Statistical Learning for Data Science: Basic techniques using R

Stefan Zohren

15 February 2016

Content of this lecture

- Model selection
  - Best subset selection
  - Forward and backward selection
- Regularisation
  - Rigid regression
  - Lasso
- High-dimensional data
  - Case study: The Arcene data set
- Summary

Regression and classification revision

Let us recall the multivariate linear regression model with p predictors $\mathbf{x}_1, ..., \mathbf{x}_p$.

$$y_i = f(x_i) + \epsilon_i = a_0 + a_1 x_{i1} + ... + a_p x_{ip} + \epsilon_i$$

We are focusing our discussion here on linear regression (lecture 1) but it can be straightforwardly applied to classification using logistic regression as well (lecture 2).

Regression example: Housing data

Recall how we implemented linear regression on the training data set for the Housing data. First splitting the data

```
n <- nrow(HousingData)
trainIndices <- sample(1:n, n*2/3)
HousingData.train <- HousingData[trainIndices,]
HousingData.test <- HousingData[-trainIndices,]
```

and then fitting the linear regression:

```
fit.lm <- lm(MedianHouseValue ~ . ,
                          data = HousingData.train)
```

Subset selection

- One question we faced was which predictors to include in the model
- In a previous exercise you investigated several choices manually
- Subset selection is an automated way to do this
- We will now discuss best subset selection, as well as forward and backward selection

Best subset selection

Best subset selection takes all subsets of predictors of a given size (running from 1 to a maximum number) and determines which subset gives the best model, e.g. by evaluation adjusted R^2 or other criteria. It is implemented in R using the function `regsubsets` in the `leaps` package:

```
library(leaps)
fit.subset <- regsubsets(MedianHouseValue ~ . ,
                         data = HousingData.train)
```

Best subset selection

For example, for 1 predictor it found:

```
coef(fit.subset,id=1)
```

```
##   (Intercept) MedianIncome
##     0.4581578    0.3985915
```

For 2 predictors:

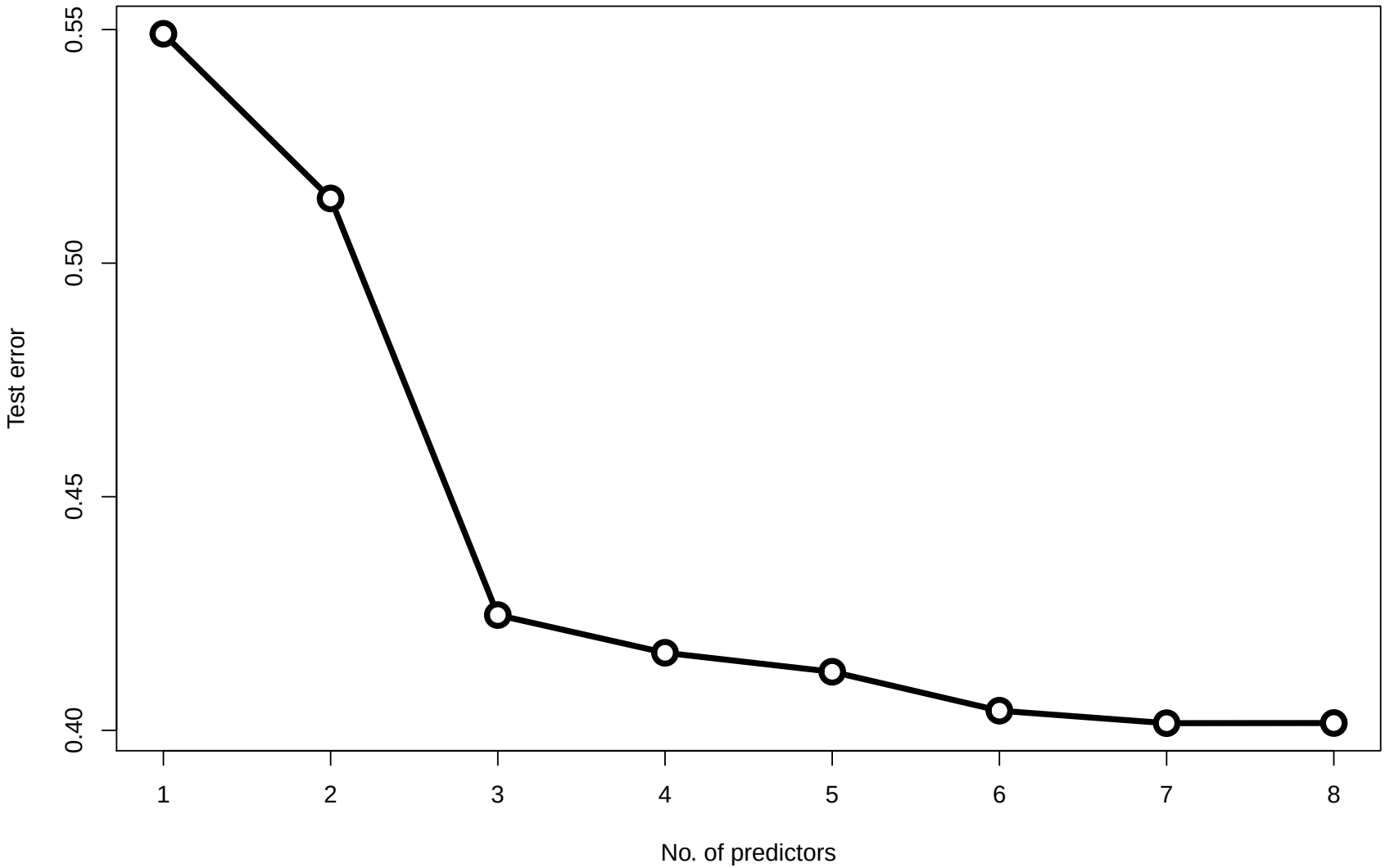
```
coef(fit.subset,id=2)
```

```
##       (Intercept)      MedianIncome HousingMedianAge
##       -0.07329656        0.42289600        0.01555377
```

etc.

Best subset selection: Test error

For all of these models we evaluate the test error on the test data set and obtain:



Best subset selection

- We would probably choose the model with 4 predictors, since for more predictors the error does not improve much.

