MS1b Statistical Data Mining
Part 2: Supervised Learning
Parametric Methods

Yee Whye Teh
Department of Statistics
Oxford

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

Outline

Supervised Learning

So far we have been interested in using EDA and clustering techniques to understand high-dimensional data, useful for hypothesis generation. If a response (or grouping) variable occured in examples, it was merely to ‘validate’ that the discovered clusters or projections are meaningful.
We now move to supervised learning where in addition to having n observations of a p-dimensional predictor variable X, we also have a response variable $Y \in \mathcal{Y}$.

- Classification: group information is given and $\mathcal{Y} = \{1, \dots, K\}$.
- Regression: a numerical value is observed and $\mathcal{Y} = \mathbb{R}$.

Given training data (X_i, Y_i), $i = 1, \dots, n$, the goal is to accurately predict the class or response Y of new observations, when only the predictor variables X are observed.

Outline

Regression example: Boston Housing Data

The original data are 506 observations on 13 variables X; medv being the response variable Y.

crim	per capita crime rate by town
zn	proportion of residential land zoned for lots over 25,000 sq.ft
indus	proportion of non-retail business acres per town
chas	Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)
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 centers
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

```
> str(X)
'data.frame': 506 obs. of  13 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 ...

> str(Y)
num[1:506] 24 21.6 34.7 33.4 36.2 28.7 22.9 27.1 16.5 18.9 ...
```

Goal: predict median house price $\hat{Y}(X)$, given 13 predictor variables X of a new district.

Classification example: Lymphoma data

Revisiting the lymphoma gene expression data. Now in the supervised setting.
We have gene expression measurements of $n = 62$ patients for $p = 4026$ genes. These form the predictor variable matrix X.
For each patient, the subtype of cancer is available in a n dimensional vector Y with entries in $\{0, 1\}$.

```
> str(X)
'data.frame': 62 obs. of  4026 variables:
 $ Gene 1   : num  -0.344 -1.188  0.520 -0.748 -0.868 ...
 $ Gene 2   : num  -0.953 -1.286  0.657 -1.328 -1.330 ...
 $ Gene 3   : num  -0.776 -0.588  0.409 -0.991 -1.517 ...
 $ Gene 4   : num  -0.474 -1.588  0.219  0.978 -1.604 ...
 $ Gene 5   : num  -1.896 -1.960 -1.695 -0.348 -0.595 ...
 $ Gene 6   : num  -2.075 -2.117  0.121 -0.800  0.651 ...
 $ Gene 7   : num  -1.8755 -1.8187  0.3175  0.3873  0.0414 ...
 $ Gene 8   : num  -1.539 -2.433 -0.337 -0.522 -0.668 ...
 $ Gene 9   : num  -0.604 -0.710 -1.269 -0.832  0.458 ...
 $ Gene 10  : num  -0.218 -0.487 -1.203 -0.919 -0.848 ...
 $ Gene 11  : num  -0.340  1.164  1.023  1.133 -0.541 ...
 $ Gene 12  : num  -0.531  0.488 -0.335  0.496 -0.358 ...
 $ Gene 13  : num   0.0846  0.4820  1.5254  0.0323 -0.7563 ...
 $ Gene 14  : num  -1.2011 -0.0505 -0.8799  0.7518 -0.9964 ...
 $ Gene 15  : num  -0.9588 -0.0554 -1.0008  0.2502 -1.0235 ...

> str(Y)
 num [1:62] 0 0 0 1 0 0 1 0 0 0 ...
```

Goal: predict ‘cancer class’ $\hat{Y}(X) \in \{0, 1\}$, given 4026 predictor variables X (gene expressions) of a new patient.

Loss

Suppose we have trained a classifier or learner so that, upon observing a new predictor variable $X \in \mathbb{R}^p$, a prediction $\hat{Y} \in \mathcal{Y}$ is made.
How good is the prediction? We can use any loss function $L : \mathcal{Y} \times \mathcal{Y} \mapsto \mathbb{R}^+$ to measure the loss incurred. Typical loss functions

- Misclassification error for classification

$$L(Y, \hat{Y}) = \begin{cases} 0 & Y = \hat{Y} \\ 1 & Y \neq \hat{Y} \end{cases} .$$

- Squared error loss for regression

$$L(Y, \hat{Y}) = (Y - \hat{Y})^2.$$

Alternative loss functions often useful. For example, non-equal misclassification error often appropriate. Or ‘likelihood’-loss $L(Y, \hat{Y}) = -\log \hat{p}(Y)$, where $\hat{p}(k)$ is the estimated probability of class $k \in \mathcal{Y}$.

Risk and empirical risk minimization

For a given loss function L, the risk R of a learner is given by the expected loss

$$R(\hat{Y}) = \mathbb{E}(L(Y, \hat{Y})),$$

where $\hat{Y} = \hat{Y}(X)$ is a function of the random predictor variable X.
Ideally, we want to find a learner or procedure that minimizes the risk. The risk is unknown, however, as we just have finitely many samples.
Empirical risk minimization can be used, where one is trying to minimize –instead of the risk $R(\hat{Y})$– the empirical risk

$$R_n(\hat{Y}) = \mathbb{E}_n(L(Y, \hat{Y}) = \frac{1}{n}\sum_{i=1}^{n} L(Y_i, \hat{Y}_i).$$

The expectation is with respect to the empirical measure and hence just a summation over the observations.

The Bayes classifier

What is the optimal classifier if the joint distribution (X, Y) were known?
The distribution f of a random predictor variable X can be written as

$$f(X) = \sum_{k=1}^{K} f_k(X) P(Y = k),$$

where, for $k = 1, \ldots, K$,

- the prior probabilities over classes are $P(Y = k) = \pi_k$
- and distributions of X, conditional on $Y = k$, is $f_k(X)$.

Given this scenario, the problem is to construct a ‘good’ classifier $\hat{Y}$ which assigns classes to observations

$$\hat{Y} : \mathcal{X} \to \left\{1, \ldots, K\right\}$$

We are interested in finding the classifier $\hat{Y}$ that minimises the risk under 0-1 loss, the *Bayes Classifier*.

$$\begin{aligned} R(\hat{Y}) &= \mathbb{E}\Big[L(Y, \hat{Y}(X))\Big] \\ &= \mathbb{E}\Big[\mathbb{E}[L(Y, \hat{Y}(x)\Big|X = x]\Big] \\ &= \int_{\mathcal{X}} \mathbb{E}\Big[L(Y, \hat{Y}(x))\Big|X = x\Big] f(x) dx \end{aligned}$$

For the Bayes classifier, minimizing $\mathbb{E}\Big[L(Y, \hat{Y}(x))\Big|X = x\Big]$ for each x suffices.

That is, given $X = x$, want to choose $\hat{Y}(x) \in \{1, \ldots, K\}$ such that the expected conditional loss is as small as possible.

Can write $\mathbb{E}\Big[L(Y,\hat{Y}(x))\Big|X=x\Big] \;=\; \sum_{k=1}^{K} L(k,\hat{Y}(x))P(Y=k|X=x)$.
Choosing $\hat{Y}(x)=m$ with $m\in\{1,\ldots,K\}$, the r.h.s. is simply

$$\mathbb{E}\Big[L(Y,\hat{Y}(x))|X=x\Big] = 1 - P(Y=m|X=x).$$

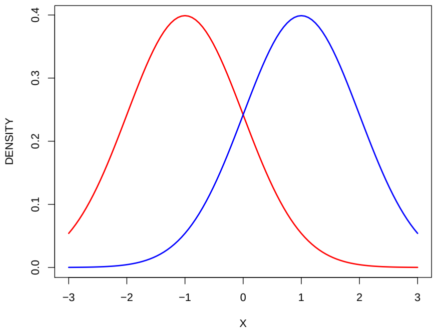
The Bayes Classifier chooses the class with the greatest posterior probability

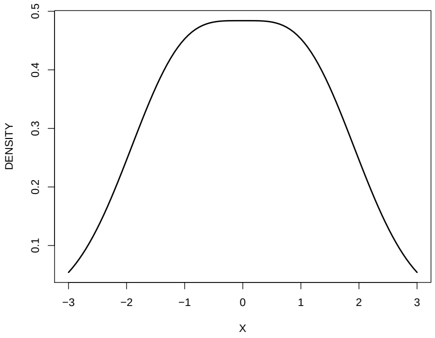
$$\begin{aligned}\hat{Y}(x) &= \operatorname*{arg\,max}_{k=1,\ldots,K} P(Y=k|X=x) = \operatorname*{arg\,max}_{k=1,\ldots,K} \frac{\pi_k f_k(x)}{\sum_{k=1}^{K}\pi_k f_k(x)} \\ &= \operatorname*{arg\,max}_{k=1,\ldots,K}\; \pi_k f_k(x).\end{aligned}$$

The Bayes classifier is optimal in terms of misclassification error.

Take a simple example, where π_k and f_k *are* known for $k = 1, \ldots, K$. Choose two classes $\{1, 2\}$.
Suppose $X \sim \mathcal{N}(\mu_Y, 1)$, where $\mu_1 = -1$ and $\mu_2 = 1$ and assume equal priors $\pi_1 = \pi_2 = 1/2$.
So $f(x) = \frac{1}{2}f_1(x) + \frac{1}{2}f_2(x)$, where

$$f_1(x) = \frac{1}{\sqrt{2\pi}} \exp(-\frac{(x-(-1))^2}{2}) \quad \text{and} \quad f_2(x) = \frac{1}{\sqrt{2\pi}} \exp(-\frac{(x-1)^2}{2}).$$

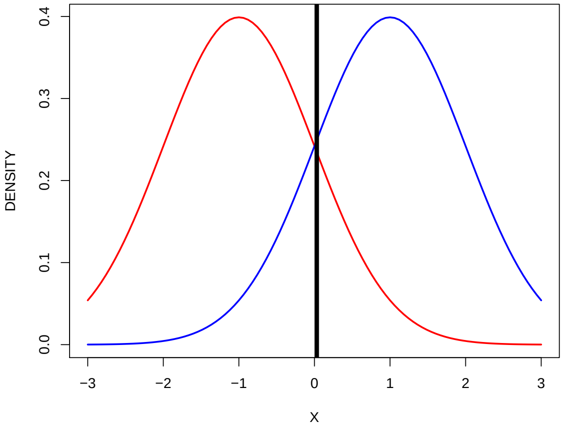




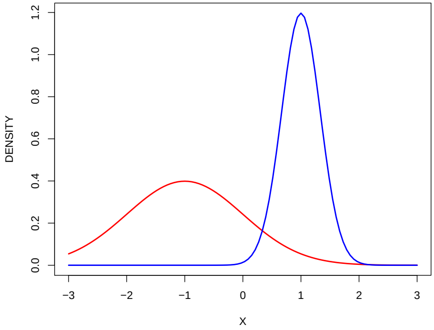
How do you classify a new observation $x = 0.1$?
Optimal classification is

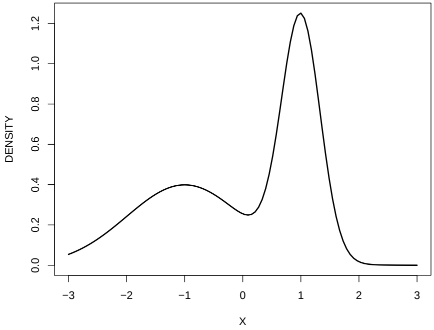
$$\hat{Y}(x) = \underset{k=1,\ldots,K}{\arg\max}\ \pi_k f_k(x),$$

which is class 1 if $x < 0$ and class 2 if $x \geq 0$.

