MS1b Statistical Data Mining
Part 3: Supervised Learning
Nonparametric Methods

Yee Whye Teh
Department of Statistics
Oxford

`http://www.stats.ox.ac.uk/~teh/datamining.html`

Outline

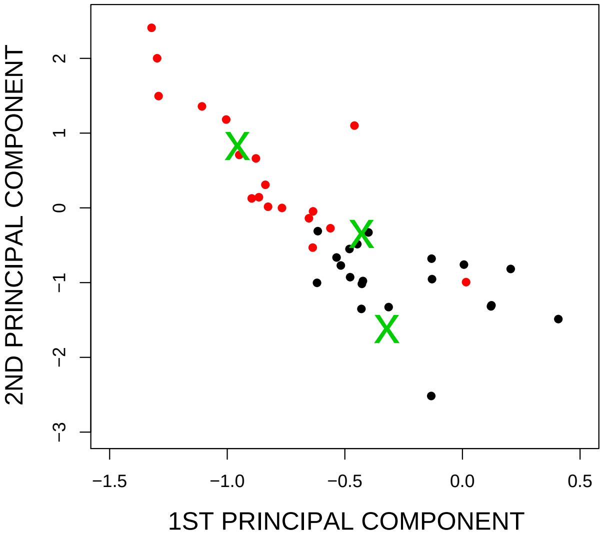
Outline

k-Nearest Neighbours

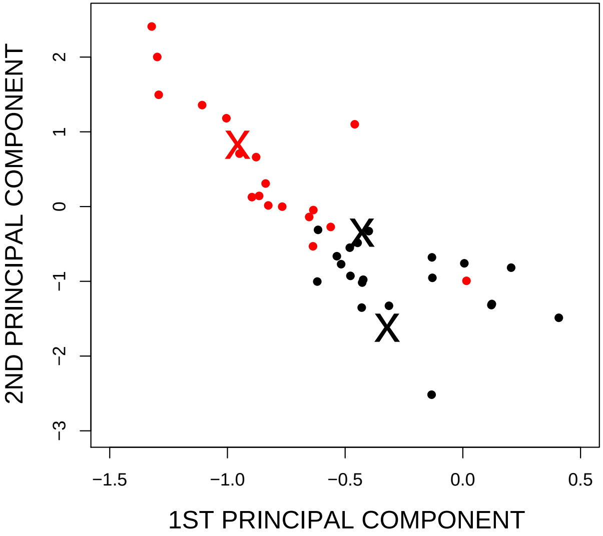
- Nearest neighbours are simple and essentially model-free methods for classification.
- These methods are not very useful for understanding relationships between attributes and class predictions.
- Makes weaker modelling assumptions than e.g. LDA, Naïve Bayes and logistic regression.
- As *black box* classification methods however, they are often reasonable performers on real life problems (at least in lower-dimensional data) and provide a good benchmark as they are trivial to set up.

Example: Spam dataset in 2 dimensions. Using first 2 principal components of the predictor variables X for illustration. Plotted are 50 emails (red: spam, black: no spam).

Task: predict category spam/no-spam for 3 new emails at the green crosses.



True categories.



Suppose again that the training data are (X_i, Y_i) for $i = 1, \ldots, n$, where as usual $X_i \in \mathbb{R}^p$ and $Y_i \in \{1, \ldots, K\}$.
Assuming $\mathcal{X} \in \mathbb{R}^p$, Euclidean distance is often used to measure distance $d(X, X') = \|X - X'\|_2$. The nearest neighbours of a point X is the set $ne_k(X) \subset \{1, \ldots, n\}$ such that

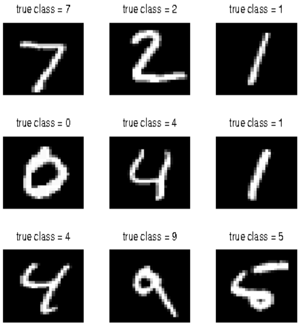
$$\max_{j \in ne_k(X)} d(X, X_j) \quad \leq \quad \min_{i \in \{1,\ldots,n\} \setminus ne_k(X)} d(X, X_i).$$

Given a new X, we find the k observations in the training data ‘closest’ to X and then classify using a majority vote amongst the k neighbours (ties are broken at random – choose k odd preferably).

$$\hat{Y}(X) = \text{argmax}_l \; |\{j \in ne_k(X) : Y_j = l\}|.$$

Application to Handwritten Character Recognition

Objective: recognizing isolated (i.e., non-overlapping) digits, as in ZIP or postal codes.

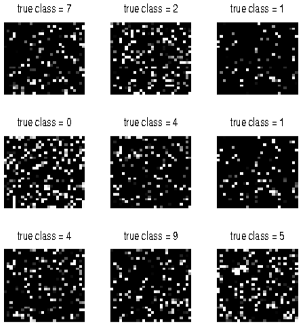


Training and Test Data: The MNIST15 dataset contains 60,000 training images and 10,000 test images of the digits 0 to 9, as written by various people.
Details: Images are 28$\times$28 and have grayscale values in the range 0:255.

Application to Handwritten Character Recognition

Results: 1-NN obtains a misclassification rate of only 3.09% on the test data using the Hamming distance!

This problem might look easy to you but remember that we do not use any spatial information. The K-NN classifier would obtain exactly the same results if the training and test data were permuted as it is invariant to the order of the features.



Asymptotic Performance of 1 NN

Let $(X_i, Y_i)_{i=1}^{n}$ be some training data where $X_i \in \mathbb{R}^p$ and $Y_i \in \{1, 2, ..., K\}$.
We define

$$\widehat{y}_{\text{Bayes}}(x) = \underset{l \in \{1,\ldots,K\}}{\arg\max}\ \pi_l f_l(x)$$

and

$$\widehat{y}_{\text{1NN}}(x) = y\left(\text{nearest neigbour of } x\right).$$

Define

$$\begin{aligned} R_{\text{Bayes}} &= \mathbb{E}\left[\mathbb{I}\left(y \neq \widehat{y}_{\text{Bayes}}(x)\right)\right], \\ R_{\text{1NN}} &= \mathbb{E}\left[\mathbb{I}\left(y \neq \widehat{y}_{\text{1NN}}(x)\right)\right], \end{aligned}$$

then, as $n \to \infty$, we have the following powerful result

$$R_{\text{Bayes}} \leq R_{\text{1NN}} \leq 2R_{\text{Bayes}} - \frac{K}{K-1} R_{\text{Bayes}}^2.$$

- Despite its simplicity, k-NN is often a very good classifiers to use in a wide range of classification problems. At the very least, it is a good 'baseline' against which classifiers can be compared.
- This method allows for extremely flexible (nonparametric) decision boundaries.
- Due to the importance of distance to k-NN, it is important to standardise the data before use and find a suitable metric.
- It is also important to determine an appropriate k.

Influence of k on Decision Boundaries

- k determines the complexity of the decision boundary.
- For $k = 1$, we have no training error but are exposed to **overfitting**.
- Increasing k yields smoother predictions, since we average over more data.
- For $k = n$, we predict the same output whatever being X.

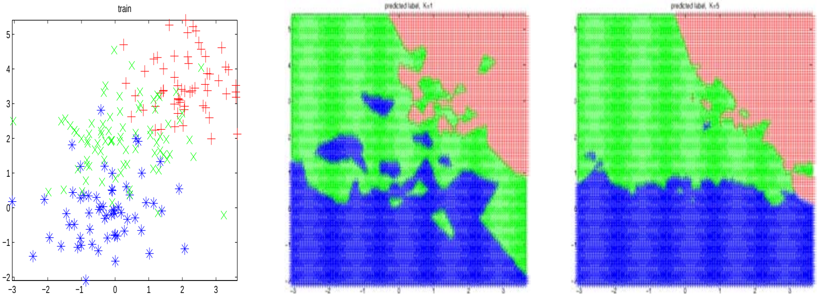


Figure: Training data (left), 1-NN (center) and 5-NN (right)

Using the SPAM dataset properly (without PC). Writing the code from scratch.

```
X <- scale(X)
knn <- 3

predicted <- numeric(length(test))
for (k in 1:length(test)){
  DIFF <- X[train,] -
           outer(rep(1,length(train)),X[test[k],],FUN="*")
  distance <- apply(DIFF^2,1,mean)
  nearestneighbors <- order(distance)[1:knn]
  predicted[k] <- mean(Y[train[nearestneighbors]])
}
```

Predict on the test set.

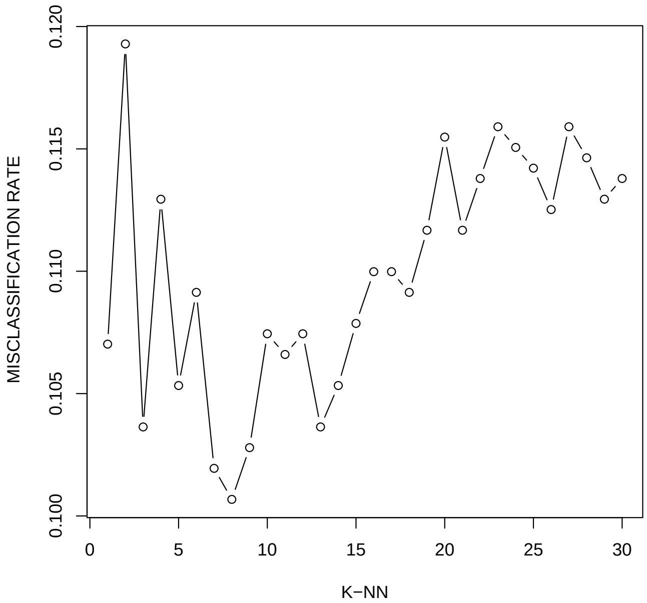
```
predicted_knn <- as.numeric(predicted > 0.5)
> table(predicted_knn, Y[test])

         actual  0    1
 predicted  0 1343  131
            1  110  818
```

Using $k = 9$ nearest neighbours.

```
predicted_knn <- as.numeric(predicted > 0.5)
> table(predicted_knn, Y[test])

          actual 0    1
 predicted  0 1349  156
            1  104  793
```

Misclassification rate is thus about 10.8% for $k = 9$.

Compute misclassification rate as a function of k.



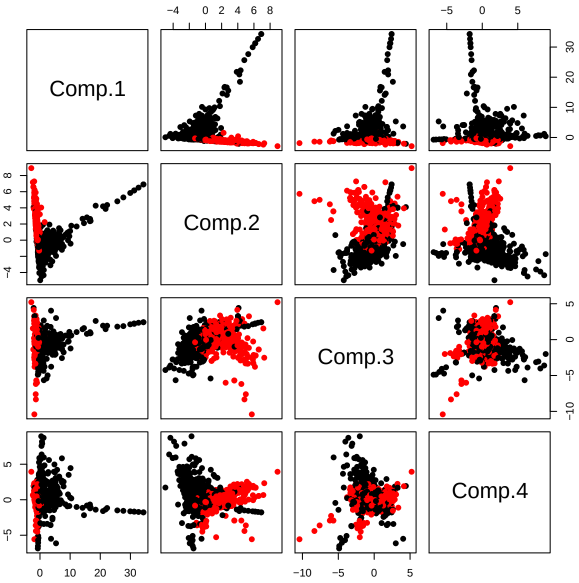
K-means clustering

- A disadvantage of k-nn is the high memory requirement as each new observations has to be matched against *all* observations to find the nearest neighbour(s).
- As discussed before, K-means is a method for finding clusters and cluster centres in a set of unlabeled data. Considering each class in isolation, we can apply the K-means algorithm (each with R clusters) to characterise observations from each class.
- These $K \times R$ labeled *prototypes* can be used to summarise the distribution of class data over $\mathcal{X}$. For a new observations X, we predict to the class with the nearest prototype. An advantage over k-NN is thus that much fewer observations/prototypes have to be kept in memory.

