clusters which can be very bad and meaningless



Partition methods

We need to find k clusters

- Create all possible classification of the n observations in k clusters (they will be huge)
- For each one calculate a score indicating how good is
- Select the one with better score

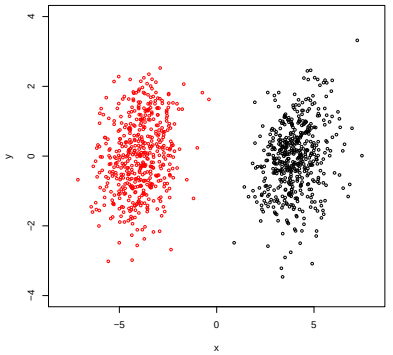
full enumeration impossible for real data sets, but what if we have clever algorithms?

Important questions

- Which method to use?
- How many cluster?
- Which variables can really used for clustering?
- How can se assess how good is our cluster?
- How do we interpret the clusters?

Are all variables Interesting

Are both X and Y useful for the clustering?



Distances

- We need to quantify "how similar" are two observation (or "how distant".
- We need to define an approrpiate "distance"/"similarity"
- There are many distances depending on the type of data, length, purpose etc

Distances

Strictly speaking a distance between observations x, y and z should satisfy

- $d(x, y) = d(y, x) \geq 0$
- $d(x, y) \leq d(x, z) + d(z, y)$
- If $d(x, y) \neq 0$ then $x \neq y$
- $d(x, x) = 0$

These are the desirable properties for a distance BUT in practice a lot of them do not satisfy all these properties.

Distances for Continuous Data

Data points $\mathbf{x} = (x_1, \ldots, x_p)'$ and $\mathbf{y} = (y_1, \ldots, y_p)'$

- Euclidean: $D(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{p} (x_i - y_i)^2$
- Manhattan: $D(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{p} |x_i - y_i|$
- Mahalanobis $D(\mathbf{x}, \mathbf{y}) = (\mathbf{x} - \mathbf{y})' S^{-1} (\mathbf{x} - \mathbf{y})$
  It takes into account the covariance (S)!
- Many many others (and more to come...)

How to choose?

- Euclidean does not take into account the scale difference. So a variable with large variance/scale will dominate.
- Manhattan downweights large deviations
- Mahalanobis is statistical preferred distance but it is more time consuming

Distances for other types of data

For binary data: Matching coefficient
Create the 2 table for each pair of observations

	1	0
1	a	b
0	c	d

i.e. d and a indicates matching between the two observations.

Distances for binary data

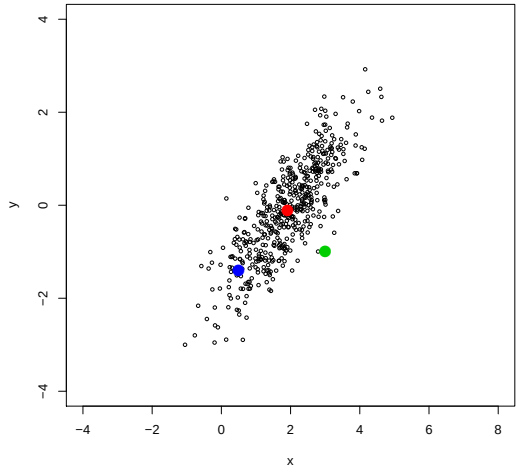
We define also the similarity $s(x, y) - 1d(x, y)$ Some similarities are

	Symmetric	Asymmetric
Matching	$\frac{a+d}{a+b+c+d}$	$\frac{a}{a+b+c+}$
Rogers	$\frac{a+d}{a+d+2(b+c)}$	$\frac{a}{a+2(b+c)}$
Sokal	$\frac{2(a+d)}{2(a+d)+b+c}$	$\frac{2a}{2a+b+c}$

Asymmetric distances ignore d. Why? Common absence of a characteristic does not contribute to their similarity.

Compare

Blue and Green points: which is closer to the red one depends on the distance.



Distances

- In more complicated circumstances it is not always obvious how to define distances
- E.g. : mixed mode data, time series data, text data
- Perhaps we need to transform the data to derive meaningful distances
- Gower distance tries to do this. It is defined as

Gower-Distance

The key idea is to measure for each variable its distance in a range between $(0, 1)$ and take the average.

This works for example if for binary/nominal variables we have simple matching distance and for continuous variables we rescale their distance to $(0, 1)$, using

$$\frac{|x_i - y_i|}{R}$$

where x_i and $y_i$is the value for the i-th variable for the observation x and y respectively and R is the range of that variable.

Be aware that each distance looks at different aspect of he data

Hierarchical clustering

Two approaches: Divisive or Agglomerative
The idea: At each step join/split the closer observations/cluster.
When joining (we move forward) we have agglomerative
When splitting (we move backward) we have divisive.

Hierarchical clustering

Given a set of N items to be clustered, and an NxN distance (or similarity) matrix, the basic process hierarchical clustering is this:

- Start by assigning each item to its own cluster, so that if you have N items, you now have N clusters, each containing just one item.
- Find the closest (most similar) pair of clusters and merge them into a single cluster, so that now you have one less cluster.
- Compute distances (similarities) between the new cluster and each of the old clusters (linkage).
- Repeat steps 2 and 3 until all items are clustered into a single cluster of size N.

What distance? What linkage? How many clusters?

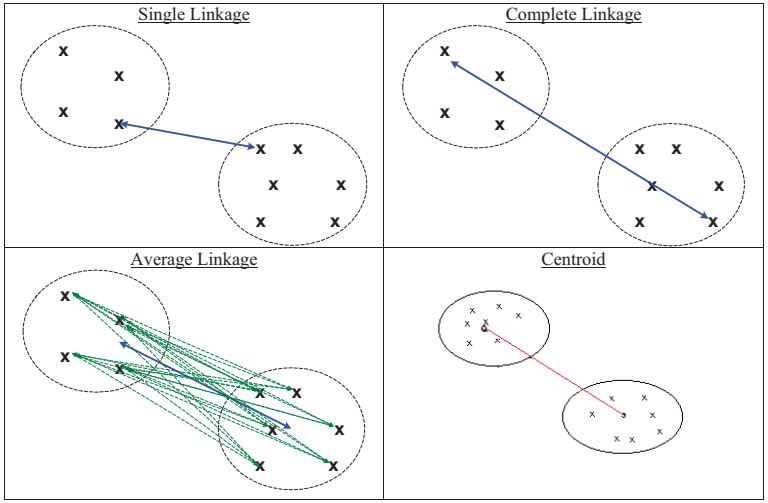
Linkage

- single-link clustering (closer neighbor) : we consider the distance between one cluster and another cluster to be equal to the shortest distance from any member of one cluster to any member of the other cluster. If the data consist ofsimilarities, we consider the similarity between one cluster and another cluster to be equal to the greatest similarity from any member of one cluster to any member of the other cluster.
- complete-link clustering (also called the diameter or maximum method): we consider the distance between one cluster and another cluster to be equal to the longest distance from any member of one cluster to any member of the other cluster.

Linkage (2)

- average-link clustering : we consider the distance between one cluster and another cluster to be equal to the average distance from any member of one cluster to any member of the other cluster.
- Ward: Ward's minimum variance criterion minimizes the total within-cluster variance. At each step the pair of clusters with minimum between-cluster distance are merged.
- Centroid: find the center of the cluster and calculate (if possible) the distance from it.

Compare



Which Linkage

- Ward method gives clusters of equal size in general
- Nearest neighbor can have the chain effect (each step just adds one observation to the existing cluster)
- Centroid: compact and elliptical clusters
- Furthest neighbor: not compact clusters

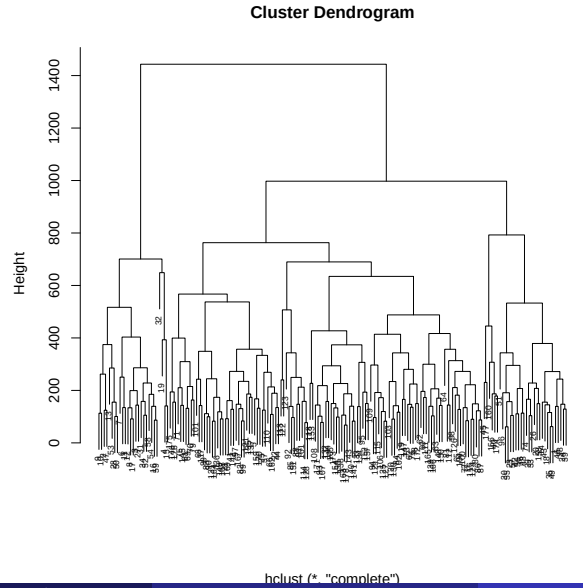
Wine dataset

Data on 27 chemical and physical properties of three types of wine (Barolo, Grignolino, Barbera) from the Piedmont region of Italy.
The study did include one further variable but the sulphur measurements were not available.

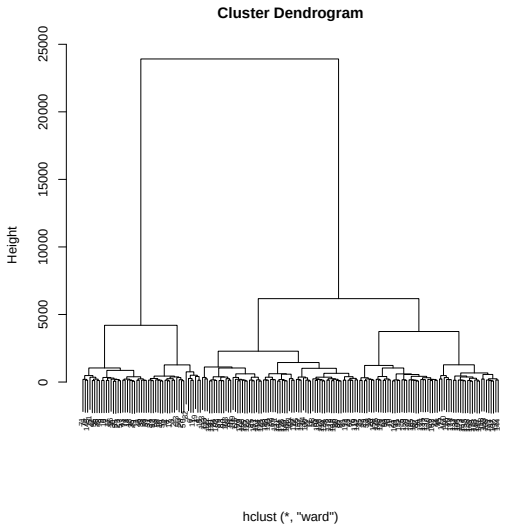
Benchmark data set in many cluster applications.

Data can be found in
`http://archive.ics.uci.edu/ml/datasets/Wine`

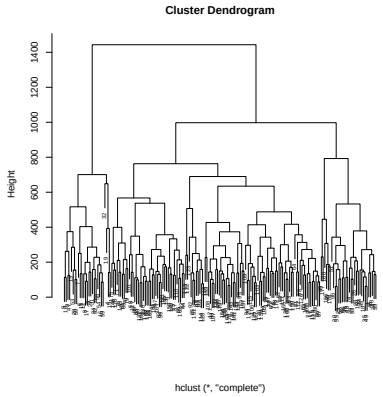
Dendrogram- Wine data Complete



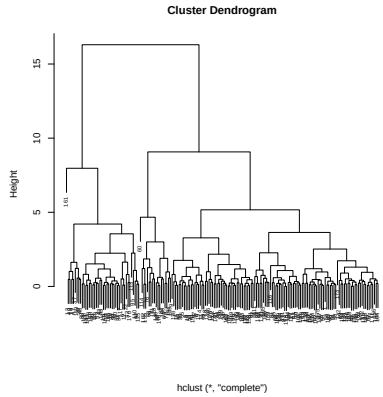
Dendrogram- Wine data Ward



Dendrogram- Wine data Complete

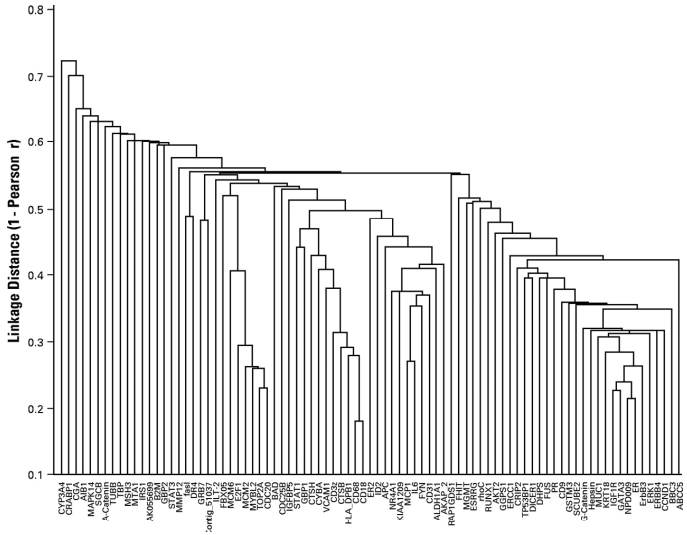


25 variables

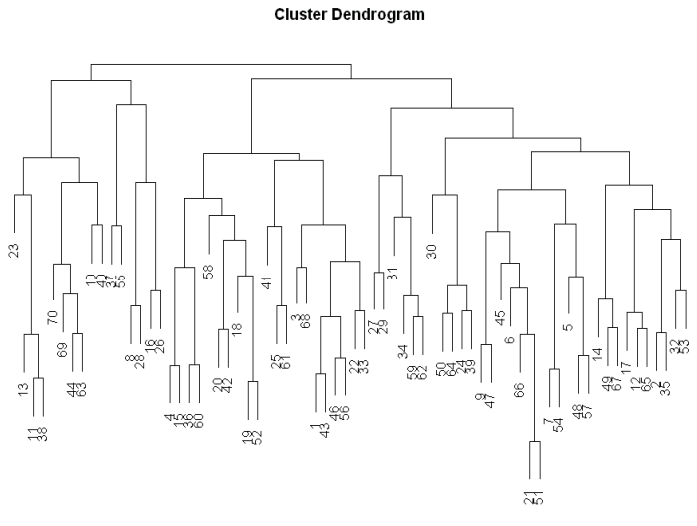


2 variables

Chain Effect



Noise



Comments

- Computationally intensive
- Observations put together early cannot split apart later
- Linkage is important
- Selecting the number of clusters is done based on after-processing of results.