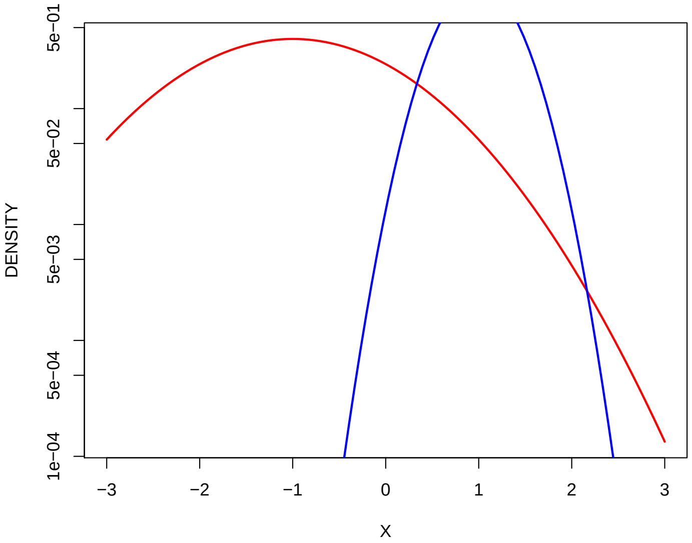


How do you classify a new observation x if now the standard deviation is still 1 for class 1 but 1/3 for class 2 ?





Looking at density in a log-scale, optimal classification is class 2 if and only if $x \in [-0.39, 2.15]$.



Plug-in classification

The Bayes Classifier chooses the class with the greatest posterior probability

$$\hat{Y}(x) \quad = \quad \underset{k=1,\dots,K}{\arg\max}\, \pi_k f_k(x).$$

Unfortunately, we usually know neither the conditional class probabilities nor the prior probabilities.
Given

- estimates $\hat{\pi}_k$ for π_k and $k = 1, \dots, K$ and
- estimates $\hat{f}_k(x)$ of conditional class probabilities,

the *plug-in classifiers* chooses the class

$$\hat{Y}(x) = \underset{k=1,\dots,K}{\arg\max}\, \hat{\pi}_k \hat{f}_k(x).$$

Linear Discriminant Analysis will be an example of plug-in classification.

Outline

Linear Discriminant Analysis

LDA is the most well-known and simplest example of plug-in classification.
Assume a parametric form for $f_k(x)$ where for each class k, the distribution of X, conditional on $Y = k$, is

$$X|Y = k \sim \mathcal{N}(\mu_k, \Sigma),$$

i.e. classes have different means with the *same* covariance matrix Σ.
For a new observation x,

$$\begin{aligned} P(Y = k|X = x) &\propto \pi_k f_k(x) \\ &\propto \frac{\pi_k}{|\Sigma|^{1/2}} \exp\left\{-\frac{1}{2}(x - \mu_k)^T \Sigma^{-1} (x - \mu_k)\right\} \end{aligned}$$

As $\arg\max_{k=1,\ldots,K} g(k) = \arg\min_{k=1,\ldots,K} -2\log g(k)$ for any real-valued function g, choose k to minimize

$$-2\log P(Y=k|X=x) \quad \propto \quad (x-\mu_k)^T\Sigma^{-1}(x-\mu_k) - 2\log(\pi_k) + \text{const.}$$

where the constant does not depend on the class k.
The quantity $(x-\mu_k)^T\Sigma^{-1}(x-\mu_k)$ is called the Malahanobis distance. It measures the distance between x and μ_k in the metric given by Σ.
Notice that if $\Sigma = I_p$ and $\pi_k = \frac{1}{K}$, $\hat{Y}(x)$ simply chooses the class k with the nearest (in the Euclidean sense) mean μ_k.

Expanding the discriminant $(x - \mu_k)^T \Sigma^{-1} (x - \mu_k)$, the term $-2 \log P(Y = k | X = x)$ is seen to be proportional to

$$\begin{aligned} & \mu_k^T \Sigma^{-1} \mu_k - 2\mu_k^T \Sigma^{-1} x + x^T \Sigma^{-1} x - 2\log(\pi_k) + \text{const} \\ & = \mu_k^T \Sigma^{-1} \mu_k - 2\mu_k^T \Sigma^{-1} x - 2\log(\pi_k) + \text{const}, \end{aligned}$$

where the constant does not depend on the class k.
Setting $a_k = \mu_k^T \Sigma^{-1} \mu_k - 2\log(\pi_k)$ and $b_k = -2\Sigma^{-1}\mu_k$, we obtain

$$-2 \log P(Y = k | X = x) = a_k + b_k^T x + \text{const}$$

i.e. a *linear discriminant function*.
Considering when we choose class k over k',

$$\begin{aligned} a_k + b_k^T x + \text{const}(x) \quad & < \quad a_{k'} + b_{k'}^T x + \text{const} \\ \Leftrightarrow a_\star + b_\star^T x \quad & < \quad 0 \end{aligned}$$

where $a_\star = a_k - a_{k'}$ and $b_\star = b_k - b_{k'}$.
Shows that the Bayes Classifier partitions $\mathcal{X}$ into regions with the same class predictions via *separating hyperplanes*. The Bayes Classifier under these assumptions is more commonly known as the *Linear Discriminant Analysis Classifier*.

Parameter Estimation and 'Plug-In' Classifiers

Remember that upon assuming a parametric form for the $f_k(x)$'s, the optimal classification procedure under 0-1 loss is

$$\hat{Y}(x) = \underset{k=1,\ldots,K}{\arg\max} \quad \pi_k f_k(x)$$

LDA proposes multivariate normal distributions for $f_k(x)$.
However, we still don't know what the parameters μ_k, $k = 1, \ldots, K$ and Σ that determine f_k. The statistical task becomes one of finding good estimates for these quantities and plugging them into the derived equations to give the *'Plug-In' Classifier*

$$\hat{Y}(x) = \underset{k=1,\ldots,K}{\arg\max} \quad \hat{\pi}_k \hat{f}_k(x).$$

The a priori probabilities $\pi_k = P(Y = k)$ are simply estimated by the empirical proportion of samples of class k, $\hat{\pi}_k = |\{i : Y_i = k\}|/n$.

For estimation of Σ and μ, looking at the log-likelihood of the training set,

$$\begin{aligned}\ell(\mu_1, \ldots, \mu_K) &= -\sum_{k=1}^{K} \sum_{j:Y_j=k} \frac{1}{2}(X_j - \mu_k)^T \Sigma^{-1} (X_j - \mu_k) \\ &\quad -\frac{1}{2} n \log |\Sigma| + \text{const.}\end{aligned}$$

Let $n_k = \#\{j : Y_j = k\}$ be the number of observations in class k. The log-likelihood is maximised by

$$\hat{\mu}_k = \frac{1}{n_k} \sum_{j:Y_j=k} X_j, \qquad \hat{\Sigma} = \frac{1}{n} \sum_{k=1}^{K} \sum_{j:Y_j=k} (X_j - \hat{\mu}_k)(X_j - \hat{\mu}_k)^T.$$

The best classifier under the assumption that $X|Y = k \sim \mathcal{N}_p(\hat{\mu}_k, \hat{\Sigma})$ with plug-in estimates of μ and Σ is therefore given by

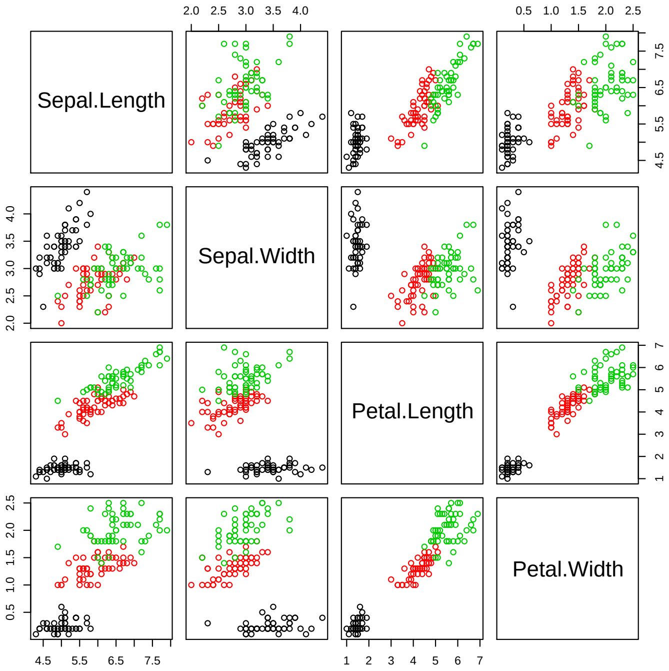
$$\hat{Y}_{lda}(x) = \underset{k=1,\dots,K}{\arg\min} \left\{ (x - \hat{\mu}_k)^T \hat{\Sigma}^{-1} (x - \hat{\mu}_k) - 2 \log(\hat{\pi}_k) \right\}$$

for each point $x \in \mathcal{X}$.
Can also be written as

$$\hat{Y}_{lda}(x) = \underset{k=1,\dots,K}{\arg\min} \left\{ \hat{\mu}_k^T \hat{\Sigma}^{-1} \hat{\mu}_k - 2\hat{\mu}_k^T \hat{\Sigma}^{-1} x - 2 \log(\hat{\pi}_k) \right\}.$$

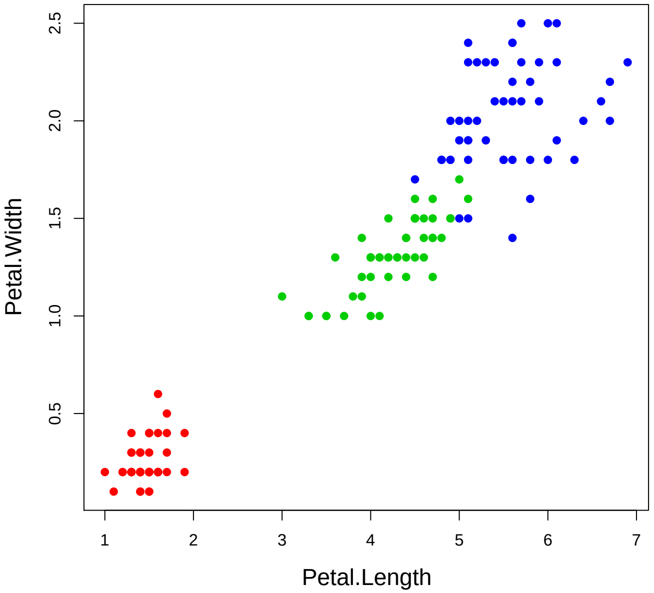
Iris example

```
library(MASS)
data(iris)

##save class labels
ct <- rep(1:3,each=50)
##pairwise plot
pairs(iris[,1:4],col=ct)

##save petal.length and petal.width
iris.data <- iris[,3:4]
plot(iris.data,col=ct+1,pch=20,cex=1.5,cex.lab=1.4)
```