Plot Spam data in the space of the first 4 Principal Components.

```
library(kernlab)
data(spam)
n <- nrow(spam)
p <- ncol(spam)-1


Y <- as.numeric(spam[, p+1])-1
X <- predict(princomp(spam[,-(p+1)] ,cor=TRUE))[,1:4]

pairs(X,col=Y+1,cex=1.5,pch=20)
```

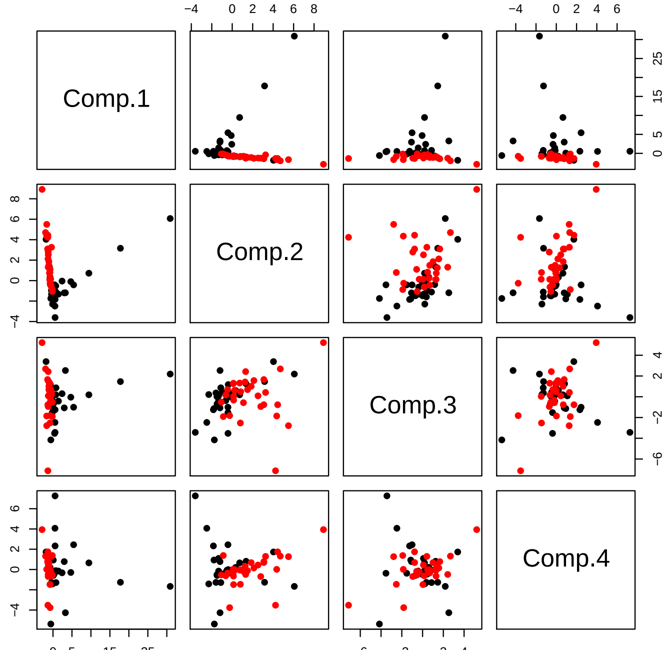
Red: spam
Black: not spam.



Now replace each class by 30 cluster centers.

```
nk <- 30
KMX <- kmeans(X[Y==0,], nk)
KMY <- kmeans(X[Y==1,], nk)


pairs( rbind(KMX$centers,KMY$centers),
            col= rep(1:2,each=nk), cex=1.5, pch=20)
```

And then use these as prototypes for k-NN classification.



Outline

Learning Vector Quantization

- Though K-means clustering helps to significantly reduce the memory load of k-NN, the most obvious shortcoming of this method is that prototypes from different classes have no say about each other's positioning.
- Recalling the VQ learning algorithm from clustering techniques for unsupervised learning, it is easy to extend it to tackle supervised learning problems.
- Recall that VQ seeks to find areas of high density in high dimensional data by strategically placing codewords (in an online or batch approach).

Consider the online version of LVQ.

1. For each of the K classes, initialise R prototypes (representative points) to model each class distribution.
2. Sample an observation X and let V_c be the Voronoi region where it falls with cluster center μ_c.
3. If the prototype is of the same class as X, move μ_c towards X

   $$mu_c \leftarrow \mu_c + \alpha(t)\left[X - \mu_c\right]$$

   and if μ_c is of a different class, move it away from X

   $$\mu_c \leftarrow \mu_c - \alpha(t)\left[X - \mu_c\right]$$

Repeat 2-3 many times and return the codebook.

Nearest Neighbours in High Dimensions

We have seen various ways to find nearest neighbors and the corresponding classification is intuitive in 2, 3 and general low-dimensional problems. The concept of a nearest neighbour is questionable, however, in high dimensions. First, look at multi-variate normal data in p dimensions,

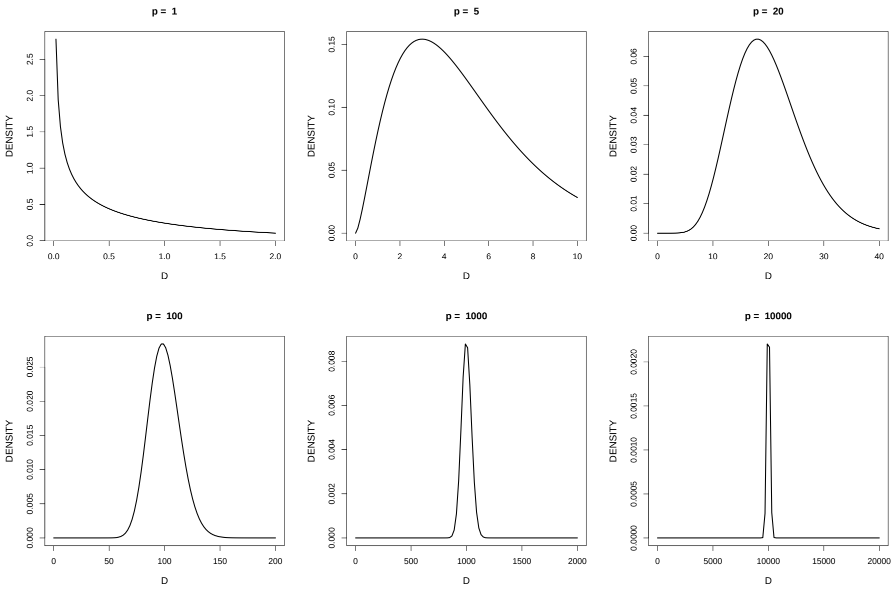
$$X \sim \mathcal{N}(\mu, \Sigma).$$

What is the distribution of the Euclidean distance D between a random observation X and the ‘cluster center’ μ if $\Sigma = 1_p$? It is

$$D = \sum_{k=1}^{p} (X^{(k)} - \mu^{(k)})^2.$$

And D has thus a χ_p^2-distribution with p degrees of freedom.

Density of distance D of observation from cluster center in p dimensions.



kNN in High Dimensional Spaces

Assume you have $\{X_i\}_{i=1}^n$ in $\mathbb{R}^p$ where $X_i \overset{\text{i.i.d.}}{\sim} f$.
Proposition. If we have

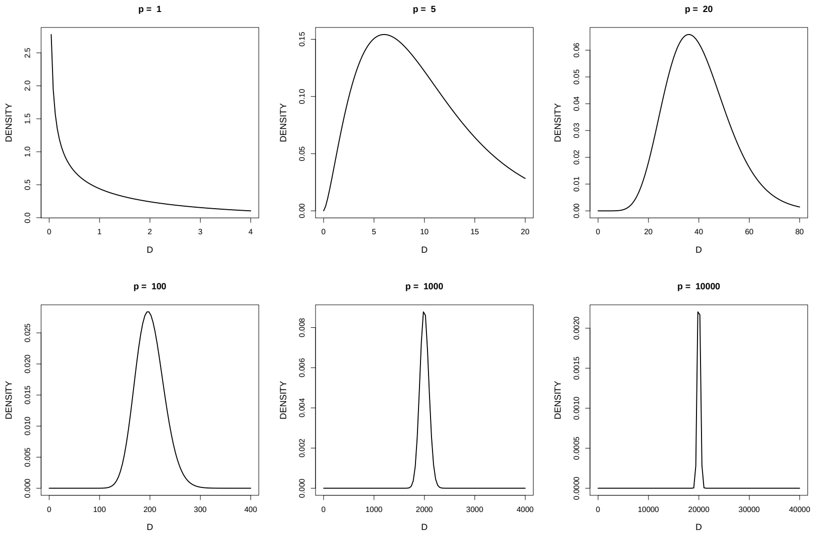
$$\lim_{p\to\infty} \frac{\mathbb{V}_f\left[d\left(X,x\right)\right]}{\mathbb{E}_f\left[d\left(X,x\right)\right]^2} = 0$$

then for any $\varepsilon > 0$

$$\lim_{p\to\infty} \mathbb{P}_{f^{\otimes n}}\left(\left|\max_{1\leq i\leq n} d\left(X_i,x\right) - \min_{1\leq i\leq n} d\left(X_i,x\right)\right| \geq \varepsilon\right) = 0.$$

Loosely speaking, in high dimensional spaces, all the points are at the same distance from the query point x so kNN is useless.
Example: Assume $d\left(X,x\right) = \sum_{l=1}^p \left(X^l - x^l\right)^2$ where $x^l = \left(\mu, ..., \mu\right)$ and $f\left(x\right) = \prod_{l=1}^{p} \mathcal{N}\left(x^l; 0, 1\right)$ then $d\left(X,x\right)$ follows a non-central chi-squared of variance $2p\left(1+2\mu^2\right)$ and mean $p\left(1+\mu\right)$ so that $\lim_{p\to\infty} \frac{\mathbb{V}_f\left[d\left(X,x\right)\right]}{\mathbb{E}_f\left[d\left(X,x\right)\right]^2} = 0.$.

Density of distance $\tilde{D}$ between two random observations in p dimensions.



There are no real ‘nearest neighbours’ in high dimensions. All points are about the same distance from each other and are sitting on the shell of the high-dimensional sphere.

Now suppose there are two groups μ_1 and μ_2, where for the two classes the distributions of $X = (X^{(1)}, \dots, X^{(p)})$ are, respectively,

$$\mathcal{N}(\mu_1, \Sigma) \qquad \text{and} \qquad \mathcal{N}(\mu_2, \Sigma),$$

with

$$\mu_1 = (2, 0, 0, 0, \dots, 0)^T \qquad \text{and} \qquad \mu_2 = (0, 0, 0, 0, \dots, 0)^T,$$

and $\Sigma = 1_p$. The two groups distinguish themselves thus just in the first component $X^{(1)}$.