Silhouette values

Silhouette refers to a method of interpretation and validation of clusters of data. The technique provides a succinct graphical representation of how well each object lies within its cluster.

Assume the data have been clustered via any technique, such as k-means, into k clusters. For each datum i, let $a(i)$ be the average dissimilarity of i with all other data within the same cluster (the smaller the value, the better the assignment).

We then define the average dissimilarity of point i to a cluster c as the the average of the distance from i to points in c. $b(i)$ be the lowest average dissimilarity of i to any other cluster which i is not a member.

Silhouette values

We now define:

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$

Which can be written as:

$$s(i) = \begin{cases} 1 - a(i)/b(i), & \text{if } a(i) < b(i) \\ 0, & \text{if } a(i) = b(i) \\ b(i)/a(i) - 1, & \text{if } a(i) > b(i) \end{cases}$$

From the above definition it is clear that

$$-1 \leq s(i) \leq 1$$

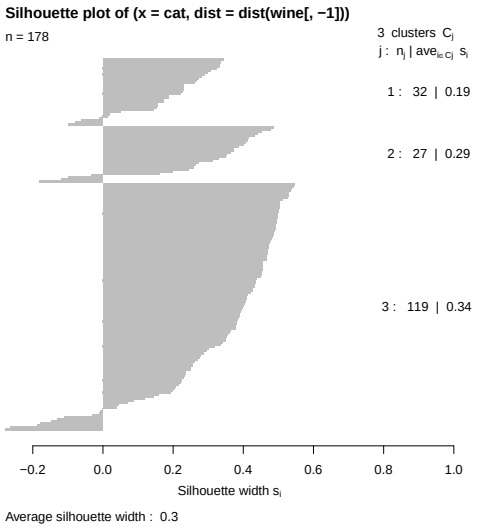
Silhouette values

- If $s(i)$ is close to one means that the datum is appropriately clustered.
- If $s(i)$ is close to negative one, then by the same logic we see that i would be more appropriate if it was clustered in its neighboring cluster.
- If $s(i)$ near zero means that the datum is on the border of two natural clusters.

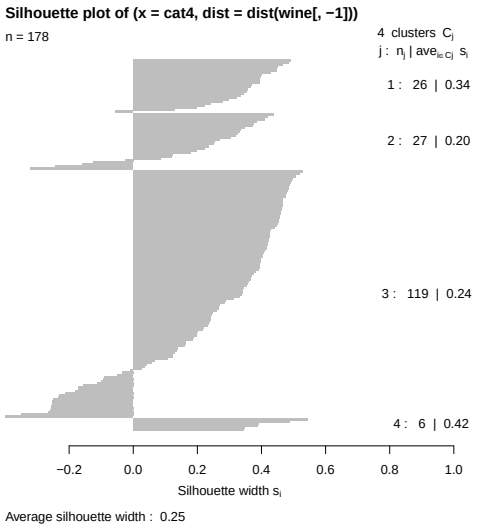
Silhouette values

The average $s(i)$ over all data of the entire dataset is a measure of how appropriately the data has been clustered.
If there are too many or too few clusters, as may occur when a poor choice of k is used, some of the clusters will typically display much narrower silhouettes than the rest.
Thus silhouette plots and averages may be used to determine the natural number of clusters within a dataset.

Wine data-Silhouettes- 3 clusters



Wine data-Silhouettes - 4 clusters



What is a good clustering

Goal: Clusters shall be well separated: observations inside cluster shall be similar but observations from different ones should be different.

We need to measure this:

How to measure this? There are several measures, but we use Wilks' Λ.

Wilks' Λ statistic

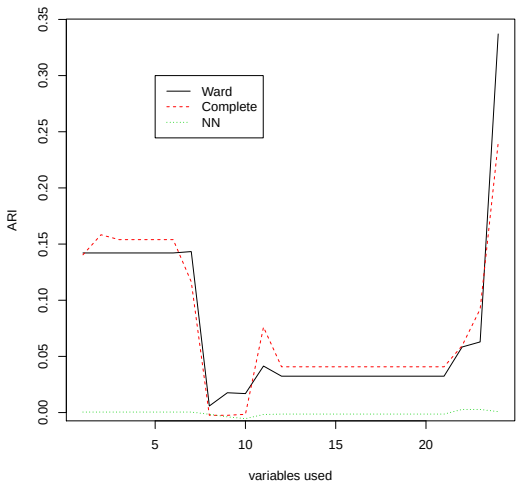
- Used in MANOVA to test equality of vectors of means.
- It measures the within variability wrt to the total variability (within+between).
- If good cluster we expect small values.
- Caution: we need continuous data

Adjusted Rand Index

Measure of partition agreement Answers the question How similar are these two ways of partitioning the data? To evaluate clusters, we compute the Rand Index between actual labels and clusters
The Rand index has a problem, the expected value for any 2 random partitions is relatively high, wed like it to be close to 0. Adjusted Rand index puts the expected value at 0, gives a more dynamic range and is probably a better metric
The adjusted form of the Rand Index, the Adjusted Rand Index, is

$$AdjustedIndex = \frac{Index - ExpectedIndex}{MaxIndex - ExpectedIndex},$$

ARI for wine data - examples



Some comments

- Distance based clustering are HARD methods : a data item is assigned to a cluster with probability 0, 1
- Density-based clustering (SOFT) : each data item is assigned to ALL clusters with a different probability or degree of membership. No harsh boundaries between clusters.
- In practice we can have overlapping clusters (more close to reality).

Some comments

- Variables to be used are important. Their selection can be based on arguments that they shall take different values for different clusters.
- Interpretation of clusters: we may investigate the characteristics of each cluster wrt the variables or other variables not used for the clustering.

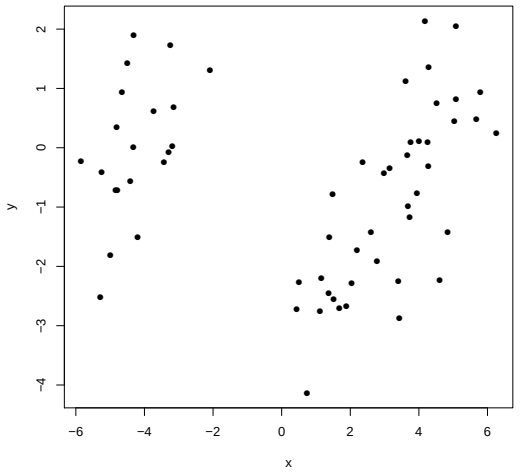
Number of clusters

- Several criteria based on different arguments and types of data
- The common thing is that we obtain some score for different values of k and compare.
- So we need to evaluate cluster several times!

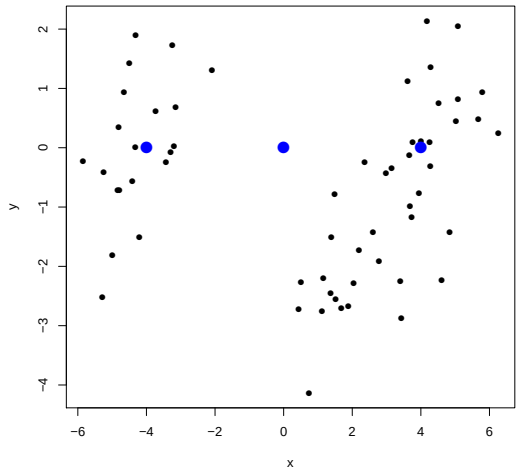
K-means

- Algorithmic approach
- Need to know the number of clusters in advance
- Assigns at each iteration each observation to the closets cluster
- Then updates the cluster centers
- Iterates until no change

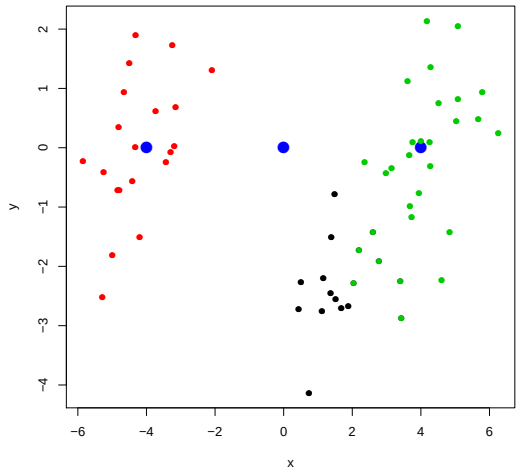
K-means - The data



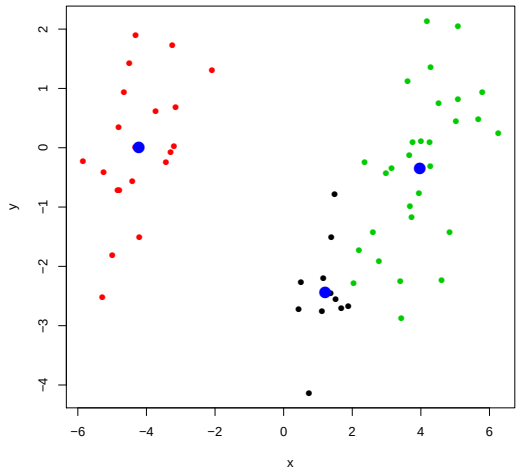
K-means - Choose initial centers



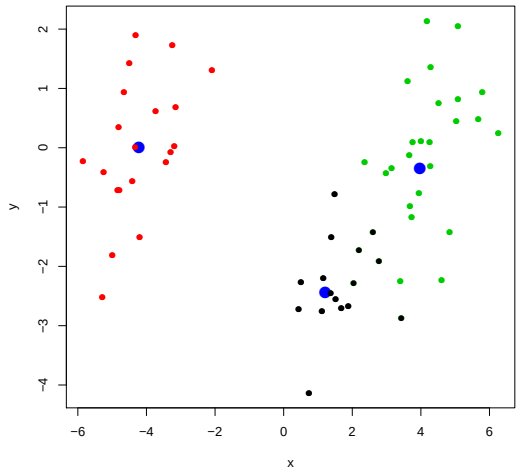
K-means - assign observations



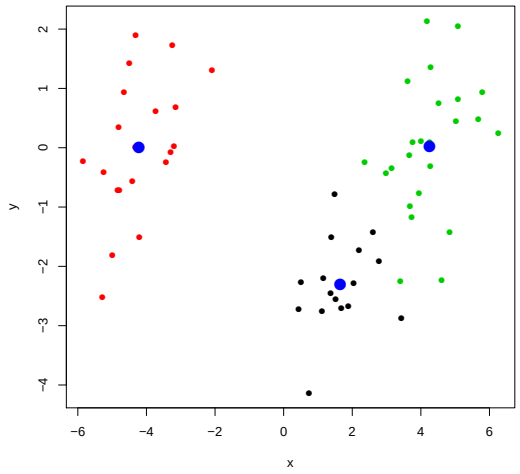
K-means - update centers



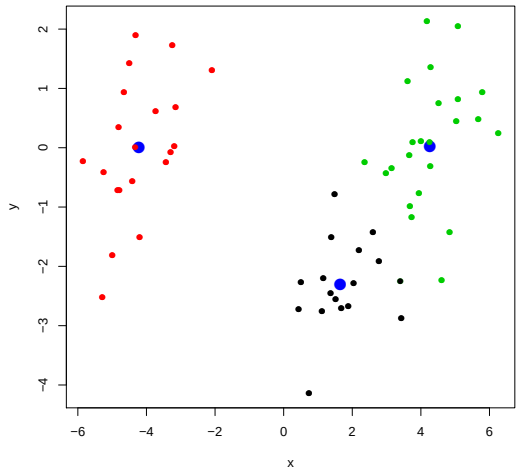
K-means - assign observations



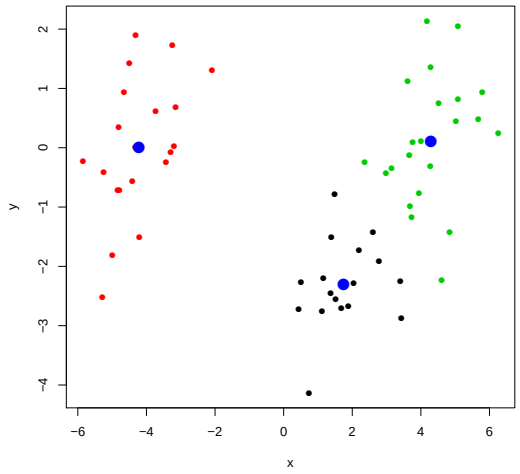
K-means - update centers



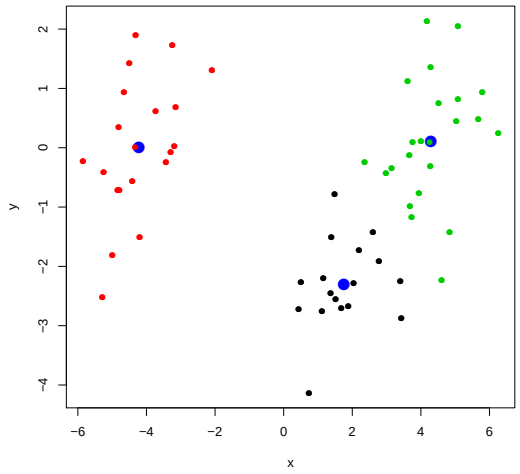
K-means - assign observations



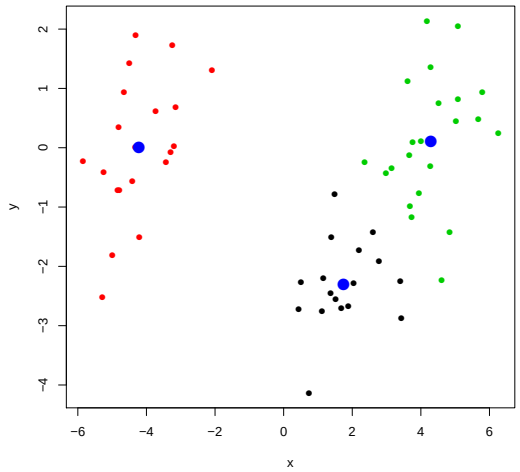
K-means - update centers



K-means - assign observations



K-means - update centers - no change



Comments

- Solution depends on the initial values
- Assignment is based on Euclidean distance
- If many variables we may use different stopping criterion
- Produced clusters are of same shape
- We have to rerun if different number of clusters is needed

