ETC3250/5250 Introduction to Machine Learning

Week 8: Support vector machines, nearest neighbours and regularisation

Professor Di Cook

etc3250.clayton-x@monash.edu

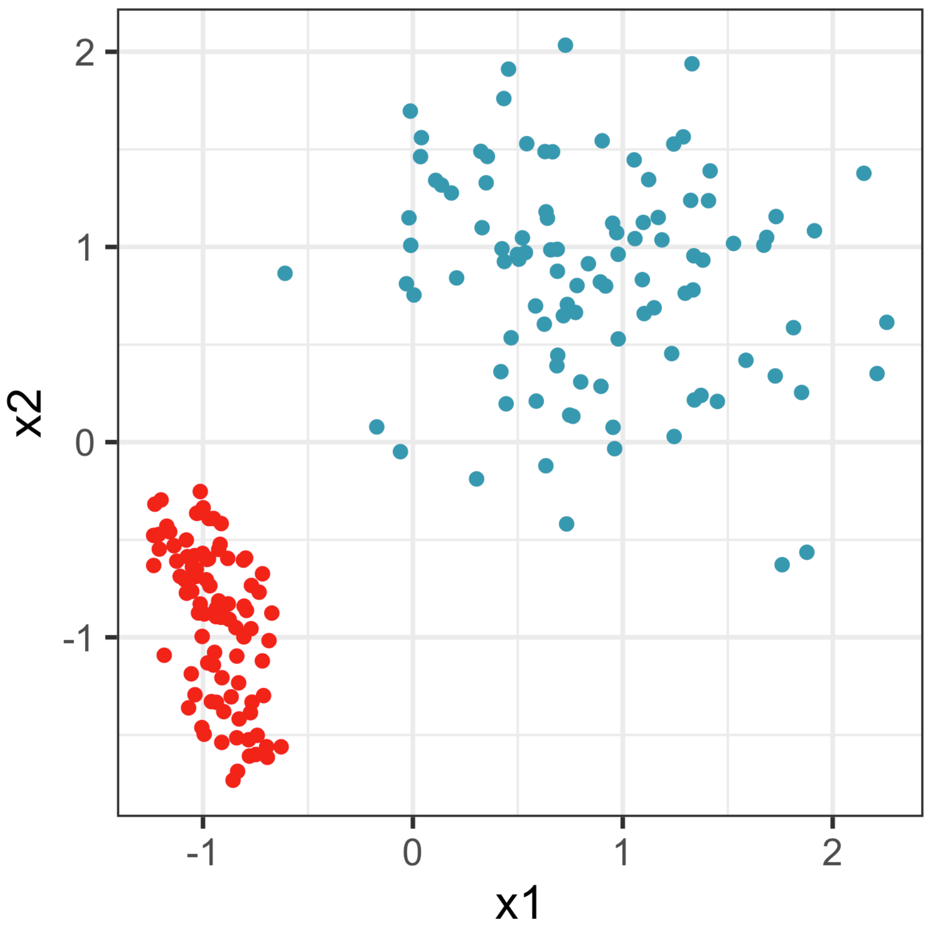
Department of Econometrics and Business Statistics

Overview

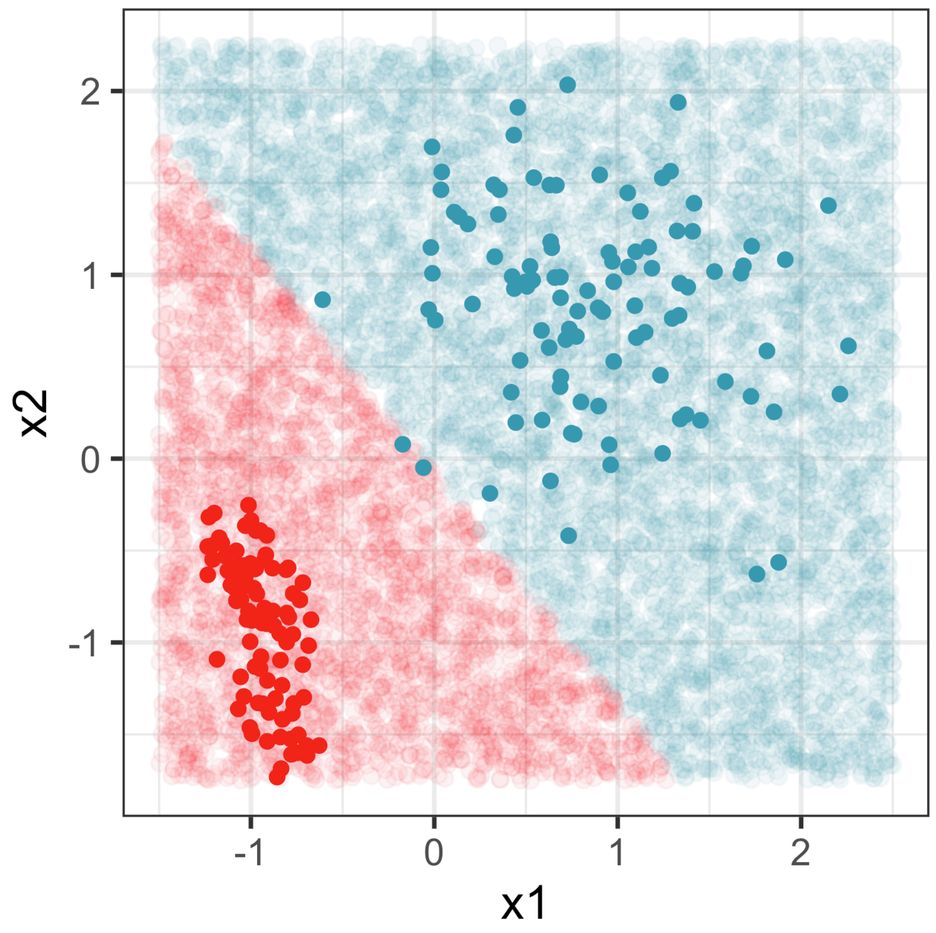
We will cover:

- Separating hyperplanes
- Non-linear kernels
- Really simple models using nearest neighbours
- Regularisation methods