Sepal.Length
Sepal.Width
Petal.Length
Petal.Width
2.0
2.5
3.0
3.5
4.0
0.5
1.0
1.5
2.0
2.5
7.5
6.5
5.5
4.5
4.0
3.5
3.0
2.5
2.0
7
6
5
4
3
2
1
2.5
2.0
1.5
1.0
0.5
4.5
5.5
6.5
7.5
1
2
3
4
5
6
7

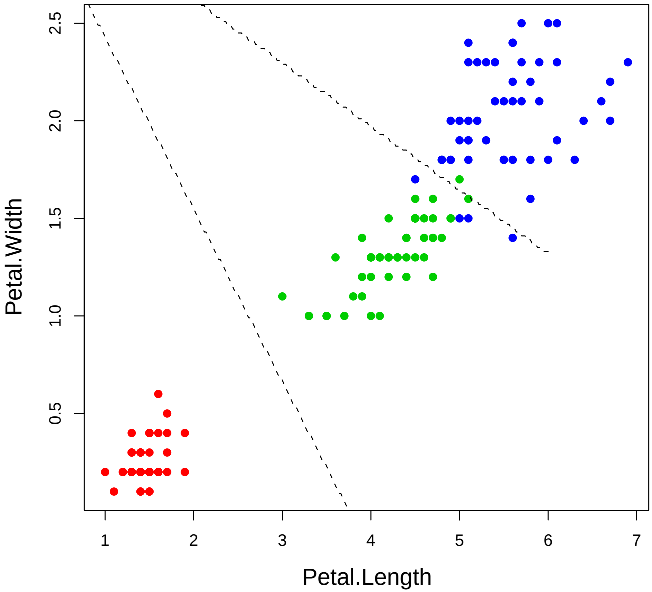
Just focus on two predictor variables.



Computing and plotting the LDA boundaries.

```
##fit LDA
iris.lda <- lda(x=iris.data,grouping=ct)

##create a grid for our plotting surface
x <- seq(-6,6,0.02)
y <- seq(-4,4,0.02)
z <- as.matrix(expand.grid(x,y),0)
m <- length(x)
n <- length(y)

##classes are 1,2 and 3, so set contours at 1.5 and 2.5
iris.ldp <- predict(iris.lda,z)$class
contour(x,y,matrix(iris.ldp,m,n),
           levels=c(1.5,2.5), add=TRUE, d=FALSE, lty=2)
```

LDA boundaries.



Fishers Linear Discriminant Analysis

We have derived LDA as the plug-in Bayes classifier under the assumption of multivariate normality for all classes with common covariance matrix.
Alternative view (without making any assumption on underlying densities):
Find a direction $a \in \mathbb{R}^p$ to maximize the variance ratio

$$\frac{a^T B a}{a^T \Sigma a},$$

where

$$\Sigma = \frac{1}{n-1} \sum_{i=1}^{n} (X_i - \mu_{Y_i})(X_i - \mu_{Y_i})^\top \qquad \text{(within class covariance)}$$

$$B = \frac{1}{n-1} \sum_{k=1}^{K} n_k (\mu_{Y_i} - \bar{X})(\mu_{Y_i} - \bar{X}))^\top \qquad \text{(between class covariance)}$$

B has rank at most $K-1$.

Discriminant Coordinates

The variance ratio satisfies

$$\frac{a^T B a}{a^T \Sigma a} = \frac{b^T (\Sigma^{-\frac{1}{2}})^T B \Sigma^{-\frac{1}{2}} b}{b^T b},$$

where $b = \Sigma^{\frac{1}{2}} a$ and $B^* = (\Sigma^{-\frac{1}{2}})^T B \Sigma^{-\frac{1}{2}}$.
The maximization over b is achieved by the first eigenvector v_1 of B^*. We also look at the remaining eigenvectors v_l associated to the non-zero eigenvalues and defined the discriminant coordinates as $a_l = \Sigma^{-\frac{1}{2}} v_l$.
These directions a_l span exactly the space of all linear discriminant functions for all pairwise comparisons and are often used for plotting (ie in the function `lda`).
Data are then projected onto these directions (these vectors are given as the “linear discriminant” functions in the R-function `lda`).

Crabs data example

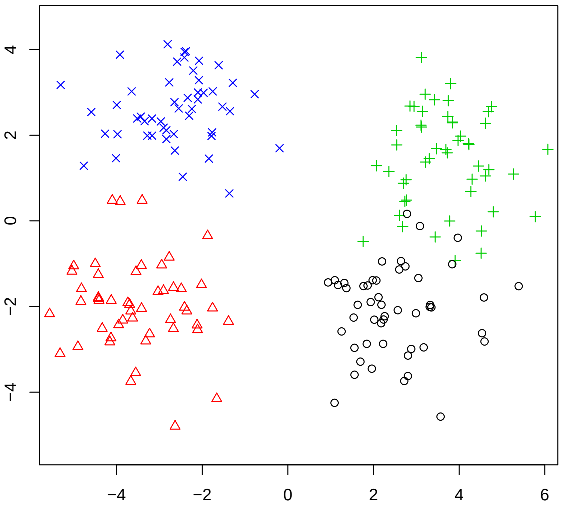
Crabs data, again.

```
library(MASS)
data(crabs)

## numeric and text class labels
ct <- as.numeric(crabs[,1])-1+2*(as.numeric(crabs[,2])-1)

## Projection on Fisher's linear discriminant directions
print(cb.lda <- lda(log(crabs[,4:8]),ct))
```

```
> > > > > > > > > Call:
lda(log(crabs[, 4:8]), ct)

Prior probabilities of groups:
   0    1    2    3 
0.25 0.25 0.25 0.25 

Group means:
        FL       RW       CL       CW       BD
0 2.564985 2.475174 3.312685 3.462327 2.441351
1 2.852455 2.683831 3.529370 3.649555 2.733273
2 2.672724 2.443774 3.437968 3.578077 2.560806
3 2.787885 2.489921 3.490431 3.589426 2.701580

Coefficients of linear discriminants:
          LD1        LD2        LD3
FL -31.217207  -2.851488  25.719750
RW  -9.485303 -24.652581  -6.067361
CL  -9.822169  38.578804 -31.679288
CW  65.950295 -21.375951  30.600428
BD -17.998493   6.002432 -14.541487

Proportion of trace:
   LD1    LD2    LD3 
0.6891 0.3018 0.0091 
```

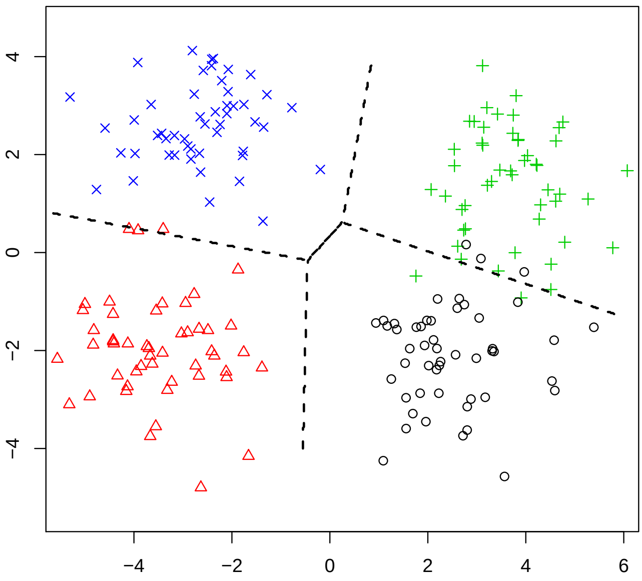
Plot predictions

```
cb.ldp <- predict(cb.lda)
eqscplot(cb.ldp$x,pch=ct+1,col=ct+1)
```



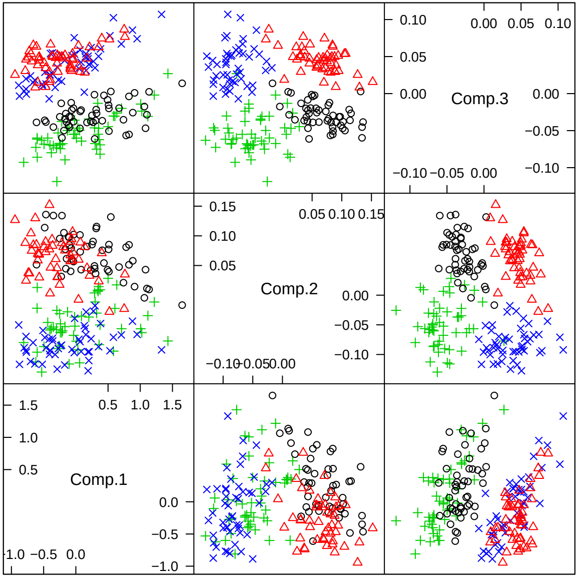
```
> ct
  [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [38] 2 2 2 2 2 2 2 2 2 2 2 2 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 [75] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 3 3 3 3 3 3 3 3
[112] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
[149] 3 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[186] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

> predict(cb.lda)
$class
  [1] 2 2 2 2 2 2 0 2 2 0 2 0 2 2 2 0 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [38] 2 2 2 2 2 2 2 2 2 2 2 2 2 0 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 [75] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 3 3 3 3 3 3 3 3
[112] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
[149] 3 3 1 3 3 1 1 1 1 1 1 1 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[186] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
Levels: 0 1 2 3

$posterior
               0            1            2            3
1   4.058456e-02 1.579991e-10 9.594150e-01 4.367517e-07
2   4.912087e-01 2.057493e-09 5.087911e-01 2.314634e-07
3   2.001047e-02 4.368642e-16 9.799895e-01 2.087757e-13
4   7.867144e-04 9.148327e-15 9.992133e-01 2.087350e-09
5   2.094626e-03 2.381970e-11 9.979020e-01 3.335500e-06
6   3.740294e-03 3.170411e-13 9.962597e-01 2.545022e-08
7   7.291360e-01 1.625743e-09 2.708639e-01 6.637005e-08
```

```
## display the decision boundaries
## take a lattice of points in LD-space
x <- seq(-6,6,0.02)
y <- seq(-4,4,0.02)
z <- as.matrix(expand.grid(x,y,0))
m <- length(x)
n <- length(y)


## predict onto the grid
cb.ldap <- lda(cb.ldp$x,ct)
cb.ldpp <- predict(cb.ldap,z)$class

## classes are 0,1,2 and 3 so set contours
## at 0.5,1.5 and 2.5
contour(x,y,matrix(cb.ldpp,m,n),
           levels=c(0.5,2.5),
           add=TRUE,d=FALSE,lty=2,lwd=2)
```

