Module 4: Classification

TMA4268 Statistical Learning V2022

Stefanie Muff, Department of Mathematical Sciences, NTNU

January 31 and February 3rd, 2022

Acknowledgements

- Thanks to Mette Langaas and her TAs who permit me to use and modify their original material.
- Some of the figures and slides in this presentation are taken (or are inspired) from James et al. (2013).

Introduction

Learning material for this module

- James et al. (2013): An Introduction to Statistical Learning. Chapter 4 + Chapter 2.2.3.

- All the material presented on these module slides.

What will you learn?

- Classification and discrimination
- Logistic regression
- Bayes classifier, Bayes risk
- KNN - majority vote or estimate posterior class probability?
- Linear discriminant analysis: model, method, results.
- Quadratic discriminant analysis: model, method, results.
- Naive Bayes - when and why?
- Sensitivity, specificity and ROC curves

What is classification?

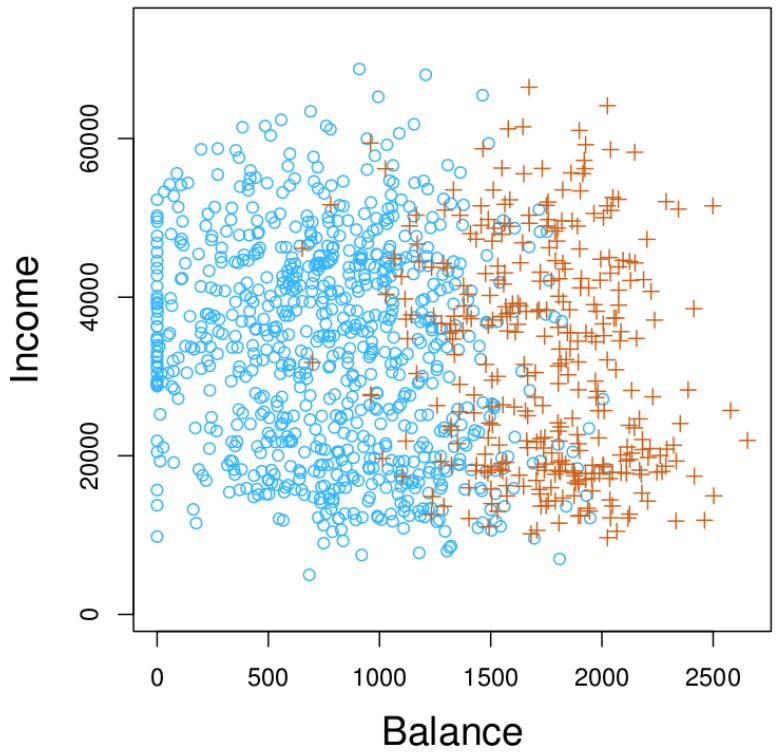
- By now our responses Y was assumed *continuous*, while covariates were allowed to be *categorical*.
- Now we allow the response to be *categorical*.
- This is even more common than continuous responses. Examples:
  - Spam filters `email` $\in$ {spam, ham},
  - `Eye color` $\in$ {blue, brown, green}.
  - `Medical condition` $\in$ {disease1, disease2, disease3}.

- Suppose we have a qualititative response value that can be a member in one of K classes $\mathcal{C} = \{c_1, c_2, \ldots, c_K\}$.
- In classification we build a function $f(X)$ that takes a vector of input variables X and predicts its class membership, such that $Y \in \mathcal{C}$.
- We would also assess the *uncertainty* in this classification. Sometimes the role of the different predictors may be of main interest.
- We often build models that **predict probabilities of categories**, *given* certain covariates X.

Example: The credit card data

The `Default` dataset is available from the ISLR package.

Aim : to predic whether an individual will default on his or her credit card payment, given the annual income and credit card balance.
Orange: default=yes, blue: default=no.



General classification setup

Set-up: Training observations $\{(x_1, y_1), ..., (x_n, y_n)\}$ where the response variable Y is categorical, e.g $Y \in \mathcal{C} = \{0, 1, ..., 9\}$ or $Y \in \mathcal{C} = \{dog, cat, ..., horse\}$.

Aim: To *build* a classifier $f(X)$ that assigns a class label from $\mathcal{C}$ to a future unlabelled observation x and to asses the *uncertainty* in this classification.

Performance measure: Most popular is the misclassification error rate (training and test version).

What are the methods?

Three methods for classification are discussed here:

- Logistic regression
- K-nearest neighbours
- Linear and quadratic discriminant analysis

Linear regression for binary classification?

Suppose we have a binary outcome, for example whether a credit card user defaults $Y =$ `yes` or `no`, given covariates X to predict Y. We could use *dummy encoding* for Y like

$$Y = \begin{cases} 0 & \text{if } \texttt{no} \ , \\ 1 & \text{if } \texttt{yes} \ . \end{cases}$$

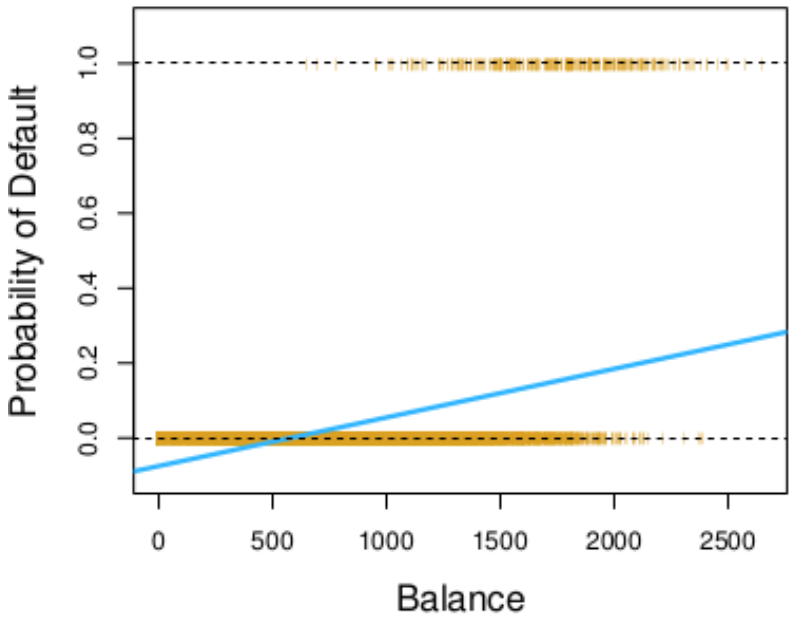
Can we simply perform a linear regression of Y on X and classify as `yes` if $\hat{Y} > 0.5$?

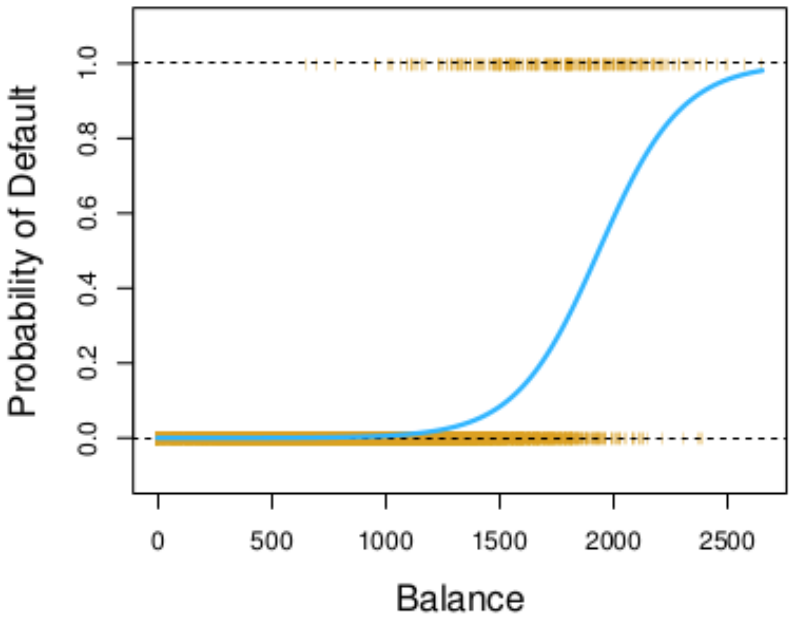
- In this case of a binary outcome, linear regression does a good job as a classifier, and is equivalent to linear discriminant analysis which we discuss later.
- However, linear regression might produce probabilities less than zero or bigger than one.

$\rightarrow$ We need to use **logistic regression**.

Linear vs. logistic regression

Let's anticipate a bit, to see why linear regression does not so well. We estimate the probability that someone defaults, given the credit card balance as predictor:





Linear regression for categorical classification?

What when there are more than two possible outcomes? For example, a medical diagnosis Y, given predictors X can be categorized as

$$Y = \begin{cases} 1 & \text{if } \texttt{stroke}\ , \\ 2 & \text{if } \texttt{drug overdose}\ , \\ 3 & \text{if } \texttt{epileptic seizure}\ . \end{cases}$$

This suggests an ordering, but this is artificial.

- Linear and logistic regression are not appropriate here.
- We need *Multiclass logistic regression* and *Discriminant Analysis*.

However:

- It is still possible to use linear regression for classification problems with two classes. It is actually not even a bad idea, and works well under some conditions. Under some standard assumptions, this linear regression (with 0 and 1 response) will in fact give the same classification as linear discriminant analysis (LDA).
- For categorical outcomes with more than two levels, it requires some extra work (multivariate Y due to the dummy variable coding).
- We leave linear regression for now.

Logistic regression

- In logistic regression we consider a classification problem with two classes.
- Assume that Y is coded ($\mathcal{C} = \{1, 0\}$ or {success, failure}), and we focus on success ($Y = 1$).
- We may assume that Y_i follows a **Bernoulli distribution** with probability of success p_i.

$$Y_i = \begin{cases} 1 \text{ with probability } p_i, \\ 0 \text{ with probability } 1 - p_i. \end{cases}$$

- **Aim**: For covariates $(X_1, \ldots, X_p)$, we want to estimate $p_i = \Pr(Y_i = 1 \mid X_1, \ldots, X_p)$.

- We need a clever way to *link* our covariates $X_1, \ldots, X_p$ with this probability p_i. Aim: want to relate the *linear predictor*

$$\eta_i = \beta_0 + \beta_1 x_{i1} + \ldots + \beta_p x_{ip}$$

  to p_i. How?

- The idea is to use a so-called *link-function* to link p_i to the linear predictor.

- In logistic regression, we use the *logistic link function*

$$\log\left(\frac{p_i}{1-p_i}\right) = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip} \ . \qquad (1)$$

- **Q:** What is the rationale behind this?

Logistic regression with one covariate

- Equation (1) can be rearranged and solved for p_i. Let's look at this for only one covariate:

$$p_i = \frac{e^{\beta_0+\beta_1 x_i}}{1 + e^{\beta_0+\beta_1 x_i}}.$$

- **Important:** These values p_i will always lie in the interval between 0 and 1, with an S-shaped curve.

Example: `Default` credit card data

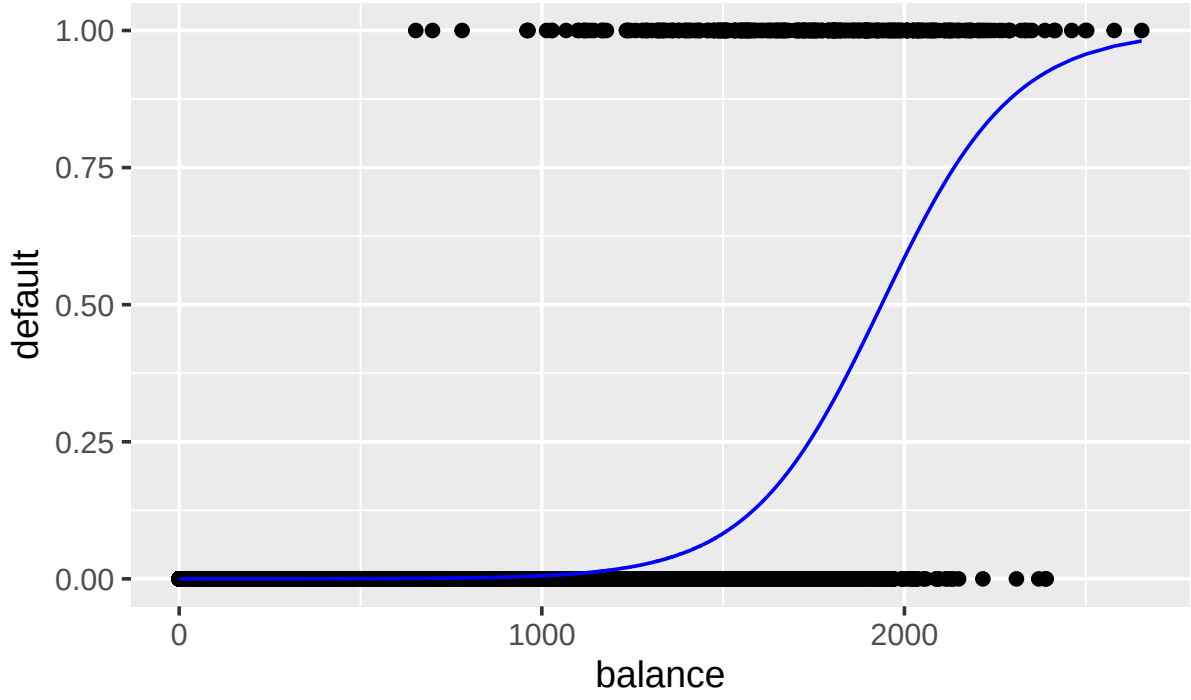
- The parameters are estimated using the method of maximum likelihood - we will look at that soon.
- Let's first do it! In R, this works with the `glm()` function, where we specify `family="binomial"`.

```
library(ISLR)
data(Default)
Default$default <- as.numeric(Default$default) - 1
glm_default = glm(default ~ balance, data = Default, family = "binomial")

summary(glm_default)$coef
```

```
##                  Estimate   Std. Error   z value      Pr(>|z|)
## (Intercept) -10.651330614 0.3611573721 -29.49221 3.623124e-191
## balance       0.005498917 0.0002203702  24.95309 1.976602e-137
```

Plotting the fitted line (in blue):



Default data: here $\hat{\beta}_0 =$ -10.65 and $\hat{\beta}_1 = 0.005$.