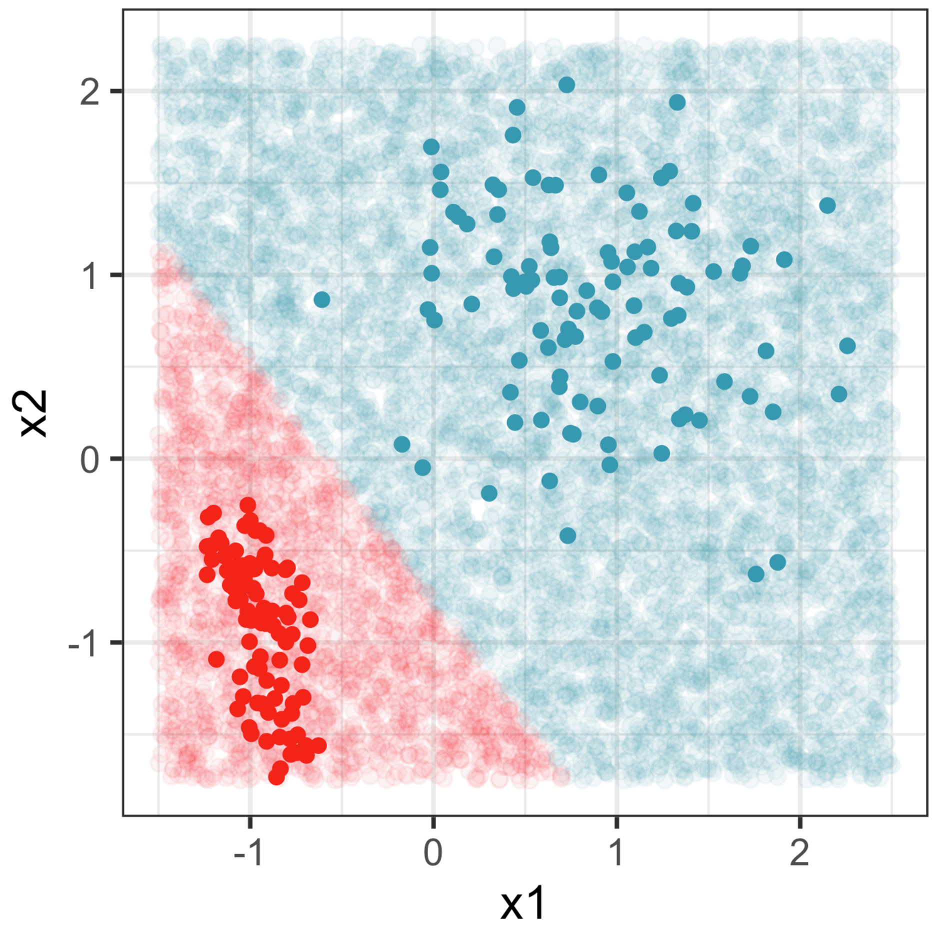
Why use support vector machines?



Where would you put the boundary to classify these two groups?



Here's where LDA puts the boundary. What's wrong with it?



Why is this the better fit?

Separating hyperplanes (1/3)

LDA is an example of a classifier that generates a separating hyperplane

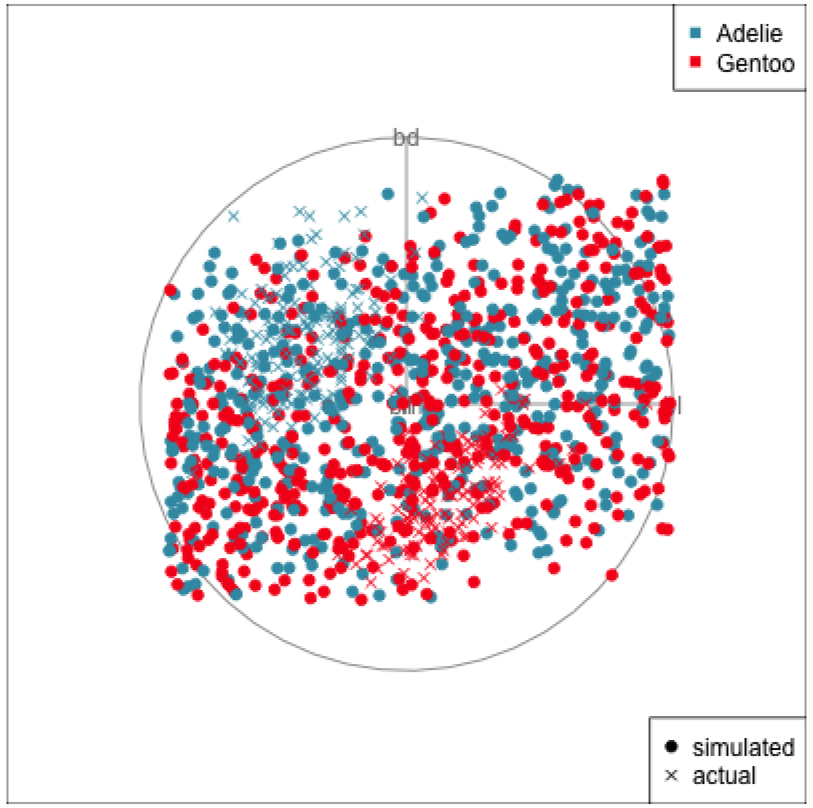
```
lda_fit$fit$scaling
```

```
    LD1
x1 -1.9
x2 -1.6
```

is defines a line orthogonal to the 1D separating hyperplane, with slope 0.81.

Equation for the separating hyperplane is $x_2 = mx_1 + b$, where $m =$ -1.24, and b can be solved by substituting in the point $((\bar{x}_{A1} + \bar{x}_{B1})/2, (\bar{x}_{A2} + \bar{x}_{B2})/2)$. (Separating hyperplane has to pass through the average of the two means, if prior probabilities of each class are equal.)

Separating hyperplane produced by LDA on 4D penguins data, for Gentoo vs Adelie. (It is 3D.)



