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

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

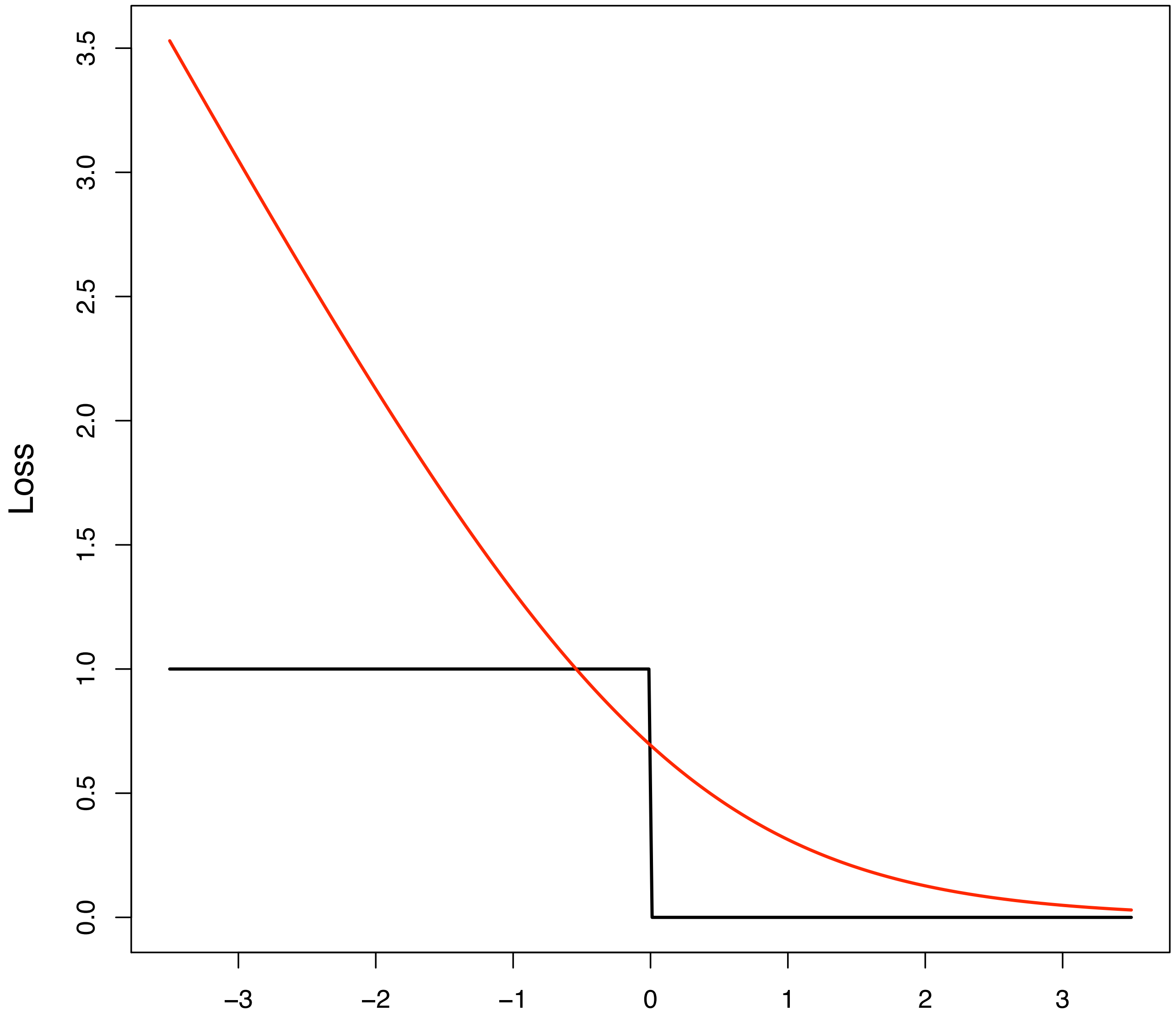
Or, shorthand for both classes, $P(Y = y|x) = \frac{1}{1+\exp(-y \cdot \beta^T x)}$.