Estimating the regression coefficients with ML

- The coefficients $\beta_0, \beta_1, ...$ are estimated with *maximum likelihood* (ML).
- Given n independent observation pairs $\{x_i, y_i\}$, the likelihood function of a logistic regression model can be written as:

$$L(\beta) = \prod_{i=1}^{n} L_i(\beta) = \prod_{i=1}^{n} f(y_i; \beta) = \prod_{i=1}^{n} (p_i)^{y_i} (1 - p_i)^{1-y_i},$$

where $\beta = (\beta_0, \beta_1, \beta_2, \ldots, \beta_p)^T$ enters into p_i

$$p_i = \frac{\exp(\beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip})}{1 + \exp(\beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip})} .$$

- The maximum likelihood estimates are found by maximizing the likelihood.
- To make the math easier, we usually work with the log-likelihood (the log is a monotone transform, thus it will give the same result as maximizing the likelihood).

$$\begin{aligned}
\log(L(\beta)) = l(\beta) &= \sum_{i=1}^{n} \Big(y_i \log p_i + (1 - y_i) \log(1 - p_i) \Big) \\
&= \sum_{i=1}^{n} \left(y_i \log \left(\frac{p_i}{1 - p_i} \right) + \log(1 - p_i) \right) \\
&= \sum_{i=1}^{n} \Big(y_i(\beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip}) - \log(1 + e^{\beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip}}) \Big).
\end{aligned}$$

- To maximize the log-likelihood function we find the $r + 1$ partial derivatives, and set equal to 0.
- This gives us a set of $p + 1$ non-linear equations in the βs.
- This set of equations does not have a closed form solution.
- The equation system is therefore solved numerically using the *Newton-Raphson algorithm* (or Fisher Scoring).

Qualitative interpretation of the coefficients

Let's again look at the regression output:

```
summary(glm_default)$coef
```

```
##                   Estimate   Std. Error   z value      Pr(>|z|)
## (Intercept) -10.651330614 0.3611573721 -29.49221 3.623124e-191
## balance       0.005498917 0.0002203702  24.95309 1.976602e-137
```

- The z-statistic is equal to $\frac{\hat{\beta}}{SE(\hat{\beta})}$, and is approximately $N(0,1)$ distributed.[1]
- The p-value is $\Pr(|Z| > |z|)$ for a $Z \sim N(0,1)$ random variable
- Check the p-value for `Balance`. Conclusion?

[1] With this knowledge we can construct confidence intervals and test hypotheses about the βs, with the aim to understand which covariate(s) contribute to our posterior probabilites and classification.

Quantitative interpretation of the coefficients

Remember from equation (1) that

$$\log\left(\frac{p_i}{1-p_i}\right) = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + ... + \beta_p x_{ip} \ ,$$

thus

$$\frac{p_i}{1-p_i} = e^{\beta_0+\beta_1 x_{i1}+\beta_2 x_{i2}+...+\beta_p x_{ip}} = e^{\eta_i} \ .$$

The quantity $p_i/(1-p_i)$ is called the *odds*. Odds represent *chances* (e.g. in betting).

Q: Answer in mentimeter:

1. You think your football team will win tonight with a probability $p = 80\%$. What is the odds that it will win?
2. The odds for the best horse in a race is $5:1$. What is the probability that this horse will win?

Why is the odds relevant?

Let's again rearrange the *odds* in the logistic regression model:

$$\frac{p_i}{1-p_i} = \frac{\mathrm{P}(Y_i = 1 \mid X = x)}{\mathrm{P}(Y_i = 0 \mid X = x)}$$

$$= \exp(\beta_0) \cdot \exp(\beta_1 x_{i1}) \cdot \ldots \cdot \exp(\beta_p x_{ip}) \ .$$

$\rightarrow$ We have a *multiplicative model* for the odds - which can help us to interpret our βs.

The odds ratio

To understand the effect of a regression coefficient β_j, let's see what happens if we increase x_{ij} to $x_{ij} + 1$, while all other covariates are kept fixed.

Using simple algebra and the formula on the previous slide, you will see that

$$\frac{\text{odds}(Y_i = 1 \mid X_j = x_{ij} + 1)}{\text{odds}(Y_i = 1 \mid X_j = x_{ij})} = \exp(\beta_j) \ . \tag{2}$$

Interpretation:
By increasing covariate x_{ij} by one unit, we change the odds for $Y_i = 1$ by a factor $\exp(\beta_j)$.

Moreover:
Taking the log on equation (2), it follows that β_j can be interpreted as a **log odds-ratio**.

Let's now fit the logistic regression model for `default`, given `balance`, `income` and the binary variable `student` as predictors:

```
glm_default2 = glm(default ~ balance + income + student, data = Default,
    family = "binomial")

summary(glm_default2)$coef
```

```
##                   Estimate   Std. Error    z value      Pr(>|z|)
## (Intercept) -1.086905e+01 4.922555e-01 -22.080088 4.911280e-108
## balance      5.736505e-03 2.318945e-04  24.737563 4.219578e-135
## income       3.033450e-06 8.202615e-06   0.369815  7.115203e-01
## studentYes  -6.467758e-01 2.362525e-01  -2.737646  6.188063e-03
```

Questions:

- What happens with the odds to default when `income` increases by 10'000 dollars?
- What happens with the odds to default when `balance` increases by 100 dollars?

Predictions

Let's catch up. Why were we doing all this in the first place?

Answer: We wanted to build a model that **predict probabilities of categories** Y, *given* certain covariates $X_1, \ldots, X_p$.

- For given parameter estimates $\hat{\beta}_0, \hat{\beta}_1, \ldots \hat{\beta}_p$ and a new observation x_0, we can estimate the probability $\hat{p}(x_0)$ that the new observation belongs to the class defined by $Y = 1$

$$\hat{p}(x_0) = \frac{e^{\hat{\eta}_0}}{1 + e^{\hat{\eta}_0}} \ ,$$

with linear predictor

$$\hat{\eta}_0 = \hat{\beta}_0 + \hat{\beta}_1 x_{01} + \ldots + \hat{\beta}_p x_{0p} \ .$$

In the case of qualitative covariates, a dummy variable needs to be introduced. This can be done as for linear regression.

So in the `Default` example, we can predict the probability that someone defaults.

For example: "What is the estimated probability that a student defaults with a balance of 2000, and an income of 40000?"

$$\hat{p}(X) = \frac{e^{\beta_0 + 2000 \cdot \beta_1 + 40000 \cdot \beta_2 + 1 \cdot \beta_3}}{1 + e^{\beta_0 + 2000 \cdot \beta_1 + 40000 \cdot \beta_2 + 1 \cdot \beta_3}} = 0.5196$$

Using R:

```
eta <- summary(glm_default2)$coef[, 1] %*% c(1, 2000, 40000, 1)
exp(eta)/(1 + exp(eta))
```

```
##              [,1]
## [1,] 0.5196218
```

(Or via the `predict()` function in R.)

Example: South African heart disease data set

- `SAheart` data set from the `ElemStatLearn` package, a retrospective sample of males in a heart-disease high-risk region in South Africa.
- 462 observations on 10 variables.
- All subjects are male in the age range 15-64.
- 160 cases (individuals who have suffered from a conorary heart disease) and 302 controls (individuals who have not suffered from a conorary heart disease).

The response value (`chd`) and covariates

- `chd` : conorary heart disease {yes, no} coded by the numbers {1, 0}
- `sbp` : systolic blood pressure
- `tobacco` : cumulative tobacco (kg)
- `ldl` : low density lipoprotein cholesterol
- `famhist` : family history of heart disease. Categorical variable with two levels: {Absent, Present}.
- `obesity` : a numerical value
- `alcohol` : current alcohol consumption
- `age` : age at onset

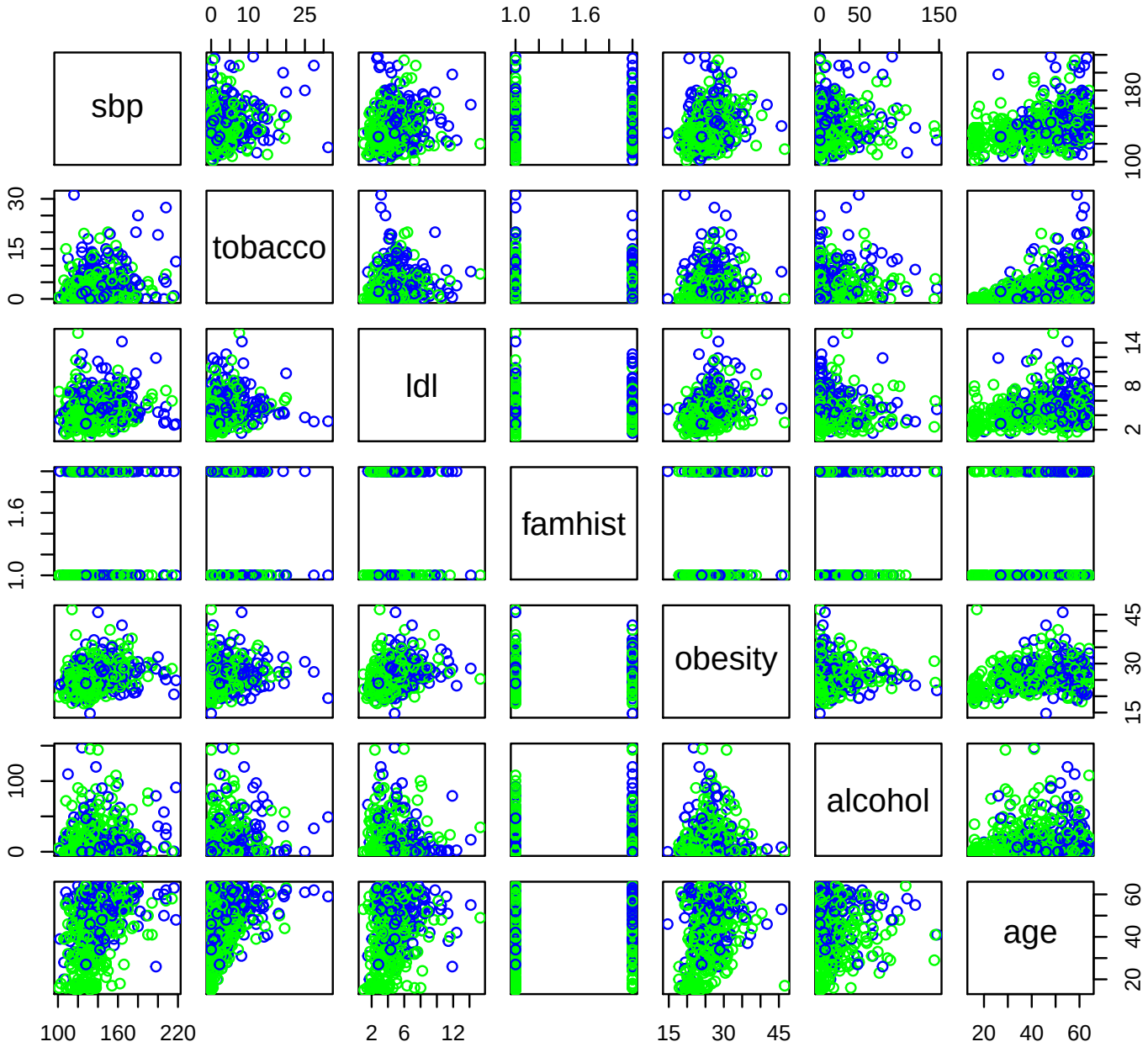
The goal is to identify important risk factors. We start by loading and looking at the data:

```r
library(ElemStatLearn)

d.heart <- SAheart
d.heart$chd <- as.factor(d.heart$chd)
d.heart <- d.heart[, c("sbp", "tobacco", "ldl", "famhist", "obesity",
    "alcohol", "age", "chd")]
head(d.heart)
```

```
##   sbp tobacco  ldl famhist obesity alcohol age chd
## 1 160   12.00 5.73 Present   25.30   97.20  52   1
## 2 144    0.01 4.41  Absent   28.87    2.06  63   1
## 3 118    0.08 3.48 Present   29.14    3.81  46   0
## 4 170    7.50 6.41 Present   31.99   24.26  58   1
## 5 134   13.60 3.50 Present   25.99   57.34  49   1
## 6 132    6.20 6.47 Present   30.77   14.14  45   0
```