Variants

To work with no continuous data we need

- to define "the center" of a cluster
- To be able to measure the distance from it so as to assign new observations

Variants K-medoid, K-centrids etc

K-means in R

R offers 4 different algorithms for implementing K-means

- Hartigan-Wong: allocates each object to one of K groups or clusters to minimize the within-cluster sum of squares
- Lloyd: Given any set of K centers Z, for each center z in Z, let $V(z)$ denote its neighborhood. That is the set of data points for which z is the nearest neighbor. Each stage of Lloyd's algorithm moves every center point z to the centroid of $V(z)$ and then updates $V(z)$ by recomputing the distance from each point to its nearest center. These steps are repeated until convergence.
- Forgy: is a simple alternating least-squares algorithm consisting of the following steps:
- MacQueen: moving all cluster centers to the mean of their respective Voronoi sets

Model Based Clustering

- Distance based methods are rather ad-hoc.
- They do not allow for inferential tools (e.g. goodness of fit, uncertainty)
- We switch to model based clustering via models aiming at having a variety of tools for inference

Total Probability Theorem

From probability we know that if $A_1, A_2, \ldots A_n$ is a partition of the sample space S, then for each event E it holds that

$$P(E) = \sum_{i=1}^{n} P(E \mid A_i) P(A_i)$$

A probability model for clustering

Let's assume that the population can be split in subpopulations.groups/clusters. For each group we know its density

$$\begin{aligned} &\text{group } 1 \quad f(x \mid \theta_1) \\ &\text{group } 2 \quad f(x \mid \theta_2) \\ &\quad \ldots \\ &\text{group } k \quad f(x \mid \theta_k) \end{aligned}$$

then pickng randomly one person from the population we got that

$$f(x) = \sum_{j=1}^{k} p_j f(x \mid \theta_j)$$

where $p_j > 0$, $\sum_{i=1}^{k} p_j = 1$

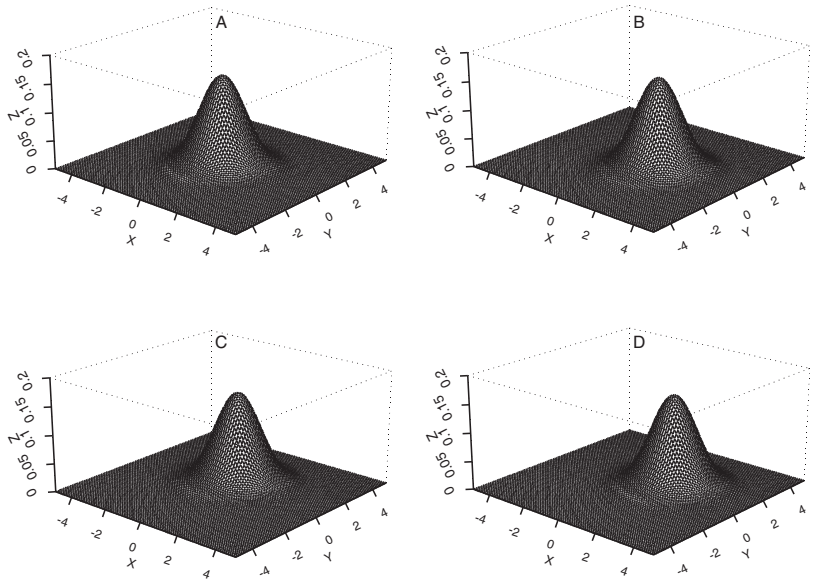
This is a finite mixture from the density f. Finite mixtures have a wide range of applications.

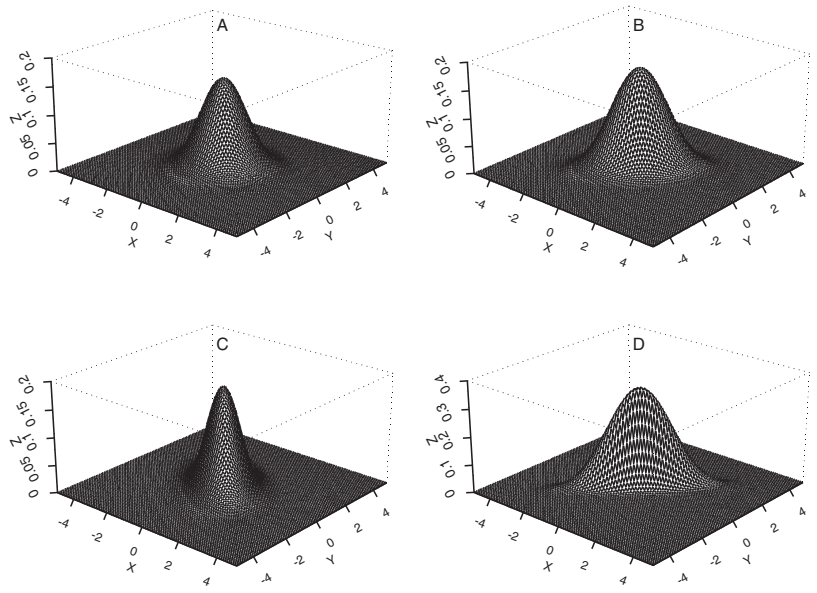
Multivariate Gaussian distribution

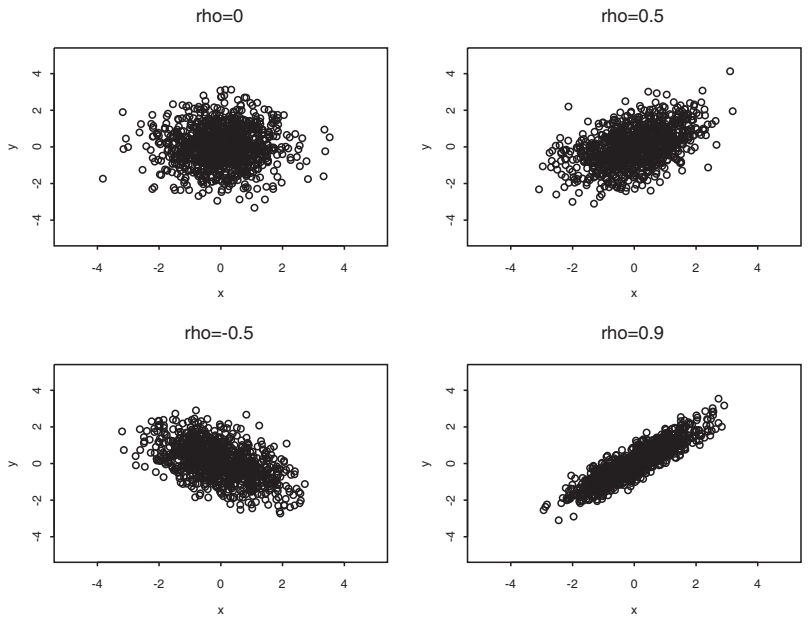
Joint density function

$$f(\mathbf{x}) = (2\pi)^{-p/2} |\mathbf{\Sigma}|^{-1/2} exp\left(-\frac{1}{2} (\mathbf{x} - \mu) \mathbf{\Sigma}^{-1} (\mathbf{x} - \mu)^T \right)$$

where $\mathbf{\Sigma}$ is the covariance matrix $(p \times p)$, μ is the $p \times 1$ vector of means and $\mathbf{x}$ is the vector of rvs
Its role is crucial in multivariate statistics
All marginals and conditional distributions are Gaussian of the corresponding dimension







Finite Mixtures of Multivariate Gaussian distributions

$$f(\mathbf{x}) = \sum_{i=1}^{k} p_j (2\pi)^{-p/2} |\mathbf{\Sigma}_j|^{-1/2} exp\left(-\frac{1}{2}(\mathbf{x}-\mu_j)\mathbf{\Sigma}_j^{-1}(\mathbf{x}-\mu_j)^T\right)$$

- Homoscedastic $\mathbf{\Sigma}_1 = \mathbf{\Sigma}_2, \ldots, \mathbf{\Sigma}_k = \mathbf{\Sigma}$
- Heteroscedastic: different covariance matrices

Parsimonious models

We may rewrite

$$\Sigma_k = \lambda_k D_k A_k D_k'$$

- D_k is the matrix of eigenvectors, determines the orientation
- A_k is a diagonal matrix proportional to the eigenvalues, explains the shape
- λ_k is ascalar, explains the volume

By constraining the parameters λ_k, D_k and A_k within and across the groups, 14 different parsimonious models can be enumerated

Different models

identifier	Model	HC	EM	Distribution	Volume	Shape	Orientation
E		•	•	(univariate)	equal		
V		•	•	(univariate)	variable		
EII	λI	•	•	Spherical	equal	equal	NA
VII	λ_k I	•	•	Spherical	variable	equal	NA
EEI	λA		•	Diagonal	equal	equal	coordinate axes
VEI	$\lambda_k A$		•	Diagonal	variable	equal	coordinate axes
EVI	λA_k		•	Diagonal	equal	variable	coordinate axes
VVI	$\lambda_k A_k$		•	Diagonal	variable	variable	coordinate axes
EEE	$\lambda D A D^T$	•	•	Ellipsoidal	equal	equal	equal
EEV	$\lambda D_k A D_k^T$		•	Ellipsoidal	equal	equal	variable
VEV	$\lambda_k D_k A D_k^T$		•	Ellipsoidal	variable	equal	variable
VVV	$\lambda_k D_k A_k D_k^T$	•	•	Ellipsoidal	variable	variable	variable

EM algorithm

In general it is hard to estimate the finite mixture of Gaussians model.
the log-likelihood is too complicated.
We can however do it easily via an iteratve algorithm called the EM algorithm
The idea is that if we can find some "missing" information structure then we can create the algorithm by estimating what is missing at the E-step and maximizing the "complete" data log-likelihood at the M-step

EM algorithm for Gaussian mixtures

Initial values for $\mathbf{p} = (p_1, \ldots, p_k)$, $\mu_1, \ldots, \mu_k$, $\mathbf{\Sigma}_1, \ldots, \mathbf{\Sigma}_k$. Then *E-step* Calculate

$$\begin{aligned} w_{ij} &= \frac{p_j f(\mathbf{x}_i \mid \theta_j)}{\sum_{i=1}^{k} p_j f(\mathbf{x}_i \mid \theta_j)} \\ &= \frac{p_j (2\pi)^{-p/2} |\mathbf{\Sigma}_j|^{-1/2} exp\left(-\frac{1}{2}(\mathbf{x}_i - \mu_j)\mathbf{\Sigma_j}^{-1}(\mathbf{x}_i - \mu_j)^T\right)}{\sum_{i=1}^{k} p_j (2\pi)^{-p/2} |\mathbf{\Sigma}_j|^{-1/2} exp\left(-\frac{1}{2}(\mathbf{x}_i - \mu_j)\mathbf{\Sigma_j}^{-1}(\mathbf{x}_i - \mu_j)^T\right)} \end{aligned}$$

EM algorithm for Gaussian mixtures (2)