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),

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

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

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

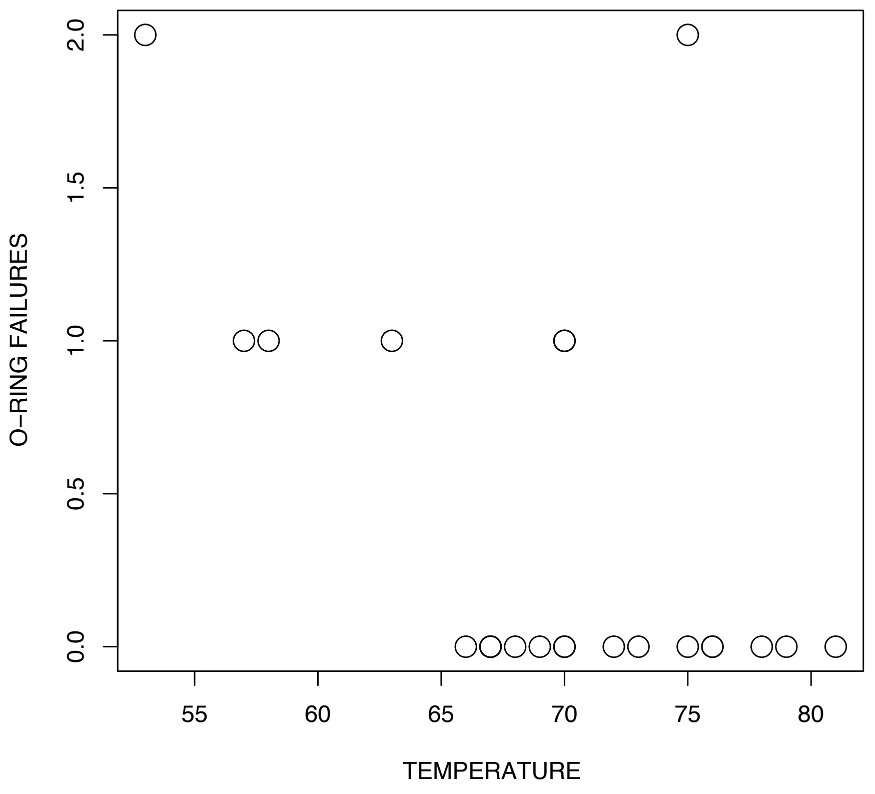
Each Newton-Raphson step can be seen as a weighted least squares step, this algorithm is more commonly known as *Iteratively Reweighted Least Squares*.
A few (even just 2 or 3) steps of the algorithm are usually sufficient.