`pairs()` plot with $Y = 1$ (case) in green and $Y = 0$ (control) in blue:



Fitting the model using all predictors in R:

```r
glm_heart = glm(chd ~ ., data = d.heart, family = "binomial")
summary(glm_heart)
```

```
##
## Call:
## glm(formula = chd ~ ., family = "binomial", data = d.heart)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.7517  -0.8378  -0.4552   0.9292   2.4434
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)    -4.1295997  0.9641558  -4.283 1.84e-05 ***
## sbp             0.0057607  0.0056326   1.023  0.30643
## tobacco         0.0795256  0.0262150   3.034  0.00242 **
## ldl             0.1847793  0.0574115   3.219  0.00129 **
## famhistPresent  0.9391855  0.2248691   4.177 2.96e-05 ***
## obesity        -0.0345434  0.0291053  -1.187  0.23529
## alcohol         0.0006065  0.0044550   0.136  0.89171
## age             0.0425412  0.0101749   4.181 2.90e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 596.11  on 461  degrees of freedom
## Residual deviance: 483.17  on 454  degrees of freedom
## AIC: 499.17
##
## Number of Fisher Scoring iterations: 4
```

- For which predictors do we have evidence that they are associated with CHD?
- How would you now calculate $\hat{p}(X)$ that someone will develop CHD, given covariates X?

R-hint: The `predict()` function can be used to get predicted probabilities using

```
predict(glm_heart, newdata = ..., type = "response")
```

The Bayes classifier

- We were going for classification, but now estimated $\hat{p}(X) = \Pr(Y \mid X)$ using logistic regression. Idea: probability can be used for classification.
- Assume that we know or can estimate the probability that a new observation x_0 belongs to class k, for K classes $\mathcal{C} = \{c_1, c_2, \ldots, c_K\}$, with elements numbered as $1, 2, ..., K$

$$p_k(x_0) = \Pr(Y = k | X = x_0), \quad k = 1, 2, ...K \ .$$

  This is the probability that $Y = k$ given the observation x_0.

- The *Bayes classifier assigns an observation to the most likely class*, given its predictor values.
- **Example** for two groups $\{A, B\}$. A new observation x_0 will be classified to A if $\Pr(Y = A | X = x_0) > 0.5$ and to class B otherwise.

Properties of the Bayes classifier

- It has the *smallest test error rate*.
- The class boundaries using the Bayes classifier is called the *Bayes decision boundary*.
- The overall Bayes error rate is given as

$$1 - \mathrm{E}(\max_j \Pr(Y = j \mid X))$$

  where the expectation is over X.
- The Bayes error rate is comparable to the *irreducible error* in the regression setting.

Some terminology

Training set: Independent observations $\{(x_1, y_1), ..., (x_n, y_n)\}$ with *qualitative response* variable $Y \in \{1, 2, ..., K\}$, used to construct the classification rule (by estimating parameters in class densities or posterior probabilites).

Test set: Independent observations of the same format as the training set, used to evaluate the classification rule.

Loss function: The misclassifications are given the loss 1 and the correct classifications loss 0 - this is called *0/1-loss*.

Training error

- **Training error rate**: The proportion of mistakes that are made if we apply our estimator $\hat{f}$ to the training observations, i.e. $\hat{y}_i = \hat{f}(x_i)$

$$\frac{1}{n} \sum_{i=1}^{n} \mathrm{I}(y_i \neq \hat{y}_i) \ ,$$

  with indicator function I, which is defined as:

$$\mathrm{I}(a \neq \hat{a}) = \begin{cases} 1 \text{ if } a \neq \hat{a} \ , \\ 0 \text{ else.} \end{cases}$$

- The training error rate is the fraction of misclassifications made on our training set.
- A very low training error rate may imply overfitting.

Test error

- **Test error rate**: The fraction of misclassifications when our model is applied on a test set

$$\text{Ave}(I(y_0 \neq \hat{y}_0)) \ ,$$

  where the average is over all the test observations (x_0, y_0).

- Again, this gives a better indication of the true performance of the classifier than the training error (remember why?).

- We assume that a *good* classifier is a classifier that has a *low test error*.

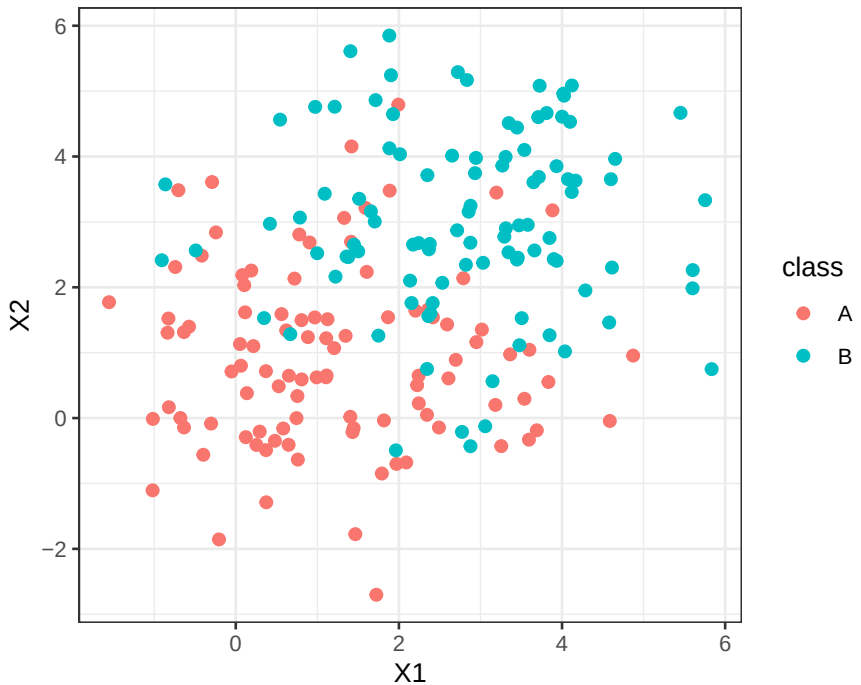
The K-nearest neighbour (KNN) classifier

- **Caveat** for the Bayes classifier: we usually don't know the true conditional distribution of $\Pr(Y|X)$ for real data.
- We have discussed how to estimate it with logistic regression (for two categories).
- Alternative: K-nearest neighbor (KNN) classifier estimates this conditional distribution *non-parametrically* and chooses the most likely cateogry (Bayes classifier).[2]

[2] Attention!! K refers to the number of neighbours used for the classifier, and *not* to the number of classes!! The latter is assumed known.

A synthetic example

- Simulate 2×100 observations from a bivariate normal distribution with mean vectors $\mu_A = (1, 1)^T$, $\mu_B = (3, 3)^T$, and covariance matrix $\Sigma_A = \Sigma_B = \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix}$.



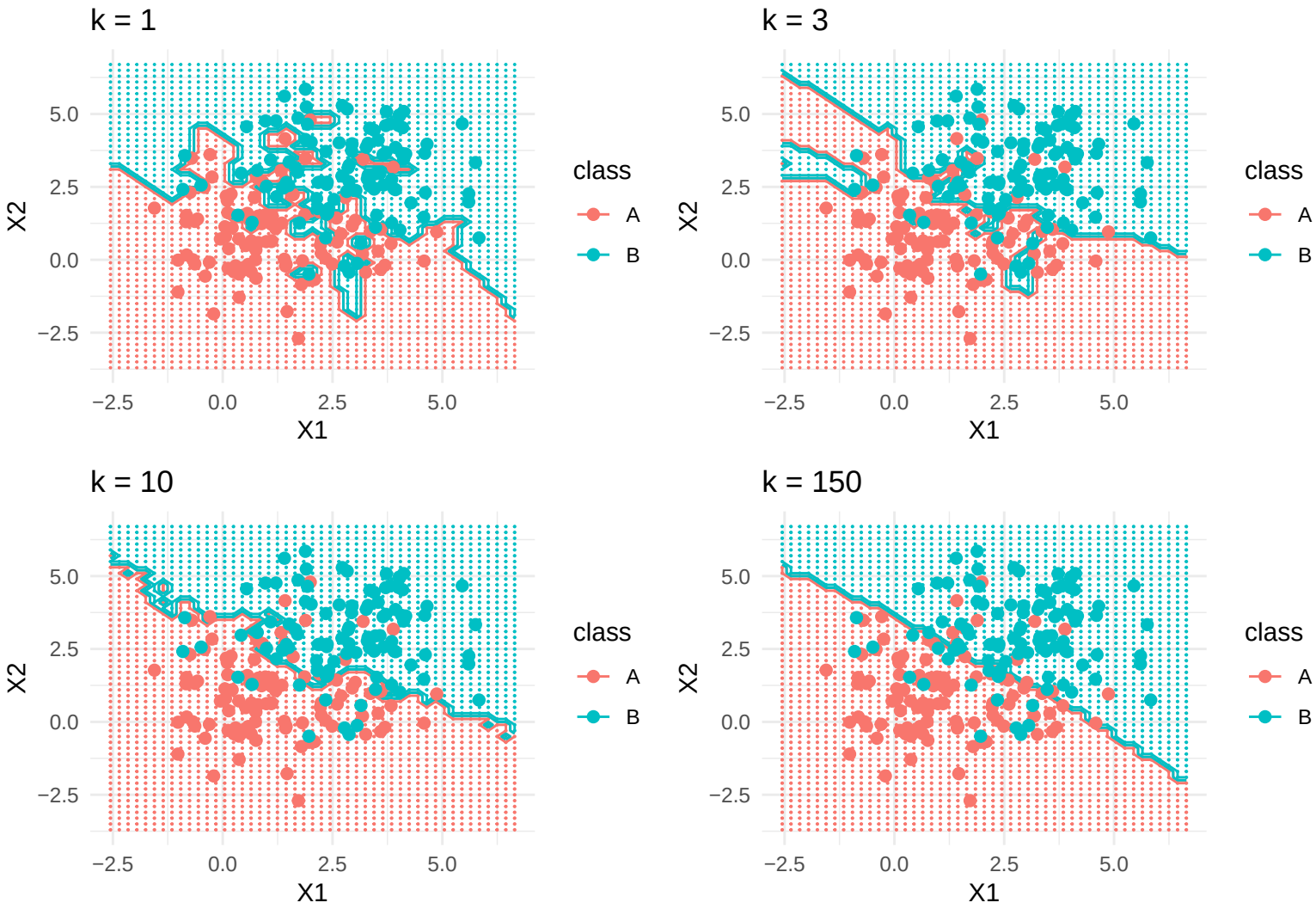
- Aim: Find a rule to classify a new observation to A or B, given only the data points (not the knowledge about the true parameters).

The K-nearest neighbour classifier (KNN) works in the following way:

- Given a new observation x_0 it searches for the K points in our training data that are closest to it (Euclidean distance).
- These points make up the neighborhood of x_0, $\mathcal{N}_0$.
- Classification is done by a *majority vote*: x_0 is classified to the most occurring class among its neighbors

$$\Pr(Y = j | X = x_0) = \frac{1}{K} \sum_{i \in \mathcal{N}_0} I(y_i = j) \ .$$

Illustration:



- The small colored dots show the predicted classes for an evenly-spaced grid.
- The lines show the decision boundaries.

How to choose K?