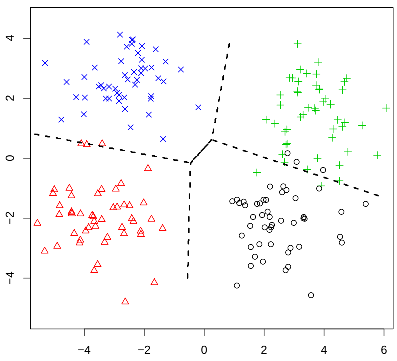

4
2
0
−2
−4
−4
−2
0
2
4
6

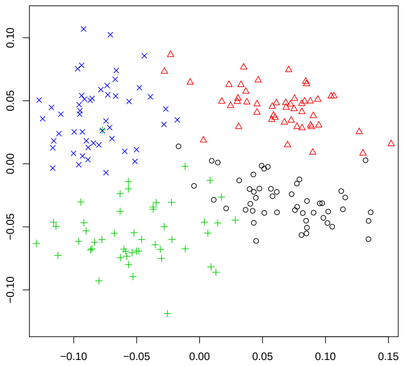
Compare with PCA plots.

```
library(lattice)
cb.pca <- princomp(log(crabs[,4:8]))
cb.pcp <- predict(cb.pca)
splom(~cb.pcp[,1:3],pch=ct+1,col=ct+1)
```



Scatter Plot Matrix





LDA separates the groups better.

Outline

Given training data with K classes, assume a parametric form for $f_k(x)$, where for each class

$$X|Y=k \;\sim\; (\mu_k, \Sigma_k),$$

i.e. instead of assuming that every class has a different mean μ_k with the *same* covariance matrix Σ, we now allow each class to have its own covariance matrix.
Considering $-2\log P(Y=k|X=x)$ as before,

$$\begin{aligned}
-2\log P(Y=k|X=x) &\propto (x-\mu_k)^T\Sigma_k^{-1}(x-\mu_k) - 2\log(\pi_k) + \text{const}_k \\
&= \mu_k^T\Sigma_k^{-1}\mu_k - 2\mu_k^T\Sigma_k^{-1}x + x^T\Sigma_k^{-1}x \\
&\qquad -2\log(\pi_k) + \text{const}_k \\
&= a_k + b_k^T x + x^T c_k x
\end{aligned}$$

i.e. we find a *quadratic* function instead (the function const_k includes the term $\log(|\Sigma_k|)$

Again, by considering when we choose class k over k',

$$\begin{aligned} 0 &> a_k + b_k^T x + x^T c_k x - (a_{k'} + b_{k'}^T x + x^T c_{k'} x) \\ &= a_\star + b_\star^T x + x^T c_\star x \end{aligned}$$

we see that the Bayes Classifier partitions $\{x : \hat{Y}(x) = k\}$ are using quadratic surfaces.
The Bayes Classifer under these assumptions is more commonly known as the *Quadratic Discriminant Analysis* Classifier.

The exact form of the QDA classifier is given by

$$\hat{Y}_{qda}(x) = \underset{k=1,\ldots,K}{\arg\min} \left\{ (x - \hat{\mu}_k)^T \hat{\Sigma}_k{}^{-1} (x - \hat{\mu}_k) - 2\log(\hat{\pi}_k) + \log(|\hat{\Sigma}_k|) \right\}$$

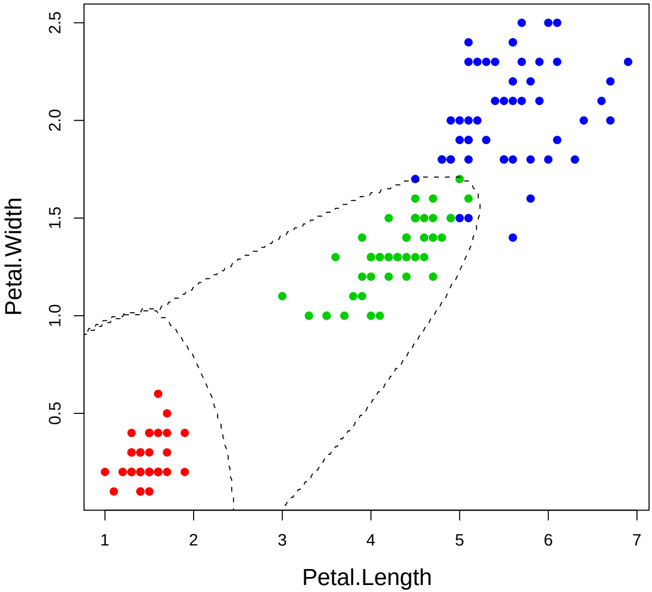
for each point $x \in \mathcal{X}$ where the plug-in estimate $\hat{\mu}_k$ is as before and $\hat{\Sigma}_k$ is (in contrast to LDA) estimated for each class $k = 1, \ldots, K$ separately:

$$\hat{\Sigma}_k = \frac{1}{n_k} \sum_{j:Y_j=k} (X_j - \hat{\mu}_k)(X_j - \hat{\mu}_k)^T.$$

Computing and plotting the QDA (and LDA) boundaries.

```
##fit LDA
iris.lda <- lda(x=iris.data,grouping=ct)
iris.qda <- qda(x=iris.data,grouping=ct)

##create a grid for our plotting surface
x <- seq(-6,6,0.02)
y <- seq(-4,4,0.02)
z <- as.matrix(expand.grid(x,y),0)
m <- length(x)
n <- length(y)


iris.qdp <- predict(iris.qda,z)$class
contour(x,y,matrix(iris.qdp,m,n),
        levels=c(1.5,2.5), add=TRUE, d=FALSE, lty=2)
```

Iris example: QDA boundaries



LDA or QDA?

Having seen both LDA and QDA in action, it is natural to ask which is the "better" classifier.
It is obvious that if the covariances of different classes are very distinct, QDA will probably have an advantage over LDA.
As parametric models are only ever approximations to the real world, allowing more flexible decision boundaries (QDA) may seem like a good idea.
However, there is a price to pay in terms of increased variance.

Regularized Discriminant Analysis

In the case where data is scarce , to fit

- LDA, need to estimate $K \times p + p \times p$ parameters
- QDA, need to estimate $K \times p + K \times p \times p$ parameters.

Using LDA allows us to better estimate the covariance matrix Σ. Though QDA allows more flexible decision boundaries, the estimates of the K covariance matrices Σ_k are more variable.
RDA combines the strengths of both classifiers by regularizing each covariance matrix Σ_k in QDA to the single one Σ in LDA

$$\Sigma_k(\alpha) = \alpha\Sigma_k + (1-\alpha)\Sigma \text{ for some } \alpha \in [0,1].$$

This introduces a new parameter α and allows for a continuum of models between LDA and QDA to be used. Can be selected by Cross-Validation for example.

Outline

Naïve Bayes

If $p > n$ (for example more genes p than patients n), LDA (and certainly QDA and RDA) runs into problems.
Recall that the covariance matrix Σ is estimated from n observations. If $p > n$, then

$$\hat{\Sigma} = \frac{1}{n} \sum_{k=1}^{K} \sum_{j:Y_j=k} (X_j - \hat{\mu}_k)(X_j - \hat{\mu}_k)^T$$

is singular. As the inverse of $\hat{\Sigma}$ is used in LDA, it will fail.
An extreme regularization is to estimate Σ as above but set all non-diagonal elements to 0, i.e. ignoring dependence between predictor variables completely. This is sometimes referred to as *Naive Bayes*. All correlations between variables are effectively ignored in this way.
Alternatively, one can estimate Σ by using the estimate $\hat{\Sigma}$ as above and adding $\lambda 1_p$ for some $\lambda > 0$, where 1_p is the p-dimensional identity matrix (makes only sense if data have been standardized initially).

Applications to Classification of Documents

Given documents such as emails, webpages, scientific articles, books etc., we might be interested in learning a classifier based on training data to automatically classify a new document. Possible classes could be spam/non-spam, academic/commercial webpages, maths/physics/biology etc.
Many popular techniques rely on simple probabilistic models for documents.
Given a prespecified dictionary, we extract high-dimensional features such as absence/presence of a word (multivariate Bernoulli), number of occurences of a word (multinomial) etc.
Parameters within in class can be estimated through Maximum Likelihood.
However Maximum Likelihood overfits so we will need to derive more robust alternative.

Outline

Limitations of Maximum Likelihood

- Given a probabilistic model

$$P(x, y = k) = \pi_k f_k(x),$$

we typically assume a parametric form for $f_k(x) = f(x | \phi_k)$ and compute the MLE $\widehat{\theta}$ of $\theta = (\pi_k, \phi_k)_{k=1}^n$ based on the training data $\{X_i, Y_i\}_{i=1}^n$.
- We then use a plug-in approach to perform classification

$$P\left(y = k | x, \widehat{\theta}\right) = \frac{\widehat{\pi}_k f\left(x | \widehat{\phi}_k\right)}{\sum_{j=1}^K \widehat{\pi}_j f\left(x | \widehat{\phi}_j\right)}.$$

Limitations of Maximum Likelihood

- Even for simple models, this can prove difficult; e.g. if $f\left(x|\,\phi_k\right) = \mathcal{N}\left(x; \mu_k, \Sigma\right)$ then the MLE estimate of Σ is not full rank for $p > n$.
- One possibility is to simplify even further the model as in Naïve Bayes; e.g.

$$f\left(x|\,\phi_k\right) = \prod_{l=1}^{p} \mathcal{N}\left(x^l; \mu_k^l, \left(\sigma_k^l\right)^2\right)$$

  but this might be too crude.
- Moreover, the plug-in approach does not take into account the uncertainty about the parameter estimate.

A Toy Example

- Consider a trivial case where $X \in \{0, 1\}$ and $K = 2$ so that
$$f(x|\phi_k) = \phi_k^x (1 - \phi_k)^{1-x}.$$
then the MLE estimates are given by
$$\widehat{\phi}_k = \frac{\sum_{i=1}^n \mathbb{I}(x_i = 1, y_i = k)}{n_k}, \ \widehat{\pi}_k = \frac{n_k}{n}$$
where $n_k = \sum_{i=1}^n \mathbb{I}(y_i = k)$.
- Assume that all the training data for class 1 are such that $x_i = 0$ then $\widehat{\phi}_1 = 0$ and
$$\begin{aligned} P\left(y = 1|\, x = 1, \widehat{\theta}\right) &= \frac{P\left(x = 1|\, y = 1, \widehat{\theta}\right) P\left(y = 1|\, \widehat{\theta}\right)}{P\left(y = 1|\, \widehat{\theta}\right)} \\ &= \frac{\widehat{\phi}_1 \widehat{\pi}_1}{P\left(y = 1|\, \widehat{\theta}\right)} = 0. \end{aligned}$$
- Hence if we have not observed such events in our training set, we predict that we will never observe them, ever!