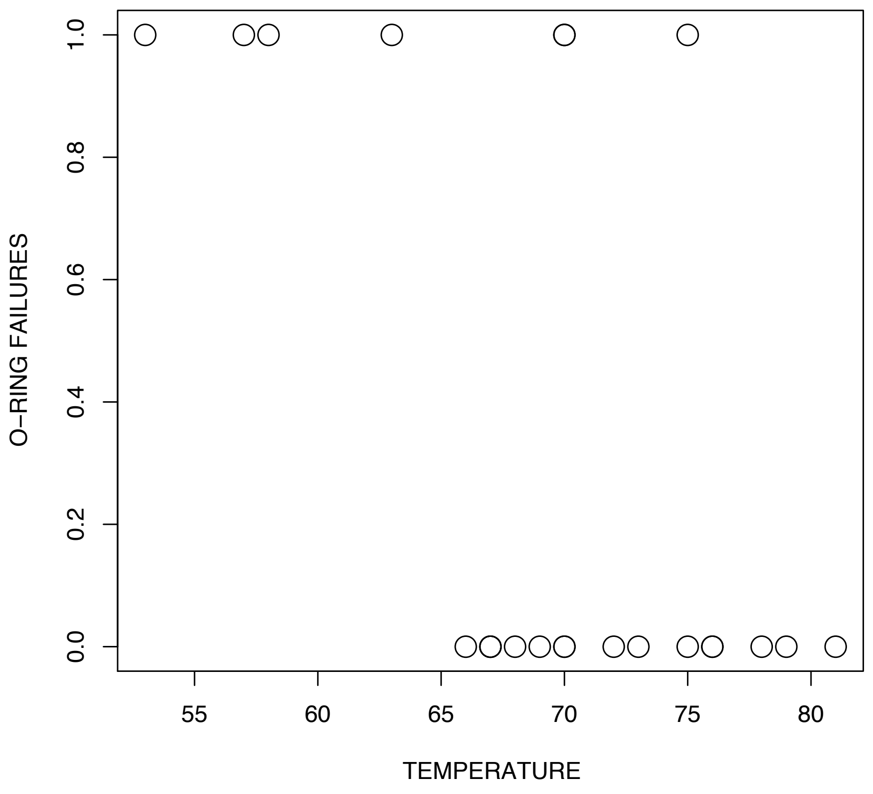
Example: O-ring failures during shuttle starts (preceeeding the Challenger incident), as a function of temperatures.

```
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)
```

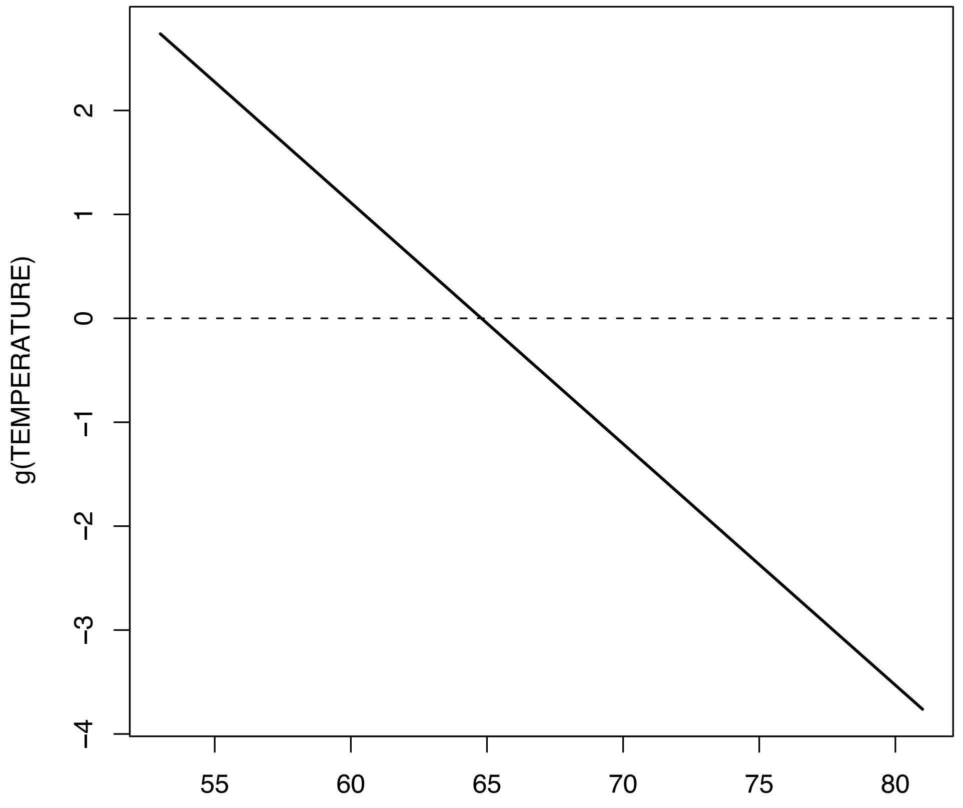
LEFT: Number of failures. Original analysis left out all measurements with 0 failures.
RIGHT: Number of O-Ring failures reduced here to “Failures Yes/No” binary variable.