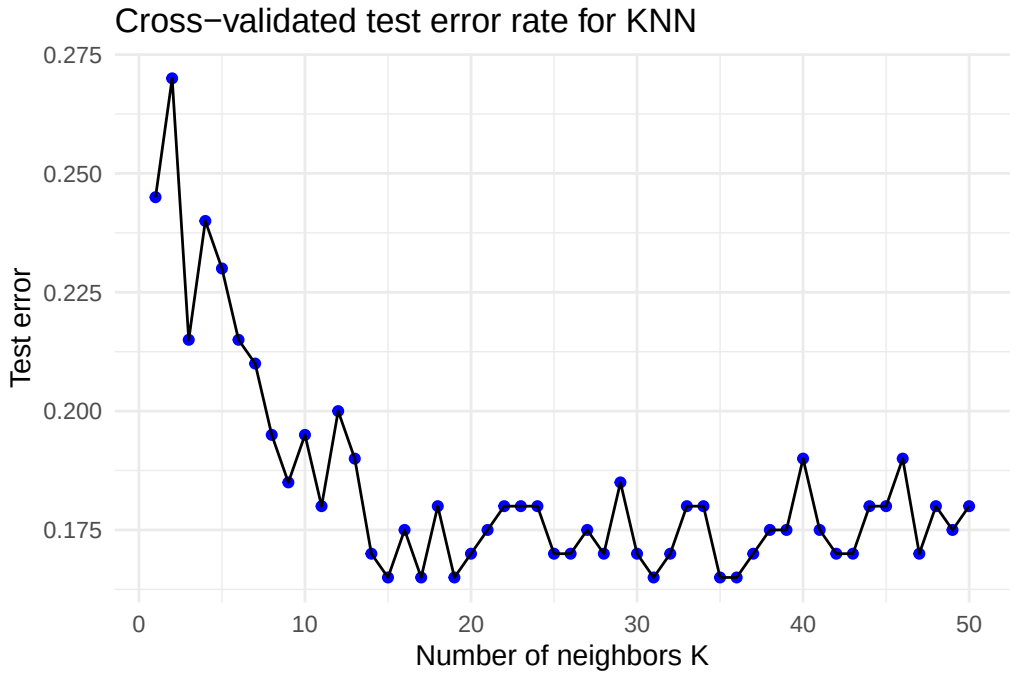
- $K = 1$: the classification is made to the same class as the one nearest neighbor.
- K large: the decision boundary tends towards a straight line (which is the Bayes boundary in this set-up).

Discuss:

- Depending on the choice of K, when is the bias large, when is the variance large?
- How to find the optimal K?

Bias-variance trade-off in a classification setting

Finding the *optimal value* of K: Test the predictive power for different K, for example by cross-validation (Module 5).



The curse of dimensionality

- KNN can be quite good if the number of predictors p is small and the number of observations n is large. We need enough close neighbors to make a good classification.
- The effectiveness of the KNN classifier falls quickly when the dimension of the preditor space is high.
- Why? Because the nearest neighbors tend to be far away in high dimensions and the method is no longer local. This is referred to as the *curse of dimensionality*.

Bayes decision rule - two paradigms

Two approaches to estimate $\Pr(Y = k \mid X = x)$:

Diagnostic paradigm

This is what we did so far: The focus was *directly* estimating the posterior distribution for the classes

$$\Pr(Y = k \mid X = x) \ .$$

Examples: Logistic regression. KNN classification.

Sampling paradigm

- Indirect approach: Model the conditional distribution of predictors $f_k(x) = \Pr(X = x \mid Y = k)$ for each class, and the prior probabilities $\pi_k = \Pr(Y = k)$.
- Then classify to the class with the maximal product $\pi_k f_k(x)$. In this paradigm, we need to model the pdf for each class.

Given a continuous X and categorical Y, and

- the probability *density* function $f_k(x) = \Pr(X = x \mid Y = k)$ for X in class k.
- the *prior* probability for class k, $\pi_k = \Pr(Y = k)$, is the prior probability.

How do we get $\Pr(Y = k|X = x_0)$? That is, how can we "flip" the conditioning around?

Bayes theorem

$$p_k(X) = \Pr(Y = k \mid X = x) = \frac{\Pr(X = x \cap Y = k)}{f(x)}$$
$$= \frac{f_k(x)\pi_k}{\sum_{l=1}^{K} f_l(x)\pi_l} \ . \tag{3}$$

Discriminant Analysis

- Discriminant analysis is relying on the *sampling paradigm*.
- The approach is to model the distribution of X in each of the classes separately, and then use Bayes theorem to flip things around and obtain $\Pr(Y \mid X)$.

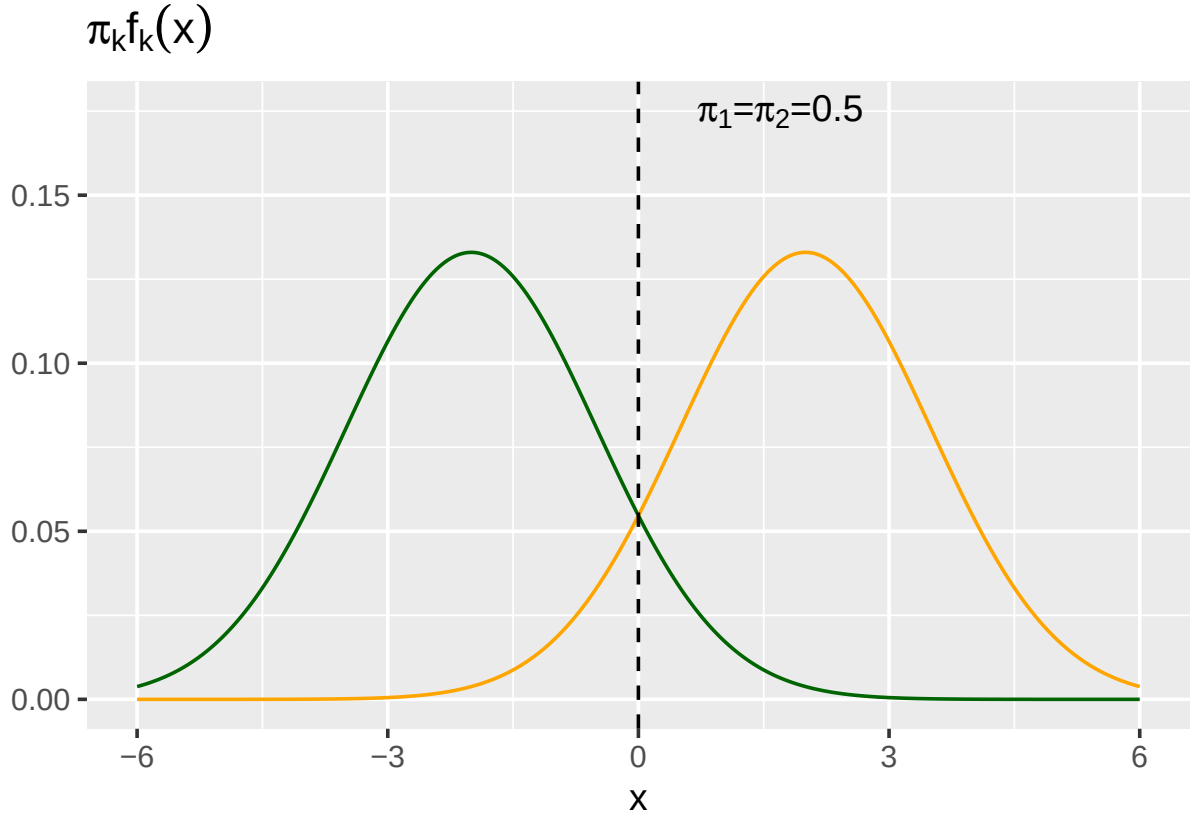
Example

Suppose we have observations coming from two classes:
{green, orange}

$$X_{\text{green}} \sim \mathcal{N}(-2, 1.5^2) \text{ and } X_{\text{orange}} \sim \mathcal{N}(2, 1.5^2)$$

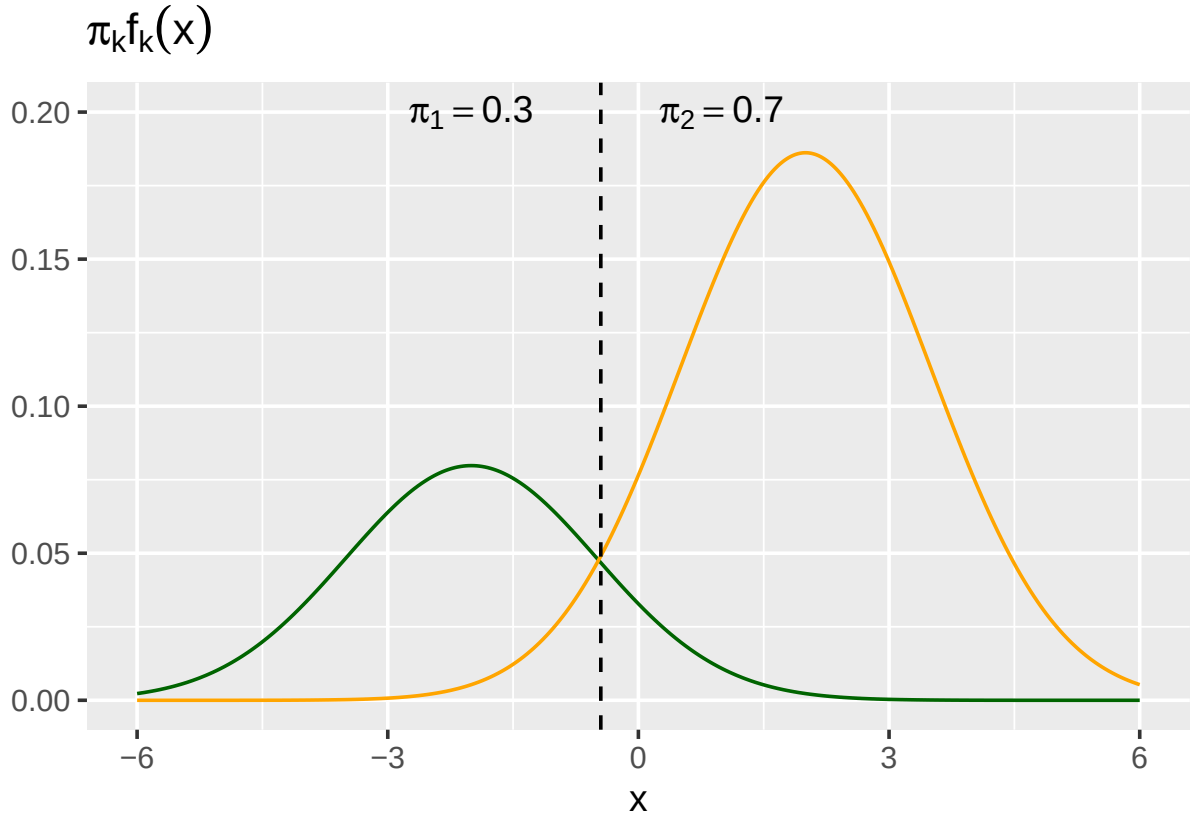
Assume probabilities to be equal, $\pi_1 = \pi_2 = 0.5$.

We plot $\pi_k f_k(x)$ for the two classes:



The decision boundary is where the point of intersection of the two lines is, because here $\pi_1 f_1(x) = \pi_2 f_2(x)$.

For different priors $\pi_1 = 0.3$ and $\pi_2 = 0.7$, the decision boundary shifts to the left:



Why discriminant analysis?

- Linear discriminant analysis is more stable than logistic regression when
  - the classes are well-separated. In that case, the parameter estimates for the logistic regression model are very unstable.
  - n is small and the distribution of the predictors X is approximately normal in each of the classes.

- Moreover, linear discriminant analysis is popular when we have more than two response classes.

Linear discriminant analysis (LDA) when $p = 1$

- Class conditional distributions $f_k(X)$ are assumed normal (Gaussian) for $k = 1, \ldots, K$, that is

$$f_k(x) = \frac{1}{\sqrt{2\pi}\sigma_k} e^{-\frac{1}{2}\left(\frac{x-\mu_k}{\sigma_k}\right)^2}$$

  has parameters μ_k (mean) and σ_k (standard deviation).

- With LDA we assume that all of the classes have the *same standard deviation* $\sigma_k = \sigma$.

- In addition we have prior class probabilites $\pi_k = \Pr(Y = k)$, so that $\sum_{k=1}^{K} \pi_k = 1$.

We can insert the expression for each class distribution into Bayes formula to obtain the posterior probability $p_k(x) = \Pr(Y = k|X = x)$

$$p_k(x) = \frac{f_k(x)\pi_k}{f(x)} = \frac{\pi_k \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}\left(\frac{x-\mu_k}{\sigma}\right)^2}}{\sum_{l=1}^{K} \pi_l \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}\left(\frac{x-\mu_l}{\sigma}\right)^2}} \ .$$

Our rule is to classify to the class for which $p_k(x)$ is largest.

