VICTORIA UNIVERSITY OF WELLINGTON
TE HERENGA WAKA

AIML427 Big Data

Week 7-8: Regression 1: Linear Regression and Shrinkage Methods

Dr Qi Chen

School of Engineering and Computer Science

Victoria University of Wellington

Qi.Chen@ecs.vuw.ac.nz

Outline

- R Resources
- The Credit Dataset
- Linear Regression Model
  - Parameter Estimation
  - Linear Regression with R
- Model Selection
- Bias-Variance Trade-off
- Shrinkage Methods or Regularization: Ridge Regression
- Shrinkage Methods or Regularization: Lasso
- Choosing the Tuning Parameter λ
- How Penalised Methods Work
- Summary

Resources

- The textbook for the next 2 weeks is "An Introduction to Statistical Learning – with Applications in R" by James et al.
- This is available for free at https://www.statlearning.com/
- The site also provides links that you might find useful
- If I refer to a section in the book, I'll write, e.g.,"ISLR Section 2.3"
- In R, make sure you install the ISLR package that was developed to go with the textbook

If you are new to R...

- ... don't worry! You will find *learning by doing* is the answer.
- First of all:
  - R is a simple language for statistical computing
  - Obtain it free from www.r-project.org or, if you prefer an IDE, www.rstudio.com
  - Work through the introductory lab in ***ISLR Section 2.3***
    - **Note** as *new versions of R available*, there might be differences between the book and the output from R
- Some notes:
  - R relies heavily on functions (often user-contributed)
  - ?... brings up the help on ...
  - = and <- both work as assignment operators in R
  - Square brackets, e.g. X[1,2], are used to reference array elements (*indexed from 1*)
  - $, e.g. *X$name*, is used to reference named elements of an object
  - Make sure any files you need are in your working directory

Regression

Regression vs Classification:

- Response variable: *quantitative or categorical/qualitative*
- *Logistic regression*?

The problems we consider will:

- be supervised – we know the outcome/response y
- be offline – the dataset is fixed
- involve a structured dataset, in particular the *matrix of predictors/features X*
  - It's easier! We know how to do it already
  - Unstructured datasets are often processed to give structured datasets, e.g., spam filter

The Credit dataset

- The Credit dataset is introduced in ISLR Section 3.3, available on the course website
- Credit card balance is the response variable

```
> Credit = read.csv("Credit.csv",header=TRUE)
> head(Credit)

   income limit rating cards age education gender student married ethnicity balance
1  14.891  3606    283     2  34        11   Male      No     Yes Caucasian     333
2 106.025  6645    483     3  82        15 Female     Yes     Yes     Asian     903
3 104.593  7075    514     4  71        11   Male      No      No     Asian     580
4 148.924  9504    681     3  36        11 Female      No      No     Asian     964
5  55.882  4897    357     2  68        16   Male      No     Yes Caucasian     331
6  80.180  8047    569     4  77        10   Male      No      No Caucasian    1151

> dim(Credit)

[1] 400  11
```