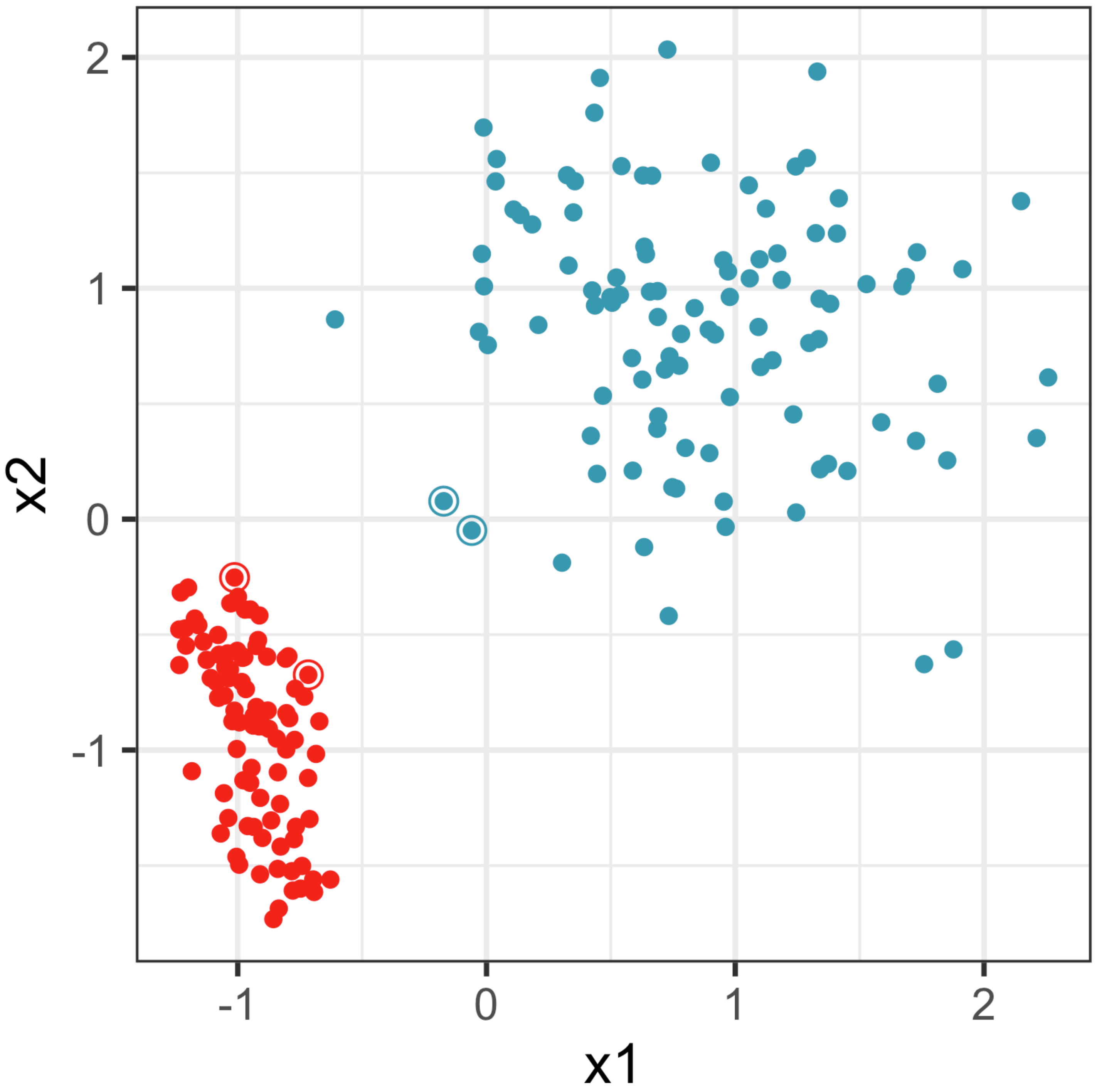
Separating hyperplanes (2/3)

The equation of p-dimensional hyperplane is given by

$$\beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p = 0$$

LDA estimates β_j based on the sample statistics, means for each class and pooled covariance.

SVM estimates β_j based on support vectors (∘, ∘), observations on the border between the two groups. Thus, the boundary will divide in the gap.



Separating hyperplanes (3/3)

LDA

$$x\, S^{-1}(\bar{x}_A - \bar{x}_B) - \frac{\bar{x}_A + \bar{x}_B}{2} S^{-1}(\bar{x}_A - \bar{x}_B) = 0$$

resulting in:

$$\hat{\beta}_0 = -\frac{\bar{x}_A + \bar{x}_B}{2} S^{-1}(\bar{x}_A - \bar{x}_B)$$

$$\begin{pmatrix} \hat{\beta}_1 \\ \hat{\beta}_2 \\ \vdots \\ \hat{\beta}_p \end{pmatrix} = S^{-1}(\bar{x}_A - \bar{x}_B)$$

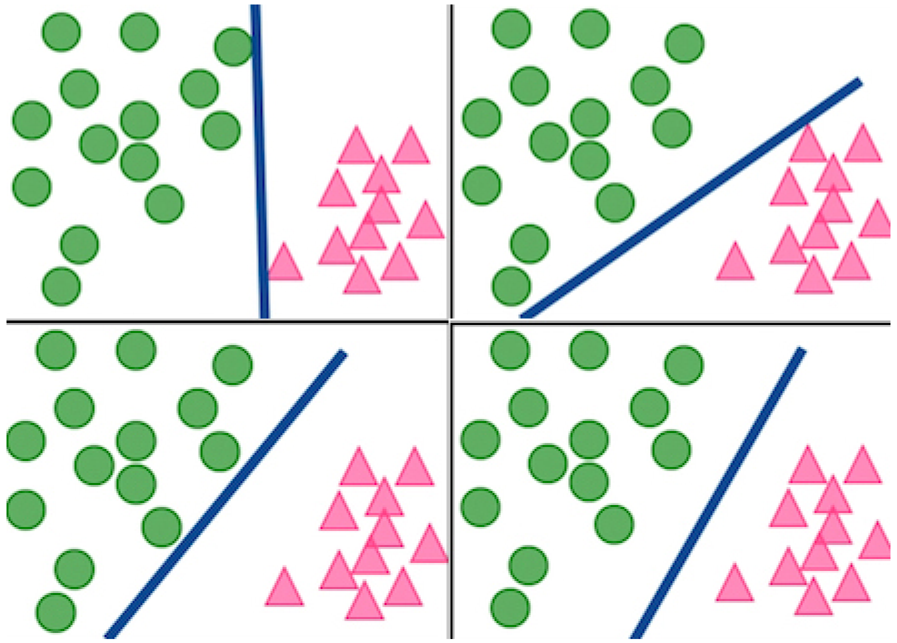
SVM

Set $y_A = 1, y_B = -1$, and x_j scaled to $[0, 1]$, $s =$ number of support vectors.