M-step Update by

$$p_j^{(new)} = \frac{\sum_{i=1}^{n} w_{ij}}{n}$$

$$\mu_j^{(new)} = \frac{\sum_{i=1}^{n} w_{ij}\mathbf{x}_i}{\sum_{i=1}^{n} w_{ij}}$$

$$\mathbf{\Sigma}_j^{(new)} = \frac{\sum_{i=1}^{n} w_{ij}\left(\mathbf{x}_i - \mu_j^{(new)}\right)\left(\mathbf{x}_i - \mu_j^{(new)}\right)^T}{\sum_{i=1}^{n} w_{ij}}$$

If convergence stop otherwise go back to the E-step

Different covariance matrices

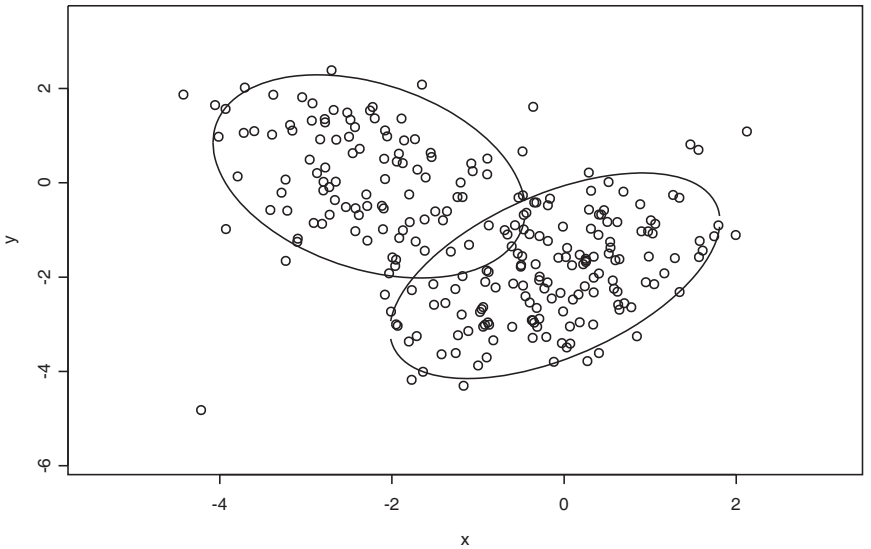
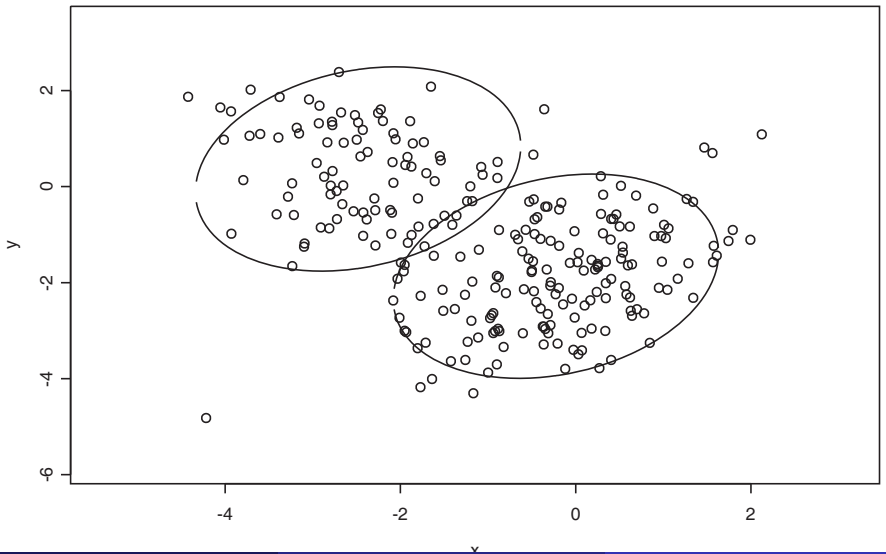


Figure: Ellipsoids are 95% confidence

Same covariance matrices



Diff Covariance	Equal Covariances
$p = (0.5865899, 0.4134101)$	$p = (0.6572255, 0.3427745)$
$\mathbf{\Sigma}_1 = \begin{bmatrix} 0.9568578 & 0.5763747 \\ 0.5763747 & 1.2372805 \end{bmatrix}$	$\mathbf{\Sigma} = \begin{bmatrix} 0.8974587 & 0.2312592 \\ 0.2312592 & 1.1782524 \end{bmatrix}$
$\mathbf{\Sigma}_2 = \begin{bmatrix} 0.8568740 & -0.3376073 \\ -0.3376073 & 1.2079469 \end{bmatrix}$	
$\mu_1 = (-0.1073719, -1.9717564)$	$\mu_1 = (-0.2259167, -1.8661791)$
$\mu_2 = (-2.2650514, 0.1351531)$	$\mu_2 = (-2.4823908, 0.3668937)$
loglikelihood $= -832.0454$	loglikelihood $= -846.2068$

Table: The fitted groups

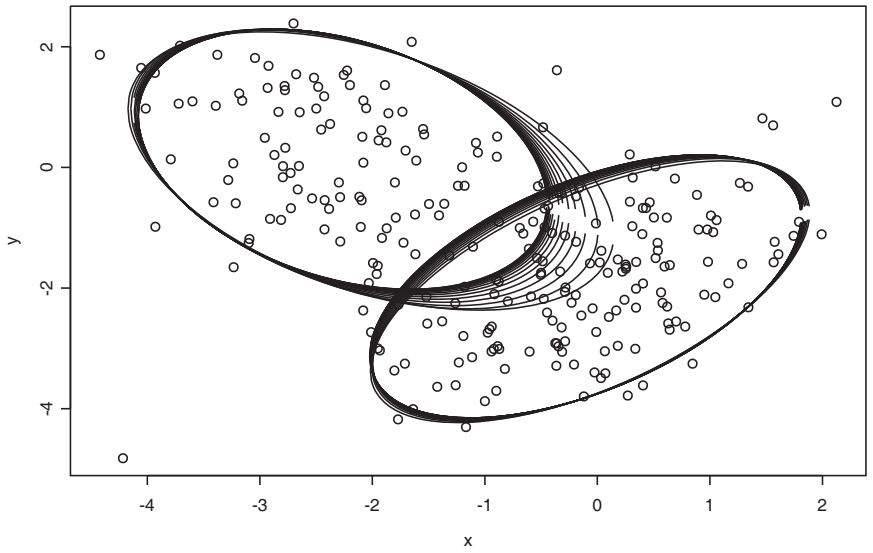


Figure: 95% ellipsoids

About EM

Advantages

- Monotonic convergence
- Parameters on the admissible range
- Easily programmable
- Byproducts of the algorithm useful for many reasons

Disadvantages

- Result depends on initial values
- Local maxima
- Slow convergence

Classification EM (CEM)

CEM algorithm is a variant. After the E-step and based on the w_{ij}, we assign each observation to the group with higher w_{ij} and then set $W_{ij} = 1$ for this and 0 otherwise. M-step remains the same. For common diagonal covariance matrix this is known as the K-means algorithm

Inference

- Selection of the number of components
- Goodness of fit
- Comparison between models

can now be based on likelihood arguments

Number of componenets/clusters

Select k to minimize

$$\begin{aligned} AIC(k) &= -2L(k) + 2d_k \\ AIC3(k) &= -2L(k) + 3d_k \\ BIC(k) &= -2L(k) + d_k log(n) \end{aligned}$$

k is the number of components, $L(k)$ maximized log-likelihood with k groups and d_k is the number of parameters for a model with k groups.

Other criteria

Normalized Entropy Criterion

$$NEC(k) = \frac{I(k)}{L(k) - L(1)}, k > 1.$$

where

$$I(k) = -\sum_{j=1}^{k}\sum_{i=1}^{n} w_{ij} ln w_{ij}$$

using $0ln0 = 0$. Note that w_{ij}'s are available from the E-step.

Factor Analyzers

As in Factor analysis we may represent

$$\Sigma = \Lambda'\Lambda + \Psi$$

where Σ is the $p \times p$ covariance matrix, Λ is a $k \times p$ matrix with $k << p$ and Ψ is a $p \times p$ matrix.

- The central idea is that the above representation allows to estimate not a $p \times p$ matrix but mainly the Λ which has much fewer elements.
- The central idea of this (as in factor analysis) is that variables share common elements so few factors can determine the structure
- We need to select the appropriate k

Recent development

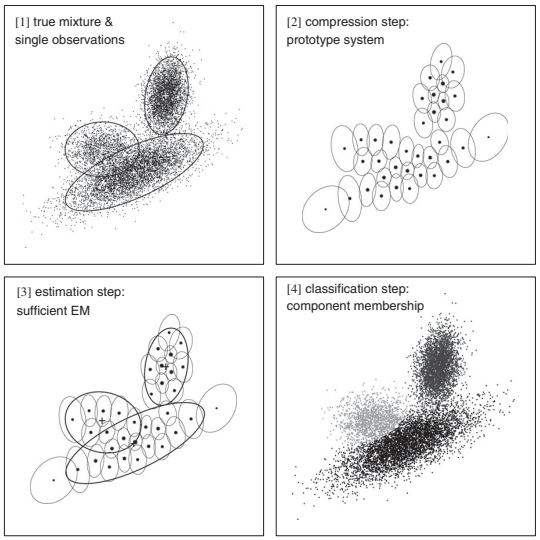
- Number of mixture components is not necessarily the number of clusters
- Example: Two normal ones can create a skew cluster
- Mixtures based on skewed densities to allow for non elliptical clusters. Several choices

Clustering large data sets

For big datasets some methods fail like the hierarchical methods or can be very hard for computing purposes. There are two main directions/approach

- Start with all data for few steps. Create a series of prototypes and then work with them. Finally classify the initial observations.
- Select one or more sub-samples and work with them

Prototypes



Sub-sampling

- How to select? Purely randomly or via supervised methods?
- How many, how large the subsamples should be?

Click Stream Data

- The page visits of users who visited msnbc.com on 28th September 1999.
- Data on 989818 users available at
- The data are recorded at the level of URL category and describe the hits in order for each user.
- Typical observations: `user A: 6 9 4 4 4 10 3 10 5 10 4 4 4`
  `user B: 1 2 1 14 14 14 14 14 14 14 14 14 14 14 1 2 2 ...`
- URL categories:

1: frontpage	6: on-air	10: living	14: bulletin board service
2: news	7: misc	11: business	15: travel
3: tech	8: weather	12: sports	16: msn-news
4: local	9: health	13: summary	17: msn-sports
5: opinion			

Average number of visits per user: 5.7 Number of URLs per category: 10 to 5000

Clustering of click stream data

- Special data structure
- Order is important
- What distance for distance based methods?
- Huge dataset
- How to model based clustering?

Distance

- **Levenshtein distance** is a string metric for measuring the difference between two sequences. Informally, the Levenshtein distance between two words is the minimum number of single-character edits (insertion, deletion, substitution) required to change one word into the other.
- Also known as a Sequence Alignment method (SAM) in biology when working with RNA sequences.
- Hard to compute, unbounded, usually a normalized version is used. Available in R

Example

For example, the Levenshtein distance between "kitten" and "sitting" is 3, since the following three edits change one into the other, and there is no way to do it with fewer than three edits:

- kitten $\rightarrow$ sitten (substitution of "s" for "k")
- sitten $\rightarrow$ sittin (substitution of "i" for "e")
- sittin $\rightarrow$ sitting (insertion of "g" at the end).

In practice use a normalized version

Distance based

- Center observation Sequence with just 1 (57552 observations visited only page 1)
- Obtain distance around this
- Select a subsample of 10000 observations using supervised selection
- Run hierarchical o the subsample and then just assing the rest observations

Model Based approach

- Mixtures of Markov chains of order 1
- We model the probability to move from site i to site j via a transition matrix.
- Assume that each group has its own transition matrix, create a finite mixture and work with this.
- To cope with huge data one may use variants of the EM for mixtures or subsampling

Example

For example the first group may have transition matrix (we use 3 sites)

$$\mathbf{P}_1 = \begin{bmatrix} 0.2 & 0.4 & 0.4 \\ 0.33 & 0.33 & 0.34 \\ 0 & 0.9 & 0.1 \end{bmatrix}$$

and the second

$$\mathbf{P}_2 = \begin{bmatrix} 0 & 0.5 & 0.5 \\ 0.9 & 0.1 & 0 \\ 0.1 & 0 & 0.9 \end{bmatrix}$$

In real circumstance we need to estimate the $\mathbf{P}_j$'s

msn data- resulted clusters

Figure 4: The frequency of appearance of each URL category in the corresponding cluster relative to its frequency in the whole data set. The size of each cluster and the minimum, average and maximum length of the sequences within each cluster are shown on the title and the subtitle of the corresponding plot.