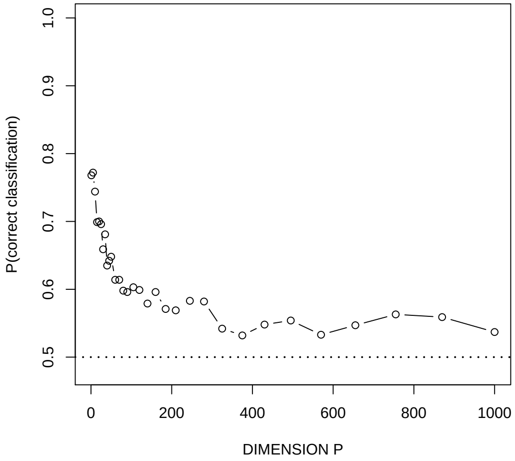
Suppose we have n observations $X_1, \dots, X_{2n}$, of which n are in class 1 and n in class 2. What is the probability $P(\text{correct classification})$ that a randomly chosen X from class 1 will have a nearest neighbor in $\{i : Y_i = 1\}$ rather than in $\{i : Y_i = 2\}$?

$$P(\text{correct classification}) = P(\min_{i:Y_i=1} \|X - X_i\|_2 \leq \min_{i:Y_i=2} \|X - X_i\|_2).$$

Answer easiest by simulation...

```
pvec <- pmax(1,unique(round((1/5)*exp(seq(0,log(1000),length=50)))*5))
nsim <- 1000
n <- 100
probability <- rep(0,length(pvec))
for (pc in 1:length(pvec)){
  p <- pvec[pc]
  for (sim in 1:nsim){

    X1 <- matrix(rnorm(n*p),nrow=n)
    X2 <- matrix(rnorm(n*p),nrow=n)
    X2[,1] <- X2[,1] + 2
    X <- rnorm(p)

    distance1 <- numeric(n)
    distance2 <- numeric(n)
    for (k in 1:n){
      distance1[k] <- mean( (X-X1[k,])^2 )
      distance2[k] <- mean( (X-X2[k,])^2 )
    }
    winningclass1 <- min(distance1)<min(distance2)
    if(winningclass1)  probability[pc] <- probability[pc] + 1/nsim
  }
  plot(pvec,probability,
    xlab="DIMENSION P",ylab="P(correct classification)",type="b")
}
```

Probability of correct classification with nearest neighbours as a function of dimension p. Misclassification probability of 0.5 can be achieved by random guessing (dotted line).



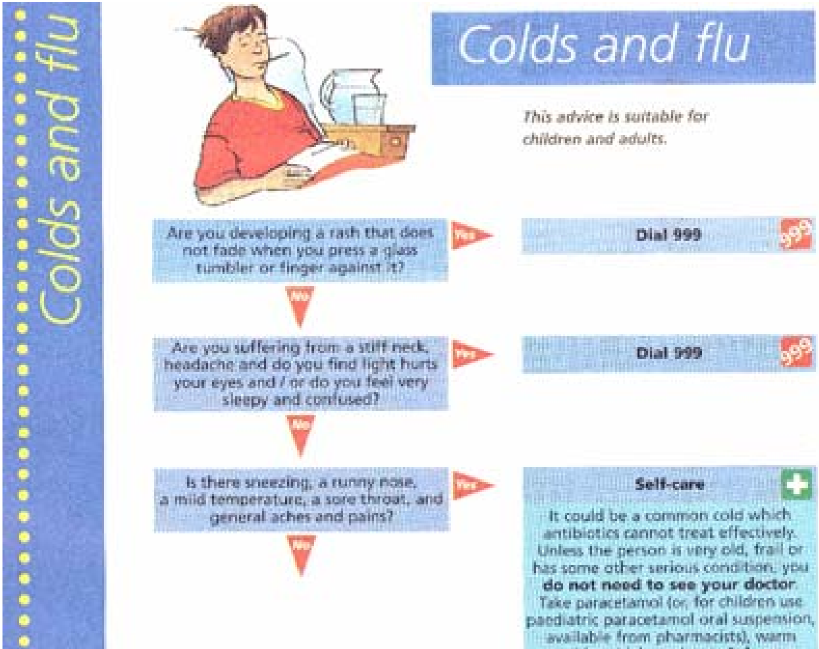
Nearest neighbor potentially performs poorly in high dimensions.

Outline

Classification and Regression Trees

CART is arguably the most widely used tree algorithm in the statistics community, but most tree algorithms are quite similar.
We suppose in the following that the p predictor variables are real-valued but extensions to categorical variables are straightforward.
A tree is then equivalent to a partition of $\mathbb{R}^p$ into R disjoint sets $\mathcal{P} = \{\mathcal{R}_1, \ldots, \mathcal{R}_R\}$, where each $\mathcal{R}_j \subset \mathbb{R}^p$ and has constant fitted values in each region $\mathcal{R}_j$, $j = 1, \ldots, R$.

Example I : NHS Direct self-help guide



Colds and flu

Colds and flu

This advice is suitable for children and adults.

Are you developing a rash that does not fade when you press a glass tumbler or finger against it? — Yes → **Dial 999**

No ↓

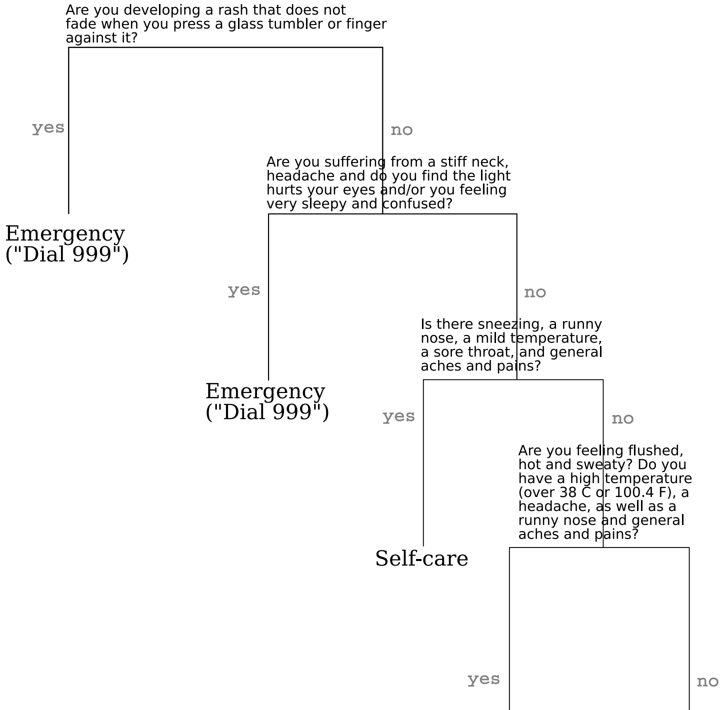
Are you suffering from a stiff neck, headache and do you find light hurts your eyes and / or do you feel very sleepy and confused? — Yes → **Dial 999**

No ↓

Is there sneezing, a runny nose, a mild temperature, a sore throat, and general aches and pains? — Yes → **Self-care**

It could be a common cold which antibiotics cannot treat effectively. Unless the person is very old, frail or has some other serious condition, you **do not need to see your doctor**. Take paracetamol (or, for children use paediatric paracetamol oral suspension, available from pharmacists), warm

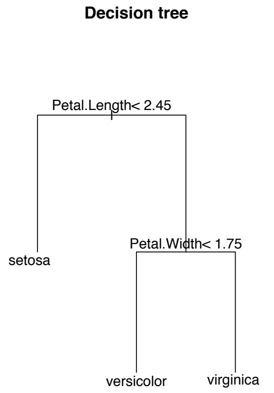
No ↓

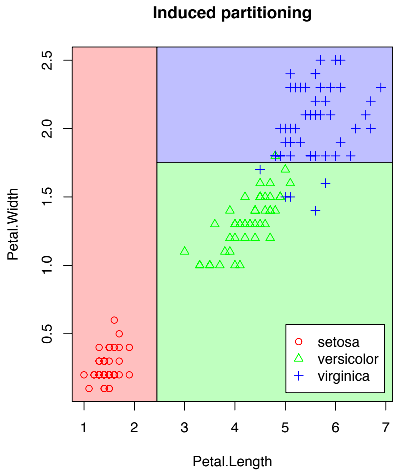

Are you developing a rash that does not fade when you press a glass tumbler or finger against it?
yes
no
Emergency ("Dial 999")
Are you suffering from a stiff neck, headache and do you find the light hurts your eyes and/or you feeling very sleepy and confused?
yes
no
Emergency ("Dial 999")
Is there sneezing, a runny nose, a mild temperature, a sore throat, and general aches and pains?
yes
no
Self-care
Are you feeling flushed, hot and sweaty? Do you have a high temperature (over 38 C or 100.4 F), a headache, as well as a runny nose and general aches and pains?
yes
no

Example II: Iris Data

```
  Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
           4.4         3.2          1.3         0.2     setosa
           5.9         3.0          5.1         1.8  virginica
           6.3         3.3          6.0         2.5  virginica
           5.3         3.7          1.5         0.2     setosa
           5.5         2.5          4.0         1.3 versicolor
           6.1         2.9          4.7         1.4 versicolor
           6.1         3.0          4.9         1.8  virginica
           5.7         2.8          4.5         1.3 versicolor
           5.4         3.0          4.5         1.5 versicolor
           4.8         3.4          1.6         0.2     setosa
           4.6         3.1          1.5         0.2     setosa
           4.9         3.1          1.5         0.2     setosa
           6.4         2.9          4.3         1.3 versicolor
           .......
```

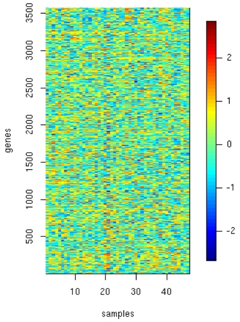
Previously seen Iris data set gives the measurements in centimeters of the variables sepal length and width and petal length and width, respectively, for 50 flowers from each of 3 species of iris. The species are Iris setosa, versicolor, and virginica.

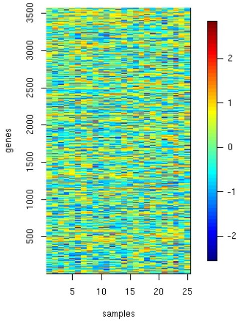


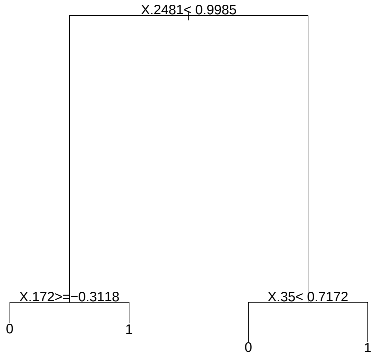


The decision tree derived from the Iris data (left) and the partitioning of feature space.

Example III: Leukemia Prediction







Leukemia Dataset: Expression values of 3541 genes for 47 patients with Leukemia ALL subtype (left) and 25 patients with AML (middle). Decision tree (right).

Some terminology

We will use the following terminology.

- **Parent** of a node c is the set of nodes (here maximally of size 1) which have an arrow pointing towards c.
- **Children** of a node c are those nodes which have node c as a parent.
- **Root node** is the top node of the tree; the only node without parents.
- **Leaf nodes** are nodes which do not have children.
- **Stumps** are trees with just the root node and two leaf nodes.
- A K**-ary tree** is a tree where each node (except for leaf nodes) has K children. Usually working with binary trees ($K = 2$).
- The **depth** of a tree is the maximal length of a path from the root node to a leaf node.