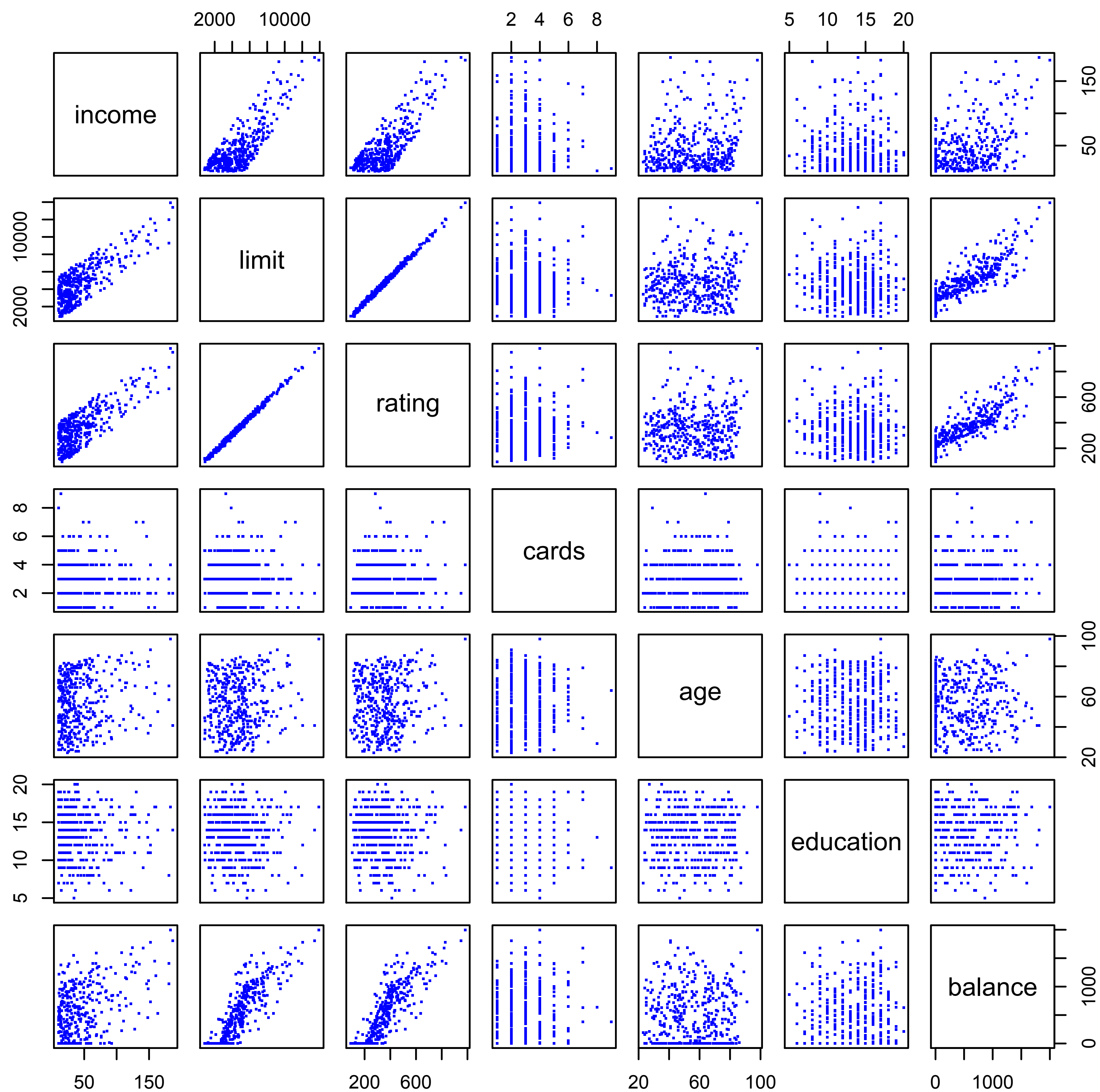
The Credit dataset

- Gender, student, married, ethnicity are categorical or qualitative *variables/predictors*

```
> summary(Credit}

     income           limit           rating          cards            age       
 Min.   : 10.35   Min.   :  855   Min.   : 93.0   Min.   :1.000   Min.   :23.00  
 1st Qu.: 21.01   1st Qu.: 3088   1st Qu.:247.2   1st Qu.:2.000   1st Qu.:41.75  
 Median : 33.12   Median : 4622   Median :344.0   Median :3.000   Median :56.00  
 Mean   : 45.22   Mean   : 4736   Mean   :354.9   Mean   :2.958   Mean   :55.67  
 3rd Qu.: 57.47   3rd Qu.: 5873   3rd Qu.:437.2   3rd Qu.:4.000   3rd Qu.:70.00  
 Max.   :186.63   Max.   :13913   Max.   :982.0   Max.   :9.000   Max.   :98.00  
   education          gender     student   married              ethnicity       balance       
 Min.   : 5.00    Male  :193   No :360   No :155   African American: 99   Min.   :   0.00  
 1st Qu.:11.00   Female:207   Yes: 40   Yes:245   Asian           :102   1st Qu.:  68.75  
 Median :14.00                                    Caucasian       :199   Median : 459.50  
 Mean   :13.45                                                           Mean   : 520.01  
 3rd Qu.:16.00                                                           3rd Qu.: 863.00  
 Max.   :20.00                                                           Max.   :1999.00  
```

The Credit dataset

- *plot*(Credit[,-(7:10)],pch=46,col="blue")
- Note that *limit* and *rating* are highly correlated



Linear Regression [ISLR Section 3.1 and 3.2]

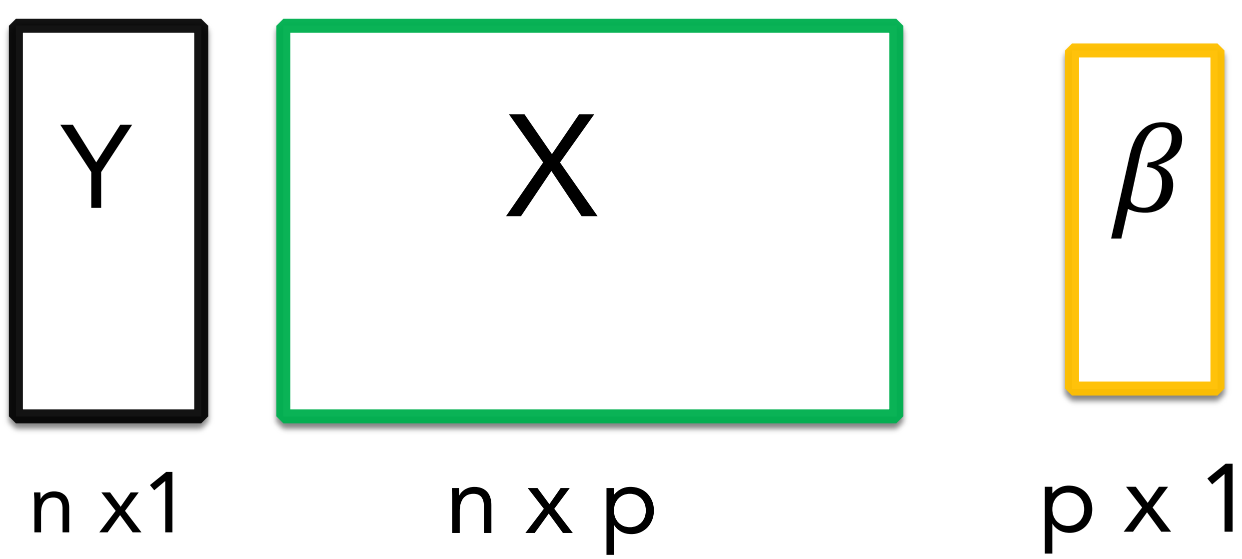
- Linear regression is a very simple approach
  - Many fancy statistical learning approaches can be seen as generalisations or extensions of LR
- For the *Advertising* data:
  - Is there a relationship between advertising *budget* and *sales*?
  - How strong is the relationship between advertising budget and sales?
  - Which (subset of) media (TV, radio, and newspaper) contribute to sales?
  - How accurately can we estimate the effect of each medium on sales?
  - How accurately can we predict future sales?
  - Is the relationship linear?
  - Is there synergy (*interaction effect*) among the advertising media?