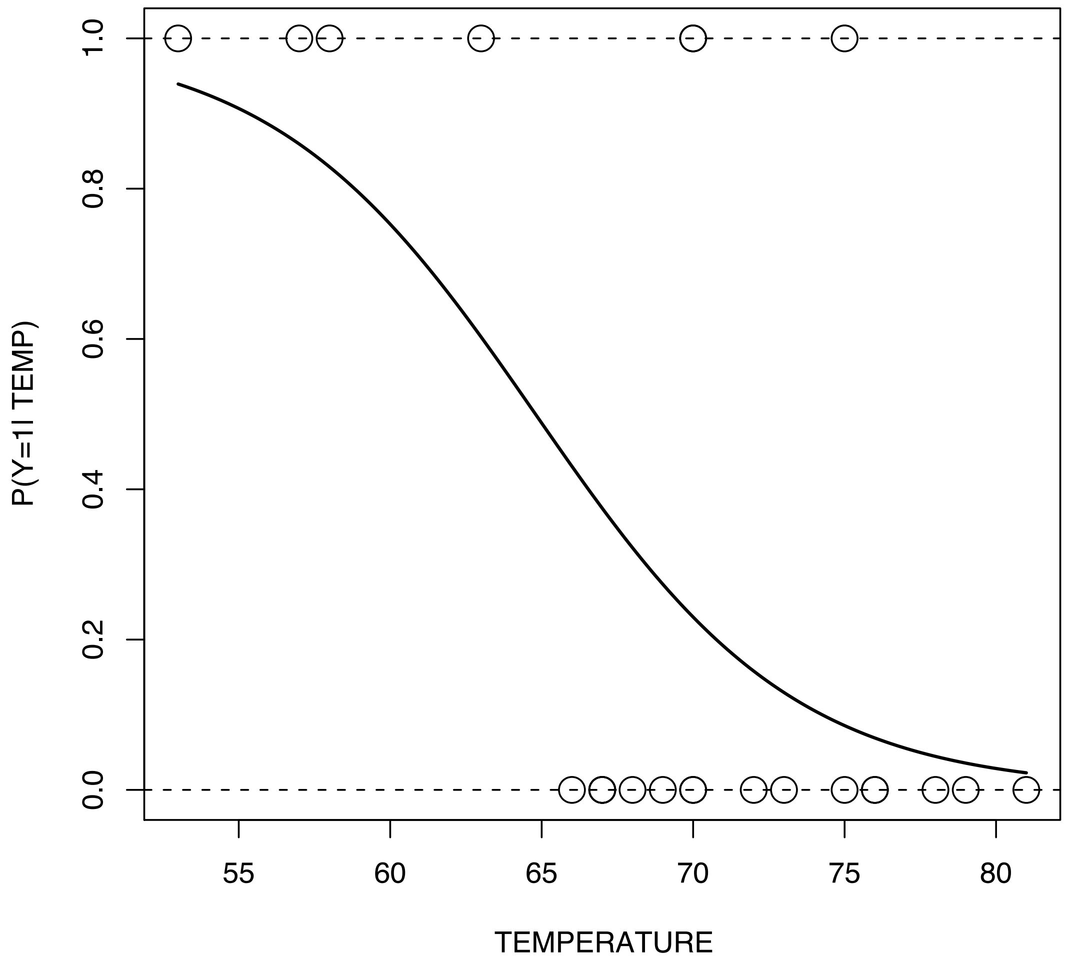
Fit logistic regression with `glm` function and plot 'link' function $f = \beta^T X$, where X is here simply temperature ($p = 1$).

```
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.
2          0             0            0.132                 0            0.
  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?