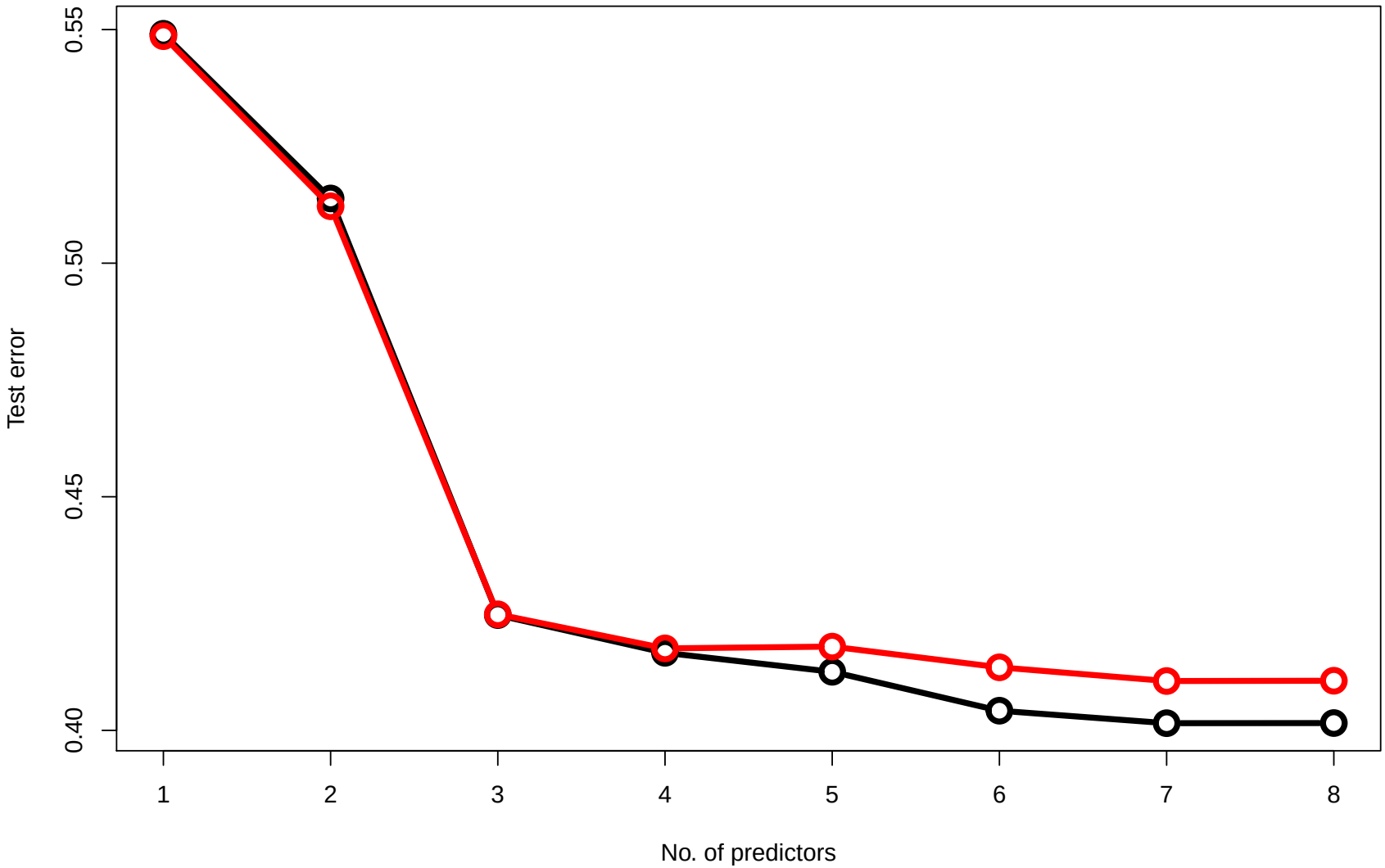
```
names(coef(fit.subset,id=4))
```

```
## [1] "(Intercept)"      "MedianIncome"     "HousingMedian
## [4] "Latitude"         "Longitude"
```

- In general we chose the model which is the least flexible model within one standard deviation from the model with minimal test error, the so-called **one standard deviation rule**.
- Therefore we would have to repeat the procedure with different choices of test sets to determine the variance of the test error under those choices

Best subset selection

As an example we redo the analysis with a different random seed and add the new points to the figure (in red).