Taking logs, and discarding terms that do not depend on k, we see that this is equivalent to assigning x to the class with the largest *discriminant score* $\delta_k(x)$:

$$\delta_k(x) = x \cdot \frac{\mu_k}{\sigma^2} - \frac{\mu_k^2}{2\sigma^2} + \log(\pi_k).$$

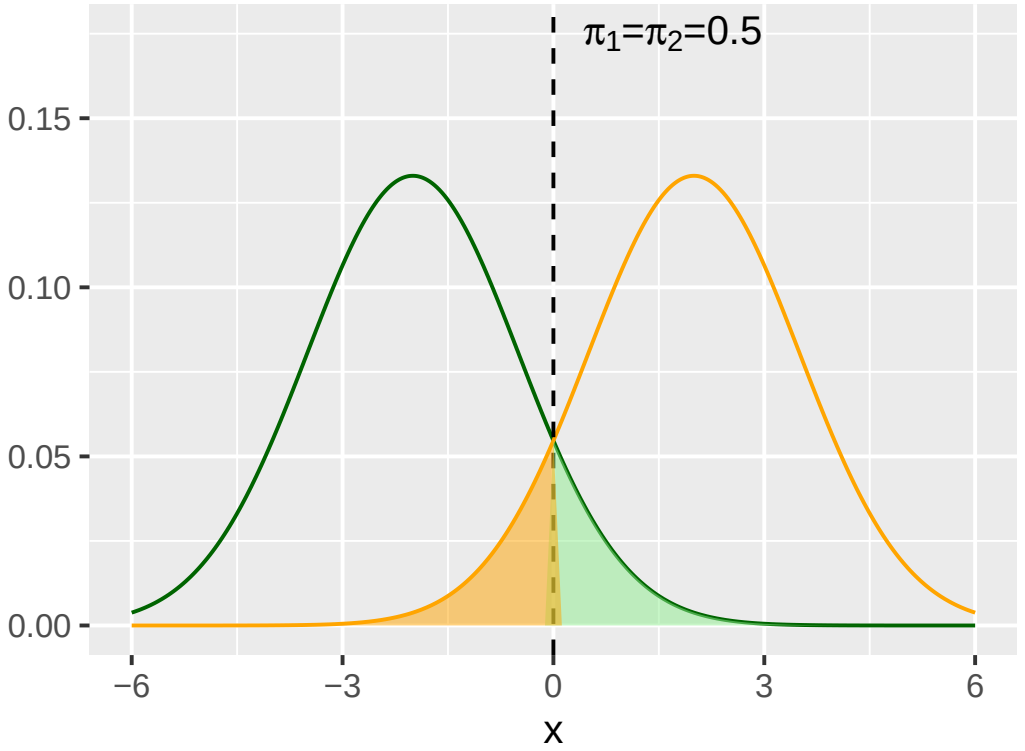
- This decision boundaries between the classes are *linear* in x.
- For $K = 2$ classes and $\pi_1 = \pi_2$, the decision boundary is at

$$x = \frac{\mu_1 + \mu_2}{2} \ .$$

(Show this by setting $\delta_1(x) = \delta_2(x)$ and resolving for x.)

Back to our example

$$X_{\text{green}} \sim \mathcal{N}(-2, 1.5^2) \text{ and } X_{\text{orange}} \sim \mathcal{N}(2, 1.5^2)$$



- The Bayes decision boundary is at $x = 0$.
- Bayes error rate: `round(pnorm(0,2,1.5))`$=0.09$.
- The Bayes classifier has the lowest test error rate.

In the above example we knew the true distributions $p_k(X)$ and the priors π_k. But typically we don't know these parameters, we only have the training data.

Idea: we simply estimate the parameters and plug them into the rule.

Parameter estimators

- Prior probability for class k is (often) estimated by taking the fraction of observations n_k (out of n) coming from class k: $\hat{\pi}_k = \frac{n_k}{n}$.
- The mean value for class k is simply the sample mean of all observations from class k:

$$\hat{\mu}_k = \frac{1}{n_k} \sum_{i:y_i=k} x_i.$$

- The standard deviation: sample standard deviation across all classes ("pooled" standard deviation):

$$\hat{\sigma}^2 = \frac{1}{n-K} \sum_{k=1}^{K} \sum_{i:y_i=k} (x_i - \hat{\mu}_k)^2 = \sum_{k=1}^{K} \frac{n_k - 1}{n-K} \cdot \hat{\sigma}_k^2.$$

  $\hat{\sigma}_k$: estimated standard deviation of all observations from class k.

How to test the goodness of the estimator?

1. Use the training set to estimate parameters and class boundary.
2. Use the test set to estimate misclassification rate.

Simulated example:

```
n = 1000
pi1 = pi2 = 0.5
mu1 = -2
mu2 = 2
sigma = 1.5
set.seed(1)
n1train = rbinom(1, n, pi1)
n2train = n - n1train
n1test = rbinom(1, n, pi1)
n2test = n - n1test
train1 = rnorm(n1train, mu1, sigma)
train2 = rnorm(n2train, mu2, sigma)
test1 = rnorm(n1test, mu1, sigma)
test2 = rnorm(n2test, mu2, sigma)
var2.1 = var(train1)
var2.2 = var(train2)
var.pool = ((n1train - 1) * var2.1 + (n2train - 1) * var2.2)/(n - 2)
```

Then set

$$\hat{\delta}_1(x) = \hat{\delta}_2(x)$$

and resolve for x to obtain a decision rule (boundary).

Exercise: Verify that the following code will give you the training and test error rates:

```
rule = 0.5 * (mean(train1) + mean(train2)) + var.pool * (log(n2train/n) -
    log(n1train/n))/(mean(train1) - mean(train2))

trainingError <- (sum(train1 > rule) + sum(train2 < rule))/n
testError <- (sum(test1 > rule) + sum(test2 < rule))/n

c(trainingError, testError)
```

```
## [1] 0.105 0.115
```

This is a rather good performance, compared to the minimal Bayes error rate. But keep in mind that the LDA classifier relies on the Normal assumption, and that $\sigma_k = \sigma$ for all classes is assumed[3].

[3] Both of which we knew were fulfilled here.

The confusion matrix

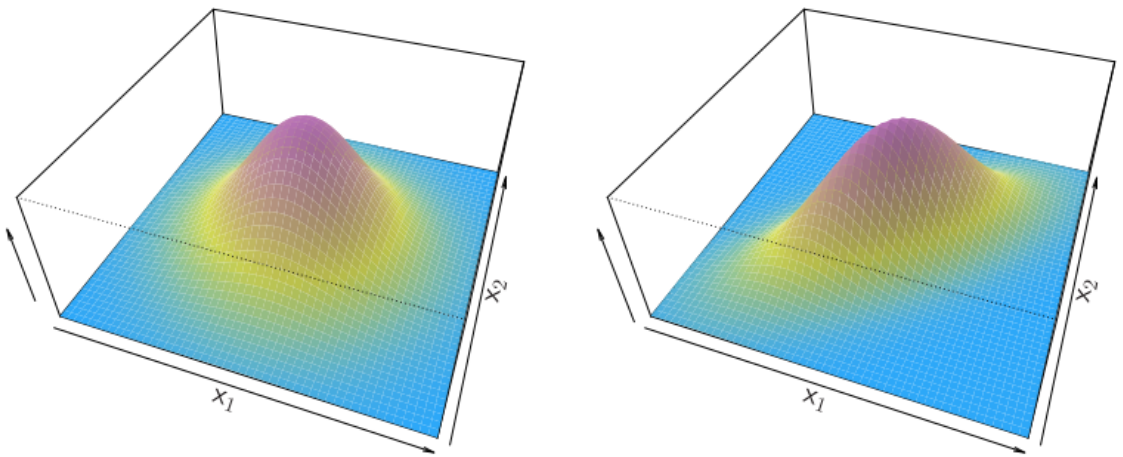
- The confusion matrix is a table that can show the performance of a classifier, given that the true values are known.
- We can make a confusion matrix from the training or test set
- The sum of the diagonal is the total number of correct classifications. The sum of all elements off the diagonal is the total number of misclassifications.

- The confusion matrix can be obtained in `R` by using the `table` function, or directly using the `caret` package.

Multivariate LDA ($p > 1$)

- LDA can be generalized to situations when $p > 1$ covariates are used. The decision boundaries are still linear.
- The multivariate normal distribution function:

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



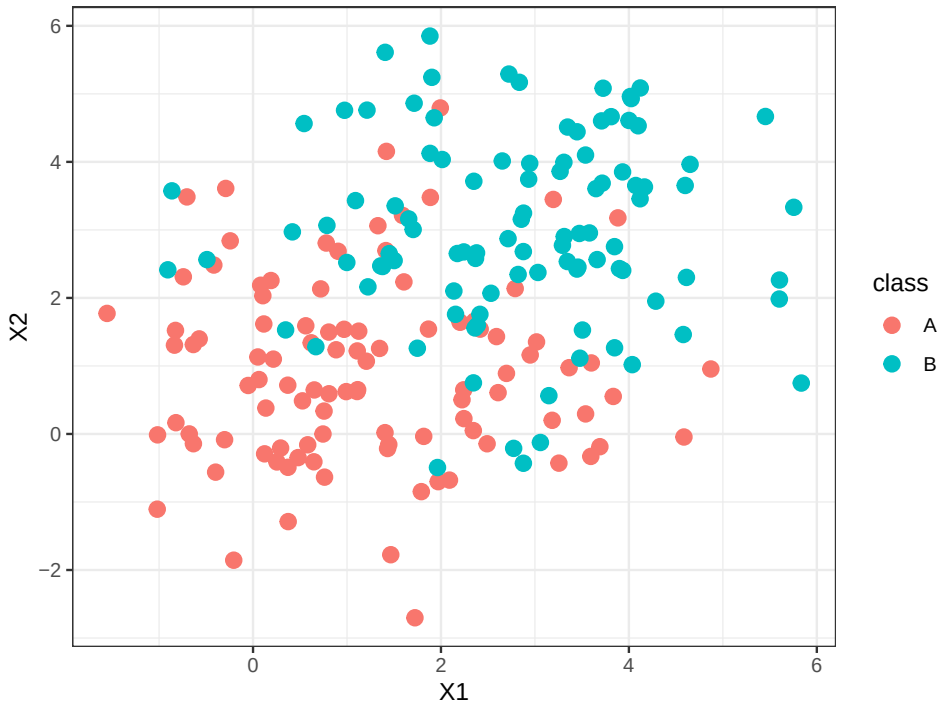
- Plugging this density into equation (3) gives the following expression for the discriminant function:

$$\delta_k(x) = x^T \Sigma^{-1} \mu_k - \frac{1}{2} \mu_k^T \Sigma^{-1} \mu_k + \log \pi_k.$$

- Note: $\delta_k(x) = c_{k0} + c_{k1} x_1 + ... + c_{kp} x_p$ is a linear function in $(x_1, ..., x_p)$.

Back to our synthetic example

- Consider again our simulation from a bivariate normal distribution with mean vectors $\mu_A = (1,1)^T$, $\mu_B = (3,3)^T$, and covariance matrix $\Sigma_A = \Sigma_B = \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix}$.
- Aim: Use LDA to classify a new observation x_0 to class A or B.



- Since *the truth is known here*, we can calculate the Bayes boundary and the Bayes error.
- Since we have bivariate normal class distributions with common covariance matrix, the optimal boundary is given by LDA, with boundary given at $\delta_A(x) = \delta_B(x)$.

$$x^T\Sigma^{-1}\mu_A - \frac{1}{2}\mu_A^T\Sigma^{-1}\mu_A + \log\pi_A = x^T\Sigma^{-1}\mu_B - \frac{1}{2}\mu_B^T\Sigma^{-1}\mu_B + \log\pi_B$$

$$x^T\Sigma^{-1}(\mu_A - \mu_B) - \frac{1}{2}\mu_A^T\Sigma^{-1}\mu_A + \frac{1}{2}\mu_B^T\Sigma^{-1}\mu_B + \log\pi_A - \log\pi_B = 0$$

Inserting numerical values gives $-x_1 - x_2 + 4 = 0$, thus a boundary with functional form

$$x_2 = 4 - x_1 \ .$$

Confusion matrix for the synthetic example

We can use the Bayes boundary to find the error rate:

```
r.pred <- ifelse(df$X2 < 4 - df$X1, "A", "B")
table(real = df$class, r.pred)
```

```
##      r.pred
## real  A  B
##    A 82 18
##    B 21 79
```

Of course, the Bayes boundary is usually not known, and we must estimate it from the data.