Regression

For regression, CART provides a piecewise constant prediction on each region $\mathcal{R}_j$,

$$\hat{Y}_{tree}(x) = \sum_{r=1}^{R} \beta_r \mathbf{1}_{[x \in \mathcal{R}_r]},$$

where β_j is the constant fitted value in $\mathcal{R}_j$.
The function $\hat{Y}(\cdot)$ is hence determined if the (a) partition and (b) the fitted values β are chosen. These choices are made such as to minimize the expected squared error loss for future observations (x, Y),

$$E(Y - \hat{Y}(x))^2.$$

Classification

For classification with two classes, the response is in $Y \in \{0,1\}$. CART chooses again a piece-wise constant function

$$\hat{Y}_{tree}(x) = \sum_{r=1}^{R} \beta_r \mathbf{1}_{[x \in \mathcal{R}_r]}.$$

This time, $\beta_r \in [0,1]$. The default classification is

$$\eta_{tree}(x) = \begin{cases} 0 & \hat{Y}_{tree}(x) \le 1/2 \\ 1 & \hat{Y}_{tree}(x) > 1/2 \end{cases} .$$

A good choice of $\hat{Y}_{tree}$ is one that leads to a small misclassification error

$$P(\eta_{tree}(x) \neq Y)$$

or, in general, to a small loss

$$E(L(Y, \eta_{tree}(X))),$$

for a loss function $\{0,1\} \times \{0,1\} \mapsto \mathbb{R}^+$.

Parameter Estimation

Recall model

$$\hat{Y}_{tree}(x) = \sum_{r=1}^{R} \beta_r \mathbf{1}_{[x \in \mathcal{R}_r]},$$

Parameter estimation $\hat{\beta}_1, \ldots, \hat{\beta}_R$ is easy if the partition $\mathcal{P} = \{\mathcal{R}_1, \ldots, \mathcal{R}_R\}$ were given.
We use

$$\begin{aligned} \hat{\beta}_r &= \sum_{i=1}^{n} Y_i \mathbf{1}_{[x_i \in \mathcal{R}_r]} / \sum_{i=1}^{n} \mathbf{1}_{[x_i \in \mathcal{R}_r]} \\ &= \text{mean}\{Y_i : X_i \in \mathcal{R}_r\}. \end{aligned}$$

for regression and binary classification (where $\hat{\beta}_r$ is just the proportion of samples from class 1 in region $\mathcal{R}_r$).

Partition Estimation

Ideally, would like to find partition that allows (with the previous parameter estimates) to achieve lowest mean-squared error loss (prediction) or misclassification rate (classification).
Number of potential partitions is too large to search exhaustively for problems of even just small to moderate size (in terms of number p of variables and number n of samples).
Need ‘greedy’ search for a good partition. First search for a good split for the root node and then work successively downwards in the tree.

Splitpoint estimation for regression trees

Given are data-points $(X_1, Y_1), \ldots, (X_n, Y_n)$, where each $X_i = (X_i^{(1)}, \ldots, X_i^{(p)})$ is a p-dimensional vector.
For continuous predictor variables, the search for a partition works as follows.

1. Start with $\mathcal{R}_1 = \mathbb{R}^p$.
2. Given a partition $\mathcal{R}_1, \ldots, \mathcal{R}_r$, split each region $\mathcal{R}_j$ into two parts $\mathcal{R}_{j_1}, \mathcal{R}_{j_2}$, where

$$\begin{aligned} \mathcal{R}_{j_1} &= \{x \in \mathbb{R}^p : x \in \mathcal{R}_j \text{ and } X^{(k)} \le c\}. \\ \mathcal{R}_{j_2} &= \{x \in \mathbb{R}^p : x \in \mathcal{R}_j \text{ and } X^{(k)} > c\}, \end{aligned}$$

   where splitpoint c and splitting variable k are found as

$$\text{argmin}_{c,k} \min_{\beta_1,\beta_2} \Big(\sum_{i:X_i \in \mathcal{R}_{j_1}} (Y_i - \beta_1)^2 + \sum_{i:X_i \in \mathcal{R}_{j_2}} (Y_i - \beta_2)^2 \Big).$$

   Let $\mathcal{R}_{1_1}, \mathcal{R}_{1_2}, \ldots, \mathcal{R}_{r_1}, \mathcal{R}_{r_2}$ be the new partition.
3. Repeat step 2) d times to get tree of depth d.

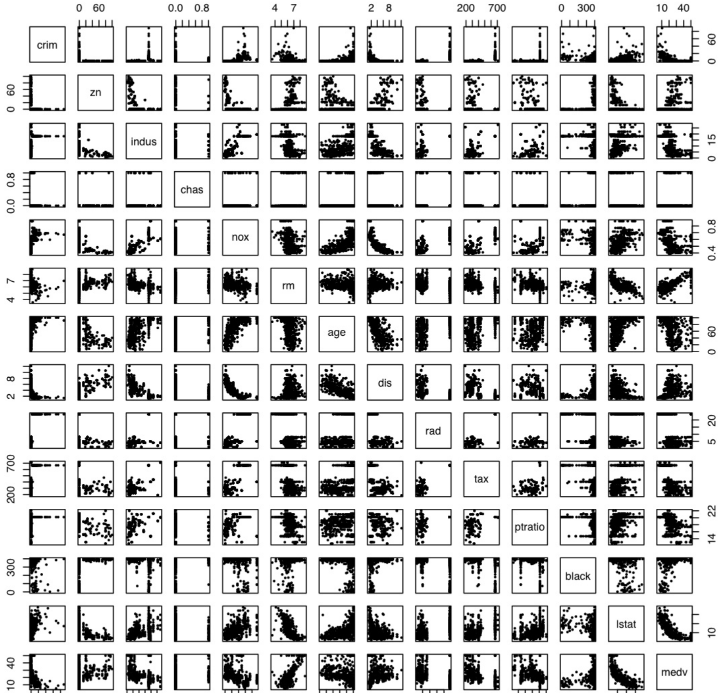
Boston Housing Data

The original data are 506 observations on 14 variables, medv being the response variable:

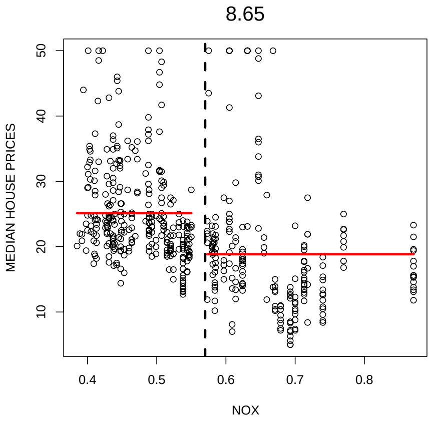
crim	per capita crime rate by town
zn	proportion of residential land zoned for lots over 25,000
indus	proportion of non-retail business acres per town
chas	Charles River dummy variable (= 1 if tract bounds river;
nox	nitric oxides concentration (parts per 10 million)
rm	average number of rooms per dwelling
age	proportion of owner-occupied units built prior to 1940
dis	weighted distances to five Boston employment centres
rad	index of accessibility to radial highways
tax	full-value property-tax rate per USD 10,000
ptratio	pupil-teacher ratio by town
b	1000(B - 0.63)^2 where B is the proportion of blacks by to
lstat	percentage of lower status of the population
medv	median value of owner-occupied homes in USD 1000's

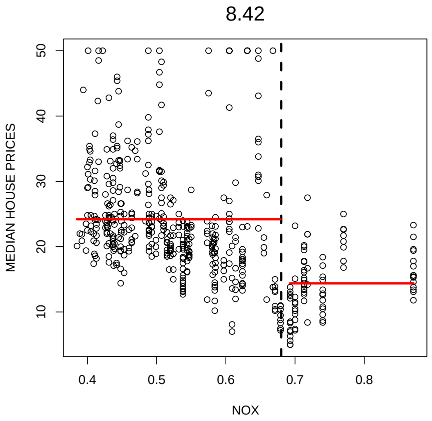
```
> library(MASS)
> data(Boston)
> str(Boston)
'data.frame': 506 obs. of  14 variables:
 $ crim   : num  0.00632 0.02731 0.02729 0.03237 0.06905 ...
 $ zn     : num  18 0 0 0 0 0 12.5 12.5 12.5 12.5 ...
 $ indus  : num  2.31 7.07 7.07 2.18 2.18 2.18 7.87 7.87 7.87 7.87
 $ chas   : int  0 0 0 0 0 0 0 0 0 0 ...
 $ nox    : num  0.538 0.469 0.469 0.458 0.458 0.458 0.524 0.524 0
 $ rm     : num  6.58 6.42 7.18 7.00 7.15 ...
 $ age    : num  65.2 78.9 61.1 45.8 54.2 58.7 66.6 96.1 100 85.9
 $ dis    : num  4.09 4.97 4.97 6.06 6.06 ...
 $ rad    : int  1 2 2 3 3 3 5 5 5 5 ...
 $ tax    : num  296 242 242 222 222 222 311 311 311 311 ...
 $ ptratio: num  15.3 17.8 17.8 18.7 18.7 18.7 15.2 15.2 15.2 15.2
 $ black  : num  397 397 393 395 397 ...
 $ lstat  : num  4.98 9.14 4.03 2.94 5.33 ...
 $ medv   : num  24 21.6 34.7 33.4 36.2 28.7 22.9 27.1 16.5 18.9 .
```

...predict median house price given 13 predictor variables.

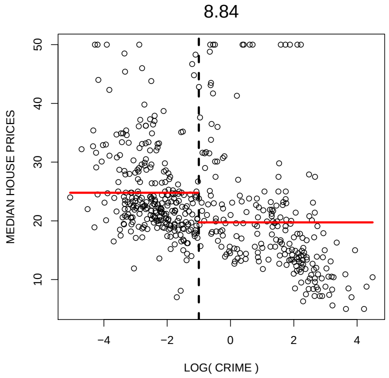

crim
zn
indus
chas
nox
rm
age
dis
rad
tax
ptratio
black
lstat
medv
0
60
0.0
0.8
4
7
2
8
200
700
0
300
10
40
15
0.4
20
5
22
14
10

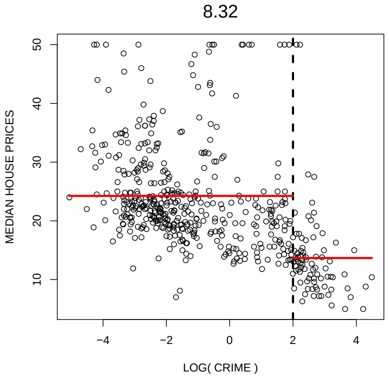
Try splitting on variable ‘NOX’ at two different splitpoints and look at the residual sum of squares.





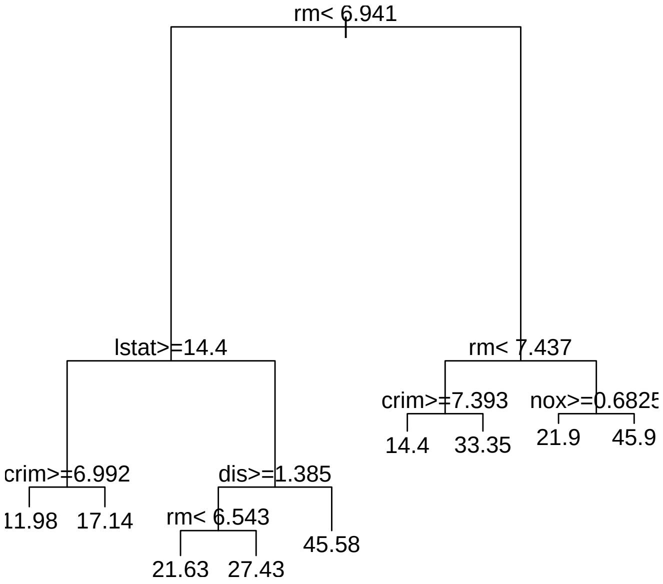
Try splitting now on variable ‘CRIME at two different splitpoints instead.