Forward and backward selection

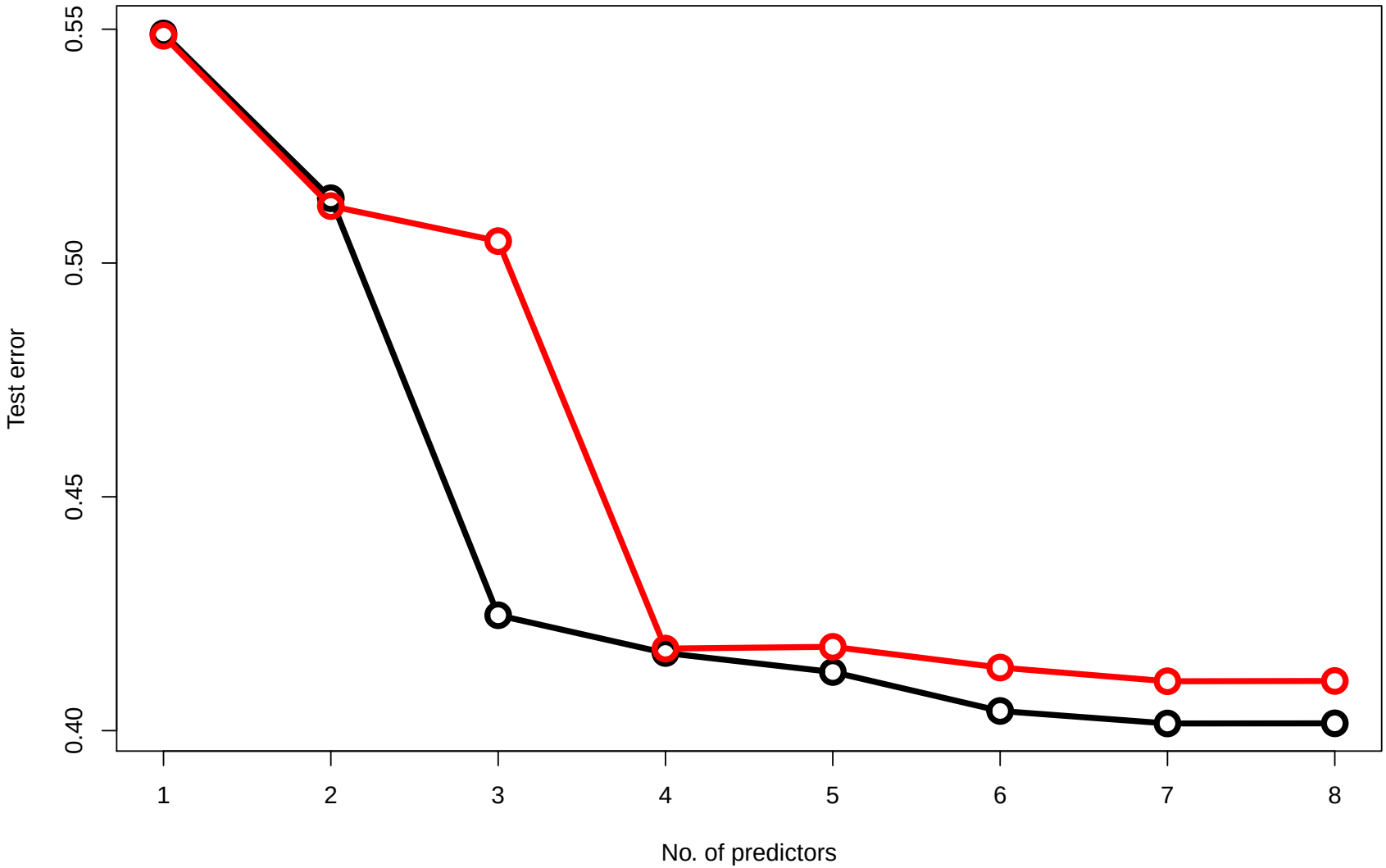
- In **best subset selection** we search through all subsets of a given size q to find the best selection of predictors.
- Instead, in a greedy fashion, we could look for the best predictor in the case q=1 and then for q=2 look which would be the best predictor to add to this one and so on. This is called **forward selection**.
- Alternatively, we could also start off with all predictors and then remove them one by one, always checking which is the best one to remove. This in turn is called **backward selection**.

Forward and backward selection

In R all those methods can be implemented by choosing different attributes for `method` in the `regsubsets` function. In particular,

- `method = "exhaustive"` is the standard option and refers to full subset selection using exhaustive search.
- `method = "forward"` refers to forward selection
- `method = "backward"` refers to backward selection.

Example: best subset selection vs forward selection



Regularisation: Rigid regression

In subset selection, all we did was basically solving the usual least-squares optimisation problem:

$$\min_{\mathbf{a}} \sum_{i=1}^{n} (y_i - f(x_i))^2.$$

subject to the constraint that at a given number of the a_i is set to zero.

Rigid regression is simply another ways to put a constraint on the coefficients a_i to force them to be closer to zero:

$$\min_{\mathbf{a}} \sum_{i=1}^{n} \left[y_i - (a_0 + a_1 x_{i1} + \ldots + a_p x_{ip}) \right]^2, \quad \text{subject to} \quad \sum_{i=1}^{p} a_i^2 \leq C$$

Here C is called the **cost**.