Estimators for p>1:

- Prior probability for class k (unchanged from $p = 1$): $\hat{\pi}_k = \frac{n_k}{n}$.
- The mean value for class k is simply the sample mean of all observations from class k (but now these are vectors):

$$\hat{\mu}_k = \frac{1}{n_k} \sum_{i:y_i=k} X_i.$$

- The covariance matrices for each class:

$$\hat{\Sigma}_k = \frac{1}{n_k - 1} \sum_{i:y_i=k} (X_i - \hat{\mu}_k)(X_i - \hat{\mu}_k)^T$$

- Pooled version:

$$\hat{\Sigma} = \sum_{k=1}^{K} \frac{n_k - 1}{n - K} \cdot \hat{\Sigma}_k.$$

Analysing the synthetic example with `lda()`

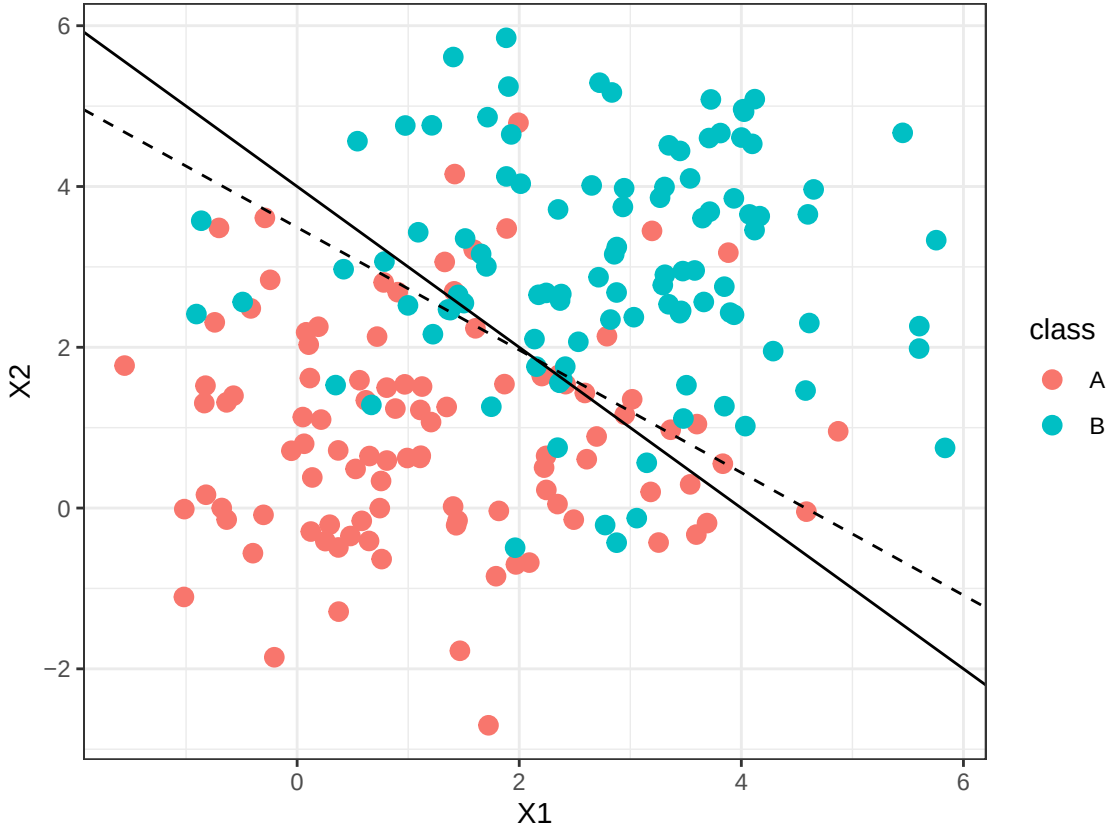
```
r.lda <- lda(class ~ X1 + X2, df)
r.pred <- predict(r.lda, df)$class
table(real = df$class, predicted = r.pred)
```

```
##     predicted
## real  A  B
##    A 87 13
##    B 18 82
```

Note: The training error is smaller than for the Bayes boundary. Why?

Comparison of the Bayes with the estimated boundary

Solid line: Bayes boundary
Dashed line: Estimated boundary



Posterior probabilites

- Sometimes the probability that an observation comes from a class k is more interesting than the actual classification itself.
- These class probabilities can be estimated from the priors and class conditional distributions, or from the discriminant functions:

$$\begin{aligned}\hat{P}(Y=k|X=x) &= \frac{\hat{\pi}_k \cdot \frac{1}{(2\pi)^{p/2}|\hat{\Sigma}|^{1/2}} \exp(-\frac{1}{2}(x-\hat{\mu}_k)^T \hat{\Sigma}^{-1}(x-\hat{\mu}_k))}{\sum_{l=1}^{K} \hat{\pi}_l \frac{1}{(2\pi)^{p/2}|\hat{\Sigma}|^{1/2}} \exp(-\frac{1}{2}(x-\hat{\mu}_l)^T \hat{\Sigma}^{-1}(x-\hat{\mu}_l))} \\ &= \frac{e^{\hat{\delta}_k(x)}}{\sum_{l=1}^{K} e^{\hat{\delta}_l(x)}}.\end{aligned}$$

Quadratic Discriminant Analysis (QDA)

- In LDA we assumed that $\Sigma_k = \Sigma$ for all classes.
- In QDA we allow different covariance matrices Σ_k for each class, while the predictors are still multivariate Gaussian

$$X \sim N(\mu_k, \Sigma_k) \ .$$

- The discriminant functions are now given by:

$$\begin{aligned}\delta_k(x) &= -\frac{1}{2}(x-\mu_k)^T\Sigma_k^{-1}(x-\mu_k) - \frac{1}{2}\log|\Sigma_k| + \log\pi_k \\ &= -\frac{1}{2}x^T\Sigma_k^{-1}x + x^T\Sigma_k^{-1}\mu_k - \frac{1}{2}\mu_k^T\Sigma_k^{-1}\mu_k - \frac{1}{2}\log|\Sigma_k| + \log\pi_k.\end{aligned}$$

- These decision boundaries are *quadratic* functions of x.

LDA vs QDA

QDA is more flexible than LDA, as it allows for group-specific covariance matrices.

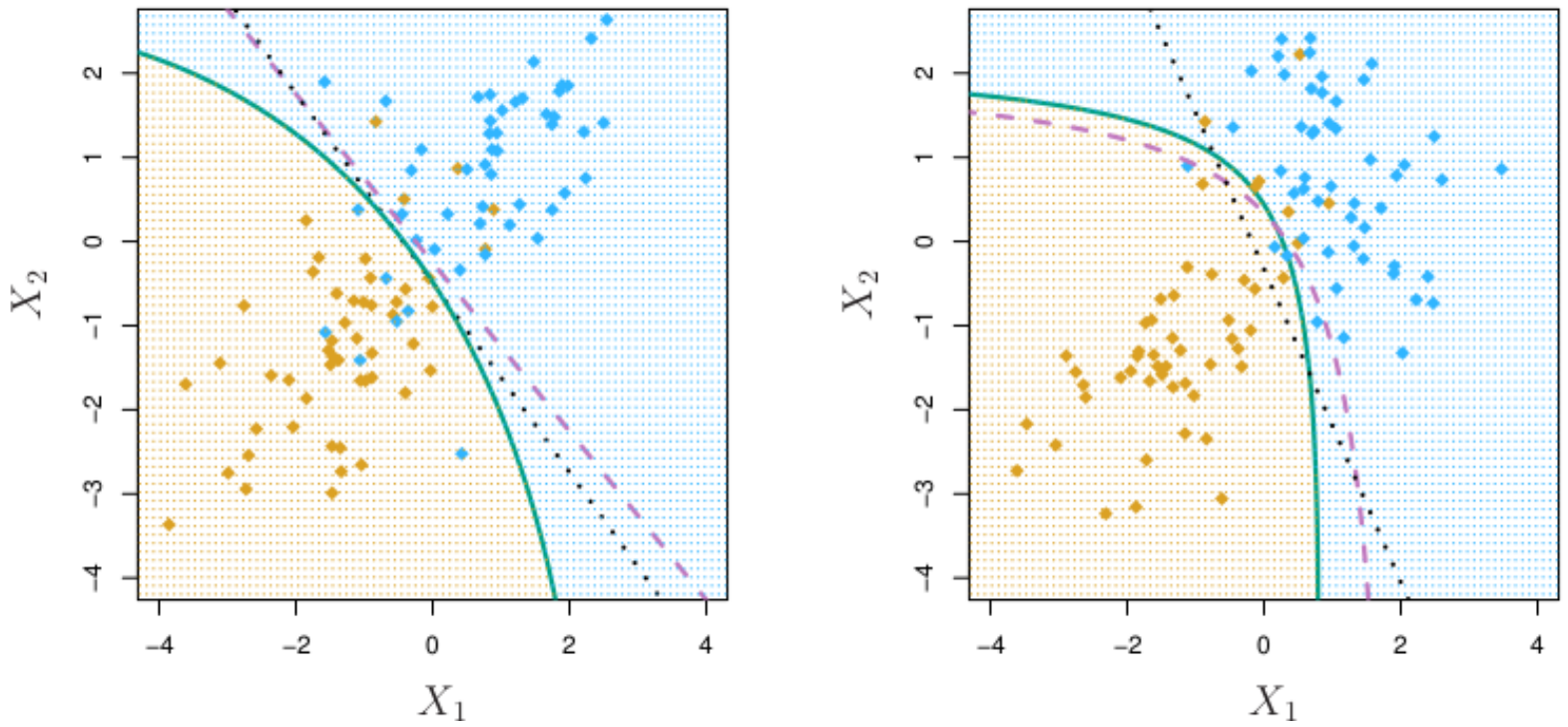
Q:

- But, if the covariance matrices in theory are equal - will they not be estimated equal?
- Should we not always prefer QDA to LDA?

A:

LDA vs QDA – Illustration

Bayes (purple dashed), LDA (black dotted) and QDA (green solid) decision boundaries for the cases where $\Sigma_1 = \Sigma_2$ (left) and $\Sigma_1 \neq \Sigma_2$ (right).



Example: Which type of iris species?

The `iris` flower data set was introduced by the British statistician and biologist Ronald Fisher in 1936.

- **Three plant species**: {setosa, virginica, versicolor}.
- **Four features**: `Sepal.Length`, `Sepal.Width`, `Petal.Length` and `Petal.Width`.