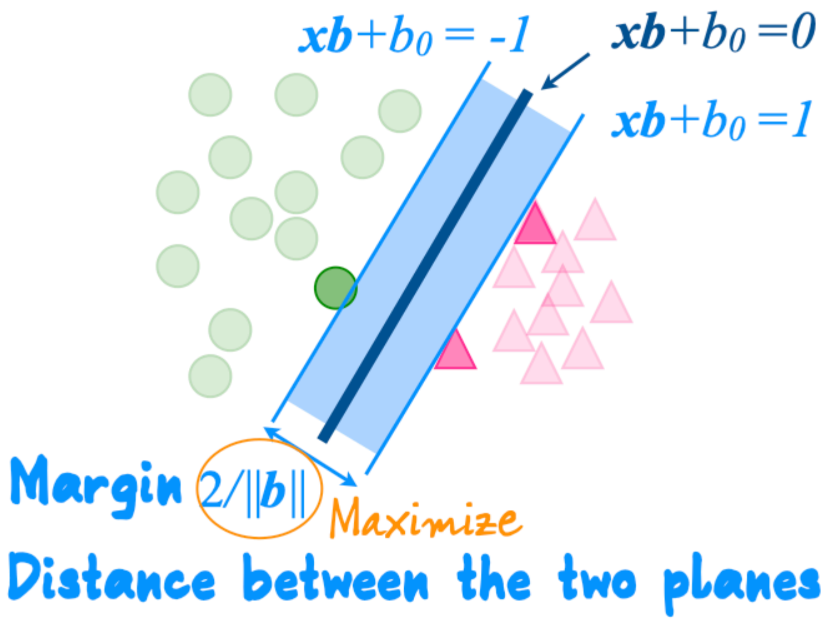
$$\begin{pmatrix} \hat{\beta}_1 \\ \hat{\beta}_2 \\ \vdots \\ \hat{\beta}_p \end{pmatrix} = \sum_{k=1}^{s} (\alpha_k y_k) x_{kj}$$

Linear support vector machine classifier (1/2)

- Many possible separating hyperplanes, which is best?



- Computationally hard. Need to find the observations which when used to define the hyperplane maximise the margin.



Minimise wrt $\beta_j, j = 0, \ldots, p$

$$\frac{1}{2}\sqrt{\sum_{i=1}^{p} \beta_j^2}$$

subject to $y_i(\sum_{j=1}^{p} x_{ij}\beta_j + \beta_0) \geq 1.$

Linear support vector machine classifier (2/2)

Classify the test observation x based on the sign of

$$s(x) = \beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p$$

- If $s(x_0) > 0$, class 1, and if $s(x_0) < 0$, class -1, i.e. $h(x_0) = \text{sign}(s(x_0))$.
- $s(x_0)$ far from zero $\rightarrow x_0$ lies far from the hyperplane + **more confident** about our classification

Note: The margin (M) is set to be equal to 1 here, but could be anything depending on scaling.

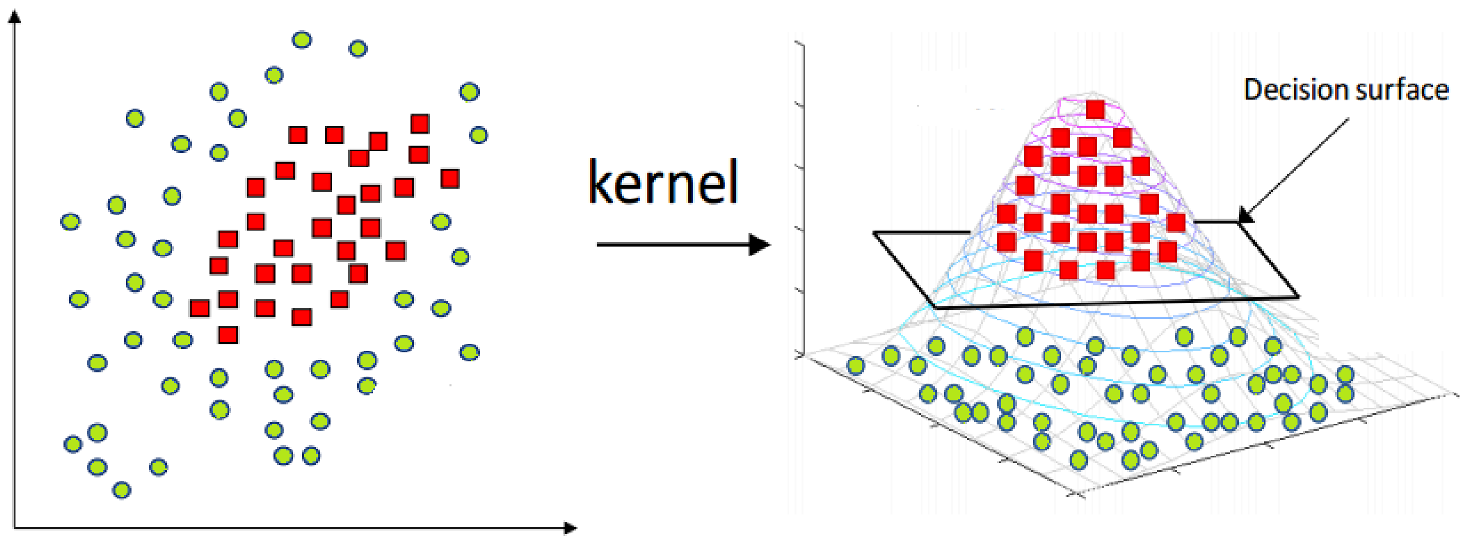
Using kernels for non-linear classification

Note: Linear SVM is

$$f(x) = \beta_0 + \sum_{i \in \mathcal{S}} \alpha_i \langle x, x_i \rangle.$$

where the inner product is defined as

$$\begin{aligned}\langle x_1, x_2 \rangle &= x_{11}x_{21} + x_{12}x_{22} + \cdots + x_{1p}x_{2p} \\ &= \sum_{j=1}^{p} x_{1j}x_{2j}\end{aligned}$$



Source: Grace Zhang @zxr.nju

A kernel function is an inner product of vectors mapped to a (higher dimensional) feature space.

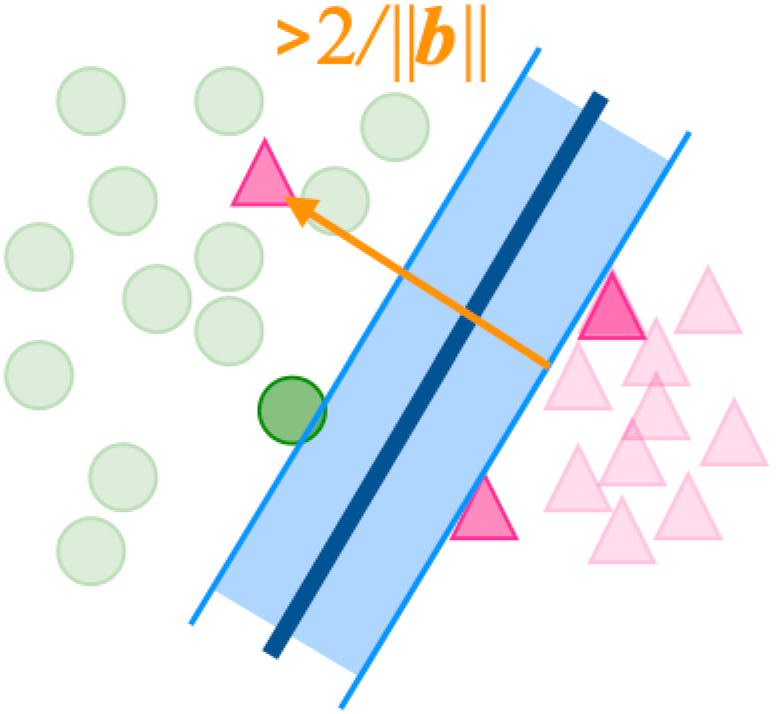
$$\mathcal{K}(x_1, x_2) = \langle \psi(x_1), \psi(x_2) \rangle$$