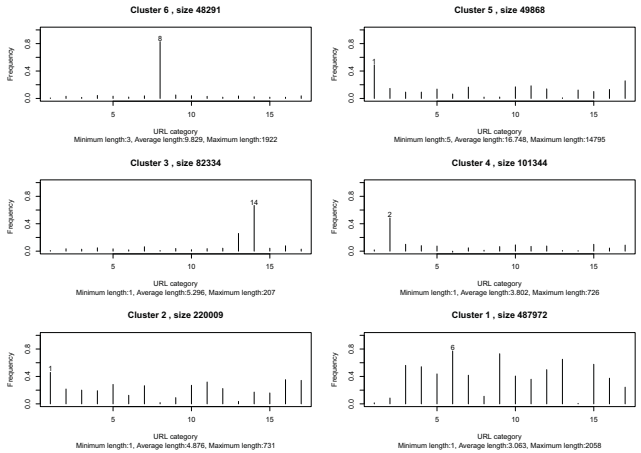


Image Segmentation

- Each pixel of the photo is characterized by a 3-dimensional RGB vector
- We want to find clusters of colors
- Example: fMRI pictures, different colors may characterize tumors
- Typically, even for small resolution a picture can have a large number of pixels. E.g. a 200 $\times$ 200 pixel (which is a very small analysis) leads to 40000 data points.

Paulia flower

RGB image of a St. Paulia flower. There are 40,656 data pixels after background removal.

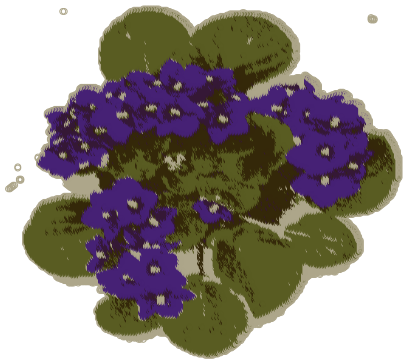
Clustering

- Use of model based clustering
- Assume finite mixture of trivariate Gaussian distributions
- Use of `mclust` package

Subsample 2000 pixels - 10 clusters

Subsample 5000 pixels - 14 clusters

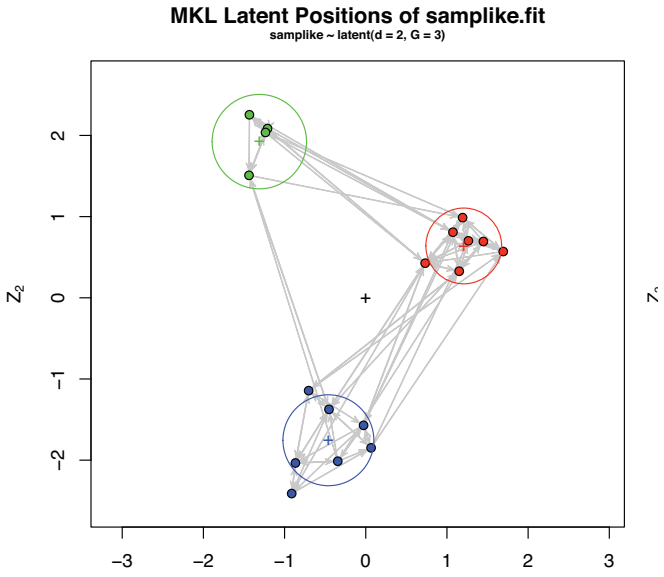
Subsample 500 pixels - 5 clusters



Clustering Social Networks

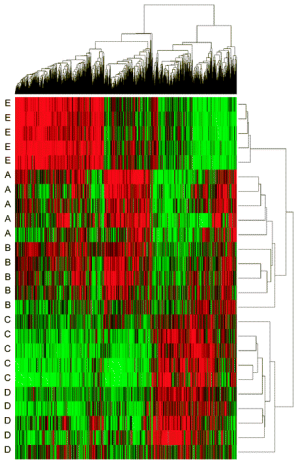
We consider the social relations between 18 monks in an isolated American monastery. The data are sociometric information by using interviews, experiments and observation. Here we focus on the social relation of liking. We say that a monk has the social relation of like to another monk if he ranked that monk in the top three monks for positive affect in any of three interviews given over a 12-month period.

Clustering Social Networks



Biclustering

- Usual clustering algorithms are based on similarities of rows or columns of data, i.e. cluster observations or (less often) variables.
- In some examples we want to be able to cluster observations AND variables at the same time
- For example in microarray data, we want to cluster both genes and patients
- This is called Biclustering (or two-way clustering)
- Goal of biclustering: identify homogeneous submatrices.
- Difficulties: computational complexity, assessing the statistical significance



PART - II
Classification

Classification

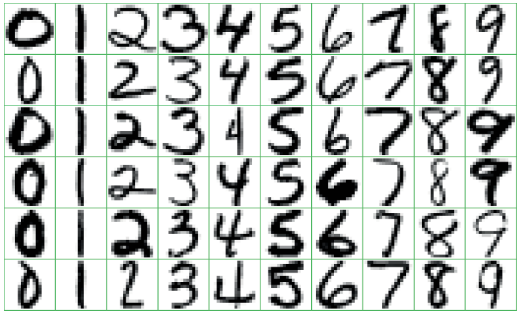
Classification is the problem of identifying to which of a set of categories (sub-populations) a new observation belongs, on the basis of a training set of data containing observations whose category membership is known.

In most applications we have available data on several covariates and we need to use this information to classify

Examples

- Credit scoring
- Medical Diagnosis
- Insurance
- Indecisive persons in political gallops

Handwritten digits



How we can recognize a new "number".

Some approaches to classification problem

1. Discriminant analysis (linear and nonlinear)
2. Linear regression
3. k-nearest neighbors
4. Logistic regression
5. Classification and regression trees)
6. Neural networks
7. Support vector machines
8. Kernels
9. Machine learning algorithms

and variants and/or combinations of the above. Keep in mind that they can lead to different classifications approached (e.g. hard versus soft)

Classification and Clustering

- Clustering is also known as **unsupervised learning** in the sense that the cluster membership is unknown and need to be inferred from the data
- Classification is also known as **supervised learning** in the sense that at least for some data we know the exact class they belong

Of course it is possible to have data where the membership is partially known: we know for some observations but not for others. Then the problem is somewhat between.

Issues to be checked

- Which variables are really useful?
- Which method?
- How to check the goodness?

Warning ...

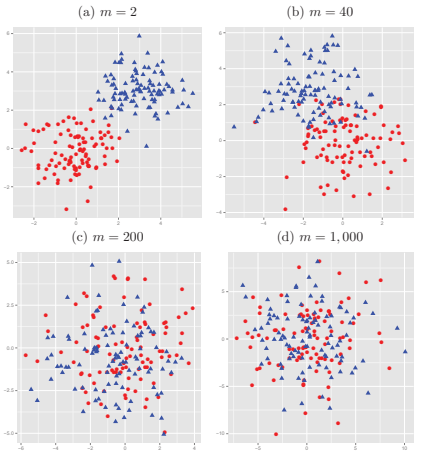


Figure 1: Scatter plots of projections of the observed data ($n = 100$ from each class) onto the first two principal components of the best m-dimensional selected feature space. A projected data with "•" indicates the first class and "▲" indicates the second class.

Hard versus Soft classification

- In Hard classification we assign each observation to a class, no uncertainty around
- In soft classification we have also available the probabilities of belonging to each class and hence we can handle the uncertainty (overlapping). This implies the usage of some probability model to handle uncertainty.

Discriminant analysis

Traditional statistical method for creating classification rules.

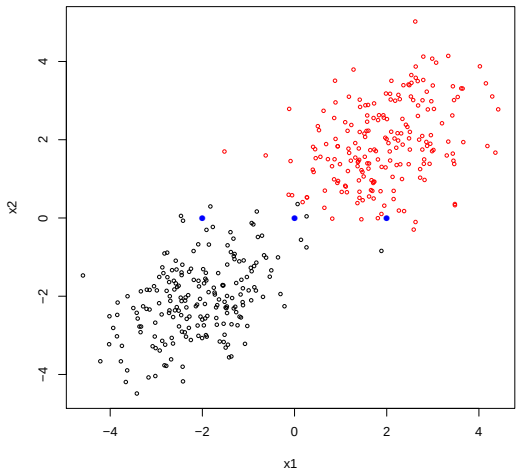
The key idea is the following:

- Let assume a binary classification problem.
- Estimate from the data from the first category a likelihood (density).
- Do the same for the second one.
- When a new observation comes just calculate which out of the two densities has larger value.
- Assign to this category

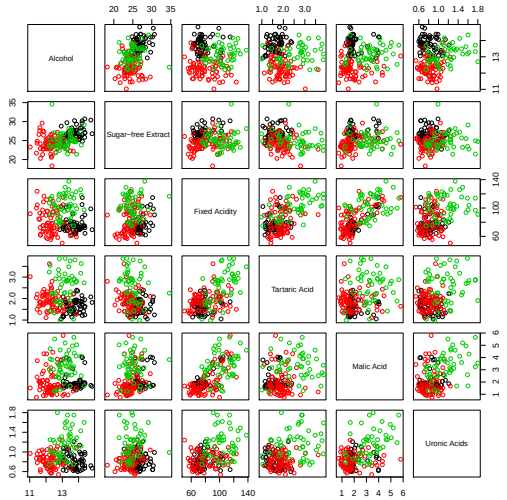
Having a statistical model we can check things like probability of misclassification, uncertainty around classification etc

Illustration

We need to classify the blue points.



Wine data



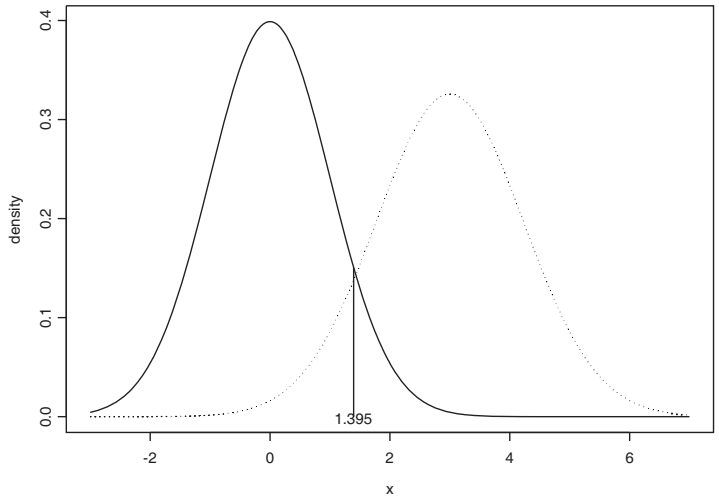
Discriminant analysis - example

Assume two groups. Assume density $f_1(x)$ and $f_2(x)$, x is a vector of covariates (information).
The classification rule is
For a new observation x_n Assign to the first group if

$$\frac{f_1(x_n)}{f_2(x_n)} \geq 1$$

and at the second otherwise.
for more than 2 groups, assign where $f_j(x_n)$ is maximum.
This is known as the maximum likelihood rule

Graphical representation

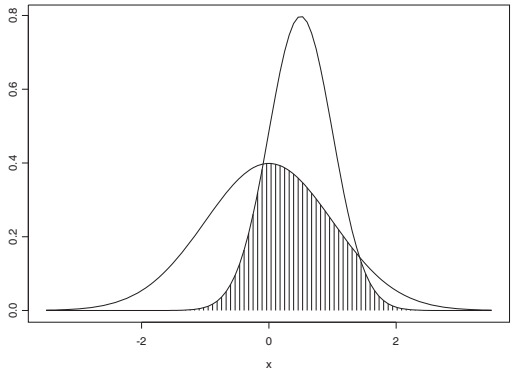


Cautions

- We need to know f. in practice this will be estimated from the available data, either parametrically (we assume a distribution and estimates its parameters) or non-parametrically (e.g with kernels).
- Since we estimate, we have uncertainty and errors. Need to be taken into account.
- The rule is sharp since it assigned to one of the groups, but in fact we can get probabilities for each group and use a soft classification

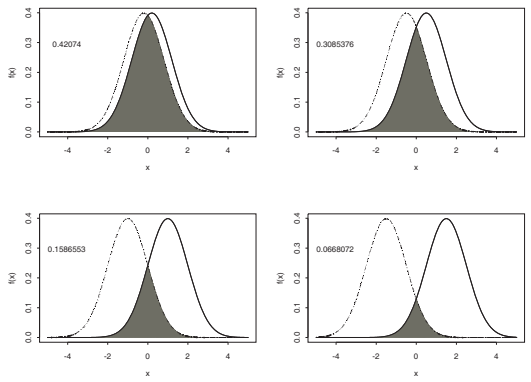
Misclassification

The probability of misclassification is the shadowed area. (Not weighted by prior evidence)

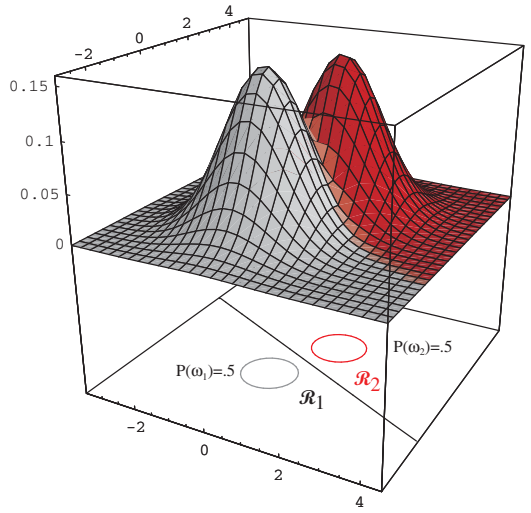


Misclassification

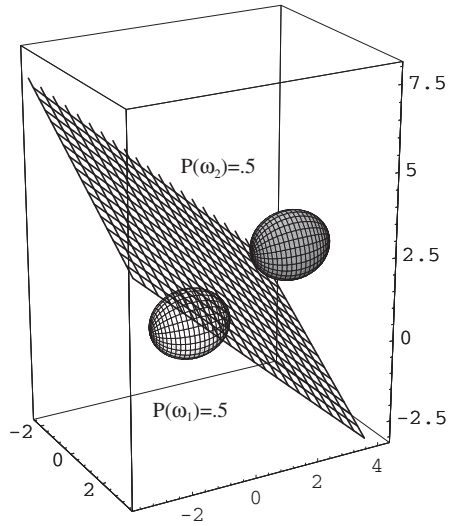
Overlapping increases the error



Larger Dimension



Larger Dimension



Bayes Rule

for each group use prior information π_j.
Classify the i-th observation to the group with higher posterior probability

$$w_{ij} = \frac{\pi_j f_j(x)}{\sum_j \pi_j f_j(x)}$$

This quantity has been also seen in model based clustering! How we estimate the prior probabilities π_j?