Regularisation: Rigid regression

Mathematically speaking we can rewrite the constraint optimisation problem above in its conjugate form:

$$\min_{\mathbf{a}} \left(\sum_{i=1}^{n} [y_i - (a_0 + a_1 x_{i1} + ... + a_p x_{ip})]^2 \quad + \quad \lambda \sum_{i=1}^{p} a_i^2 \right)$$

Here λ is conjugate to C, i.e. small λ corresponds to large C and vice versa.

Regularisation: Rigid regression in R

In R rigid regression is implemented in the package glmnet.

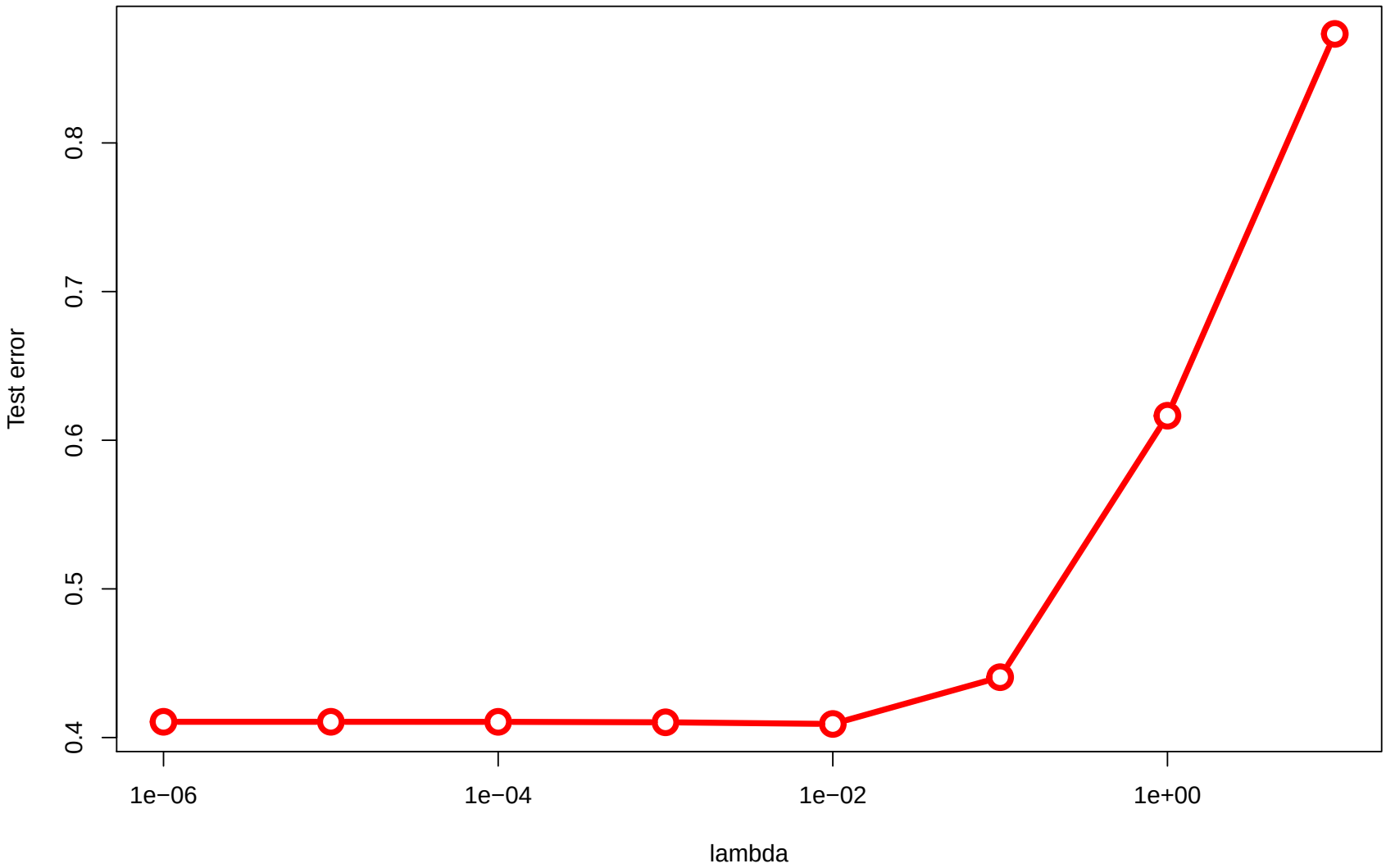
```
library(glmnet)
```

The syntax is slightly different, by passing X and y, as opposed to a formula:

```
X.train <- as.matrix(HousingData.train[,-1])
y.train <- as.matrix(HousingData.train[,1])
fit.rigid <- glmnet(X.train,y.train,alpha=0,lambda=0.1)
```

Regularisation: Rigid regression example

Here we see the test error as a function of the shrinkage parameter.