We can generalise SVM to a non-linear classifier by replacing the inner product with the kernel function as follows:

$$f(x) = \beta_0 + \sum_{i \in \mathcal{S}} \alpha_i \mathcal{K}(x, x_i).$$

Common kernels: polynomial, radial

Soft threshold, when no separation



Distance observation i is on wrong side of boundary is $\xi_i/||b||$.

Minimise wrt $\beta_j, j = 0, \dots, p$

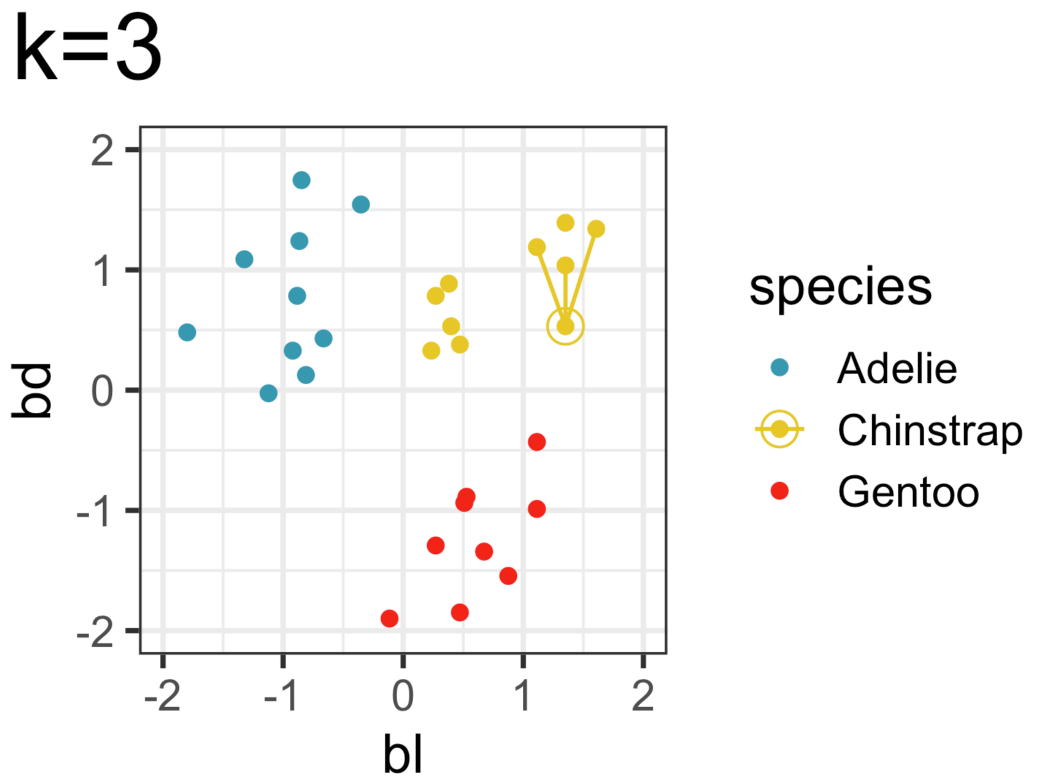
$$\frac{1}{2}\sqrt{\sum_{i=1}^{p}\beta_j^2} + C\sum_{i=1}^{s^*}\xi_i$$

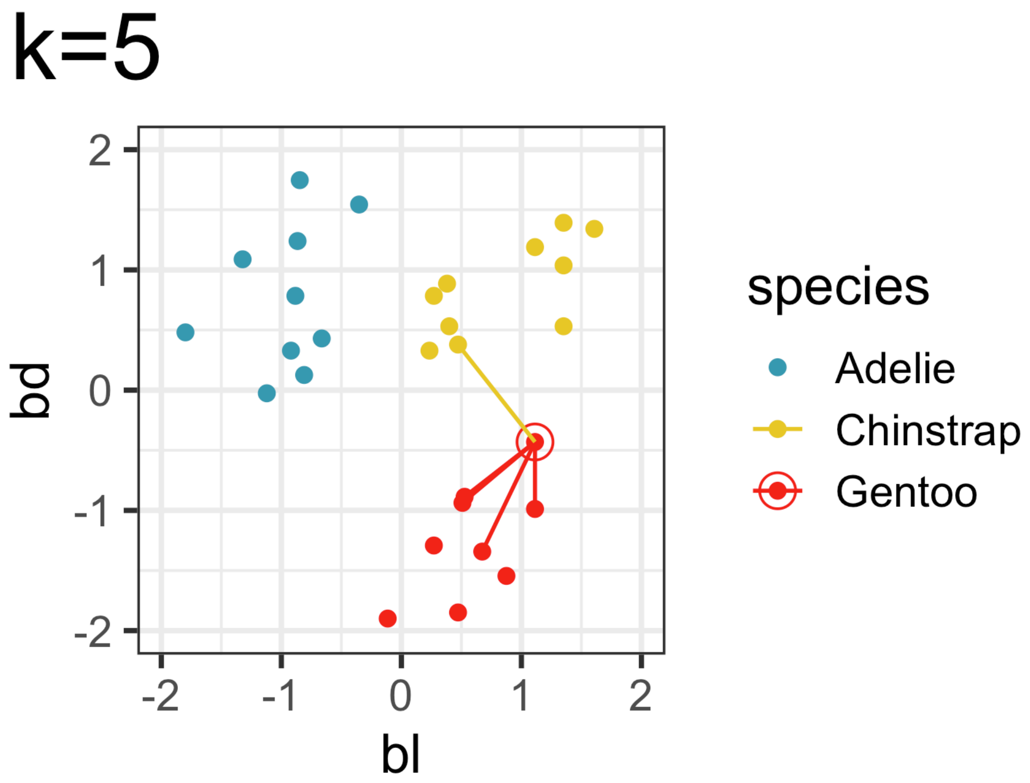
- subject to $y_i(\sum_{j=1}^{p} x_{ij}\beta_j + \beta_0) \geq 1,$
- where C is a regularisation parameter that controls the trade-off between maximizing the margin and minimizing the misclassifications $\sum_{i=1}^{s^*}\xi_i$, for s^* misclassified observations.