Linear Regression Model

Assume the *true* relationship between X and Y:

$$\boldsymbol{y_i} = \boldsymbol{\beta_0} + \sum_{\boldsymbol{j=1}}^{\boldsymbol{p}} \boldsymbol{x_{ij}\beta_j} + \epsilon_i \qquad \text{for } i = 1, \dots, n$$

- *LR* : $\boldsymbol{\hat{y}_i} = \boldsymbol{\beta_0} + \sum_{\boldsymbol{j=1}}^{\boldsymbol{p}} \boldsymbol{x_{ij}\beta_j}$
- $\beta_0, \beta_1, \dots, \beta_p$: the model coefficients or parameters, *unknown constants*
  - β_0 intercept --- the expected value of y_i given all $x_{ij} = 0$
  - $\beta_1, \dots, \beta_p$ are the slope terms --- the average increase in y_i associated with one-unit increase in x_{ij}
- ϵ_i is the error term, $\epsilon_i \sim N(0, \sigma^2)$



Linear Regression: Parameter Estimation

- Estimating parameters $\beta_0, \beta_1, \ldots, \beta_p$ by minimising the Residual Sum of Squares (RSS) on a given set of data/observations:

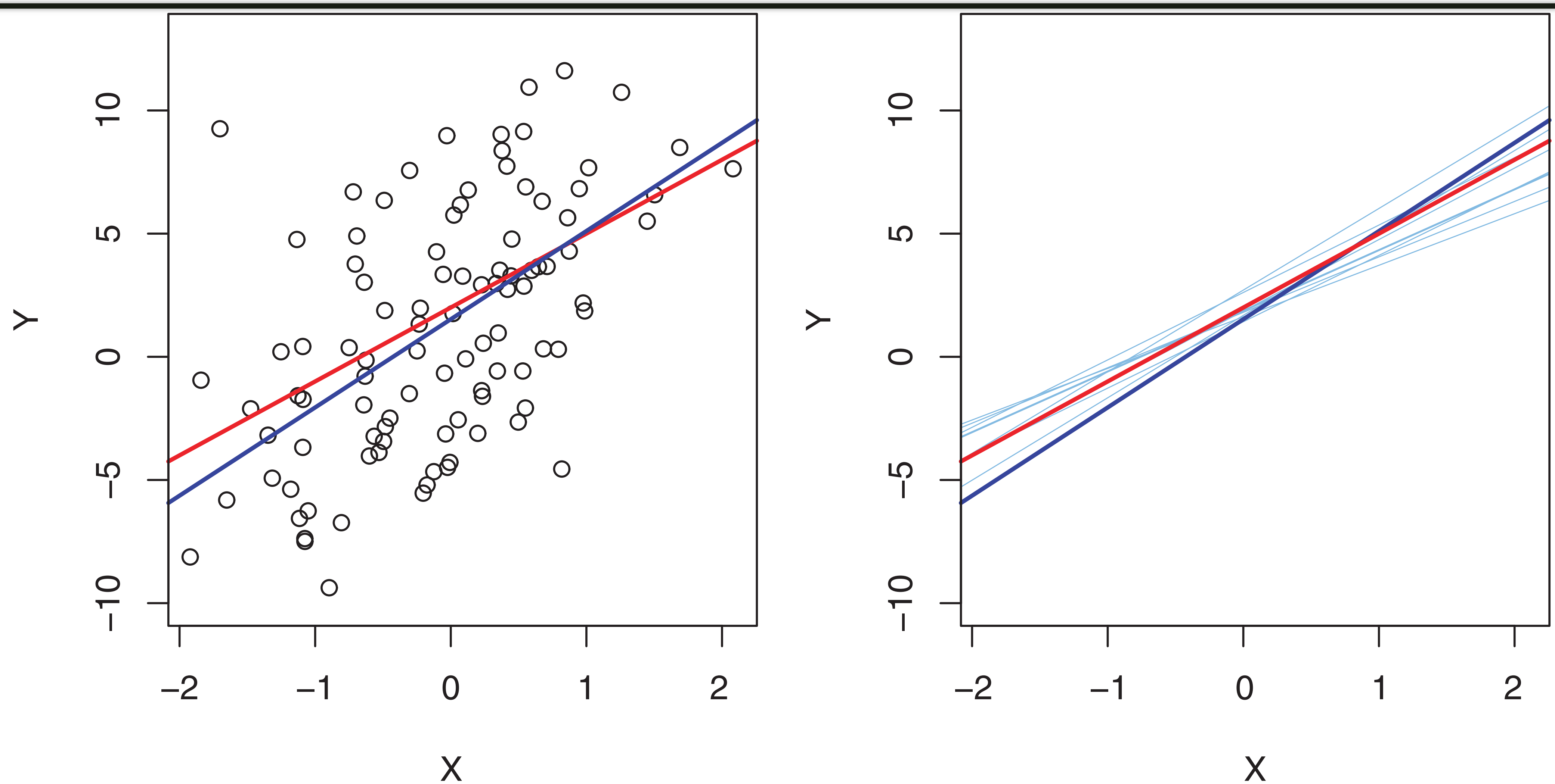
$$RSS = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{n} (y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2$$

  - RSS is an error measure.
    - There are other error measures, MSE, RMSE, R Squared