Text Classification

- Assume we are interested in classifying documents; e.g. scientific articles or emails.
- A basic but standard model for text classification consists of considering a pre-specified dictionary of p words (including say physics, calculus.... or dollars, sex etc.) and summarizing each document by $X = \left(X^1, ..., X^p\right)$ where
$$X^l = \begin{cases} 1 & \text{if word } l \text{ is present in document} \\ 0 & \text{otherwise.} \end{cases}$$
- To implement a probabilistic classifier, we need to model $f_k(x)$ for $k = 1, ..., K$.
- A Naïve Bayes approach ignores features correlations and assumes $f_k(x) = f(x|\phi_k)$ where
$$f(x|\phi_k) = \prod_{l=1}^{p} \left(\phi_k^l\right)^{x^l} \left(1 - \phi_k^l\right)^{1-x^l}$$

Maximum Likelihood for Text Classification

- Given training data, the MLE is easily obtained

$$\widehat{\pi}_k = \frac{n_k}{n}, \ \widehat{\phi}_k^l = \frac{\sum_{i=1}^n \mathbb{I}\left(X_i^l = 1, Y_i = k\right)}{n_k}$$

- If word l never appears in the training data for class k then $\widehat{\phi}_k^l = 0$ and

$$P\left(y = k \middle| x = \left(x^{1:l-1}, x^l = 1, x^{l+1:p}\right), \widehat{\theta}\right) = 0;$$

  i.e. we will never attribute a new document containing word l to class k.
- In many practical applications, we have $p \gg n$ and this problem often occurs.

A Bayesian Approach

- An elegant way to deal with the problem consists of using a Bayesian approach.
- We start with the very simple case where

$$f\left(x|\,\phi\right)=\phi^{x}\left(1-\phi\right)^{1-x}$$

and now set a Beta prior on $p\left(\phi\right)$ on ϕ

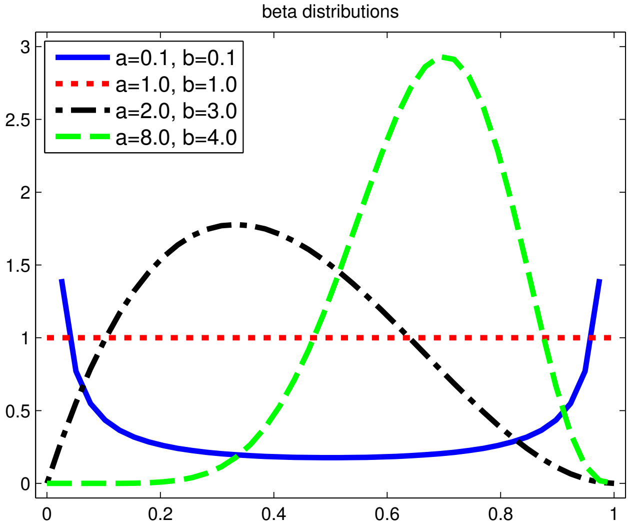
$$p\left(\phi\right)=Beta\left(\phi;a,b\right)$$

where

$$Beta\left(\phi;a,b\right)=\frac{\Gamma\left(a+b\right)}{\Gamma\left(a\right)\Gamma\left(b\right)}\phi^{a-1}\left(1-\phi\right)^{b-1}1_{[0,1]}\left(\phi\right)$$

with $\Gamma\left(u\right)=\int_{0}^{\infty}t^{u-1}e^{-t}dt$. Note that $\Gamma\left(u\right)=\left(u-1\right)!$ for $u\in\mathbb{N}$.
(a,b) are *fixed* quantities called *hyperparameters*. For $a=b=1$, the Beta density corresponds to the uniform density.

Beta Distribution



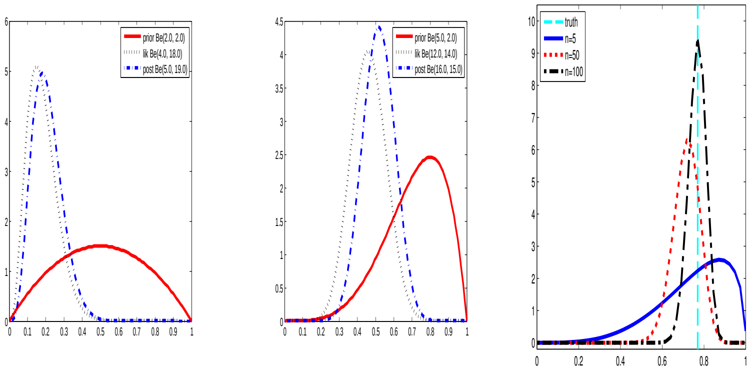
A Bayesian Approach

- Given a realization of $X_{1:n} = (X_1, ..., X_n)$, inference on ϕ is based on the posterior

$$
\begin{aligned}
p\left(\phi \middle| x_{1:n}\right) &= \frac{p\left(\phi\right) \prod_{i=1}^{n} f\left(x_i \middle| \phi\right)}{\pi\left(x_{1:n}\right)} \\
&= Beta\left(\theta; a + n_s, b + n - n_s\right)
\end{aligned}
$$

  with $n_s = \sum_{i=1}^{n} \mathbb{I}\left(x_i = 1\right)$.
- The prior on θ can be conveniently reinterpreted as an imaginary initial sample of size $(a + b)$ with a observations “1” and b observations “0”. Provided that $(a + b)$ is small with respect to n, the information carried by the data is prominent.

Beta Posteriors



(left) Updating a Beta(2,2) prior with a Binomial likelihood with $n_s = 3$, $n = 20$ to yield a Beta(5,19); (center) Updating a Beta(5,2) prior with a Binomial likelihood with $n_s = 11$, $n = 24$ to yield a Beta(16,15) posterior. (right) Sequentially updating a Beta distribution starting with a Beta(1,1) and converging to a delta function centered on the true value.

Posterior Statistics

- We have
$$\mathbb{E}\left(\phi \middle| x_{1:n}\right) = \frac{a + n_s}{a + b + n}$$
and the posterior means behave asymptotically like n_s/n (the ‘frequentist’ estimator) and converge to ϕ^*, the ‘true’ value of ϕ.
- We have
$$\begin{aligned}\mathbb{V}\left(\phi \middle| x_{1:n}\right) &= \frac{(a + n_s)(b + n - n_s)}{(a + b + n)^2 (a + b + n + 1)} \\ &\approx \frac{\widehat{\phi}\left(1 - \widehat{\phi}\right)}{n} \text{ for large } n\end{aligned}$$
- The posterior variance decreases to zero as $n \to \infty$, at rate n^{-1}: the information you get on ϕ gets more and more precise.
- For n large enough, the prior is washed out by the data. For a small n, its influence can be significant.

Prediction: Plug-in Estimate vs Bayesian Approaches

- Assume you have observed $X_1 = \cdots = X_n = 0$, then the plug-in prediction is

$$P\left(x = 1 | \widehat{\phi}\right) = \widehat{\phi}$$

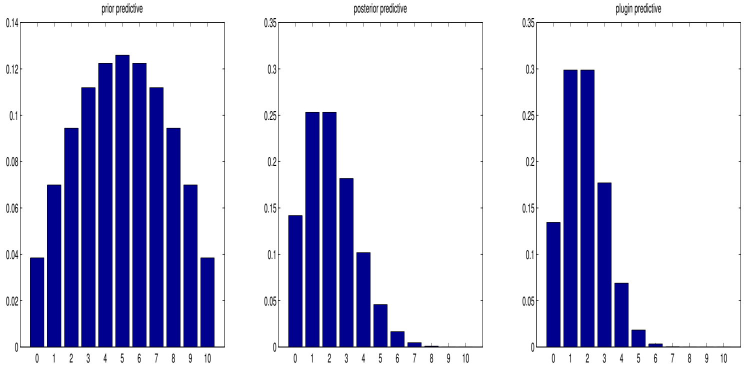
  which does not account whatsoever for the uncertainty about ϕ.

- In a Bayesian approach, we will use the predictive distribution

$$\begin{aligned} P\left(x = 1 | x_{1:n}\right) &= \int P\left(x = 1 | \phi\right) p\left(\phi | x_{1:n}\right) d\phi \\ &= \frac{a + n_s}{a + b + n} \end{aligned}$$

  so even if $n_s = 0$ then $P\left(x = 1 | x_{1:n}\right) > 0$ and our prediction takes into account the uncertainty about ϕ.

Beta Posteriors



(left) Prior predictive dist. for a Binomial likelihood with $n = 10$ and a Beta(2,2) prior. (center) Posterior predictive after having seen $n_s = 3, n = 20$. (right) Plug-in approximation using $\widehat{\phi}$.

Bayesian Inference for the Multinomial

- Assume we have $Y_{1:n} = (Y_1, ..., Y_n)$ where $Y_i = \left(Y_i^1, ..., Y_i^K\right) \in \{0,1\}^K$, $\sum_{k=1}^K Y_i^k = 1$ and

$$P\left(y|\,\pi\right) = \prod_{k=1}^{K} \pi_k^{y^k}$$

  for $\pi_k > 0$, $\sum_{k=1}^K \pi_k = 1$.
- We have seen that the MLE estimate is

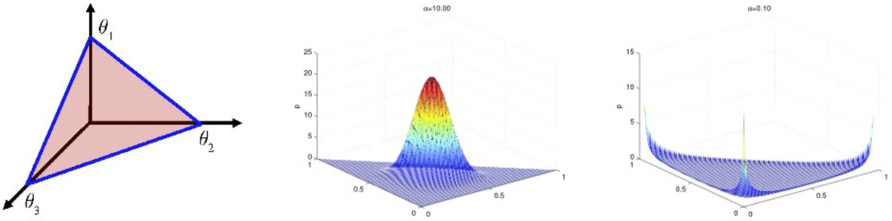
$$\widehat{\pi}_k = \frac{\sum_{i=1}^n \mathbb{I}\left(y_i^k = 1\right)}{n} = \frac{n_k}{n}$$

- We introduce the Dirichlet density

$$p\left(\pi\right) = \text{Dir}\left(\pi; \alpha\right) = \frac{\Gamma\left(\sum_{k=1}^K \alpha_k\right)}{\prod_{k=1}^{K} \Gamma\left(\alpha_k\right)} \prod_{k=1}^{K} \pi_k^{\alpha_k - 1}$$

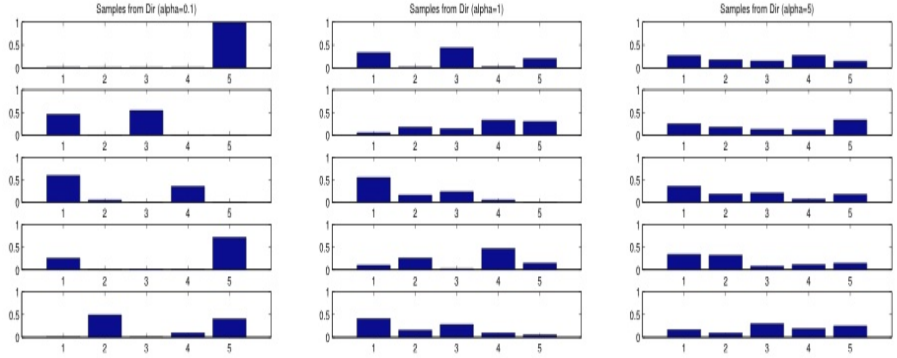
  for $\alpha_k > 0$ defined on $\left\{\pi : \pi_k > 0 \text{ and } \sum_{k=1}^K \pi_k = 1\right\}$.