Regularisation: Lasso

The **lasso** is nearly the same as rigid regression with the only difference that in the penalty term we consider $|a_i|$ as opposed to a_i^2:

$$\min_{\mathbf{a}} \left(\sum_{i=1}^{n} [y_i - (a_0 + a_1 x_{i1} + \ldots + a_p x_{ip})]^2 \quad + \quad \lambda \sum_{i=1}^{p} |a_i| \right)$$

Regularisation: Lasso vs rigid regression

```
fit.rigid <- glmnet(X.train,y.train,alpha=0,lambda=0.05)
fit.lasso <- glmnet(X.train,y.train,alpha=1,lambda=0.05)
```

Here are the estimated coefficients for rigid regression:

```
(fit.rigid$beta)[1:4]
```

```
## [1]  0.40944422  0.01046672 -0.07407463  0.41995763
```

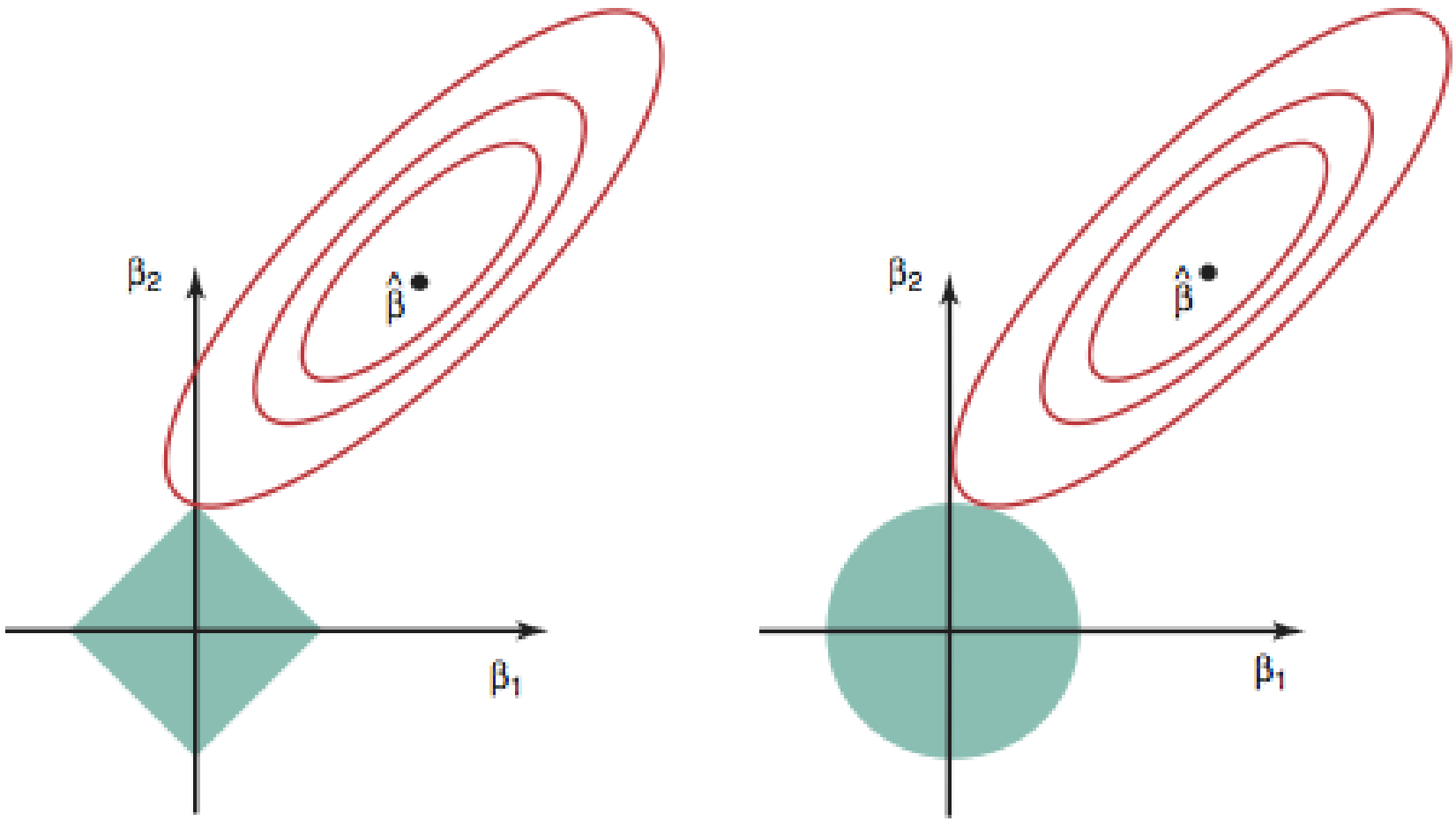
and here are the estimated coefficients for the lasso:

```
(fit.lasso$beta)[1:4]
```

```
## [1] 0.369265264 0.008869284 0.000000000 0.000000000
```