- Residual: the difference between the observed response value and the predicted response value

- The estimation procedure is often called *lest squares (coefficient) estimation*
- The line generated is the *least squares line*

True Relationship vs Least Squares Line



- **Black**: the observed/given data instances
- **Red**: the true relationship f(X), known as the *population regression line*
- **Dark Blue**: the *least squares line*, the *least squares estimation* for f(X) based on the observed data

- **Red**: the true relationship f(X), known as the *population regression line*
- **Dark Blue**: the *least squares line*, the *least squares estimation* for f(X) based on the observed data
- **Light Blue**: the ten *least squares lines,* each computed on the basis of a separate random set of observations/instances

Different Sets of Observations

- The true relationship is generally not known for real data, but the least squares line can always be computed using the coefficient estimates
- In real applications, we have access to a set of observations (training data) from which to compute the least squares line
  - the *population regression line is unobserved*
- Notice that *different data sets generated from the **same** true model* result in slightly different least squares lines

- If estimated on the basis of a particular data set, the least squares line won't be exactly the same as the true population regression line, but if estimates obtained over *a huge number of data sets*, then the *average of these estimates* would be spot on!

Linear Regression with R

- Here is how to fit the linear model in R
- Note how the categorical variables have been recoded as *indicator or dummy* variables, taking 0 or 1
- For the Credit dataset, n = 400, p = 11

```
> X = model.matrix(balance~.,Credit)[,-1]
> y = Credit$balance
> head(X)

   income limit rating cards age education genderFemale studentYes marriedYes
1  14.891  3606    283     2  34        11            0          0          1
2 106.025  6645    483     3  82        15            1          1          1
3 104.593  7075    514     4  71        11            0          0          0
4 148.924  9504    681     3  36        11            1          0          0
5  55.882  4897    357     2  68        16            0          0          1
6  80.180  8047    569     4  77        10            0          0          0
  ethnicityAsian ethnicityCaucasian
1              0                  1
2              1                  0
3              1                  0
4              1                  0
5              0                  1
6              0                  1
```

Linear Regression with R

```
> linear.mod = lm(y~X)
> summary(linear.mod)

Coefficients:
                      Estimate Std. Error t value Pr(>|t|)
(Intercept)         -479.20787   35.77394 -13.395  < 2e-16 ***
Xincome               -7.80310    0.23423 -33.314  < 2e-16 ***
Xlimit                 0.19091    0.03278   5.824 1.21e-08 ***
Xrating                1.13653    0.49089   2.315   0.0211 *
Xcards                17.72448    4.34103   4.083 5.40e-05 ***
Xage                  -0.61391    0.29399  -2.088   0.0374 *
Xeducation            -1.09886    1.59795  -0.688   0.4921
XgenderFemale        -10.65325    9.91400  -1.075   0.2832
XstudentYes          425.74736   16.72258  25.459  < 2e-16 ***
XmarriedYes           -8.53390   10.36287  -0.824   0.4107
XethnicityAsian       16.80418   14.11906   1.190   0.2347
XethnicityCaucasian   10.10703   12.20992   0.828   0.4083
```

Model Selection

In classical statistics, we often adopt the following rather restricted approach to model selection:

- consider *nested models*, $M_1 \subset M_2$
- assume the null hypothesis H_o: M_1 is sufficient to explain the data against M_2
- reject H_o if the P-value of an appropriate statistical test (e.g. ANOVA) is less than some threshold

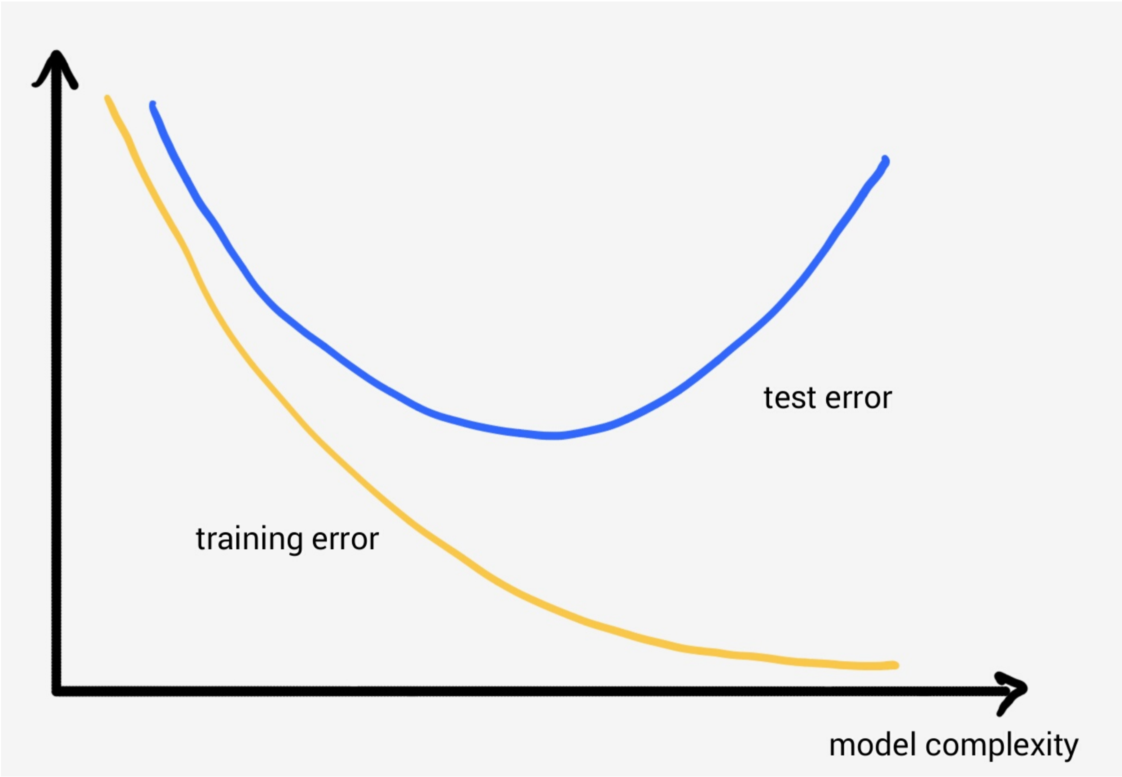
There is, however, a deeper and more useful way:

training error vs test error

- a model that underfits the training data will have a large error on test data
- a model that overfits the training data will also have a large error on test data

Model Selection: training error vs test error

- The **test error** should be contrasted with the **training error**
  - The training error can be made arbitrarily small by making the model more complex, but this is seldom what we want



  - An example for the test error in R for the Credit dataset
    - This is the test error for the model with all features included; we could compare it to test errors from models with fewer features

```
> set.seed(987654312)
> train = sample(1:nrow(X),nrow(X)/2)
> test = -train
> linear.mod = lm(y[train]~X[train,])
> linear.pred = coef(linear.mod)[1]+X[test,] %*% coef(linear.mod)[-1]
> mean((linear.pred-y[test])^2)

[1] 10446.33
```

Model Selection: training error vs test error

Need to estimate the test error, two common approaches:

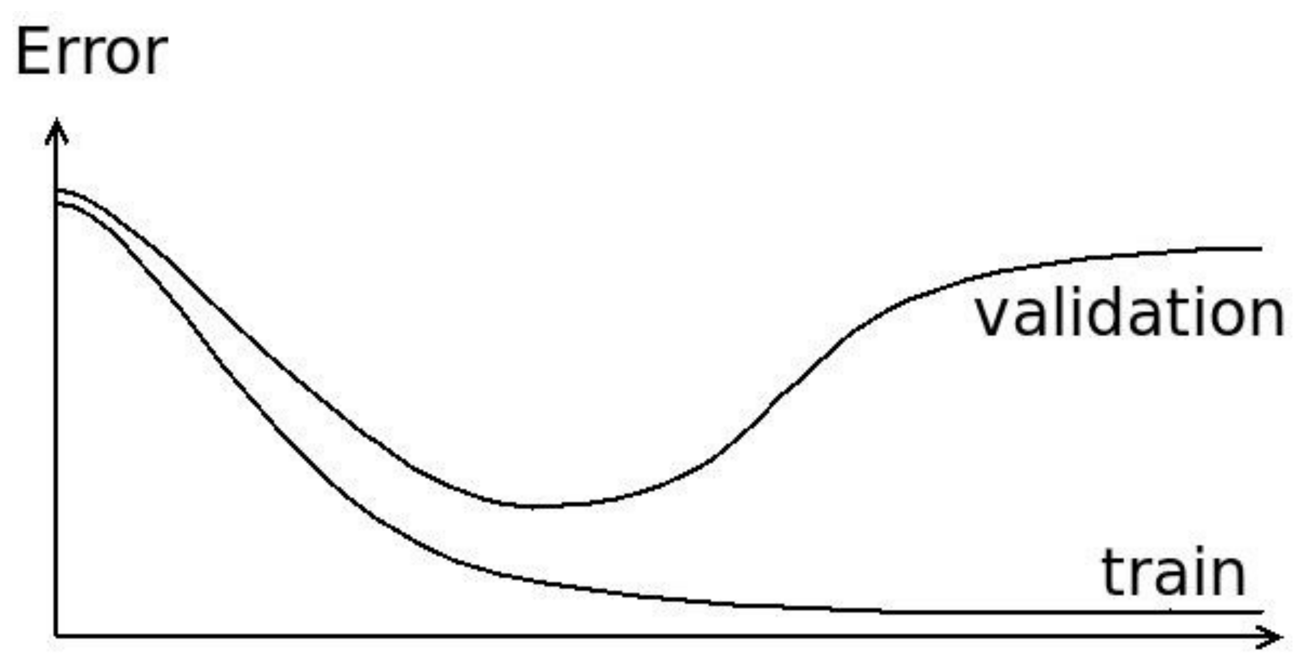
- *Indirectly* estimate test error by making an *adjustment* to the training error to account for the bias due to overfitting,
  - e.g. Akaike information criterion (AIC), Bayesian information criterion(BIC), etc
- *Directly* estimate the test error, using either a validation set approach or a cross-validation approach
  - *Test set should remain unseen.*

- Using a validation set
  - Use y_{train} and X_{train} to find estimates $\hat{\beta}$
  - Predict the outcomes in the validation set $\hat{y} = X_{validation}\hat{\beta}$
  - Compute the *validation* error – typically the mean squared error (MSE), i.e. the mean squared difference between $\hat{y}$ and $y_{validition}$

Model Selection using a Validation Set

- Subset Selection: using different subsets of features to build models.
  - select the model with the smallest validation error:

- If plot training error against validation error:

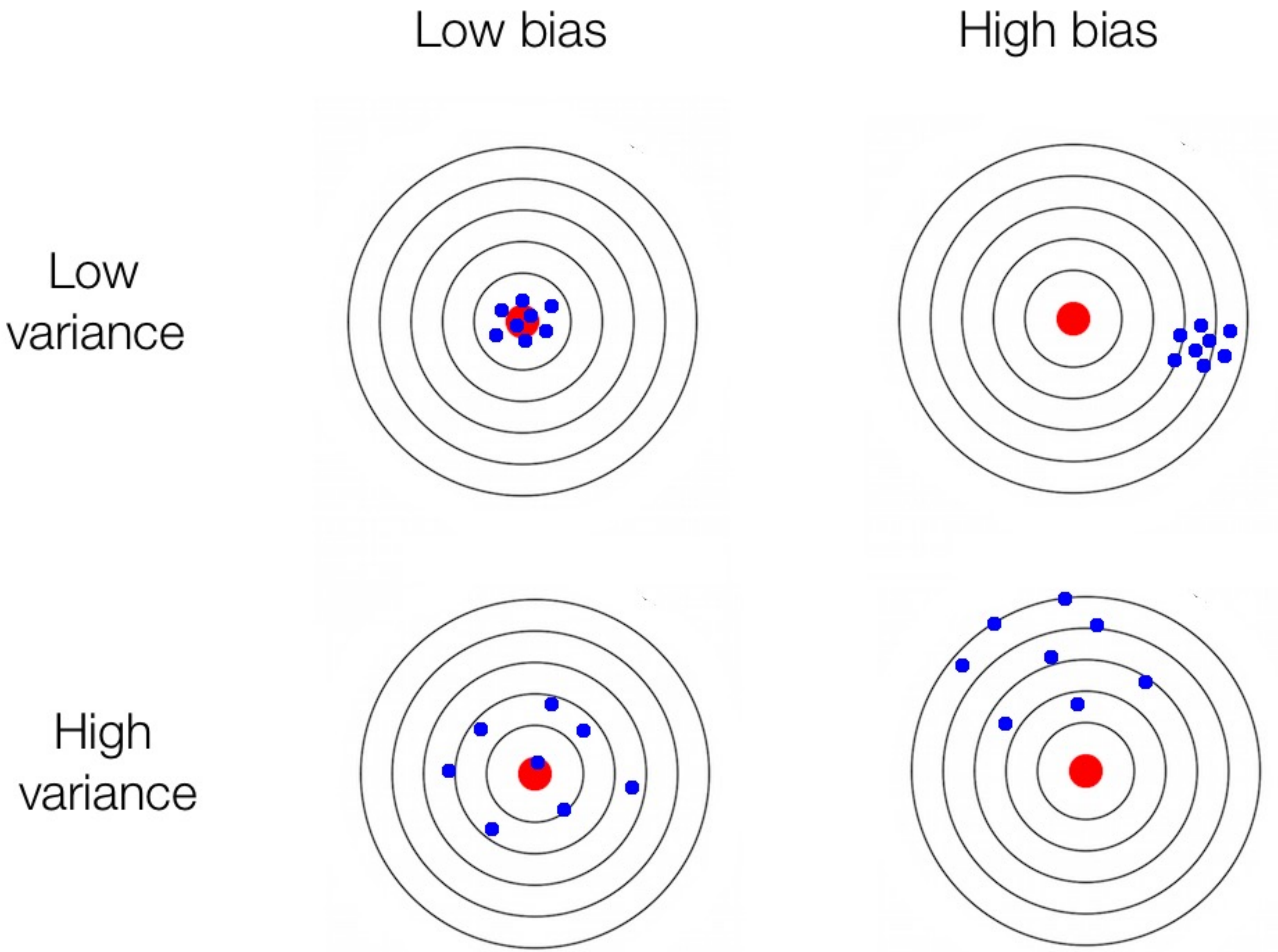


Bias-Variance Trade-off

- All models suffer from ***bias and variance***.
- Typically, the test error is a combination of the bias (squared) and the variance
  - The Short Story: generalization error = bias^2 + variance + noise.
- Bias refers to the error that is introduced by approximating a real-life problem, which may be extremely complicated, by a much simpler model.
  - It is unlikely that any real-life problem truly has a simple *linear* relationship, so undoubtedly result in some bias
- Variance: refers by what amount $\hat{y}$ will change if estimating it using a different training set
  - if using a different training data set to estimate it