Discriminant analysis - in practice

- Most of the existing theory is based on assuming multivariate normal subpopulations.
- This leads to a linear classification rule (i.e. categories are split by a straight line) or a quadratic rule (the line is quadratic).
- Fisher showed that this can be seen as calculating some score for each subpopulation via a linear function and then classify to the category with larger score!
- It allows for simple interpretation but linearity is usual a large restriction.

Fisher Discriminant analysis

- Fisher worked the idea to create linear scores from the data, and use the scores to classify the new observations.
- For k groups we need $k-1$ such functions.
- The idea is to create the score (discriminant function) such as it can better discriminate between groups, i.e. the coefficients of the linear functions are derived so as to give largest difference between the groups!
- The second one is uncorrelated to the first one and discriminated toward different direction.
- This is shown to be equivalent to the LDA when multivariate normal with equal variance matrices are assumed.

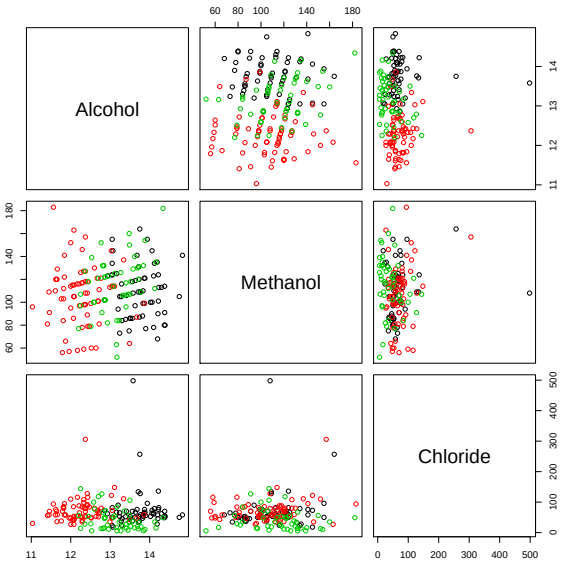
Quadratic Discriminant Analysis

- Generalized the LDA
- We do not assume equal covariance matrices. (but still assume multivariate normality)
- Now each group can have its own covariance 9need to be estimated)
- The rule is not any more linear but quadratic

Example: Wine Data

- We want to create a rule to be able to classify a new wine to one of he three categories.
- Available 27 variables, for illustration we use only three Alcohol, Methanol and Chloride.
- Apply Fisher's discriminant anlayis (i.e. LDA)

Data



Group means

Type	Alcohol	Methanol	Chloride
1	13.744	109.3220	71.711
2	12.278	106.5352	71.690
3	13.153	115.5417	41.270

Coefficients of linear discriminants

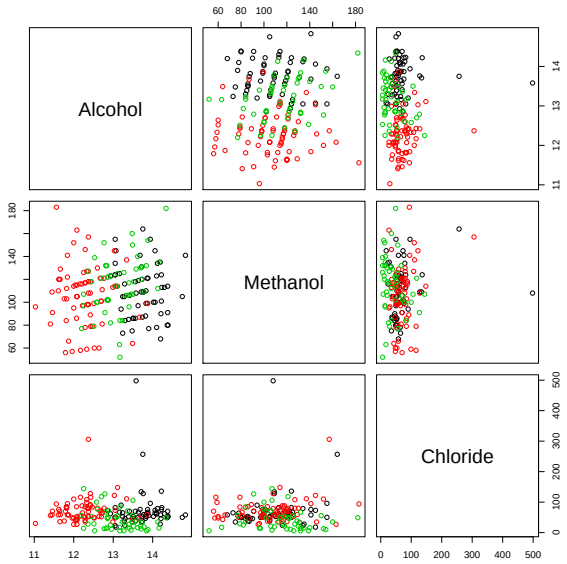
Variable	LD1	LD2
Alcohol	-1.952898e+00	-0.11723877
Methanol	7.489453e-05	0.02101704
Chloride	3.118705e-04	-0.01896483

Classification

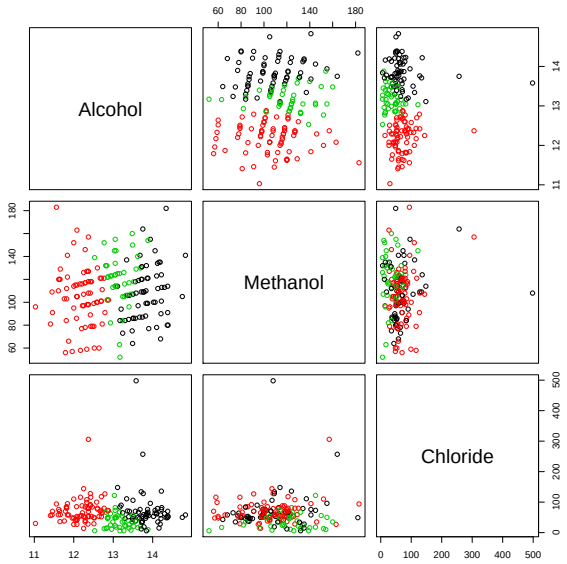
		Reality		
		1	2	3
Class	1	48	5	10
	2	1	62	11
	3	10	4	27

In total 137 (out of 178) observations were correctly classified. Accuracy = 0.769.
But this is "in-sample".

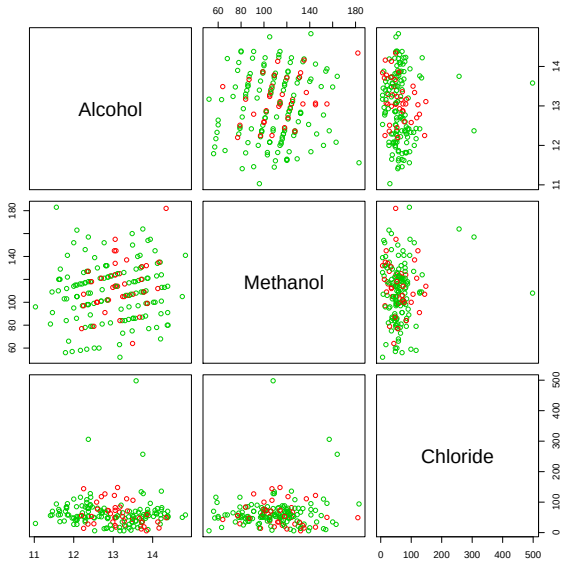
Data



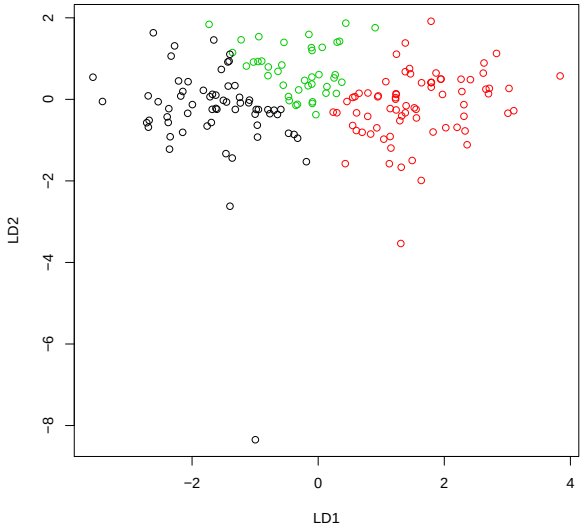
Fit



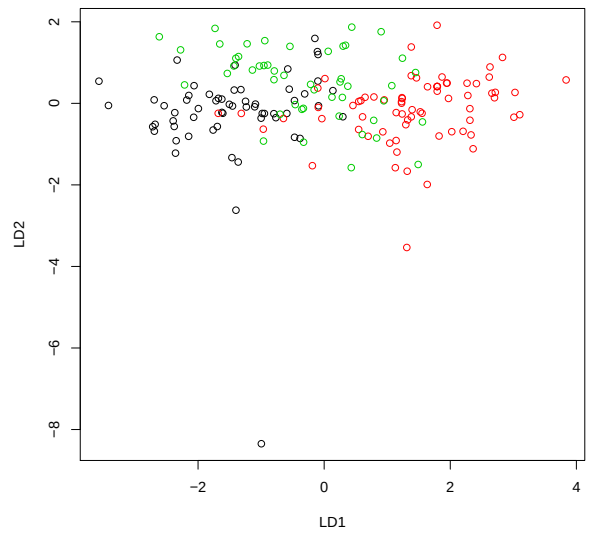
Correct classification = green



Discriminant functions - Fitted



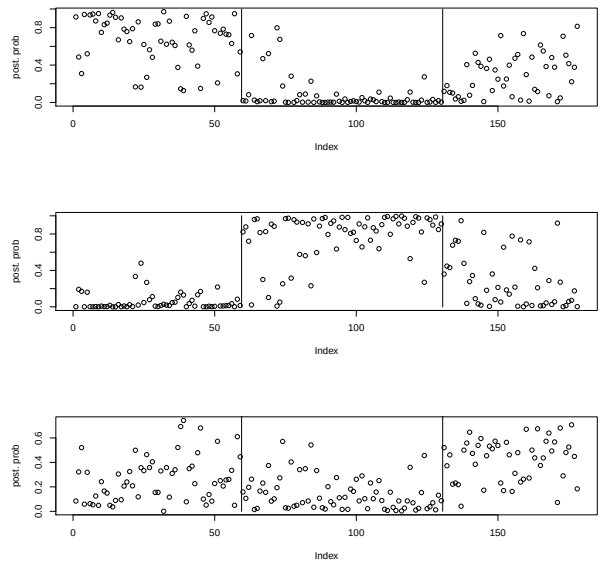
Discriminant functions - color=Truth



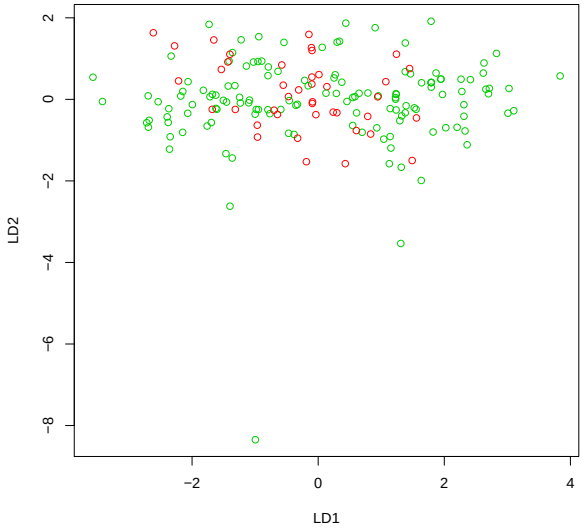
Posterior Probabilities - Soft classification

	Post. Probab for Group			
Obs	1	2	3	Type
1	0.91428	0.00119	0.08452	1
2	0.48594	0.19187	0.32219	1
60	0.01877	0.82279	0.15844	2
61	0.01593	0.87790	0.10618	2
165	0.61422	0.01069	0.37508	3
170	0.37720	0.05583	0.56697	3
77	0.28098	0.31548	0.40354	2

Posterior Probabilities



Discriminant functions - green= Correct classification



Variable Selection in LDA

Selection of variables is important. Usage of too many variables can lead to very bad classifications.
All we need is:

- a criterion to be able to "identify" better models: e.g. Wilks Λ
- an algorithm to find the best subset, e.g. forward algorithm