Regularisation: Lasso vs rigid regression

- The **lasso chooses several coefficient exactly to zero**. This is for example good for interpretation.
- It can be explained through the following figure:



High-dimensional data

- When there are many more predictors than observations, $p \gg n$, we speak of **high-dimensional data**.

In general, for $p > n$. The solution to ordinary least squares:

$$\hat{\mathbf{a}} = (X^T X)^{-1} X^T \mathbf{y}$$

does not exist anymore. When using for example rigid regression the pseudo-inverse gets regularised solving the existence issue:

$$\hat{\mathbf{a}} = (X^T X + \lambda I)^{-1} X^T \mathbf{y}$$

Thus regularisation is essential for high-dimensional data!

Case study: The Arcene data set

- The Arcene data set is data for a classification problem where we want to infer cancer (label 1) or no cancer (label -1) from mass-spectroscopy data.
- The training set consists of only n=100 observations. However, the mass-spectrometric data gives rise to p=10000 predictors.
- Thus we have a typical example of high-dimensional data with $p \gg n$.

Case study: The Arcene data set

We read in the training data which is in a separate file:

```
X.train <- read.csv("arcene_train.data.txt", sep=" ",heade
X.train$V10001 <- NULL  # this is an artefact from the lin
X.train <- as.matrix(X.train)
y.train <- read.csv("arcene_train.labels.txt", sep=" ",hea
y.train <- as.factor(y.train$label)
```

The same is done for the CV/test data.