Really simple models

k-nearest neighbours

Predict y using the $k-$ nearest neighbours from observation of interest.



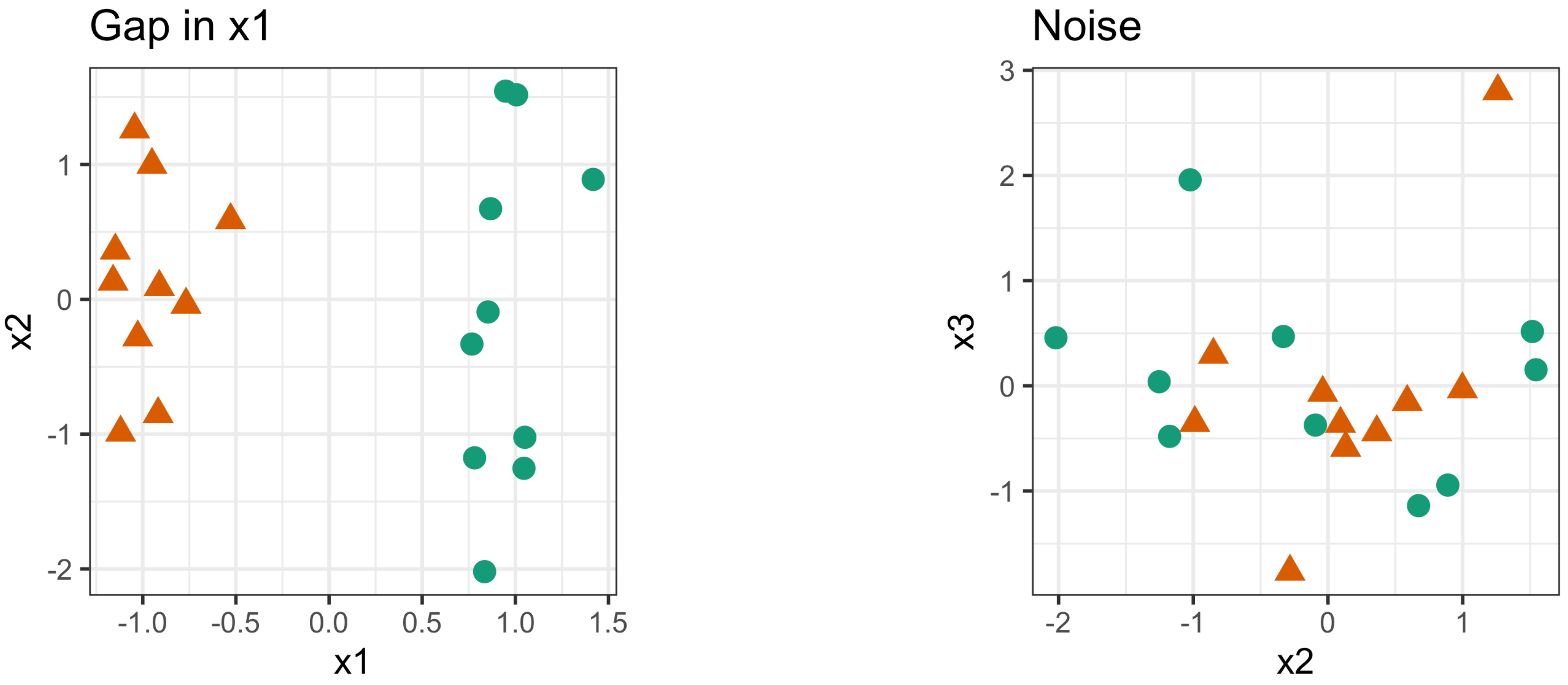


- standardise your data, to compute distances between points accordingly
- fails in high dimensions because data is too sparse

Regularisation

What is the problem in high-high-D? (1/3)

20 observations and 2 classes: A, B. One variable with separation, 99 noise variables



What will be the optimal LDA coefficients?

What is the problem in high-high-D? (2/3)

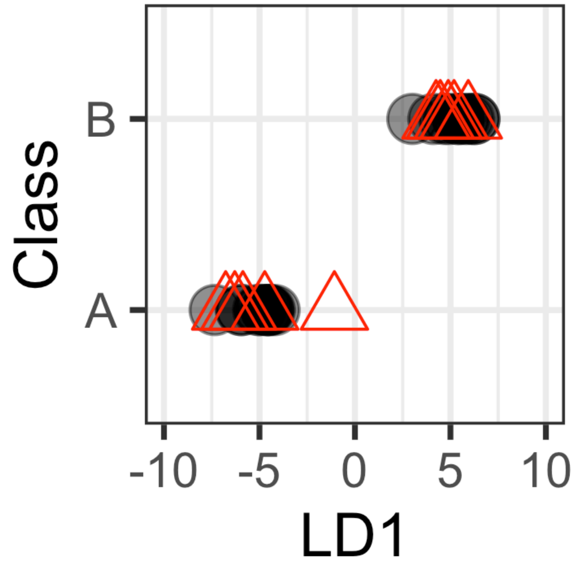
Fit linear discriminant analysis on first two variables.

```
Call:
lda(cl ~ ., data = tr[, c(1:2, 101)], prior = c(0.5, 0.5))

Prior probabilities of groups:
  A   B 
0.5 0.5 

Group means:
     x1    x2
A  0.96 -0.13
B -0.96  0.13

Coefficients of linear discriminants:
     LD1
x1 -5.36
```

Coefficient for x1 MUCH higher than x2. As expected!