Dirichlet Distributions



(left) Support of the Dirichlet density for $K = 3$ (center) Dirichlet density for $\alpha_k = 10$ (right) Dirichlet density for $\alpha_k = 0.1$.

Samples from Dirichlet Distributions



Samples from a Dirichlet distribution for $K = 5$ when $\alpha_k = \alpha_l$ for $k \neq l$.

Bayesian Inference

- We obtain

$$\begin{aligned} p\left(\pi \middle| y_{1:n}\right) &= \frac{p(\pi)\prod_{i=1}^{n} P\left(y_i \middle| \pi\right)}{p\left(y_{1:n}\right)} \\ &= Dir\left(\pi; \alpha_1 + n_1, \ldots, \alpha_K + n_K\right) \end{aligned}$$

- We have

$$\begin{aligned} P\left(y = k \middle| y_{1:n}\right) &= \int P\left(y = k \middle| \pi\right) p\left(\pi \middle| y_{1:n}\right) d\pi \\ &= \frac{\alpha_k + n_k}{\sum_{j=1}^{K} \alpha_j + n}. \end{aligned}$$

Bayesian Text Classification

- We have $\theta = \left(\pi_k, \left(\phi_k^1, ..., \phi_k^p\right)\right)_{k=1,...,K}$ with $\pi \sim$Dir(α) and $\phi_k^l \sim Beta\left(a, b\right).$
- Given data $D = (x_i, y_i)_{i=1,\ldots,n}$, classification is performed using

$$P\left(y = k \middle| D, x\right) = \frac{P\left(x \middle| D, y = k\right) P\left(y = k \middle| D\right)}{P\left(y = k \middle| D\right)}$$

where

$$P\left(y = k \middle| D\right) = \frac{\alpha_k + n_k}{\sum_{j=1}^{K} \alpha_j + n}$$

and $P\left(x \middle| D, y = k\right) = \prod_{l=1}^{p} P\left(x^l \middle| D, y = k\right)$ with

$$P\left(x^l \middle| D, y = k\right) = \frac{a + \sum_{i=1}^{n} \mathbb{I}\left(x_i^l = 1, y_i = k\right)}{a + b + n_k}.$$

- A popular alternative for text data consists of using as features the number of occurrences of words in document and using a multinomial model for $P\left(x \middle| \phi_k\right)$.

Bayesian QDA

- Let us come back to the QDA model where

$$f\left(x\middle|\,\phi_k\right)=\mathcal{N}\left(x;\mu_k,\Sigma_k\right).$$

- We set improper priors on (μ_k, Σ_k) where

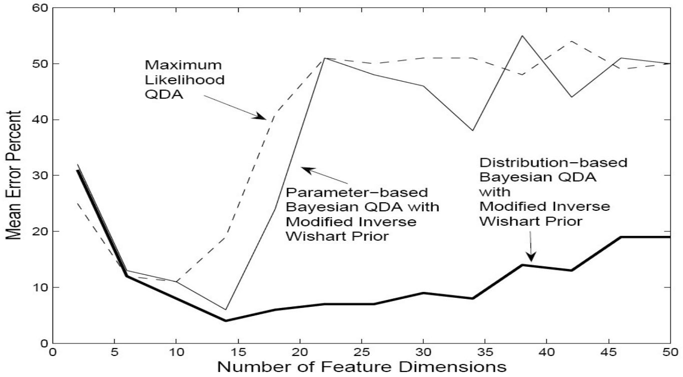
$$p\left(\mu_k,\Sigma_k\right)\propto\frac{\exp\left(-\frac{1}{2}tr\left(\Sigma_k^{-1}B_k\right)\right)}{\left|B_k\right|^{q/2}}$$

  where $B_k > 0$ (e.g. $B_k = \lambda I_p$ with $\lambda > 1.$) ; i.e. flat prior on μ_k and inverse-Wishart on Σ_k. Unimodal prior on Σ_k with mode B_k/q.
- It follows that

$$\begin{aligned} f\left(x\middle|\,D,y=k\right) &= \int\mathcal{N}\left(x;\mu_k,\Sigma_k\right)p\left(\mu_k,\Sigma_k\middle|\,D\right)d\mu_k d\Sigma_k \\ &= \left(\frac{n_k}{n_k+1}\right)^{p/2}\frac{\Gamma\left(\frac{n_k+q+1}{2}\right)}{\Gamma\left(\frac{n_k+q-p+1}{2}\right)}\frac{\left|\frac{S_k+B_k}{2}\right|^{\frac{n_k+q}{2}}}{\left|A_k\right|^{\frac{n_k+q+1}{2}}},\end{aligned}$$

$$A_k=\tfrac{1}{2}\left(S_k+\tfrac{n_k(x-\mu_k)(x-\mu_k)^T}{n_k+1}+B_k\right),$$
$$S_k=\textstyle\sum_{i=1}^{n}I\left(y_i=k\right)\left(x_i-\widehat{\mu}_k\right)\left(x_i-\widehat{\mu}_k\right)^T.$$

Bayesian QDA



Mean error rates are shown for a two-class problem where the samples from each class are drawn from a Gaussian distribution with the same mean but different, highly ellipsoidal covariance matrices. 40 training examples, 100 test samples.

Outline

Logistic Regression

Recall that for LDA, upon assuming that $X|Y = k \sim N(\mu_k, \Sigma)$, the Bayes Classifier classified to class 1 over class k if

$$\begin{aligned} 0 &> 2\log P(Y=k|x) - 2\log P(Y=1|x) \\ &= \mu_k^T\Sigma^{-1}\mu_k - 2\mu_k^T\Sigma^{-1}x - 2\log(\pi_k) \\ &\quad -(\mu_1^T\Sigma^{-1}\mu_1 - 2\mu_1^T\Sigma^{-1}x - 2\log(\pi_1)) \\ &= a_k + b_k^T x \end{aligned}$$

i.e. hyperplanes separate classes in the feature space $\mathcal{X}$.
The *separating hyperplane* can be rewritten more clearly as

$$2\log\frac{P(Y=k|x)}{P(Y=1|x)} = a_k + b_k^T x.$$

For QDA, $X|Y = k \sim N(\mu_k, \Sigma_k)$, we in turn found a quadratic function $0 > a_k + b_k^T x + x^T c_k x$ i.e.

$$2 \log \frac{P(Y = k|x)}{P(Y = 1|x)} = a_k + b_k^T x + x^T c_k x.$$

The exact value of the parameters a_k and b_k (c_k) had expressions which could be evaluated once the parameters μ_k and Σ (Σ_k) were in turn found by plug-in estimation (via ML estimation)

We can model these *decision boundaries* directly instead. This is called *logistic discrimination*.

Logistic discrimination model posterior probabilities $P(Y = k|x)$ directly. Assuming a parametric family of discriminant functions $g_\beta(x)$, we model the conditional probabilities as

$$\hat{P}(Y = k|x) = \frac{\exp g_{\beta_k}(x)}{\sum_{j=1}^{K} \exp g_{\beta_j}(x)}.$$

Note that the log probability of a class k, with respect to a reference class 1 is:

$$\log \frac{P(Y = k|x)}{P(Y = 1|x)} = g_{\beta_k}(x) - g_{\beta_1(x)}$$

This reduces to LDA and QDA for linear and quadratic discriminant functions (assuming also that the parameters β_k were estimated as before).

The parameter $\hat{\beta} = (\hat{\beta}_1, \ldots, \hat{\beta}_K)$ is typically chosen by computing the (Conditional) Maximum Likelihood estimate.
Given a training set, the likelihood of the model is given by

$$L(\beta) = \prod_{i=1}^{n} P(Y = y_i | x_i) = \prod_{i=1}^{n} \frac{\exp g_{\beta_{y_i}}(x_i)}{\sum_{j=1}^{K} \exp g_{\beta_j}(x_i)}$$

and so the (conditional) log-likelihood is

$$\ell(\beta) = \sum_{i=1}^{n} \log P(Y = y_i | x_i).$$

Choosing $g_\beta(x) = \beta^T x$ results in linear decision boundaries and ensures that $\ell(\beta)$ is concave.
This particular logistic discrimination model is known as *logistic regression* and is an example of empirical risk minimization, where the risk is measured in terms of the 'logistic' loss function.

For the case of $K = 2$ classes (*binomial logistic regression*), the log-likelihood collapses into a much simpler form than when $K > 2$ (*multinomial logistic regression*). We concentrate on the case where $K = 2$ though it should be noted that the theory still applies for $K > 2$.
Looking at $K = 2$, we can derive an explicit expression for the log-likelihood as follows.
For the following let $Y \in \{-1, 1\}$. Let $g_\beta = \beta^T x$ and $\beta_{-1} \equiv 0$ (so class -1 is the reference class). Let $\beta = \beta_1$. Then

$$\begin{aligned} P(Y = 1|x) &= \frac{\exp(\beta^T x)}{\exp(\beta^T x) + 1} = \frac{1}{1 + \exp(-\beta^T x)} \\ P(Y = -1|x) &= \frac{1}{1 + \exp(\beta^T x)}. \end{aligned}$$

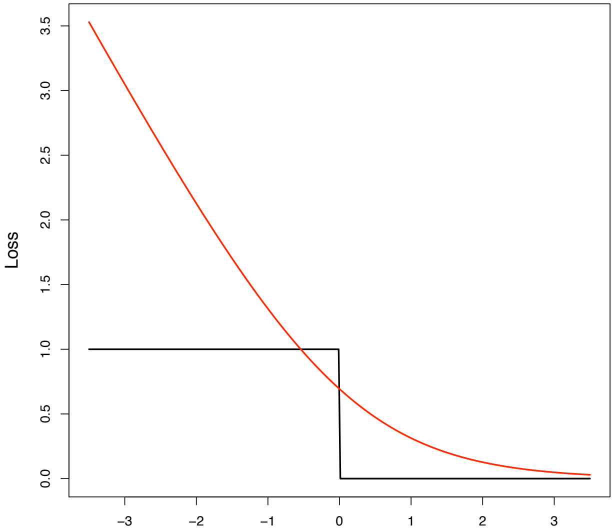
Or, shorthand for both classes, $P(Y = y|x) = \frac{1}{1+\exp(-y \cdot \beta^T x)}$.