...last split is most favourable among the four considered splits as it leads to largest reduction in MSE. Choose this split.

Overall, the best first split is on variable `rm`, average number of rooms per dwelling. Final tree contains predictions in leaf nodes.



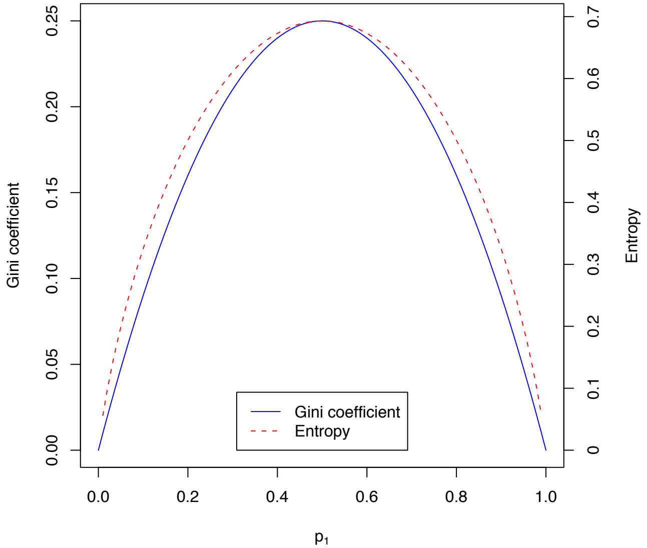
Classification

Remember that for binary classification,

$$\hat{p}_r = \hat{\beta}_r = \sum_{i=1}^{n} Y_i \mathbf{1}_{[x_i \in \mathcal{R}_r]} / \sum_{i=1}^{n} \mathbf{1}_{[x_i \in \mathcal{R}_r]}$$

is just the estimated probability for class $Y = 1$ in each partition $\mathcal{R}_r$.
The tree growth algorithm is identical to the regression case, except that one is using a different measure of node impurity. For regression, the residual sum of squares $\sum_{i:\mathcal{R}_r} (Y_i - \hat{\beta}_r)^2$ was used for each region $\mathcal{R}_r$.
For classification, try instead to minimize a measure of node impurity:

- Misclassification error: $1 - \max\{\hat{p}_r, 1 - \hat{p}_r\}$.
- Gini Index: $2\hat{p}_r(1 - \hat{p}_r)$.
- Cross-entropy: $-\hat{p}_r \log \hat{p}_r - (1 - \hat{p}_r) \log(1 - \hat{p}_r)$.



Misclassification error?

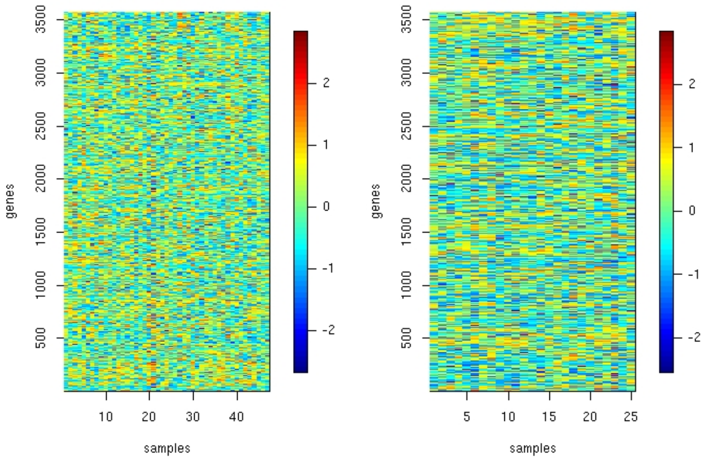
All three criteria of misclassification error are similar, but Gini Index and Cross-entropy usually preferred. Gini Index and Cross-entropy are differentiable; misclassification error not.
Gini Index and Cross-entropy also favour purer nodes. Consider example where 400 observations of class 0 and 400 observations of class 1 are present. Possible splits into
A: (300,100) and (100,300) vs.
B: (200,400) and (200,0).

A: (300,100) and (100,300) vs.
B: (200,400) and (200,0).

Split A	$\hat{\beta}$	Misclassification error	Gini Index
Node 1	1/4	1/4	$2\frac{1}{4}(1-\frac{1}{4}) = 3/8$
Node 2	3/4	1/4	$2\frac{3}{4}(1-\frac{3}{4}) = 3/8$
Total		$\frac{400}{800} \cdot 1/4 + \frac{400}{800} \cdot 1/4 = 1/4$	$\frac{400}{800} \cdot 3/8 + \frac{400}{800} \cdot 3/8 = 3/8$

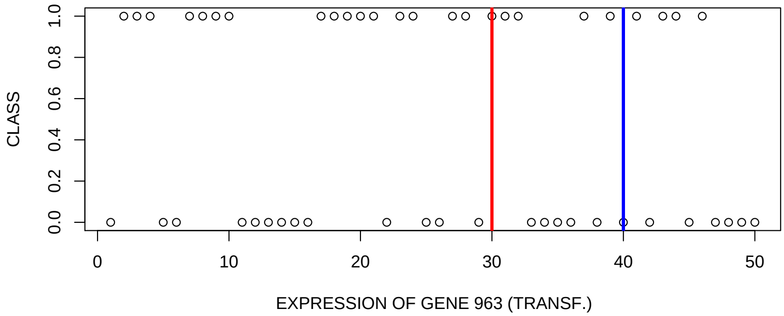
Split B	$\hat{\beta}$	Misclassification error	Gini Index
Node 1	2/3	1/3	$2\frac{2}{3}(1-\frac{2}{3}) = 4/9$
Node 2	0	0	0
Total		$\frac{600}{800} \cdot 1/3 + \frac{200}{800} \cdot 0 = 1/4$	$\frac{600}{800} \cdot 4/9 + \frac{200}{800} \cdot 0 = 1/3$

Example: Leukemia Prediction



Leukemia Dataset: Expression values of 3541 genes for 47 patients with Leukemia ALL subtype (left) and 25 patients with AML (right).

Compare two potential splits (red and blue) on gene 963.



	$X \geq 30$	$X < 30$	
Y=0	12	13	25
Y=1	9	16	25
$\hat{\beta}$	0.42	0.55	

	$X \geq 40$	$X < 40$	
Y=0	7	18	25
Y=1	4	21	25
$\hat{\beta}$	0.36	0.53	

	$X \geq 30$	$X < 30$
Y=0	12	13
Y=1	9	16
$\hat{\beta}$	0.42	0.55

Misclassification error:

$$\frac{12+9}{50} \cdot 0.42 + \frac{13+16}{50} \cdot (1-0.55) = 0.4374$$

Gini Index:

$$2\frac{12+9}{50} \cdot 0.42(1-0.42) + 2\frac{13+16}{50} \cdot 0.55(1-0.55) = 0.4917$$

	$X \geq 40$	$X < 40$
Y=0	7	18
Y=1	4	21
$\hat{\beta}$	0.36	0.53

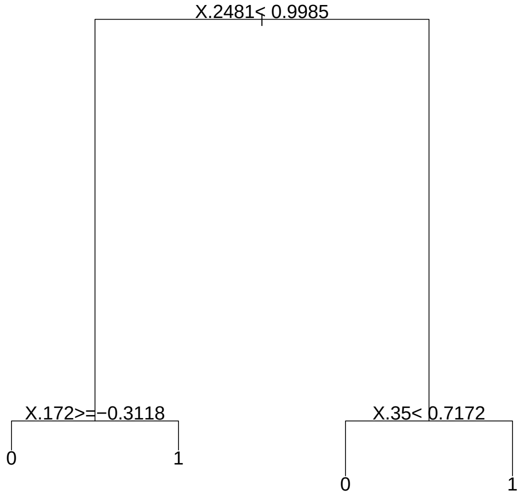
Misclassification error:

$$\frac{7+4}{50} \cdot 0.36 + \frac{18+21}{50}(1-0.53) = 0.445$$

Gini Index:

$$2\frac{7+4}{50} \cdot 0.36(1-0.36) + 2\frac{18+21}{50} 0.53(1-0.53) = 0.4899$$

Final tree is of depth 2. This tree is very interpretable as it selects 3 out of 4088 genes and bases prediction only on these (as opposed to LDA, QDA, LR and k-NN which perform no variable selection).



Extension to multi-class problems

Let $Y \in \{1, \dots, K\}$. The empirical probability of class m in a region $\mathcal{R}_r$ is

$$\hat{p}_{m,r} = \frac{1}{N_r} \sum_{i: X_i \in \mathcal{R}_r} \mathbf{1}\{Y_i = m\},$$

where $N_r = \sum_{i: X_i \in \mathcal{R}_r} 1$ are the number of samples in region $\mathcal{R}_r$. Let m_r^* be the class with the highest probability

$$m_r^* = \text{argmax}_m \, \hat{p}_{m,r}.$$

The measures of node impurity generalize then as

- Misclassification error: $1 - \hat{p}_{m_r^*,r}$.
- Gini Index: $\sum_{m \neq m'} \hat{p}_{m,r}\hat{p}_{m',r} = \sum_{m=1}^{k} \hat{p}_{m,r}(1 - \hat{p}_{m,r})$.
- Cross-entropy: $-\sum_{m=1}^{k} \hat{p}_{m,r} \log \hat{p}_{m,r}$.

Outline

Model complexity

Which size of the tree is optimal?
Can grow tree until every leaf node contains only 1 original observation.
Clearly one should stop before. But where?
Example: Pima Indians Dataset.

The subjects were women who were at least 21 years old, of Pima Indian heritage and living near Phoenix, Arizona. They were tested for diabetes according to World Health Organisation criteria.

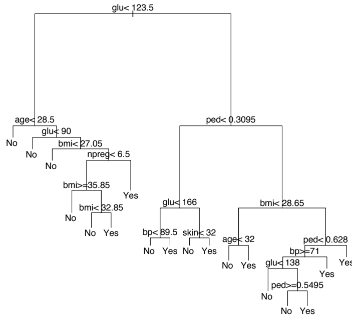
The variables measured were the number of pregnancies (npreg), the plasma glucose concentration in an oral glucose tolerance test (glu), the diastolic blood pressure in mmHg (bp), the triceps skin fold thickness in mm(skin), the body mass index(bbi), the diabetes pedigree function (ped), and the age (age).