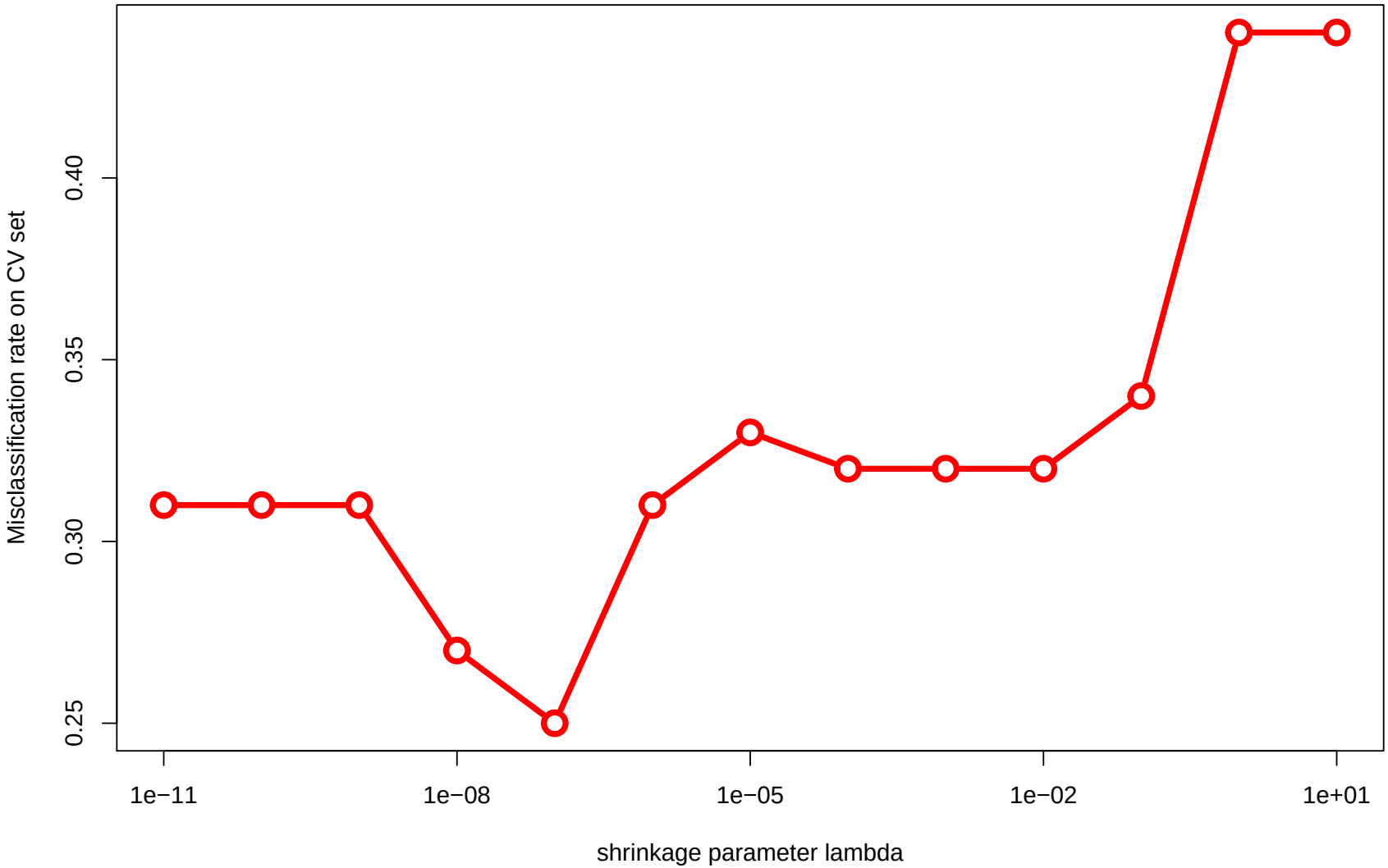
Case study: The Arcene data set

You can fit a logistic regression with lasso penalty using the `glmnet` function by including the attribute `family="binomial"` to indicate that we are doing logistic regression. This is the same attribute as in the `glm` function. For example:

```
fit.lasso <- glmnet(X.train, y.train, alpha=1,
                    lambda=0.05, family="binomial")
```

Case study: The Arcene data set

Test error and best fit:



You will continue this analysis in the exercises.

Discussion

- We discussed model selection using the `leap` package; this included: **best subset selection** (exhaustive search), as well as **forward and backward selection**.
- We introduced the highly important technique of **regularisation**, in particular, **rigid regression and the lasso**.
- The lasso had the advantage over rigid regression, that it chooses several coefficients exactly equal zero, which was particularly good for interpretation of the final model
- Regularisation is essential for **high-dimensional data**. We look at the Arcene data set as an example.