Continuing with this notation, the (conditional) log-likelihood is

$$\begin{aligned}
\ell(\beta) &= \sum_{i=1}^{n} \log P(Y = y_i | x_i) \\
&= \sum_{i=1}^{n} \log \frac{1}{1 + \exp(-y_i \cdot \beta^T x_i)} \\
&= \sum_{i=1}^{n} - \log(1 + \exp(-y_i \cdot \beta^T x_i)),
\end{aligned}$$

where $L(y, f) = \log(1 + \exp(-y \cdot f))$ is the so-called logistic loss, using notation $f = \beta^T x_i$.
(Note that for under 0-1 loss, the optimal classification is 1 if $f > 0$ and -1 if $f \leq 0$.)

Compare the logistic loss $L(y,f) = \log(1 + \exp(-y \cdot f))$ with the 0-1 misclassification loss $L(y,f) = 1\{\text{sign}(y) \neq \text{sign}(f)\} = 1\{y \cdot f < 0\}$.



Loss L as a function of $y \cdot f = y \cdot \beta^T x$.

As shown above, ML estimation is (in the case $Y \in \{-1, 1\}$ equivalent to solving the equations),

$$\hat{\beta} = \text{argmin}_{\beta} \quad \sum_{i=1}^{n} \log(1 + \exp(-y_i \cdot \beta^T x_i)),$$

numerical methods must be applied. A high-dimensional version of the Newton-Raphson algorithm is typically used, where locally the objective function is approximated by a quadratic function and the solution is then found by iterated least squares.
When using the univariate Newton-Raphson approach, we need information about the slope of the curve, in our case we need the Hessian matrix

$$\frac{\partial^2 \ell(\beta)}{\partial\beta\partial\beta^T} = -\sum_{i=1}^{n} x_i x_i^T p(x_i|\beta)\left[1 - p(x_i|\beta)\right].$$

Extending Newton-Raphson to higher dimensions, starting with β^{old}, a single Newton-Raphson update is given by

$$\beta^{new} = \beta^{old} - \left(\frac{\partial^2 \ell(\beta)}{\partial\beta\partial\beta^T}\right)^{-1} \frac{\partial \ell(\beta)}{\partial\beta}$$

where the derivatives are evaluated at β^{old}.

Logistic Regression

- Writing everything in vectorial form,
  - $\mathbf{c} = (Y_i)_{i=1}^n$, a vector of the classes
  - $\mathbf{p} = \left(P(Y_i = 1 | X_i, \beta^{old})\right)_{i=1}^n$, the vector of fitted probabilities
  - $\mathbf{X}$, an $n \times p$ matrix with i^{th} row as X_i
  - $\mathbf{W}$, a diagonal matrix with i^{th} diagonal as $P(Y_i = 1 | X_i, \beta^{old}) \left(1 - P(Y_i = 1 | X_i, \beta^{old})\right)$
- Lets us write $\frac{\partial \ell(\beta)}{\partial \beta} = \mathbf{X}^T(\mathbf{c} - \mathbf{p})$ and $\frac{\partial^2 \ell(\beta)}{\partial \beta \partial \beta^T} = -\mathbf{X}^T\mathbf{W}\mathbf{X}$ so

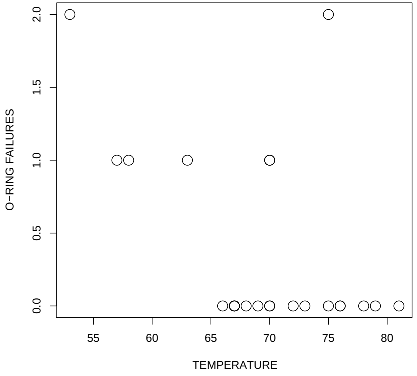
$$\begin{aligned} \beta^{new} &= \beta^{old} - \left(\frac{\partial^2 \ell(\beta)}{\partial \beta \partial \beta^T}\right)^{-1} \frac{\partial \ell(\beta)}{\partial \beta} \\ &= \beta^{old} + (\mathbf{X}^T\mathbf{W}\mathbf{X})^{-1}\mathbf{X}^T(\mathbf{c} - \mathbf{p}) \\ &= (\mathbf{X}^T\mathbf{W}\mathbf{X})^{-1}\mathbf{X}^T\mathbf{W}\left[\mathbf{X}\beta^{old} + \mathbf{W}^{-1}(\mathbf{c} - \mathbf{p})\right] \end{aligned}$$

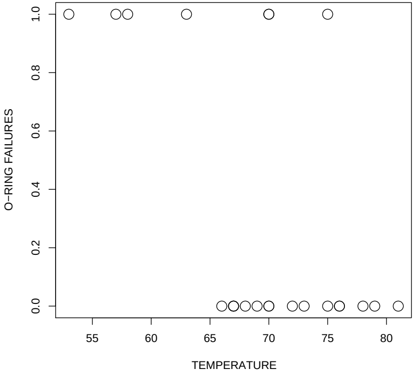
Each Newton-Raphson step can be seen as a weighted least squares step, this algorithm is more commonly known as *Iteratively Reweighted Least Squares*.
A few (even just 2 or 3) steps of the algorithm are usually sufficient.

Example: O-ring failures during shuttle starts (preceeeding the Challenger incident), as a function of temperatures.

```
library(alr3)
data(challeng)
temp <- challeng[,1]
failure <- challeng[,3]
Y <- as.numeric(failure>0)

plot(temp,Y,xlab="TEMPERATURE",
      ylab="O-RING FAILURES",cex=2)
```

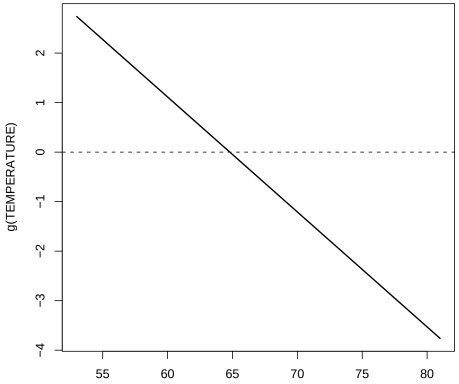
LEFT: Number of failures.
RIGHT: Number of O-Ring failures reduced here to “Failures Yes/No” binary variable.





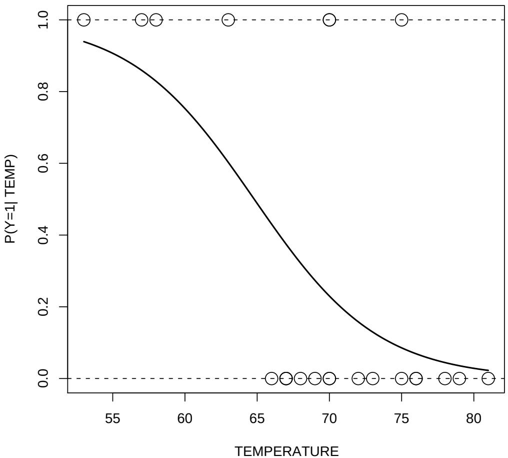
Fit logistic regression with `glm` function and plot 'link' function $f = \beta^T X$, where X is here simply temperature ($p = 1$).

```
log_reg <- glm( Y ~  temp ,family=binomial)
xvec <- seq(min(temp),max(temp),length=200)
g <- predict(log_reg,newdata=data.frame(temp=xvec),
        type="link")
plot(xvec, g ,
        type="l",lwd=1.8,
        xlab="TEMPERATURE",ylab="g(TEMPERATURE)")
```



Now plot $P(Y = 1|X) = 1/(1 + \exp(-\beta^T X))$.

```
prob <- predict(log_reg,newdata=data.frame(temp=xvec),
       type="response")
plot(xvec, prob ,
       type="l",lwd=1.8,
       xlab="TEMPERATURE",ylab="P(Y=1| TEMP)",ylim=c(0,1))
points(temp,Y,cex=2)
```



Logistic Regression or LDA?

Both LR and LDA possess linear decision boundaries

- LDA as a consequence of assuming $X|Y = k \sim \mathcal{N}_p(\mu_k, \Sigma)$ and
- Logistic Regression by construction of the log-odds. However, we can easily replace a, say, two-dimensional predictor with intercept, $x = (1, x^{(1)}, x^{(2)})$ with $\tilde{x} = (1, x^{(1)}, x^{(2)}, (x^{(1)})^2, (x^{(2)})^2)$ to model non-linear decision boundaries.

However, actual decision boundaries for both models differ and do so because of differences in how the coefficients of class decision boundaries (hyperplanes) are estimated, which approach is 'better'?

- Where $X|Y = k \sim N_p(\mu_k, \Sigma)$ is true, LDA seems better positioned.
- It can be shown that where $X|Y = k \sim \mathcal{N}_p(\mu_k, \Sigma)$, using LR results in a $\sim$30% reduction in the efficiency.
- However, if the assumptions are far from true LDA will suffer.

In support of Logistic Regression over LDA, it can be noted that Logistic Regression is simply a generalised linear model (GLM).
Knowing this, we can take advantage of all of the theory developed for GLMs.

- assessment of fit via deviance and plots,
- interpretation of β_k's via *odds-ratios*,
- fitting categorical data (code it via indicator functions),
- well founded approaches to removing insignificant terms (via the drop-in deviance test and the Wald test),
- model selection via AIC/BIC.

Ultimately, we have to let the data speak!