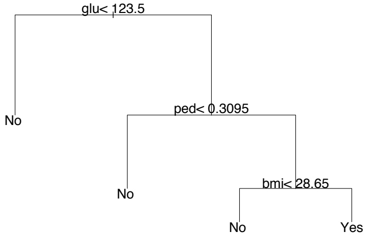
```
> library(rpart)
> library(MASS)
> data(Pima.tr)
> str(Pima.tr)

> > Pima.tr
    npreg glu  bp skin  bmi   ped age type
1       5  86  68   28 30.2 0.364  24   No
2       7 195  70   33 25.1 0.163  55  Yes
3       5  77  82   41 35.8 0.156  35   No
4       0 165  76   43 47.9 0.259  26   No
5       0 107  60   25 26.4 0.133  23   No
6       5  97  76   27 35.6 0.378  52  Yes
7       3  83  58   31 34.3 0.336  25   No
8       1 193  50   16 25.9 0.655  24   No
9       3 142  80   15 32.4 0.200  63   No
10      2 128  78   37 43.3 1.224  31  Yes
11      0 137  40   35 43.1 2.288  33  Yes
12      9 154  78   30 30.9 0.164  45   No
13      1 189  60   23 30.1 0.398  59  Yes
...
```

```
> rp <- rpart(Pima.tr[,8] ~ ., data=Pima.tr[,-8])
> rp
n= 200

node), split, n, loss, yval, (yprob)
      * denotes terminal node

 1) root 200 68 No (0.66000000 0.34000000)
   2) glu< 123.5 109 15 No (0.86238532 0.13761468)
     4) age< 28.5 74  4 No (0.94594595 0.05405405) *
     5) age>=28.5 35 11 No (0.68571429 0.31428571)
      10) glu< 90 9  0 No (1.00000000 0.00000000) *
      11) glu>=90 26 11 No (0.57692308 0.42307692)
        22) bp>=68 19  6 No (0.68421053 0.31578947) *
        23) bp< 68 7  2 Yes (0.28571429 0.71428571) *
   3) glu>=123.5 91 38 Yes (0.41758242 0.58241758)
     6) ped< 0.3095 35 12 No (0.65714286 0.34285714)
      12) glu< 166 27  6 No (0.77777778 0.22222222) *
      13) glu>=166 8  2 Yes (0.25000000 0.75000000) *
     7) ped>=0.3095 56 15 Yes (0.26785714 0.73214286)
      14) bmi< 28.65 11  3 No (0.72727273 0.27272727) *
      15) bmi>=28.65 45  7 Yes (0.15555556 0.84444444) *
```

Two possible trees.

```
> rp1 <- rpart(Pima.tr[,8] ~ ., data=Pima.tr[,-8])
> plot(rp1);text(rp1)


> rp2 <- rpart(Pima.tr[,8] ~ ., data=Pima.tr[,-8],
                        control=rpart.control(cp=0.05))
> plot(rp2);text(rp2)
```





Model Complexity

What influence has the size of the tree on predictive performance?

- The larger the tree is (the more final leaf nodes), the better is the prediction on the **training samples**.
- However, performance on **new data** / **test data** is deteriorating –in general– after a certain complexity (size) of the tree is surpassed.

Want to find the optimal complexity / tree size, giving best predictive performance for new (unseen) data.

Training and test error rate

Let $L(Y, \hat{Y})$ be a loss function that measures the loss when observing Y under a predition $\hat{Y}$.

- For regression trees,
$$L(Y, \hat{Y}) = (Y - \hat{Y})^2.$$
- For classification trees
$$L(Y, \hat{Y}) = 1\{Y \neq \hat{Y}\}.$$

There are two important error rates, when using observations $(X_1, Y_1), \ldots, (X_n, Y_n)$ and a predictor $\hat{Y} = \hat{Y}(x)$. The fitted values at the n observations are $\hat{Y}_i := \hat{Y}(X_i)$.

- **Training error** rate R (or apparent error rate) is the loss for the training sample,

$$R_{train} \ = \ n^{-1} \sum_{i=1}^{n} L(Y_i, \hat{Y}_i).$$

- True error is the expected error rate/risk for new data (X, Y)

$$R_{test} \ = \ E\big(L(Y, \hat{Y})\big),$$

  where the expectation is with respect to drawing new random pairs (X, Y) and using the predictor $\hat{Y} = \hat{Y}(X)$ at the newly observed X.

Cross-Validation

Suppose we had

- **training data** (X_i, Y_i), $i = 1, \ldots, n$
- and a separate set of **test data** $(\tilde{X}_j, \tilde{Y}_j)$, $j = 1, \ldots, n_{test}$.

One possibility of estimating the true error rate is to

- **fit** the predictor $\hat{Y}$ (a tree here) on the **training data** and then
- **evaluate** the error rate on the **test data** (which have not been used for fitting of the tree),

$$\hat{R}_{test} = n_{test}^{-1} \sum_{j=1}^{n_{test}} L(\tilde{Y}_j, \hat{Y}(\tilde{X}_j)).$$

Disadvantage: if we have n_{test} additional samples, we could have used test data to get a larger training set and thus a better predictor.

Leave-one out cross-validation (LOO-CV)

For all $i = 1, \dots, n$:

- fit the tree $T^{(-i)}$ by using all n observations **except** the i-th observation.
- compute prediction $\hat{Y}^{(-i)}(X_i)$ by running X_i down this tree.

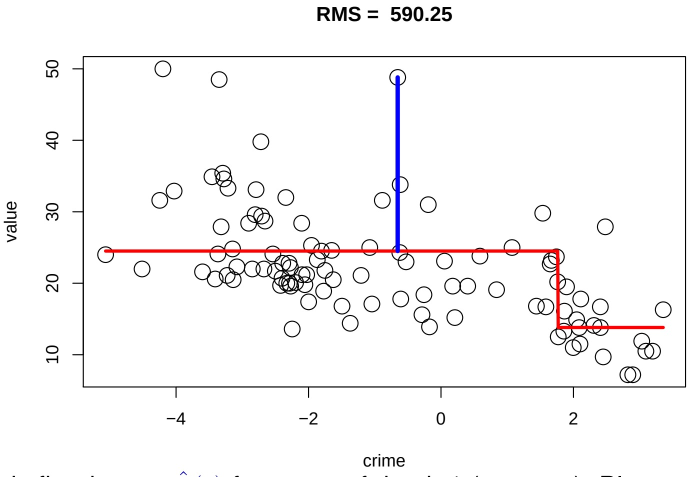
Compute the LOO-CV estimate of generalization error as

$$\hat{R}_{test} = n^{-1} \sum_{i=1}^{n} (\hat{Y}^{(-i)}(X_i) - Y_i)^2$$

for regression and mis-classification error or entropy criterion for classification. LOO-CV is a nearly unbiased estimate of generalization error. It can be expensive to compute as the tree (or other predictor) needs to be re-computed n times.

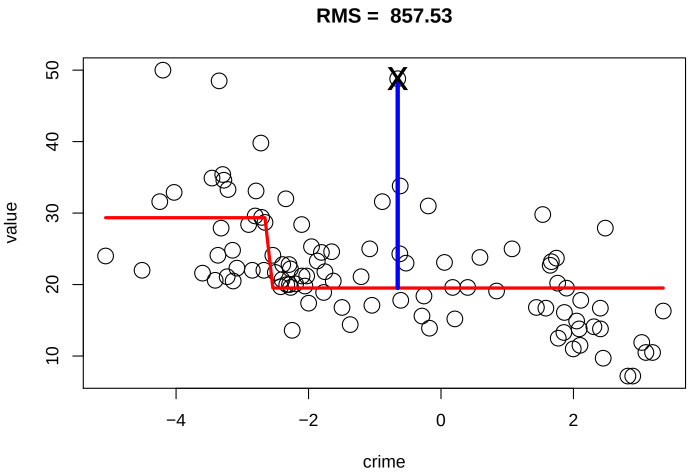
Example: Boston Housing Data

Again try to predict median house prices by using for simplicity just a single predictor variable, (logarithm of) crime rate.



Red line is fitted curve $\hat{Y}(x)$ for a tree of depth 1 (a stump). Blue vertical bar corresponds to residual of $i = 54$th observation with a squared residual of 590. Observation i was used to fit $\hat{Y}$ here !

Do the same fit but leave-out observation $i = 54$.



Red line is fitted curve $\hat{Y}^{(-54)}(x)$. Blue vertical bar corresponds to LOO-CV residual of $i = 54$th observation with a squared residual of 590. Observation $i = 54$ was now NOT used to fit $\hat{Y}^{(-54)}$ here!
Repeat for all $i = 1, \ldots, n$.

V-fold cross-validation

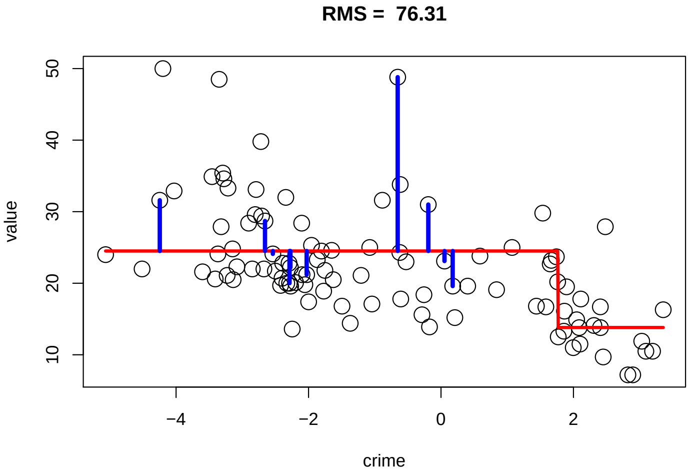
Is computationally cheaper than LOO-CV and yields comparable results.
V-fold cross-validation works by splitting the dataset randomly into V sets of equal size $S_1, \ldots, S_V$, so that $S_k \cap S_{k'} = \emptyset$ for all $k \neq k'$ and $\cup_k S_k = \{1, \ldots, n\}$.
For each $v = 1, \ldots, V$

- compute the predictor (tree) using samples $\{1, \ldots, n\} \setminus S_v$.
- predict the response for samples in set S_v with the found predictor
- record the test error for the set S_v.

Average the test error over all V sets.
Typical choices are $V = 5$ or $V = 10$.

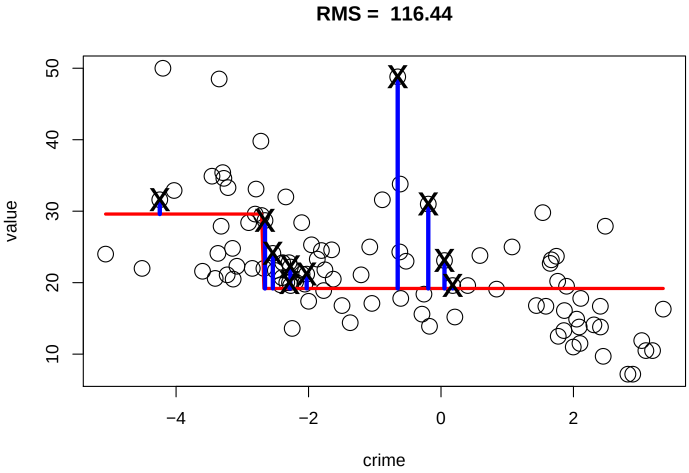
Example: Boston Housing Data

Assess now a whole block S_v of about $n/10$ of all n observations (V=10 fold CV).