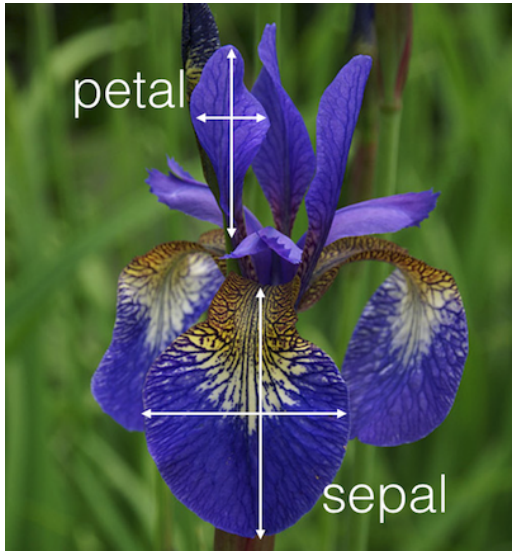


Figure 1: Iris plant with sepal and petal leaves

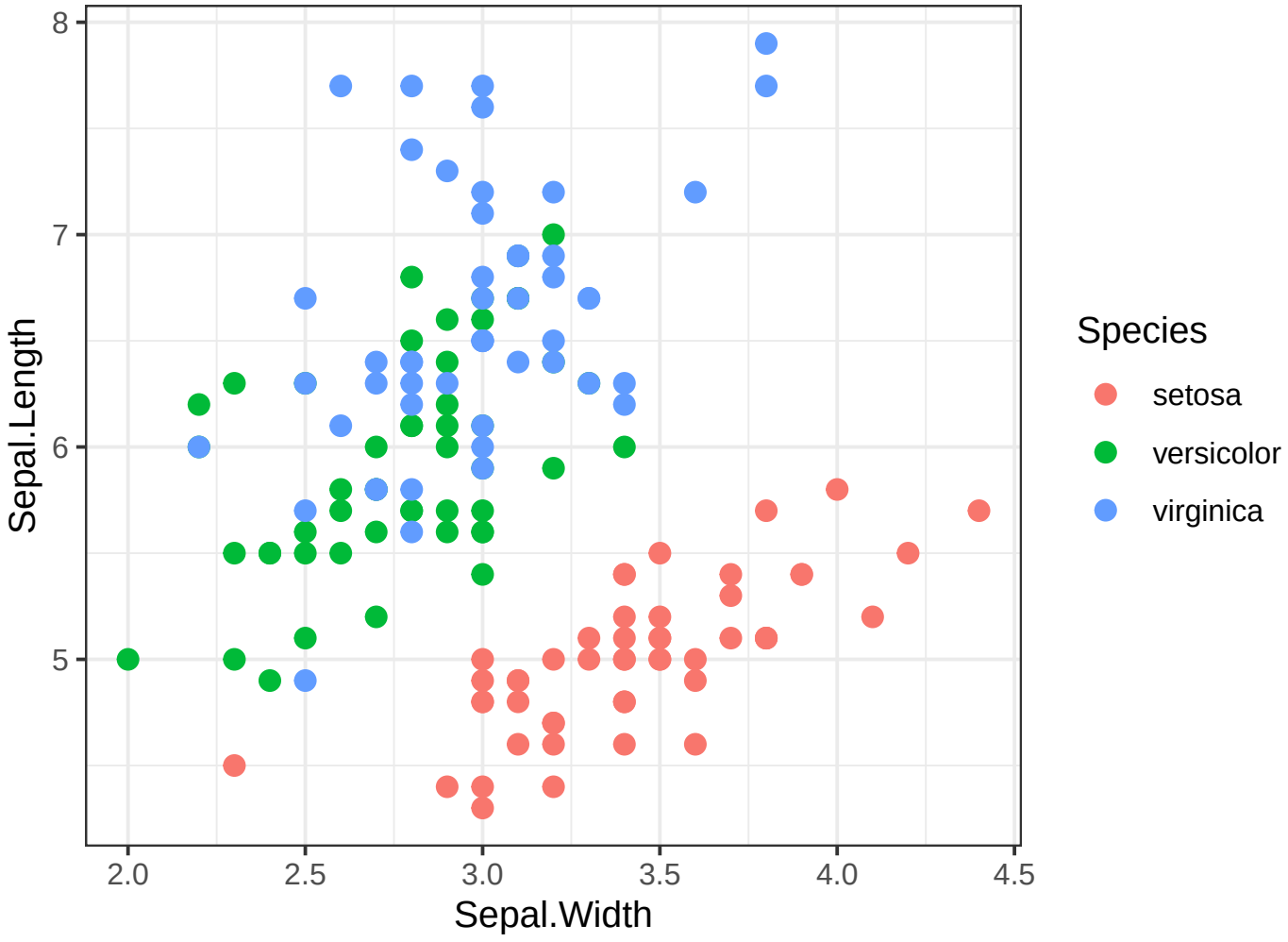
http://blog.kaggle.com/2015/04/22/scikit-learn-video-3-machine-learning-first-steps-with-the-iris-dataset/

Example: Classification of iris plants

We will use `sepal width` and `sepal length` to build a classificator. We have 50 observations from each class.

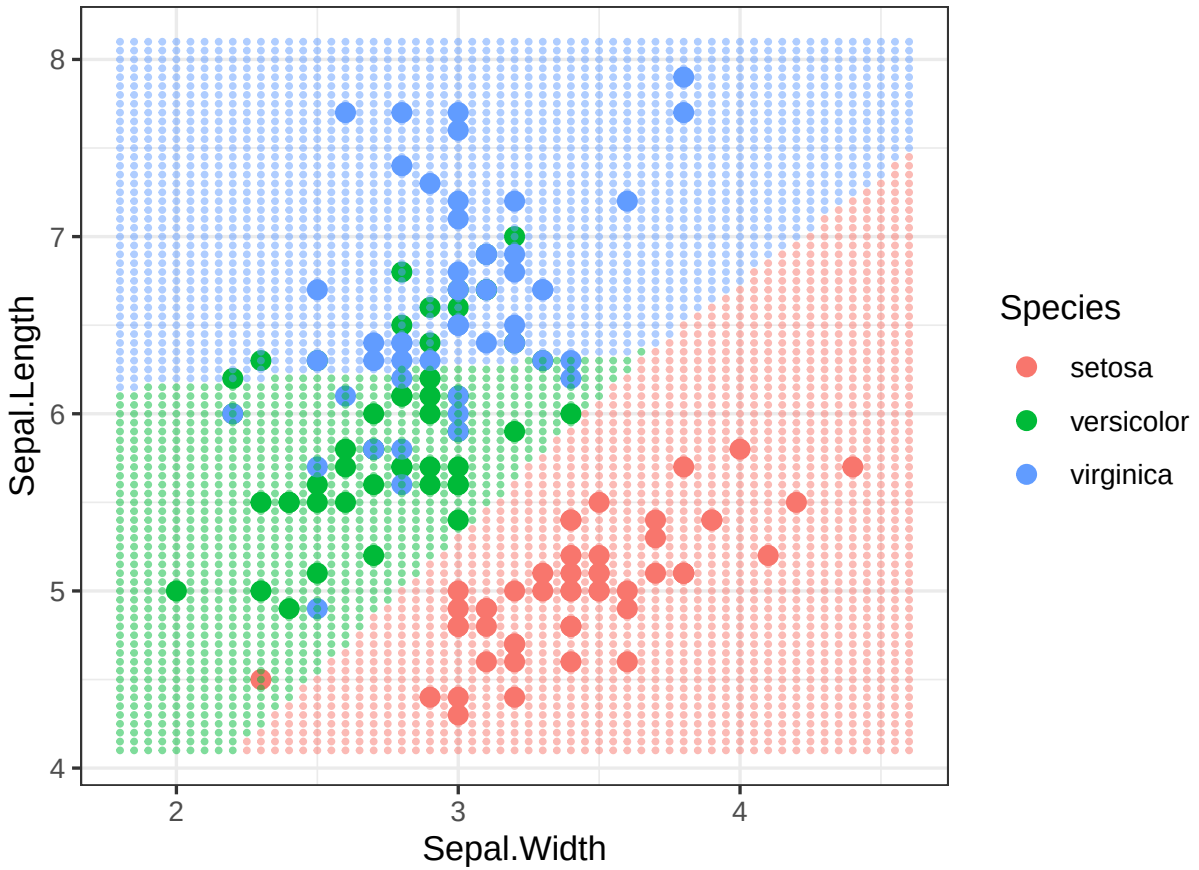
```
attach(iris)
head(iris)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```


Sepal.Length
8
7
6
5
2.0
2.5
3.0
3.5
4.0
4.5
Sepal.Width
Species
setosa
versicolor
virginica

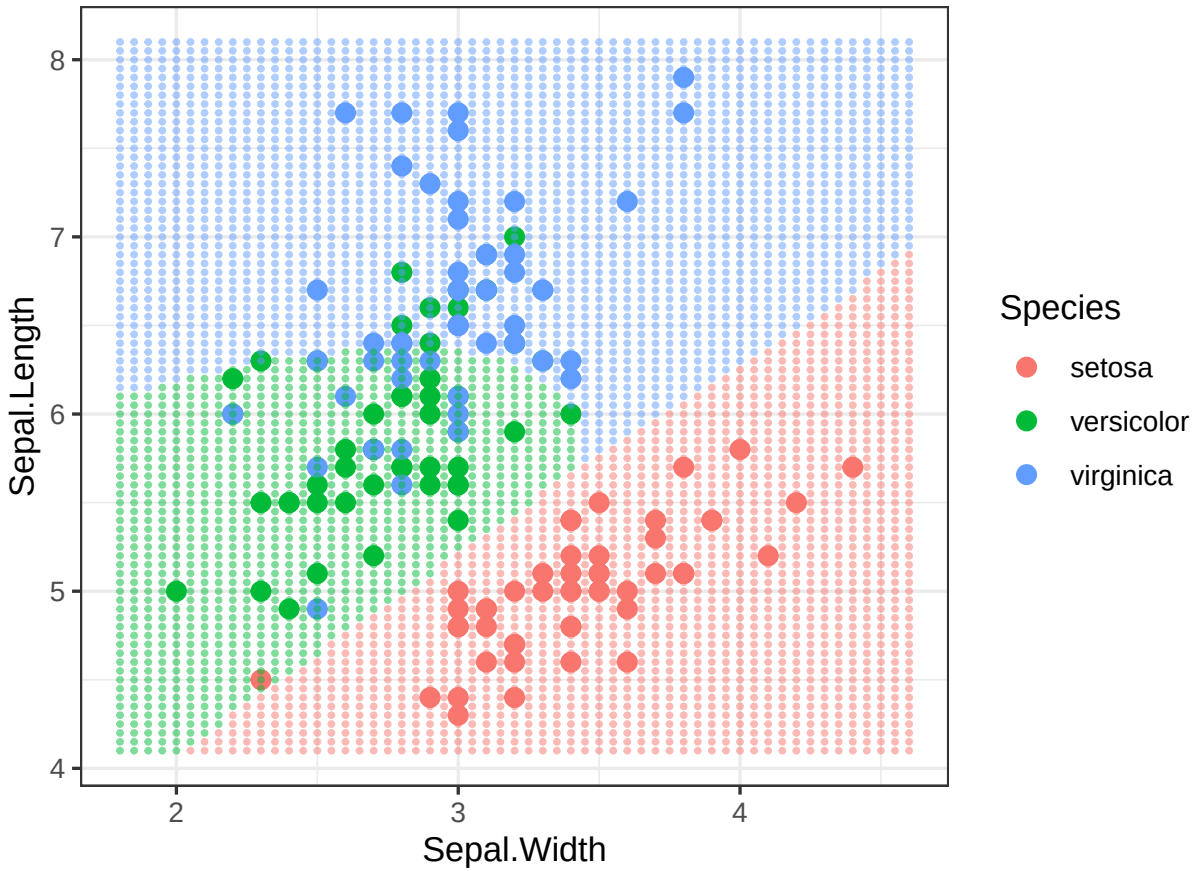
Iris: LDA

```
iris_lda = lda(Species ~ Sepal.Length + Sepal.Width, data = iris, prior = c(1,
    1, 1)/3)
```



Iris: QDA

```
iris_qda = qda(Species ~ Sepal.Length + Sepal.Width, data = iris, prior = c(1,
    1, 1)/3)
```



Iris: compare LDA and QDA

To compare the *predictive performance* of our two classifiers we divide the original `iris` data set randomly into train and test samples of equal size:

```
set.seed(1)
train = sample(1:150, 75)

iris_train = iris[train, ]
iris_test = iris[-train, ]
```

Run LDA and QDA *on the same training set*:

```
iris_lda2 = lda(Species ~ Sepal.Length + Sepal.Width, data = iris_train,
    prior = c(1, 1, 1)/3)

iris_qda2 = qda(Species ~ Sepal.Length + Sepal.Width, data = iris_train,
    prior = c(1, 1, 1)/3)
```

LDA training error: $\frac{14}{75} = 0.19$

```
table(predict(iris_lda2, newdata = iris_train)$class, iris_train$Species)
```

```
##
##              setosa versicolor virginica
##   setosa         27          0         0
##   versicolor      1         15         8
##   virginica       0          5        19
```

LDA test error: $\frac{19}{75} = 0.26$.

```
iris_lda2_predict = predict(iris_lda2, newdata = iris_test)
table(iris_lda2_predict$class, iris$Species[-train])
```

```
##
##              setosa versicolor virginica
##   setosa         22          0         0
##   versicolor      0         22        11
##   virginica       0          8        12
```

QDA training error: $\frac{13}{75} = 0.17$.

```
table(predict(iris_qda2, newdata = iris_train)$class, iris_train$Species)
```

```
##
##              setosa versicolor virginica
##   setosa         28          0         0
##   versicolor      0         16         9
##   virginica       0          4        18
```

QDA test error: $\frac{24}{75} = 0.32$.