Predict the training and test sets

```
   A  B
A 10  0
B  0 10

  A B
A 5 0
B 0 5
```

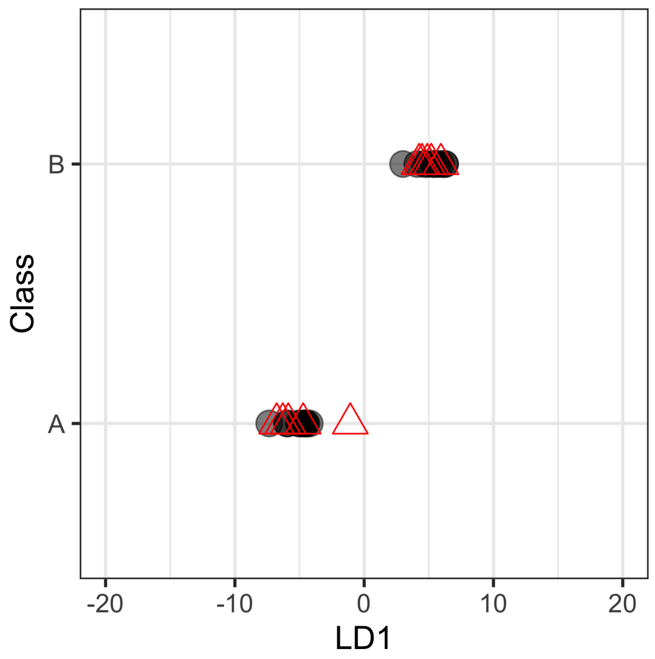


Perfect!

What is the problem in high-high-D? (3/3)

What happens to test set (and predicted training values) as number of noise variables increases?

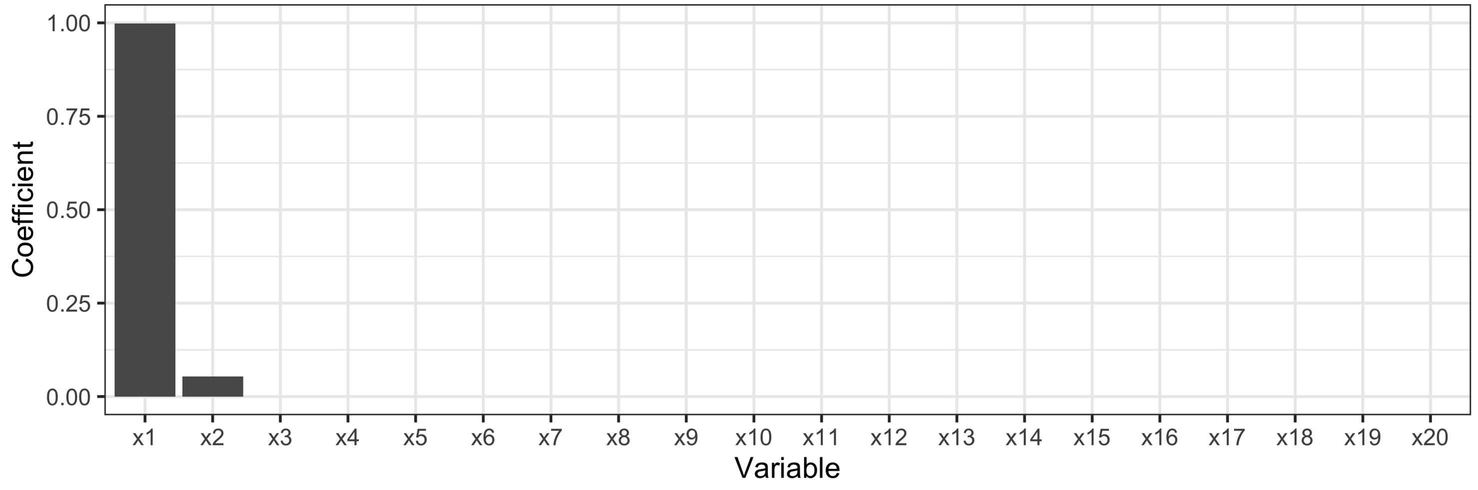
p = 2 train = 0 test = 0



What happens to the estimated coefficients as dimensions of noise increase?

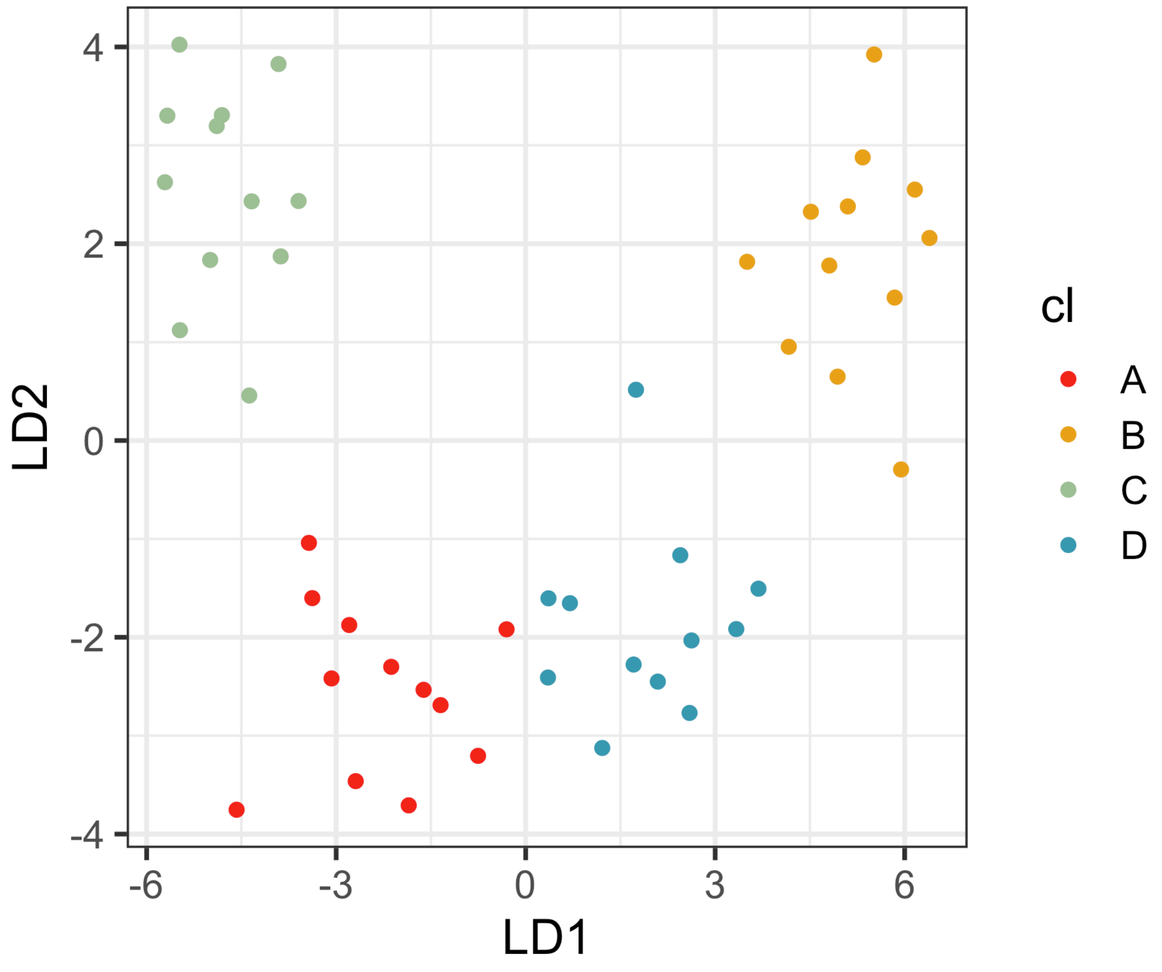
Remember, the noise variables should have coefficient = ZERO.