Spam dataset: Look at examples of spam emails and non-spam emails. The predictor variables count occurrence of specific words/characters. Look at the first 2 emails in the database (which are spam).

```
> library(kernlab)
> data(spam)
> dim(spam)
[1] 4601   58

> spam[1:2,]
  make address  all num3d  our over remove internet order mail receive wil
1 0.00    0.64 0.64     0 0.32 0.00   0.00     0.00     0 0.00    0.00 0.6
2 0.21    0.28 0.50     0 0.14 0.28   0.21     0.07     0 0.94    0.21 0.7
  people report addresses free business email  you credit your font num000
1   0.00   0.00      0.00 0.32     0.00  1.29 1.93      0 0.96    0   0.00
2   0.65   0.21      0.14 0.14     0.07  0.28 3.47      0 1.59    0   0.43
  money hp hpl george num650 lab labs telnet num857 data num415 num85
1  0.00  0   0      0      0   0    0      0      0    0      0     0
2  0.43  0   0      0      0   0    0      0      0    0      0     0
  technology num1999 parts pm direct cs meeting original project re edu ta
1          0    0.00     0  0      0  0       0        0       0  0   0
2          0    0.07     0  0      0  0       0        0       0  0   0
  conference charSemicolon charRoundbracket charSquarebracket charExclamat
1          0             0            0.000                 0        0.778
2          0             0            0.132                 0        0.372
  charDollar charHash capitalAve capitalLong capitalTotal type
1       0.00    0.000      3.756          61          278 spam
2       0.18    0.048      5.114         101         1028 spam
>
```

Fit a GLM to the data (look at `?glm` for help on the command).

```
library(kernlab)
data(spam)

## let Y=0 be non-spam and Y=1 be spam.
Y <- as.numeric(spam[, ncol(spam)])-1
X <- spam[ ,-ncol(spam)]

gl <- glm(Y ~ ., data=X,family=binomial)
```

Which predictor variables seem to be important? Can for example check which ones are significant in the GLM.

```
summary(gl)
```

```
> summary(gl)

Call:
glm(formula = Y ~ ., family = binomial, data = X)

Deviance Residuals:
       Min          1Q      Median          3Q         Max
-4.127e+00  -2.030e-01  -1.967e-06   1.140e-01   5.364e+00

Coefficients:
                    Estimate Std. Error z value Pr(>|z|)
(Intercept)       -1.569e+00  1.420e-01 -11.044  < 2e-16 ***
make              -3.895e-01  2.315e-01  -1.683 0.092388 .
address           -1.458e-01  6.928e-02  -2.104 0.035362 *
all                1.141e-01  1.103e-01   1.035 0.300759
num3d              2.252e+00  1.507e+00   1.494 0.135168
our                5.624e-01  1.018e-01   5.524 3.31e-08 ***
over               8.830e-01  2.498e-01   3.534 0.000409 ***
remove             2.279e+00  3.328e-01   6.846 7.57e-12 ***
internet           5.696e-01  1.682e-01   3.387 0.000707 ***
order              7.343e-01  2.849e-01   2.577 0.009958 **
mail               1.275e-01  7.262e-02   1.755 0.079230 .
receive           -2.557e-01  2.979e-01  -0.858 0.390655
will              -1.383e-01  7.405e-02  -1.868 0.061773 .
people            -7.961e-02  2.303e-01  -0.346 0.729557
report             1.447e-01  1.364e-01   1.061 0.288855
addresses          1.236e+00  7.254e-01   1.704 0.088370 .
...
```

```
...
business            9.599e-01  2.251e-01   4.264 2.01e-05 ***
email               1.203e-01  1.172e-01   1.027 0.304533
you                 8.131e-02  3.505e-02   2.320 0.020334 *
credit              1.047e+00  5.383e-01   1.946 0.051675 .
your                2.419e-01  5.243e-02   4.615 3.94e-06 ***
font                2.013e-01  1.627e-01   1.238 0.215838
num000              2.245e+00  4.714e-01   4.762 1.91e-06 ***
money               4.264e-01  1.621e-01   2.630 0.008535 **
hp                 -1.920e+00  3.128e-01  -6.139 8.31e-10 ***
hpl                -1.040e+00  4.396e-01  -2.366 0.017966 *
george             -1.177e+01  2.113e+00  -5.569 2.57e-08 ***
num650              4.454e-01  1.991e-01   2.237 0.025255 *
lab                -2.486e+00  1.502e+00  -1.656 0.097744 .
labs               -3.299e-01  3.137e-01  -1.052 0.292972
telnet             -1.702e-01  4.815e-01  -0.353 0.723742
num857              2.549e+00  3.283e+00   0.776 0.437566
data               -7.383e-01  3.117e-01  -2.369 0.017842 *
num415              6.679e-01  1.601e+00   0.417 0.676490
num85              -2.055e+00  7.883e-01  -2.607 0.009124 **
technology          9.237e-01  3.091e-01   2.989 0.002803 **
num1999             4.651e-02  1.754e-01   0.265 0.790819
parts              -5.968e-01  4.232e-01  -1.410 0.158473
pm                 -8.650e-01  3.828e-01  -2.260 0.023844 *
direct             -3.046e-01  3.636e-01  -0.838 0.402215
cs                 -4.505e+01  2.660e+01  -1.694 0.090333 .
meeting            -2.689e+00  8.384e-01  -3.207 0.001342 **
original           -1.247e+00  8.064e-01  -1.547 0.121978
```

```
...
project           -1.573e+00  5.292e-01  -2.973 0.002953 **
re                -7.923e-01  1.556e-01  -5.091 3.56e-07 ***
edu               -1.459e+00  2.686e-01  -5.434 5.52e-08 ***
table             -2.326e+00  1.659e+00  -1.402 0.160958
conference        -4.016e+00  1.611e+00  -2.493 0.012672 *
charSemicolon     -1.291e+00  4.422e-01  -2.920 0.003503 **
charRoundbracket  -1.881e-01  2.494e-01  -0.754 0.450663
charSquarebracket -6.574e-01  8.383e-01  -0.784 0.432914
charExclamation    3.472e-01  8.926e-02   3.890 0.000100 ***
charDollar         5.336e+00  7.064e-01   7.553 4.24e-14 ***
charHash           2.403e+00  1.113e+00   2.159 0.030883 *
capitalAve         1.199e-02  1.884e-02   0.636 0.524509
capitalLong        9.118e-03  2.521e-03   3.618 0.000297 ***
capitalTotal       8.437e-04  2.251e-04   3.747 0.000179 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 6170.2  on 4600  degrees of freedom
Residual deviance: 1815.8  on 4543  degrees of freedom
AIC: 1931.8

Number of Fisher Scoring iterations: 13
```

How good is the classification?

```
> proba <- predict(gl,type="response")
> predicted_spam <- as.numeric( proba>0.5)
> table(predicted_spam,Y)
              Y
predicted_spam    0    1
             0 2666  194
             1  122 1619

> predicted_spam <- as.numeric( proba>0.99)
> table(predicted_spam,Y)
              Y
predicted_spam    0    1
             0 2776 1095
             1   12  718
```

So out of 730 emails marked as spam, 12 were actually not spam. Would you expect a similar success rate for future classifications?

Outline

Training and Test error

Important distinction:

- **Training error** is the empirical risk

$$n^{-1}\sum_{i=1}^{n} L(y_i, \hat{y}_i)$$

  For 0-1 loss in classification, this is the misclassification error on the training data, **which were used in fitting** $\hat{y}$.

- **Test error** is the empirical risk on **new, previously unseen**, observations

$$m^{-1}\sum_{i=1}^{m} L(y_i, \hat{y}_i)$$

  which were NOT used in fitting.

The test error is in general larger than the training error (as we are fitting partially noise – depending on the complexity of the classifier). It is a much better gauge of how well the method will do on future data.

Success rate is calculated on the same data that the GLM is trained on!
Separate in training and test set.

```
n <- length(Y)
intrain <- sample( rep(c(TRUE,FALSE),each=n/2) ,
                              round(n/2) ,replace=TRUE )
train <- (1:n)[intrain]
test  <- (1:n)[!intrain]
```

Fit only on training set and predict on both training and test set.

```
gl <- glm(Y[train] ~ ., data=X[train,],family=binomial)

proba_train <- predict(gl,newdata=X[train,],type="response")
proba_test  <- predict(gl,newdata=X[test,],type="response")

predicted_spam_train <- as.numeric(proba_train > 0.95)
predicted_spam_test  <- as.numeric(proba_test > 0.95)
```

Results for training and test set:

```
> table(predicted_spam_train, Y[train])
predicted_spam_train    0    1
                   0 1403  354
                   1   11  567
```

```
> table(predicted_spam_test, Y[test])
predicted_spam_test    0    1
                  0 1346  351
                  1   28  541
```

Its no coincidence that the success rate is worse on the test data.

Compare with LDA.

```
library(MASS)
ldares <- lda(x=X[train,],grouping=Y[train])
```

With following result

```
> Call:
lda(X, grouping = Y)

Prior probabilities of groups:
        0         1
0.6059552 0.3940448

...
```

```
...

Coefficients of linear discriminants:
                          LD1
make              -0.2053433845
address           -0.0496520077
all                0.1618979041
num3d              0.0491205095
our                0.3470862316
over               0.4898352934
remove             0.8776953914
internet           0.3874021379
order              0.2987224576
mail               0.0621045827
receive            0.2343512301
will              -0.1148308781
people             0.0490659059
....
charHash           0.1141464080
capitalAve         0.0009590191
capitalLong        0.0002751450
capitalTotal       0.0003291749
```

Compare prediction on test set.

```
library(MASS)
lda_res <- lda(x=X[train,],grouping=Y[train])

proba_lda <- predict(lda_res,newdata=X[test,])$posterior[,2]
predicted_spam_lda <- as.numeric(proba_lda > 0.95)

> table(predicted_spam_test, Y[test])
predicted_spam_test    0    1
                  0 1346  351
                  1   28  541



> table(predicted_spam_lda, Y[test])
predicted_spam_lda    0    1
                 0 1364  533
                 1   10  359
```

It seems as if LDA beats Linear Regression here, but would need to adjust cutpoint to get proper comparison. Use ROC curves.

ROC curves

We can change the cutpoint c

```
predicted_spam_lda <- as.numeric(proba_lda > c)
```

to get different tradeoffs between

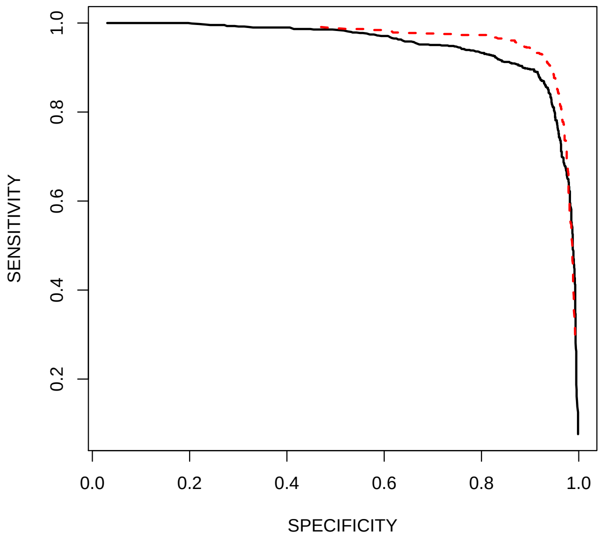
- Sensitivity: probability of predicting spam given true state is spam
- Specificity: probability of predicting non-spam given true state is non-spam

```
      TRUE STATE 0    1                           0       1
 PREDICTION   0 1364  533   normalize     0  0.9972  0.5975
              1   10  359    ---->        1  0.0072  0.4024
     TOTAL      1374  892                         1       1
```

ROC curve is sensitivity versus specificity

```
cvec <- seq(0.001,0.999,length=1000)
specif <- numeric(length(cvec))
sensit <- numeric(length(cvec))

for (cc in 1:length(cvec)){
  sensit[cc] <- sum( proba_lda> cvec[cc] & Y[test]==1)/sum(Y[test]==1)
  specif[cc] <- sum( proba_lda<=cvec[cc] & Y[test]==0)/sum(Y[test]==0)
}
plot(specif,sensit,
            xlab="SPECIFICITY",ylab="SENSITIVITY",type="l",lwd=2)
```

ROC curve for LDA and Logistic Regression classification of spam dataset.
LDA = unbroken black line; LR = broken red line.



Obvious now that LR is better for this dataset than LDA, contrary to the first impression.