Simple Linear Regression

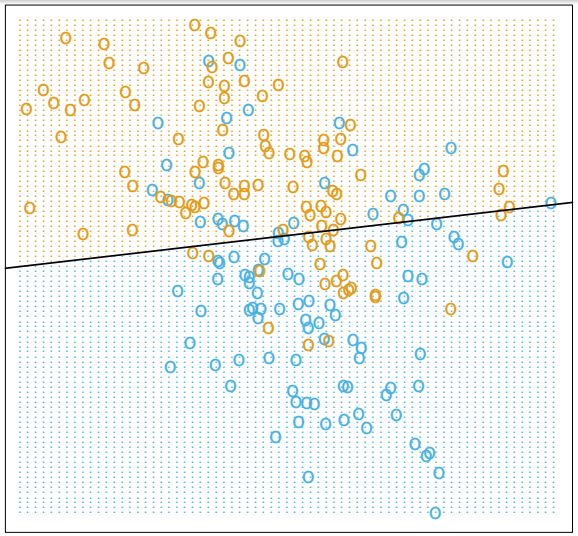
Consider a two-categories classification problem. Let Y_i be the target variable taking values only 0 and 1 (subscript denotes observation) Let also $\mathbf{x}_i$ a vector with characteristics of the i-th observation. We fit a linear regression model

$$Y = \mathbf{x}_i\beta$$

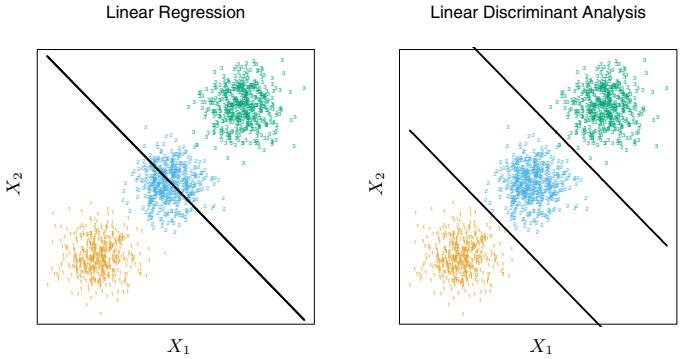
and we estimate by $\hat{Y}_i = \mathbf{x}_i\hat{\beta}$.
The classification rule will be:
if $\hat{Y}_i > 0.5$ we classify in the first population (corresponding to 0) otherwise to the second (corresponding to 1).

- We do not care if some $\hat{Y}_i$ lies outside the $(0, 1)$ interval
- We split by drawing a linear line
- Very simplistic but in practice can work well.
- does not coincide with Linear Discriminant analysis (due to different objective functions) but usually they are close)
- It may fail for special structures.

Example



Linear Regression vs LDA



Multivariate Regression

The previous idea can be generalize to more than two categories
When $K > 2$ groups

- Define K dummies and use them as responses.
- Fit K simple linear regressions and obtain predicted values
- Assign to the class with larger predicted value.

Logistic Regression

Let say that we have n individuals. The group of the i-th individual is denoted as C_i, $i = 1, \ldots, n$.
Assume that

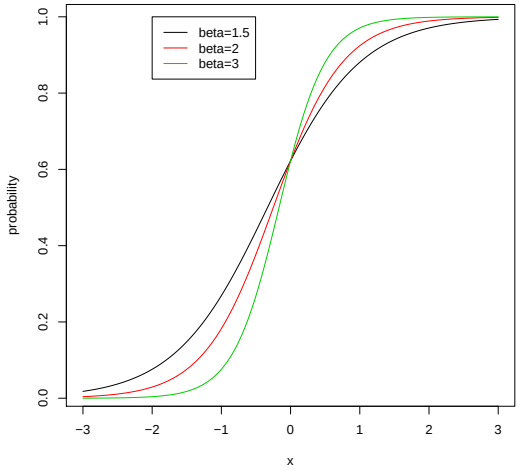
$$p_i = P(C_i = 1) = \frac{exp(\beta x_i)}{1 + exp(\beta x_i)}$$

where x_i is a vector with covariates related to the i-th individual and β a vector of regression coefficients.
Note that the function selected makes sure that the probability will be in the unit interval, i.e. $[0, 1]$

Example

One covariate x, and

$$p_i = P(C_i = 1) = \frac{exp(0.5 + \beta x)}{1 + exp(0.5 + \beta x)}$$



Some more details

- Is the above functional form the only available?
- How can we check things like goodness of fit, significance of the parameters etc as we did in standard regression?
- What kind of interpretation can we give?
- Is there some theory behind all these?

Variants

- Probit model
- t-link model
- multinomial logistic
- A lot of discrete choice models

Multinomial logit model

Suppose that we have N groups, i.e. $\mathcal{C} = \{1, 2, \ldots, N\}$
We assume that

$$P(C_i = k) = \frac{exp\left(\beta_k x_i\right)}{1 + \sum_{k=1}^{N-1} exp\left(\beta_k x_i\right)}, \; k = 1, \ldots, N-1$$

Notes:

- We need $N-1$ since they shall to 1.
- We have one vector of coefficients per group

Multinomial logit model

- It holds that

$$P(C_i = N) = \frac{1}{1 + \sum_{k=1}^{N-1} exp\left(\beta_k x_i\right)}, \; k = 1, \ldots, N-1$$

- It is also true that

$$\log\left(\frac{P(C_i = k)}{P(C_i = N)}\right) = \beta_k x_i$$

i.e. all the probabilities are expressed via the "baseline" one (which is the N-th in my notation (IIA assumption)

Nearest Neighbors (K-nn)

The basic idea is that if we want to predict the value of a new observation we look for observations similar to this one and we use as predictor their values (or a function of them).
Formally for a new observation with characteristics x we use as prediction

$$\hat{y} = \frac{1}{k} \sum_{x_i \in N_k(x)} y_i$$

where $N_k(x)$ is the set with the k nearest neighbors to x. The above function implies that we get the k nearest observation and use their average (or mode) as prediction

Nearest Neighbors (K-nn)

- Method is non-parametric
- For classification problems we may use the mode
- Searching for the nearest neighbors can be time consuming. Special algorithms exist
- Choice of k: in practice we use a range of values and select the k based on some criterion
- The classification rule is very flexible and non-linear

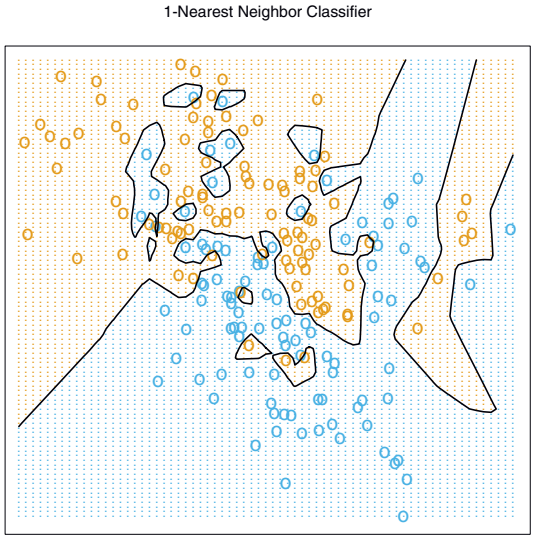


Figure: k-nn with $k = 1$ for a toy-example

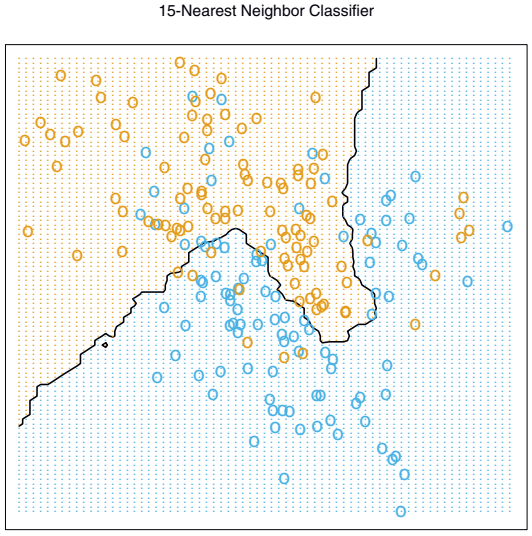


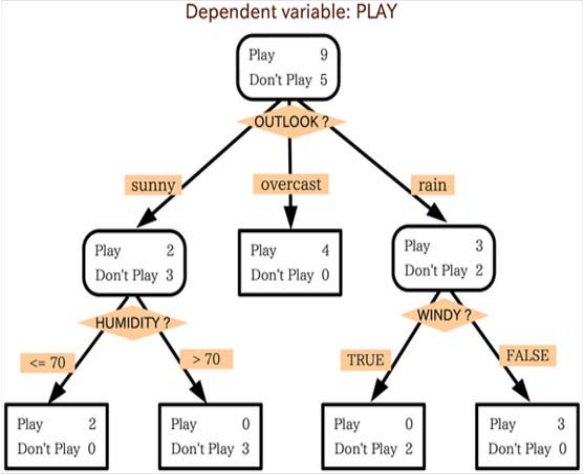
Figure: k-nn with $k = 15$

Decision Trees

There are two broad categories

- Regression Trees: the aims is to predict a value for the attribute under consideration, typically a continuous one
- Classification trees: the aim is to classify the observation to a given number of categories

Example - Tennis data



Decision Trees

Strengths:

- Can cope with missing data;
- Generates easy to understand rules
- Non-parametric and flexible.

Weaknesses:

- Requires large sample sizes
- Can become over-complex
- Tendency to over-fit the data.

Basic ingredients

Decision tree is a classifier in the form of a tree structure

- Decision node: specifies a test on a single attribute
- Leaf node: indicates the value of the target attribute
- Arc/edge: split of one attribute
- Path: a disjunction of test to make the final decision

Decision trees classify instances or examples by starting at the root of the tree and moving through it until a leaf node.

Important characteristics

- Predictors must be discrete if not we must discretize them. E.g. age can be put in categories 18-45, 45-64, above 65
- Which variables shall be used first (or in which order)?
- When to stop?
- The tree can grow huge and become hard to understand. Larger trees are typically less accurate than smaller trees.
- We need to prune the tree to become parsimonious.

Attribute selection

- Selection of an attribute to test at each node - choosing the most useful attribute for classifying examples.
- Information gain measures how well a given attribute separates the training examples according to their target classification
- This measure is used to select among the candidate attributes at each step while growing the tree

Decision Trees

- Decision trees are powerful and popular tools for classification and prediction.
- Decision trees represent rules, which can be understood by humans and used in knowledge system such as database.
- Decision trees can handle continuous and categorical variables
- Decision trees can provide a clear indication of which fields are most important for prediction or classification

Decision Trees

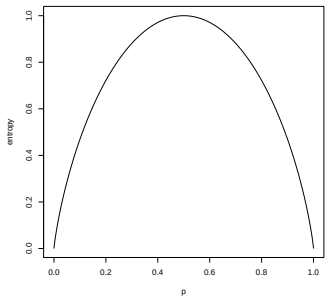
- Not suitable for prediction of continuous attribute.
- Perform poorly with many class and small data.
- Computationally expensive to train.
- Do not treat well non-rectangular regions.

Decision Trees

Let a characteristic T that can take two values say A and B. Let p_A and p_B be the empirical proportions. Then we may define the entropy as

$$H(T) = H(p_A, p_B) = p_A \log_2 p_A + p_B \log_2 p_B$$

Obviously the entropy is maximized if $p_A = p_B = 1/2$ and it is zero if $p_A = 0$ or 1.



Information Gain

Information gain: In general terms, the expected information gain is the change in information entropy from a prior state to a state that takes some information as given namely as:

$$IG(T,\alpha) = H(T) - H(T|\alpha)$$

namely conditional on a characteristic α

$$IG(T,\alpha) = H(T) - \sum_{u \in \mathcal{A}} H(T|\alpha = u)$$

where $\mathcal{A}$ is the set of values for the characteristic α
Information gain measures how well a given attribute separates the training examples according to their target classification
This measure is used to select among the candidate attributes at each step while growing the tree

Example

Let say that we care about a characteristic T taking values A and B and we have two candidate variables say X and Y being binary. The two contigency tables are

	X_1	X_2	*Total*
A	6	4	10
B	6	4	10
Total	12	8	

and

	Y_1	Y_2	*Total*
A	10	0	10
B	2	8	10
Total	12	8	

Is X or Y better to predict T?