p = 2



How do you check? (1/2)

```r
w <- matrix(runif(48*40), ncol=40) |>
  as.data.frame() |>
  mutate(cl = factor(rep(c("A", "B", "C", "D"), rep(12, 4))))
w_lda <- lda(cl~., data=w)
w_pred <- predict(w_lda, w, dimen=2)$x
w_p <- w |>
  mutate(LD1 = w_pred[,1],
         LD2 = w_pred[,2])
```

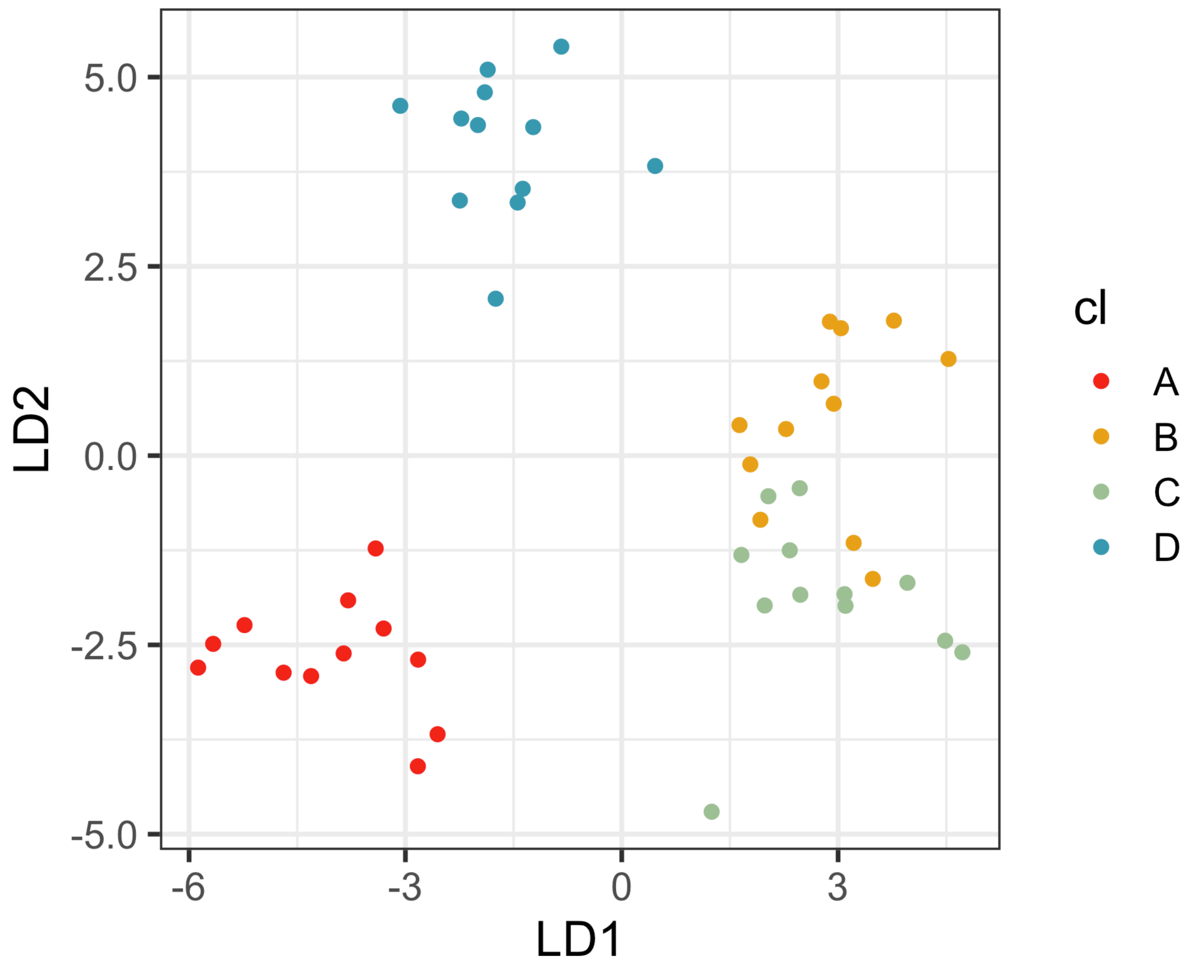


$n = 48, p = 40$ Class labels are randomly generated

Permutation is your friend, for high-dimensional data analysis.

Permute the class labels.

```r
set.seed(951)
ws <- w |>
  mutate(cl = sample(cl))
```



How do you check? (2/2)

- Permuting response, repeating the analysis, then make model summaries and diagnostic plots
- Comparing with test set is critical.
- If results (error/accuracy, low-d visual summary) on test set are very different than training, it could be due to high-dimensionality.

How can you correct?

- Subset selection: reduce the number of variables before attempting to model
- Penalisation: change the optimisation criteria to include another term which makes it worse when there are more coefficients

Penalised LDA

Recall: LDA involves the eigen-decomposition of $\Sigma^{-1}\Sigma_B$. (Inverting Σ is a problem with too many variables.)

The eigen-decomposition is an analytical solution to an optimisation:

$$\underset{\beta_k}{\text{maximize}} \;\; \beta_k^T \hat{\Sigma}_B \beta_k$$

$$\text{subject to } \beta_k^T \hat{\Sigma} \beta_k \leq 1, \;\; \beta_k^T \hat{\Sigma} \beta_j = 0 \; \forall i < k$$

Fix this by:

$$\underset{\beta_k}{\text{maximize}} \left(\beta_k^T \hat{\Sigma}_B \beta_k + \lambda_k \sum_{j=1}^{p} |\hat{\sigma}_j \beta_{kj}| \right)$$

$$\text{subject to } \beta_k^T \tilde{\Sigma} \beta_k \leq 1$$

Next: K-nearest neighbours and hierarchical clustering