Red line is fitted curve $\hat{Y}(x)$ for a tree of depth 1 (a stump). Blue vertical bar corresponds to residuals of ith observation, where i is in the to be assessed block v.

Do the same fit but leave-out observation the whole block of observations S_v.



Red line is fitted curve without using observations in block v. Blue vertical bar corresponds to LOO-CV residuals.
Repeat for all $V = 10$ blocks of all observations, each containing about $n/10$ of all samples.

Choosing the optimal tree

We would like to choose the tree that minimizes the true error rate. We dont have the test error, but can use CV-approximation $\hat{R}_{test}$ instead and choose the optimal tree T^* as

$$T^* = \text{argmin}_T \hat{R}_{test}(T).$$

This would require searching across all possible trees T and is clearly infeasible.
With CV, we can however search for the optimal value of one-dimensional so-called 'tuning' parameter. Here, we use tuning parameter α for tree pruning and find α by CV.

Pruning

Let $R_{train}(T)$ be the training error as a function of tree T (squared error on the training set for regression, mis-classification or entropy for classification). Minimizing $R_{train}(T)$ leads to a tree with maximal size. Minimize instead

$$(*) \qquad R_{train}(T) + \alpha \cdot \text{size}(T),$$

where the size of a tree T is measured by the number of leaf nodes.

- Either grow the tree from scratch and stop once the criterion $(*)$ starts to increase.
- Or first grow the full tree and start to delete nodes (starting at the leaf nodes), until the criterion $(*)$ starts to increase.

Second option is preferred as the choice of tree is less sensitive to "wrong" choices of splitpoints and variables to split on in the first stages of tree fitting.

Choice of α

Which value of α should be chosen ? Let T_α for $\alpha \in \mathbb{R}^+$ be the tree that is the minimizer of

$$T_\alpha = \text{argmin}_T \; \{R_{train}(T) + \alpha \cdot \text{size}(T)\}.$$

Want to pick α^* such that the resulting tree has minimal test error:

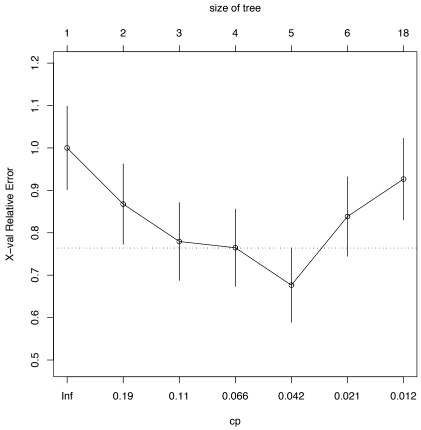
$$T_{\alpha^*} = \text{argmin}_{T_\alpha ; \alpha \in \mathbb{R}^+} \; \hat{R}_{test}(T_\alpha).$$

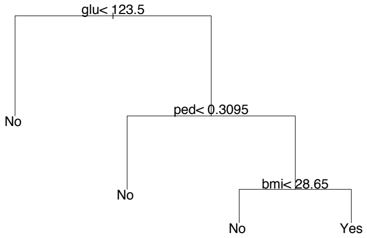
where we compute $\hat{R}_{test}$ using CV.
Its best to visualize $\hat{R}_{test}(T_\alpha)$ as a function of α.

Can plot the generalization error $\hat{R}_{test}$ of the optimal tree under criterion

$$R_{train}(T) + \alpha \cdot \text{size}(T)$$

as a function of α and pick the value of α which yields the smallest estimate of the generalization error.
For Pima Indians example:





Bias-Variance Tradeoff

Suppose

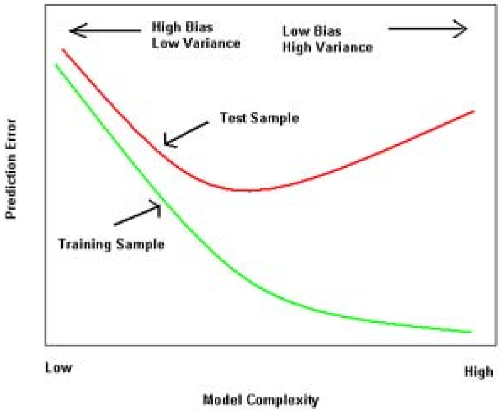
$$y = f^*(x) + \mathcal{N}(0, \sigma^2)$$

Given a dataset (X, Y), train a model $f(x; X, Y)$. How did we do, averaging over datasets?

$$\begin{aligned}
& E_{X,Y}[(y - f(x; X, Y))^2] & \\
= \; & (\bar{f}(x) - f^*(x))^2 & \text{bias}^2 \\
& + E_{X,Y}[(\bar{f}(x) - f(x; X, Y))^2] & \text{variance} \\
& + (y - f^*(x))^2 & \text{noise}
\end{aligned}$$

where $\bar{f}(x) = E_{X,Y}[f(x; X, Y)]$ is average prediction (averaged over datasets).

Choosing Model Complexity



The training error will always decrease if the model is made more complex (the tree grown larger). The test error will have reach a minimum at a certain model complexity (tree size) and grow if the tree is made either larger or smaller.

Pitfalls of using the training error rate

How deceptive can the training error be ?

- Assume we have n data samples $(X_1, Y_1), \ldots, (X_n, Y_n)$ and

$$Y_i \sim \mathcal{N}(0, 1)$$

  so there is no information about Y in the predictor variables X.
- Assume we take a tree with size d (the size is the number of leaf nodes), which is chosen independently of Y, so that each leaf node contains the same number of samples.

What is the expected training (apparent) error rate, as a function of tree size d?

Assume

- In total d leaf nodes.
- In each final leaf node, there are n/d samples $j_1, \ldots, j_{n/d}$.

The value of $\hat{\beta}_k$ in each leaf node k is simply the mean $\overline{Y}_k$ over all observations in node k.
The test error rate in each leaf node k is

$$R_{test} = E((Y - \overline{Y}_k)^2) = E((Y - E(Y))^2) + E((\overline{Y}_k - E(Y))^2) = 1 + \overline{Y}_k^2.$$

Averaged over independent realizations of the new test data, the expected test error rate is

$$E(R_{test}) = 1 + E(\overline{Y}_k^2)$$

The training error in each node is

$$R_{train} = \frac{d}{n} \sum_{j_1}^{j_{n/d}} (Y_i - \overline{Y}_k)^2 = \left(\frac{d}{n} \sum_{j_1}^{j_{n/d}} (Y_i - E(Y))^2 \right) \; - \; \overline{Y}_k^2.$$

The expected value of the training error rate is

$$E(R_{train}) = 1 - E(\overline{Y}_k^2).$$

The mean $\overline{Y}_k$ has a distribution $\sim \mathcal{N}(0, d/n)$. Then $\frac{n}{d}\overline{Y}_k^2 \sim \chi_1^2$ and $E(\overline{Y}_k^2) = d/n$. The expected value of the test error rate is thus

$$E(R_{test}) = 1 + d/n$$

The expected value of the training error rate is

$$E(R_{train}) = 1 - d/n$$

In this extreme example, choosing the number of leaf nodes according to the

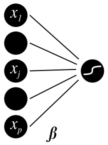
- training error rate leads you to choose maximal tree size $d = n$,
- test error rate leads you choose minimal tree size $d = 0$.

Outline

Neural Networks

The term *Neural Network* has evolved to encompass a large class of models and learning methods. We describe the most widely used neural network called the *single hidden layer back-propagation network*.
Initially motivated by biological processes, NN are simply another nonlinear method which can be used to find good predictions for classification and regression problems.

Each node of the network receives input from other nodes.

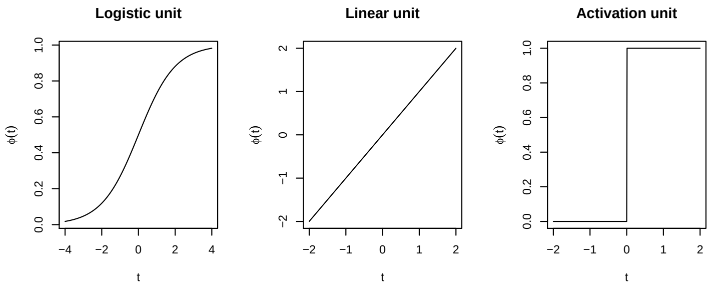


will pass a a signal along itself to other nodes in the network. For some so-called activation function f,

$$\text{total inputs to node } j: \qquad x_j = \sum_{i \to j} w_{ij} y_i$$

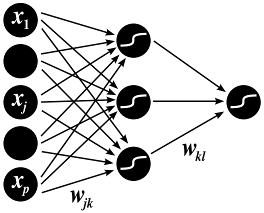
$$\text{output from node } j: \qquad y_j = f(x_i).$$

Many *activation functions* f are possible.



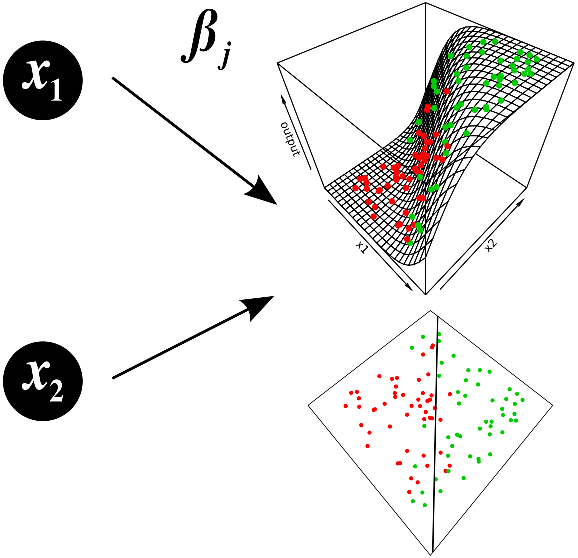
The ‘activation unit’ is not differentiable rendering it difficult to use for modelling. The linear unit is uninteresting in a network.

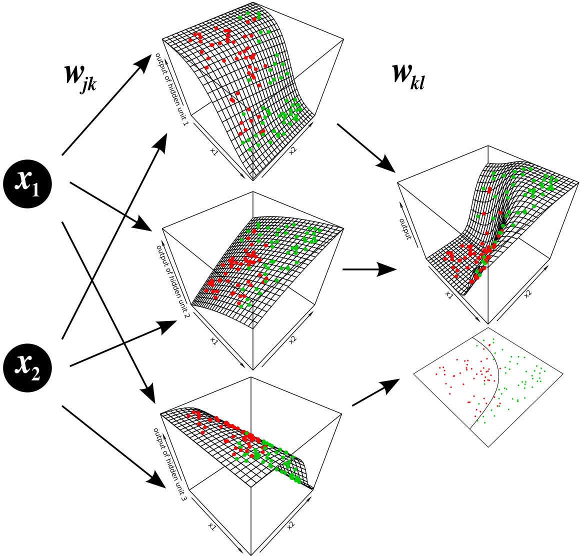
Neural networks combine many layers of nodes to allow for a very flexible model, a single hidden layer neural network in particular consists of 3 layers of nodes, below is an example of one.