```
iris_qda2_predict = predict(iris_qda2, newdata = iris_test)
table(iris_qda2_predict$class, iris$Species[-train])
```

```
##
##              setosa versicolor virginica
##   setosa         22          0         0
##   versicolor      0         18        12
##   virginica       0         12        11
```

Result: The LDA classifier has given the smallest test error[4] for classifying iris plants based on sepal width and sepal length for our test set and should be preferred in this case.

But:

1. Would another division of the data into training and test set give the same conclusion (that LDA is better than QDA for this data set)?
   A: Not necessarily, but probably.
   → We will look into such questions in Module 5 (cross-validation).

2. What about the other two covariates? Would adding them to the model (4 covariates) give a better classification rule?
   A: Probably. Try if you want.

[4] Note that the training error is of much less interest; it could be low due to *overfitting* only.

Different forms of discriminant analysis

- LDA
- QDA
- Naive Bayes ("Idiot's Bayes"): Assume that each class density is the product of marginal densities - i.e. inputs are conditionally independent in each class

$$f_k(x) = \prod_{j=1}^{p} f_{kj}(x_j) \ .$$

  This is generally not true, but it simplifies the estimation dramatically.

- Other forms by proposing specific density models for $f_k(x)$, including nonparametric approaches.

Naive Bayes

- Naive Bayes is method that is popular when p is large.
- The *original naive Bayes*: univariate normal marginal distributions. Consequently

$$\delta_k(x) \propto \log\left[\pi_k \prod_{j=1}^{p} f_{kj}(x_j)\right] = -\frac{1}{2}\sum_{j=1}^{p} \frac{(x_j - \mu_{kj})^2}{\sigma_{kj}^2} + \log(\pi_k) \ ,$$

  thus Σ_k is assumed diagonal, and only the diagonal elements are estimated.
- Arbitraty generalizations can be made. For example, mixed features (qualitative and quantitative predictors).
- This method often produces good results, even though the joint pdf is not the product of the marginal pdf. This might be because we are not focussing on estimation of class pdfs, but class boundaries.

Summary of Classification Methods

- Logistic regression
- KNN
- Linear discriminant analysis
- Quadratic discriminant analysis
- Naive Bayes

Remember:

- Logistic regression and KNN *directly estimate* $\Pr(Y = k \mid X = x)$ (diagnostic paradigm).
- LDA, QDA and naive Bayes *indirectly estimate* $\Pr(Y = k \mid X = x) \propto f_k(x) \cdot \pi_k$ (sampling paradigm).

Which classification method is the best?

Advantages of discriminant analysis

- Discriminant analysis is more stable than logistic regression when the classes are well-separated.
- Discriminant analysis is more stable than logistic regression if the number of observations n is small and the distribution of the predictors X is approximately (multivariate) normal.

Linearity

Assume a binary classification problem with one covariate.

- Recall that logistic regression can be written:

$$\log\left(\frac{p(x)}{1-p(x)}\right) = \beta_0 + \beta_1 x \ .$$

- For a two-class problem, one can show that for LDA

$$\log\left(\frac{p_1(x)}{1-p_1(x)}\right) = \log\left(\frac{p_1(x)}{p_2(x)}\right) = c_0 + c_1 x_1 \ ,$$

  thus the same linear form. The difference is in how the parameters are estimated.

LDA vs logistic regression

- In practice the results are often very similar[5], but
  - LDA is "more available" in the multi-class setting.
  - if the class conditional distributions are multivariate normal then LDA (or QDA) is preferred.
  - logistic regression makes no assumptions about the covariates and is therefore to be preferred in many practical applications.
  - in medicine for two-class problems logistic regression is often preferred (for interpretability) and (always) together with ROC and AUC (for model comparison).

and KNN?

- KNN is used when the class boundaries are non-linear.

[5] logistic regression can also fit quadratic boundaries like QDA, by explicitly including quadratic terms in the model.

So: Which classification method is the best?

The answer is: **it depends!**

- Logistic regression is very popular for classification, especially when $K = 2$.
- LDA is useful when n is small, or the classes are well separated, and Gaussian assumptions are reasonable. Also when $K > 2$.
- Naive Bayes is useful when p is very large.
- KNN is nonparametric, thus no assumptions about the decision boundary nor the distribution of the variables. Expected to work better than LDA and logistic regression when boundary is very non-linear. Caveats:
  - No interpretation of the effect of the covariates possbile.
  - Curse of dimensionality.

Please read Section 4.5 of our coursebook (James et al. 2013).

Two-class problems: sensitivity, specificity

- Problems with only two classes (binary classifiers) have a special status, e.g. in medicine or biology.
- Assume the classes (Y) are labelled "-" (non disease, or $Y = 0$) and "+" (disease, or $Y = 1$), and that a diagnostic test is used to predict Y given $X = x$.
- **Sensitivity** is the proportion of correctly classified positive observations:

$$\frac{\#\text{True Positive}}{\#\text{Condition Positive}} = \frac{\text{TP}}{\text{P}} \ .$$

- **Specificity** is the proportion of correctly classified negative observations:

$$\frac{\#\text{True Negative}}{\#\text{Condition Negative}} = \frac{\text{TN}}{\text{N}} \ .$$

- We would like that a classification rule (or a diagnostic test) have both a high sensitivity and a high specificity.

- However, in an imperfect test (i.e., an imperfect predictor) one usually comes at the cost of the other.

2×2 table shows data from a simple diagnostic study:

		Predicted		
		$\hat{Y} = 0$	$\hat{Y} = 1$	
True	$Y = 0$	True negative (TN)	False positive (FN)	N
	$Y = 1$	False negative (FP)	True positive (TP)	P
		$N^\star$	$P^\star$	Tot

Example Continued: South African heart disease

We evaluate our multiple logistic model for the **SAheart** data set. We divide the original data set randomly into a training and test dataset of equal size.

```
set.seed(20)
train_ID = sample(1:nrow(d.heart), nrow(d.heart)/2)
train_SA = d.heart[train_ID, ]
test_SA = d.heart[-train_ID, ]
```

Fit a logistic regression model, **using the training set only**:

```
glm_SA = glm(chd ~ ., data = train_SA, family = "binomial")
summary(glm_SA)$coef
```

```
##                     Estimate  Std. Error     z value    Pr(>|z|)
## (Intercept)    -3.7674748108 1.439119254 -2.61790314 0.008847191
## sbp             0.0045154473 0.008907046  0.50695226 0.612188315
## tobacco         0.1253783872 0.043828325  2.86067029 0.004227464
## ldl             0.0484641791 0.080453775  0.60238540 0.546917627
## famhistPresent  0.8186419234 0.338067955  2.42153068 0.015455296
## obesity        -0.0226110199 0.040483169 -0.55852890 0.576483276
## alcohol        -0.0007383597 0.008039590 -0.09184047 0.926824793
## age             0.0446073935 0.014781896  3.01770439 0.002546972
```

- The estimated probability of a `chd` event ($Y = 1$) is then given as

$$\hat{p}(\mathbf{X}) = \frac{e^{\eta}}{1 + e^{\eta}},$$

with $\eta = \beta_0 + \beta_1 \cdot x_{\text{sbp}} + \beta_2 \cdot x_{\text{tobacco}} + ... + \beta_7 \cdot x_{\text{age}}$.
Remember that $x_{famhist} \in \{0, 1\}$ is a binary covariate.

- We are interested in the **predictions for the test set**.

- The `predict` function does these calculations for us. When specifying `type="response"` the function returns the probabilities for $Y = 1$.

```
probs_SA = predict(glm_SA, newdata = test_SA, type = "response")
```

- Here we have chosen a threshold value of 0.5. By using the `ifelse` function we specify that all probabilities larger than 0.5 are to be classified as 1, while the remaining probabilities are to be classified as 0.

```r
pred_SA = ifelse(probs_SA > 0.5, 1, 0)

predictions_SA = data.frame(probs_SA, pred_SA, test_SA[, "chd"])
colnames(predictions_SA) = c("Estim. prob. of Y=1", "Predicted class",
    "True class")
head(predictions_SA)
```

```
##    Estim. prob. of Y=1 Predicted class True class
## 1            0.7741048               1          1
## 4            0.7141807               1          1
## 7            0.2258178               0          0
## 11           0.5216856               1          1
## 12           0.7112496               1          1
## 15           0.6702761               1          0
```

The confusion matrix is used to count the number of misclassifications in the test set:

```
table(predicted = pred_SA, true = d.heart[-train_ID, "chd"])
```

```
##           true
## predicted   0   1
##         0 118  45
##         1  27  41
```

- The logistic model has correctly classified 118+41 times, and misclassified 27+45 times, thus

$$\text{Test error rate} = \frac{27+45}{27+45+118+41} \approx 0.31 \ .$$

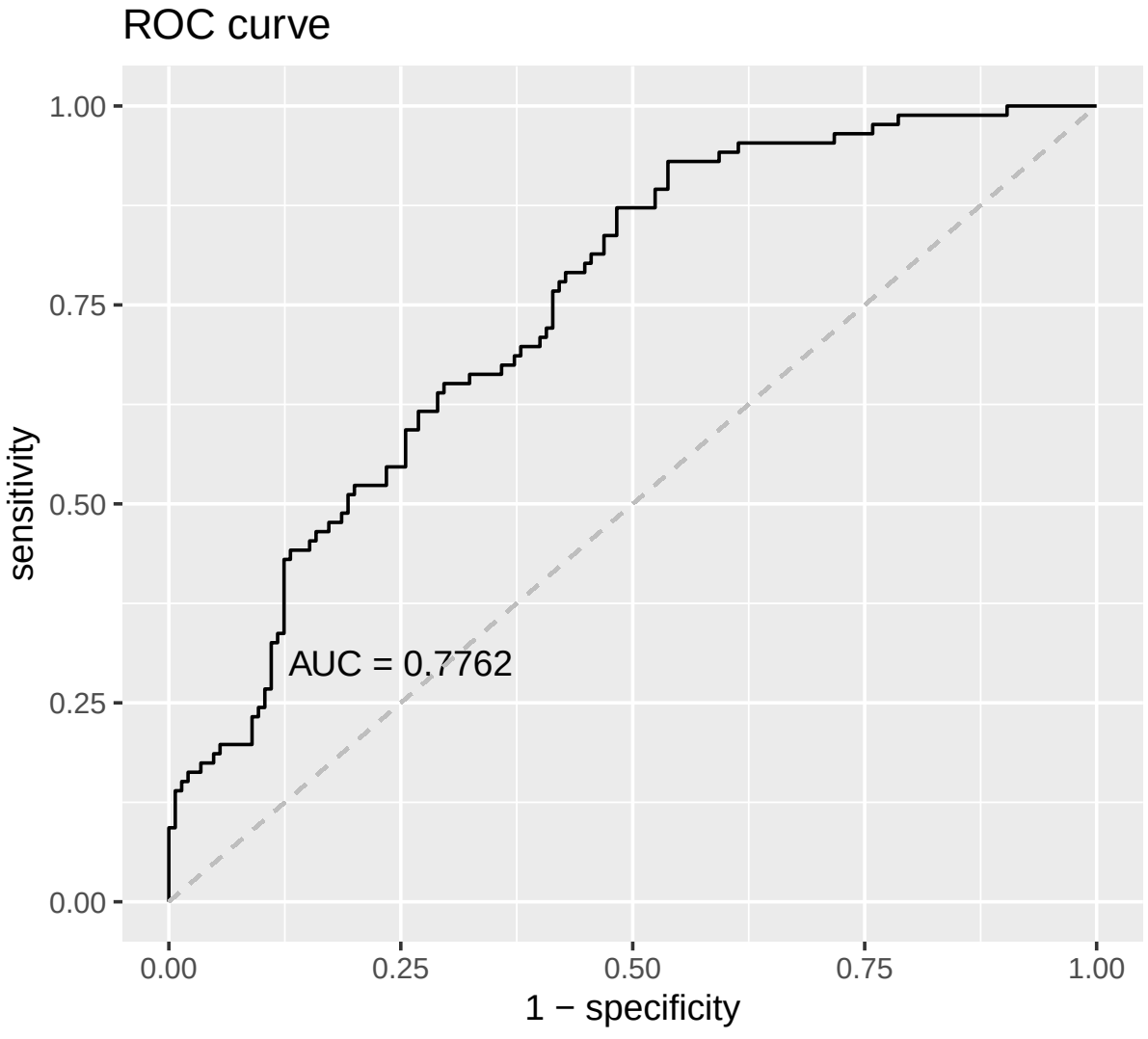
- Sensitivity: $\frac{41}{41+45} = 0.477$, specificity $= \frac{118}{118+27} = 0.814$.

ROC curves and AUC

- The receiver operating characteristics (ROC) curve gives a graphical display of the sensitivity against (1-specificity), as the threshold value (cut-off on probability) is moved from 0 to 1.
- An ideal classifier will give a ROC curve which hugs the top left corner (sensitivity = specificity = 1), while a straight line represents a classifier with a random guess of the outcome.
- The **AUC** score is the area under the AUC curve. It ranges between the values 0 and 1, where a higher value indicates a better classifier.
- The AUC score is useful for comparing the performance of different classifiers, as all possible threshold values are taken into account.

Example Continued: South African heart disease

In order to see how our model performs for different threshold values, we can plot a ROC curve:



To check where in the plot we find the default cut-off on 0.5, we need to calculate sensitivity and specificity for this cut-off:

```
res = table(pred_SA, d.heart[-train_ID, "chd"])
sens = res[2, 2]/sum(res[, 2])
spec = res[1, 1]/sum(res[, 1])
sens
```

```
## [1] 0.4767442
```

```
spec
```

```
## [1] 0.8137931
```

Observe that the value 0.477 (on y-axis) and 0.186 (1-specificity on x-axis) is on our ROC curve.

The ROC-curve is made up of all possible cut-offs and their associated sensitivity and specificity.

Further reading

- Videos on YouTube by the authors of ISL, Chapter 4,

References

James, G., D. Witten, T. Hastie, and R. Tibshirani. 2013. *An Introduction to Statistical Learning with Applications in R*. New York: Springer.