Example

We calculate the entropy for T as

$$H(T) = -0.5 \log_2 0.5 - 0.5 \log_2 0.5 = 1$$

For X we have that $H(T|X = X_1) = 1$ and $H(T|X = X_2) = 1$ (since they have equal proportions) and hence the information gain is

$$IG(T,X) = 1 - \left[\frac{12}{20}1 + \frac{8}{20}1\right] = 0$$

Similarly for Y we get

$$IG(T,Y) = 1 - \left[\frac{12}{20}0.65 + \frac{8}{20}0\right] = 0.35$$

Hence attribute Y provides more information for the characteristic T. for a decision tree it shall be used firstly

Decision Tree - Algorithm

So, for a decision tree

- We grow the tree by selecting every time the variable that has the larger information gain and we create a new node for this variable until
  - either no variable is still remaining
  - or the training examples associated with this leaf node all have the same target attribute value (i.e., their entropy is zero).
- Alternatively we may stop when new nodes are not statistically significant.
- Clearly for large number of variables this can be very time consuming.
- In most real applications we *prune* the tree by cutting nodes that improve only slightly the performance to end up with a more parsimonious tree

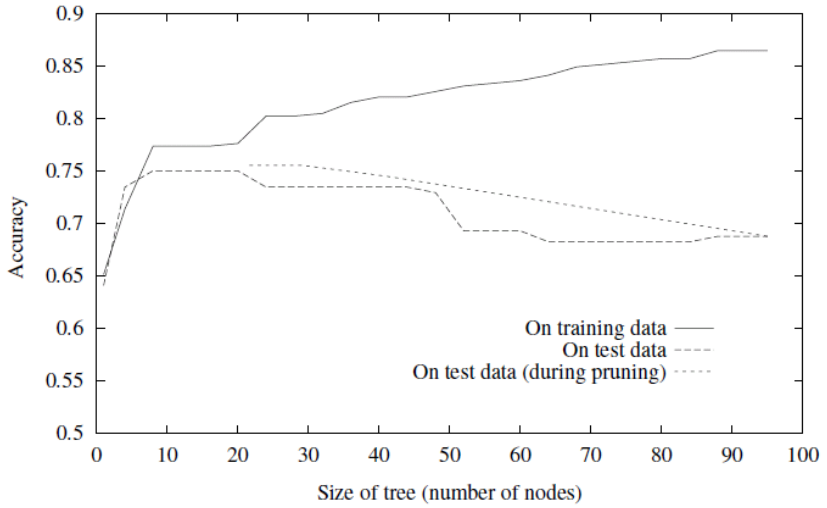
Pruning

Split data into training and validation set

- Repeat until further pruning is harmful:
  1. Evaluate impact on validation set of pruning each possible node (plus those below it)
  2. Greedily remove the one that most improves validation set accuracy
- Produces smallest version of most accurate subtree

Pruning helps improving wrt over-fitting

Overfitting



Decision Tree - Other Considerations

- Continuous variables need to be discretize, usually become binary. there are several procedures to do so.
- Either for categorical variable we may need to make them binary in order not to create a large tree.
- Cross validation is used to find the performance of the tree and for pruning purposes.
- DT split the instance space in rectangular areas (hence not appropriate for non-rectangular rules)

Random forest

Instead of building one decision tree,

- Built several of them to create a forest
- Take one predictions from each tree and
- Combine them all together to create a generic prediction

Kernel based methods

- Estimate the underlying densities in a non-parametric way, i.e. not assuming a specific parametric form (e.g. multivariate normal).
- Great flexibility: the separating line/hyperplane is not of any parametric form but fully flexible.
- In one dimension define

$$\hat{f}(x) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{h}\,K\left(\frac{x-x_i}{h}\right)$$

where $K(x)$ is a pdf with $K(x) \geq 0$, $\int_{-\infty}^{+\infty} K(x)dx = 1$, $\int_{-\infty}^{+\infty} xK(x)dx = 0$ $0 < \int_{-\infty}^{+\infty} x^2K(x)dx < \infty$, and h is the bandwidth.

Kernel based methods - many dimensions

The joint pdf is estimated by

$$\hat{f}(x_1, x_2, ..., x_d) = \hat{f}(\mathbf{x}) = \frac{1}{n\,|H|}\sum_{i=1}^{n} K_d\left[\mathbf{H^{-1}}(\mathbf{x}-\mathbf{x}_i)\right],$$

where $\mathbf{x}_i$ are the vector valued observations and $\mathbf{H}$ is the bandwidth matrix.

Choose $\mathbf{H}$

- $\mathbf{H} = \begin{bmatrix} h & 0 & ... & 0 \\ 0 & h & ... & 0 \\ ... & ... & ... & ... \\ 0 & 0 & ... & h \end{bmatrix}$, diagonal, smooth the same to all dimensions.
- $\mathbf{H} = \begin{bmatrix} h_1 & 0 & ... & 0 \\ 0 & h_2 & ... & 0 \\ ... & ... & ... & ... \\ 0 & 0 & ... & h_d \end{bmatrix}$, smooth separately each dimension $|H| = h_1h_2...h_d$.
- $\mathbf{H} = \begin{bmatrix} h_{11} & h_{21} & ... & h_{d1} \\ h_{21} & h_2 & ... & h_{d2} \\ ... & ... & ... & ... \\ h_{d1} & h_{d2} & ... & h_{dd} \end{bmatrix}$ take into account all dimensions

What kernels

- Product of independent kernels.
- Some multivariate ones like Epaneshnikov

$$K_d(\mathbf{x}) = \begin{cases} \left[\frac{d(d+2)}{4}\right]\Gamma(d/2)\pi^{-d/2}(1-\mathbf{x}'\mathbf{x}) & \mathbf{x}'\mathbf{x} \leq 1 \\ 0 & \text{otherwise} \end{cases}$$

- Multivariate Gaussian.

Support Vector Machines (SVM)

- Supervised learning models with associated learning algorithms that analyze data and recognize patterns, used for binary classification and regression analysis.
- The simple idea is that we may separate the two classes by hyperplane.
- A special property is that they simultaneously minimize the empirical classification error and maximize the geometric margin; hence they are also known as maximum margin classifiers.
- It can generalize to nonlinear SVMs.
- For more than two classes: the dominant approach for doing so is to reduce the single multiclass problem into multiple binary classification problems.

Support Vector Machines -Example

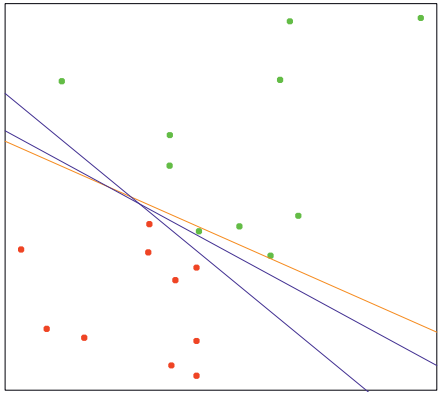


FIGURE 4.14. *A toy example with two classes separable by a hyperplane. The orange line is the least squares solution, which misclassifies one of the training points. Also shown are two blue separating hyperplanes found by the* perceptron learning algorithm *with different random starts.*

Support Vector Machines -Example

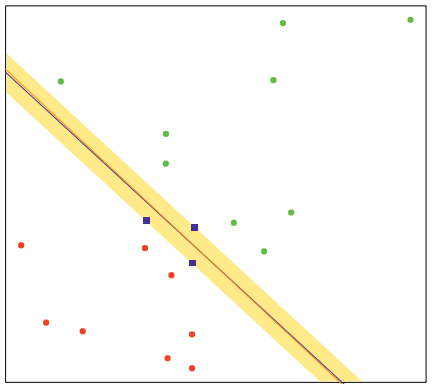
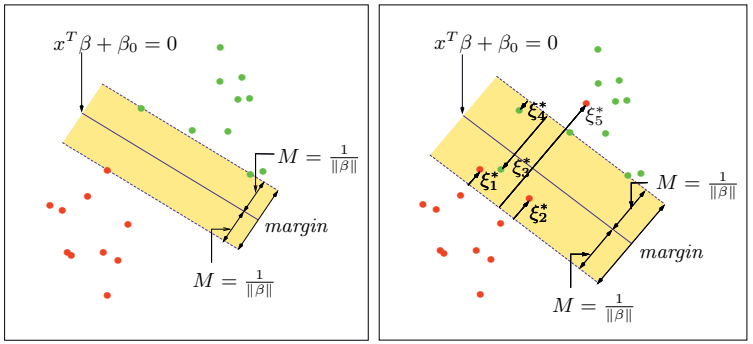


FIGURE 4.16. *The same data as in Figure 4.14. The shaded region delineates the maximum margin separating the two classes. There are three support points indicated, which lie on the boundary of the margin, and the optimal separating hyperplane (blue line) bisects the slab. Included in the figure is the boundary found using logistic regression (red line), which is very close to the optimal separating hyperplane (see Section 12.3.3).*

Support Vector Machines - Illustration



Support Vector Machines

Our data consists of N pairs $(x_1, y_1), \ldots, (x_N, y_N)$, $x_i \in R^p$ and $y_i \in \{-1, 1\}$. Define a hyperplane by

$$\{x : f(x) = x'\beta + \beta_0 = 0\}$$

where β is a unit vector. A classification rule can be derived as

$$sign[x'\beta + \beta_0]$$

The problem to be solved is maximize M with respect to β_0 and β (but $||\beta|| = 1$ subject to

$$y_i(x_i'\beta + \beta_0) \geq M, \quad i = 1, \ldots, N$$

M is the margin between the hyperplane and the closest observation

Support Vector Machines

If case is not separable then the constraint becomes

$$y_i(x_i'\beta + \beta_0) \geq M - \xi_i, \quad i = 1, \ldots, N$$

$$xi_i \geq 0, \sum \xi \leq c$$

for some constant c.
In fact ξ_i refers to how much inside the other class we allow to enter.
Constant c is specified by the user depending on hiw much error is allowed.

Boosting

Idea: Combine several classifiers to create a better one

- Use weights, weighting more good classifiers
- Remove bad classifiers

An example: diabetes data

768 women from the Pima tribe.

- Number of times pregnant (PREGN)
- Plasma glucose concentration a 2 hours in an oral glucose tolerance test (PLASMA)
- Diastolic blood pressure (mm Hg) (BLP)
- Triceps skin fold thickness (mm) (TRICEP)
- 2-Hour serum insulin (mu U/ml)(INSUL)
- Body mass index (weight in kg/(height in m)^2) (BMI)
- Diabetes pedigree function (PEDIG)
- Age (years) (AGE)
- Class variable (0 no diabetes or 1 diabetes)

We want to classify the data

An example: diabetes data - RESULTS

Predictive ability based on 20-fold cross-validation

Method	All variables	Remove BLP, TRICEP
LDA	0.753	0.744
QDA	0.734	0.730
Log Regr	0.751	0.751
Knn 4	0.772	0.795
Knn 5	0.772	0.788
Knn 6	0.744	0.772
Tree	0.781	0.770

Summarizing

- There are several other approaches that can be used of various complexity and suitability for specific data.
- There is no unique "good" classifier, several methods may be applied and compared to select the "best"
- Often we may need to switch to a different method
- Different methods are based on very different rational!
- They also offer different properties and opportunities!
- Prediction (classification) is a very difficult and delicate task
- New challenges based on new data and model often come in practice

Cross-validation

The idea is to split the data in

- Training set: the observations to be used to built the model
- Test set: the observations to be used to check if the predictions are good enough.

Obviously

- There is a question how to split the data in training-test sets.
- Typicaly we need to redo this a couple of times to avoid the case of an "unfortunate" split.
- This can have more computational burden.

Leave one out Cross-validation

Every time we leave out just one observation.

- We fit the model with all the remaining $n-1$ and then we predict the one left out.
- We repeat this by leaving each observation out once.
- We calculate some score based on these predictions

BUT

Leaving just one observation out we have too similar training sets which perhaps leads to very similar models: overfitting

k-fold Cross-validation

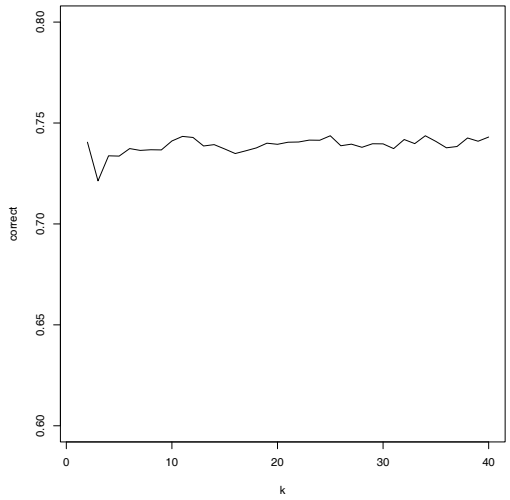
Instead of one observation we leave out more.

In practice we create k folds of observations of almost equal size. E.g having 1000 observations we have 10 folds of 100 observations each. Now we leave out one fold each time, use the rest for model fitting and the one left out for prediction.

BUT

1. How can we select an optimal k.
2. Is the choice of k important?
3. How do we split in folds?

Example



k-fold Cross-validation

- **Unsupervised**: We select randomly. For rare characteristics there is some danger that we do not have enough (or even not at all) observations from some groups which may lead to spurious prediction and perhaps runtime errors.
- **Supervised**: We take care that in each fold we have a good representation of at least the basic characteristics

Random Cross-validation

- Each time we select randomly $n - k$ observations to represent the training set
- Leave the rest for the test set.
- We repeat this a number of times and take as score the average over replications.

This is too simple to run and avoid problems of creating the folds.

Bootstrap

Bootstrap is a well known statistical approach for error estimation.

- The central idea is that if we sample with replacement from our data we get a sample with good properties as being a sample from the underlying population.
- In our case we take bootstrap sample as a training set and use the observations not taken in the training set to be the test set.
- We replicate a number of times.