A bias term is also incorporated at each neuron which provides a constant output of 1, so at node j

$$x_j = \sum_{i \to j} w_{ij} y_i = w_0 + \sum_{i \to j, i \neq 0} w_{ij} y_i.$$


β_j
x_1
x_2
output
x1
x2



w_{jk}
w_{kl}
x_1
x_2
output of hidden unit 1
output of hidden unit 2
output of hidden unit 3
output
x1
x2

There is nothing to stop us from adding more hidden layers.
It can be shown that single hidden layer neural networks can model arbitrarily complex surfaces (if the number of nodes in the hidden layer is large enough).
But more layers (“deep” networks) can sometimes model complex surfaces more easily.
When using linear activation units, the neural network collapses to a linear model. Logistic activation units are preferred as they are nonlinear and differentiable.

For inputs x_j, the output of the node l in the final layer can be expressed explicitly as

$$y_l = f\Big(\sum_{k \to l} w_{kl} f(\sum_{j \to k} w_{jk} x_j) \Big) \quad =: y_l(x, w)$$

for a neural network with a single hidden layer.
The statistical part is to find good weights w, given some training data.

Neural Networks can be used for both regression and classification. In the regression framework of Rumelhart and McClelland, for observations (X_i, Y_i), $i = 1, \ldots, n$, we train neural networks with a single output node y_1 by seeking weights w to minimize

$$E(w) = \sum_{i=1}^{n} |Y_i - y_1(X_i, w)|^2.$$

A generic drawback of Neural Networks is that $E(w)$ is a non-convex function and can have multiple minima. It is thus not easy to find a good solution.

In a classification setting, we now have K output nodes $y_1, \ldots, y_K$, each representing one of the classes. Let $Y_{i,k} := 1\{Y_i = k\}$ for $k = 1, \ldots, K$.
By augmenting the final outputs, it is straightforward to enforce that each output node returns probability predictions $p_{i,k} := P(Y = k|X_i)$ via the *softmax* transformation

$$p_{i,k} = \frac{\exp y_k(X_i)}{\sum_{l=1}^{K} \exp y_l(X_i)}.$$

A measure of fit is via likelihoods, using

$$L(w) \propto \prod_{i=1}^{n} \prod_{\text{outputs } k} (p_{i,k})^{Y_{i,k}}.$$

We can seek weights to maximise the log-likelihood

$$\ell(w) \propto \sum_{i=1}^{n} \sum_{k} Y_{i,k} \log p_{i,k}.$$

As the log-likelihood attains a maximum at

$$\sum_i \sum_k Y_{i,k} \log Y_{i,k},$$

it is conventional to consider finding weights to minimize the cross-entropy

$$E(w) = \sum_p \sum_k Y_{i,k} \log \frac{Y_{i,k}}{p_{i,k}} = \sum_p \sum_k Y_{i,k} \log Y_{i,k} - \sum_p \sum_k Y_{i,k} \log p_{i,k},$$

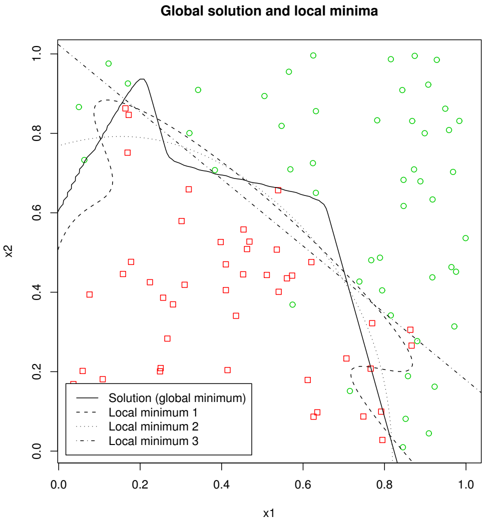
so $E(w) \geq 0$ with equality iff we can find w so that $p_{i,k} = Y_{i,k}$, i.e. a perfect fit (on the training data).
For both criteria, E can be minimised via gradient descent with update rule

$$w_{ij} \leftarrow w_{ij} - \eta \frac{\partial E}{\partial w_{ij}}.$$

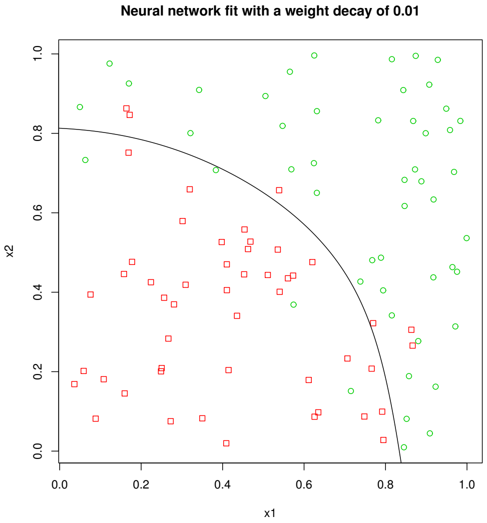
Corresponding algorithm called *back-propagation*.

- Due to the modular nature of the nodes, partial derivatives are easily calculated via the chain rule, which leads to an algorithm called *back-propagation*.
- In back-propagation, predictions are made in a “forward-pass” through the network, while derivatives are computed in a “backward-pass” propagating error information towards earlier layers of network.
- As the high-dimensional likelihood surface need not be convex, we often find suboptimal maxima.
- With large numbers of nodes in the network, we have to be careful not to overfit. Regularisation is obtained bys a combination of
  - not fitting until convergence
  - Using weight decay, a regularization penalty on the weights w.
  - Choosing a simple but suitable network structure.

Neural Networks



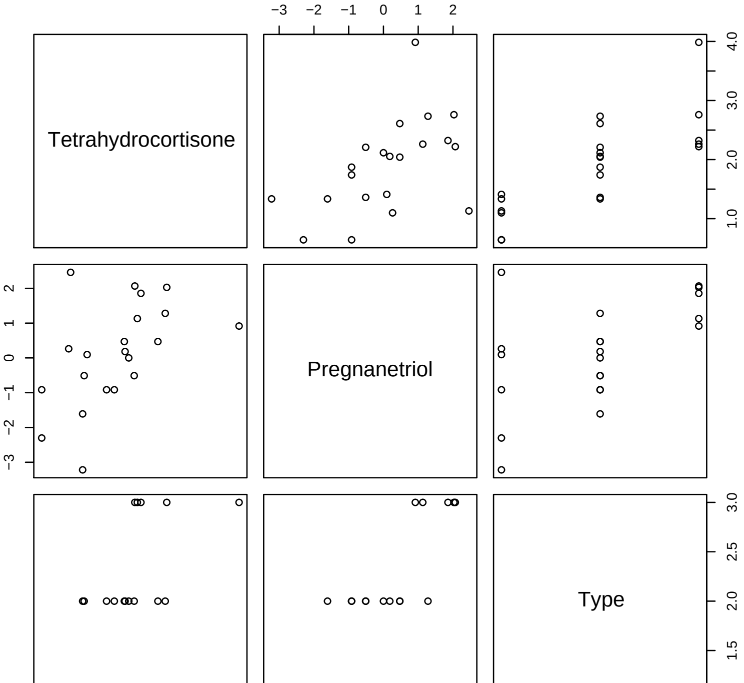
Neural Networks



Cushings data (load with `data(Cushings)` in package `MASS`).

```
> ?Cushings
Cushings                package:MASS                R Documentation

Diagnostic Tests on Patients with Cushing’s Syndrome

Description:
     Cushing’s syndrome is a hypertensive disorder associated with
     over-secretion of cortisol by the adrenal gland. The observations
     are urinary excretion rates of two steroid metabolites.

Format:
     The ’Cushings’ data frame has 27 rows and 3 columns:

     ’Tetrahydrocortisone’ urinary excretion rate (mg/24hr) of
          Tetrahydrocortisone.

     ’Pregnanetriol’ urinary excretion rate (mg/24hr) of
          Pregnanetriol.

     ’Type’ underlying type of syndrome, coded ’a’ (adenoma) , ’b’
          (bilateral hyperplasia), ’c’ (carcinoma) or ’u’ for unknown.
```

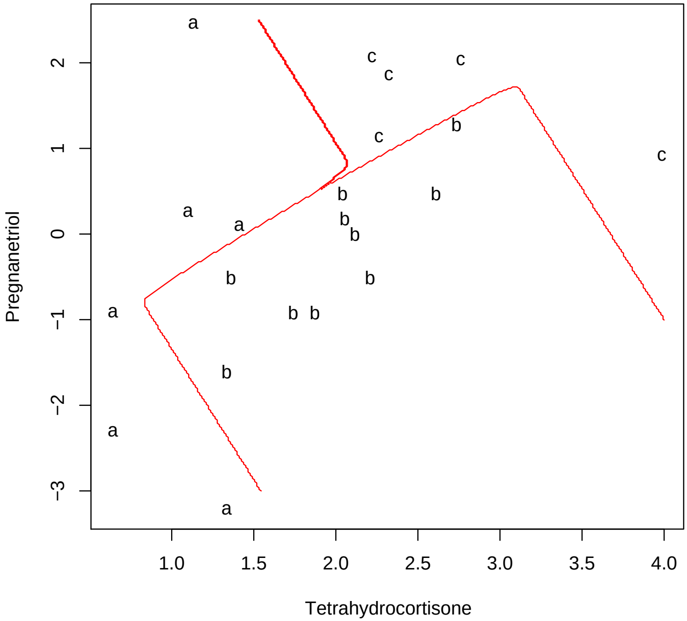
```r
## we will not deal with the untyped data here
cush <- Cushings[Cushings$Type!="u",]
cush[,1:2] <- log(cush[,1:2])

## plot
pairs(cush)


## fit neural network with 5 nodes in the hidden layer
cush.nnet <- nnet(Type ~ . , data=cush, size=5)
```



Tetrahydrocortisone
Pregnanetriol
Type
−3
−2
−1
0
1
2
1.0
2.0
3.0
4.0
1.5
2.0
2.5
3.0

Display the decision boundaries.

```
## take a lattice of points
## 100 by 100 lattice
m <- 100
x <- seq(0,4,length.out=m)
y <- seq(-3,2.5,length.out=m)
z <- data.frame(expand.grid(
             Tetrahydrocortisone=x,
             Pregnanetriol=y))
cush.nnp <- predict(cush.nnet,z)

## plot the data and decision boundaries
## classes are a,b,c =1,2,3 so set contours at 1.5 and 2.5
plot(cush[,1:2], pch=as.character(cush$Type))
contour(x, y, matrix(max.col(cush.nnp),m,m), levels=c(1.5,2
        add=TRUE, d=FALSE, lty=1, col=2)
```


Pregnanetriol
Tetrahydrocortisone
1.0
1.5
2.0
2.5
3.0
3.5
4.0
−3
−2
−1
0
1
2
a
b
c