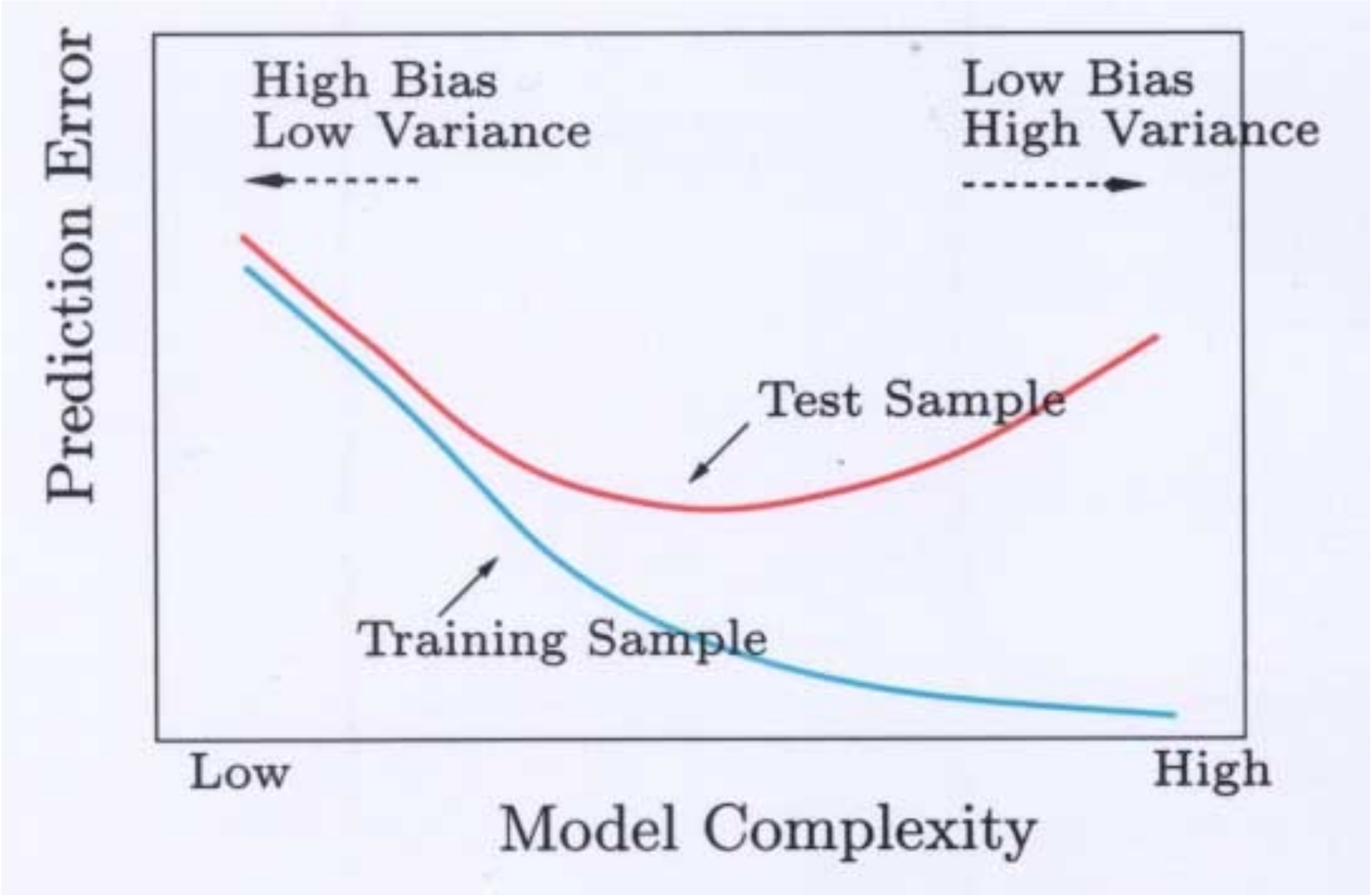
[See also ISLR Section 2.2.2]

Bias-Variance Trade-off



Bias-Variance Trade-off

- Models that underfit tend to have high bias and low variance
- Models that overfit tend to have low bias and high variance
- Model selection is finding the model that best balances between bias and variance



from the book of "Elements of Statistical Learning"

Regularisation/ Shrinkage Methods

- If the $\widehat{\beta}_j$ s are unconstrained...
  - They can explode
  - And hence are susceptible to very high variance

- Using a technique that ***constrains or regularises*** the coefficient estimates, or equivalently, that ***shrinks*** the coefficient estimates towards zero.
  - Regularisation, shrinkage penalty, *constraints*
  - Shrinking the coefficient estimates can significantly reduce their variance
  - attempt to *automate the bias-variance trade-off*.
- Two best-known techniques for shrinking the regression coefficients towards zero are ridge regression and the lasso
- See also ISLR Section 6.2

Ridge Regression

- Ridge regression is very similar to least squares, except that the coefficients $\beta_0,\beta_1,\ldots,\beta_p$ are estimated by minimising:

$$\sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda\sum_{j=1}^{p}\beta_j^2 = RSS + \lambda\sum_{j=1}^{p}\beta_j^2$$

where $\lambda \geq 0$ is called the **tuning parameter**

- $\lambda\sum_{j=1}^{p}\beta_j^2$ is *shrinkage penalty* term
  - If $\lambda = 0$, we revert to ordinary linear regression;
  - If $\lambda \to \infty$, we get an intercept-only model
  - λ controls the size of the coefficients, shrinking the estimates of β_j towards zero
  - Model complexity goes down as λ goes up

- Solution is indexed by the tuning parameter λ:
  - So for each λ, we have a solution, λ is trace out a path of solutions
- Important, by tradition: Matrix X should be **standardized** (mean 0, standard deviation 1); y is assumed to be **centered**

Ridge regression

- A convenient package for doing penalized regression in R is *glmnet*

```
> library(glmnet)
> grid = 10^seq(5,-2,length=100)
> ridge.mod = glmnet(X,y,alpha=0,lambda=grid)
```

- *grid* is a *decreasing* sequence of values for the *tuning* parameter λ
- *glmnet* does penalised regression for each value of the tuning parameter λ
- *alpha=0* means to do ridge regression
- *glmnet* automatically standardises X
- See also ISLR Section 6.6

Ridge Regression

- When λ is small, ridge regression gives similar answers to ordinary regression:

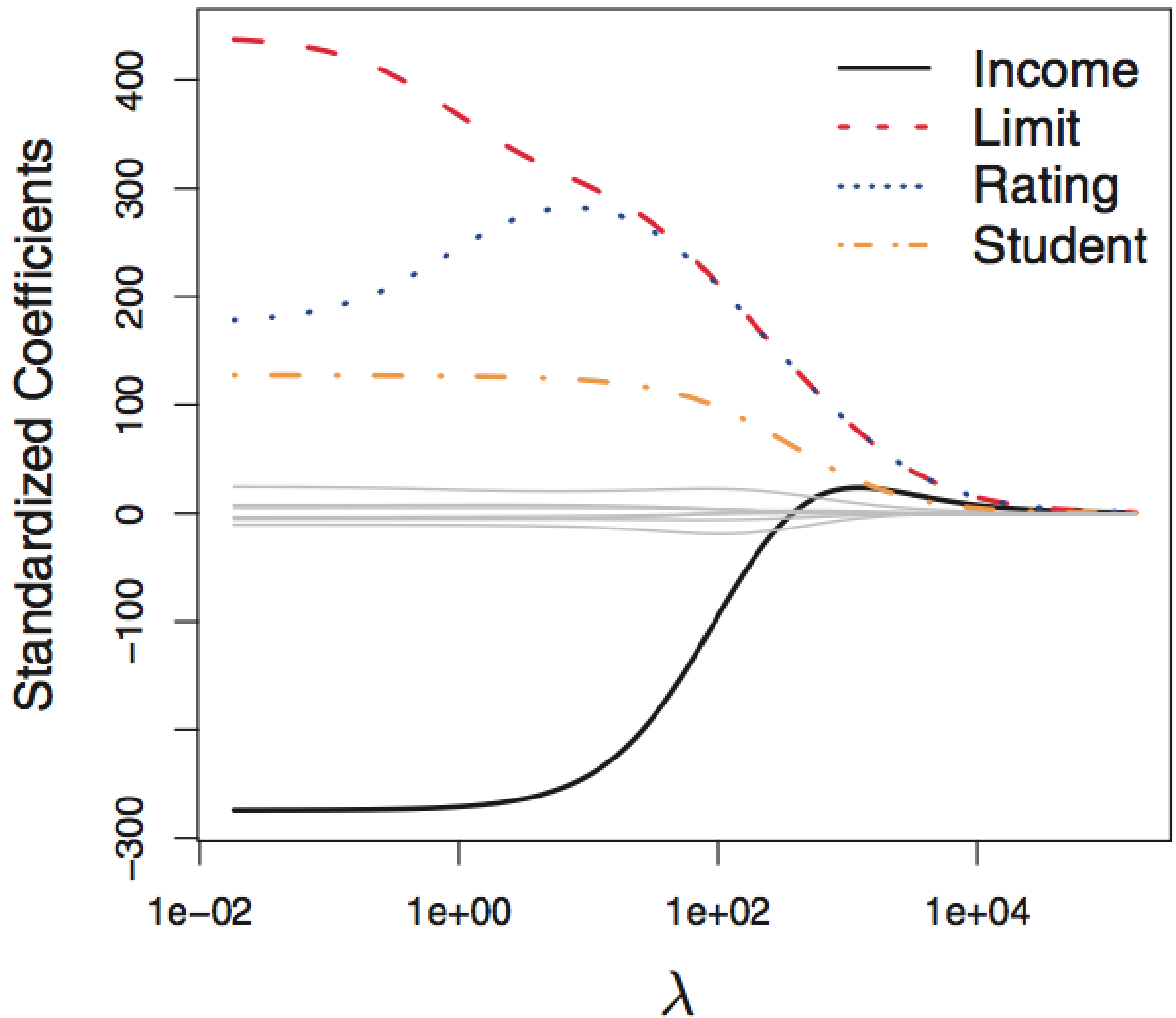
```
> ridge.mod$lambda[100]
[1] 0.01
> coef(ridge.mod)[,100]
 (Intercept)          income           limit          rating
-484.5225957      -7.8000469       0.1763554       1.3529988
       cards             age       education    genderFemale
  16.6769557      -0.6161700      -1.0438640     -10.6555578
  studentYes      marriedYes  ethnicityAsian ethnicityCaucasian
 425.0254323      -9.0360841      17.1807594      10.1483725
```

Ridge Regression

- When λ is large, ridge regression shrinks the parameter estimates when compared to the least squares estimates:

```
> ridge.mod$lambda[40]
[1] 174.7528
> coef(ridge.mod)[,40]
  (Intercept)          income           limit          rating
-231.85315960     -1.66321463      0.08128525      1.20615729
        cards             age       education    genderFemale
  15.78664431     -1.07643143     -0.02942012      2.29848286
   studentYes      marriedYes  ethnicityAsian ethnicityCaucasian
 292.41955822    -11.70027752      5.86343217      5.82465221
```

Ridge Regression



ISLR Figure 6.4: Penalized methods are shrinkage methods

Test Error in Ridge Regression

- When λ is small, we get only small improvement in the test error over linear regression:
  - Setting thresh to a smaller value (default is 10^{-7}) is often advisable; better numerical accuracy at the cost of compute time
  - NB s – not lambda! – sets the value of the tuning parameter
  - s doesn't have to be one of the values of grid; glmnet will happily interpolate

```
> ridge.mod = glmnet(X[train,],y[train],alpha=0,lambda=grid,thresh=1e-12)
> ridge.pred = predict(ridge.mod,s=0.01,newx=X[test,])
> mean((ridge.pred-y[test])^2)

[1] 10438.68
```

- See also ISLR Section 6.6

Test Error in Ridge Regression

- If λ is a little larger, we see definite improvement:

```
> ridge.pred = predict(ridge.mod,s=7,newx=X[test,])
> mean((ridge.pred-y[test])^2)

[1] 10126.62
```

- But λ mustn't get too big...

```
> ridge.pred = predict(ridge.mod,s=20,newx=X[test,])
> mean((ridge.pred-y[test])^2)

[1] 10823.96
```

- Ridge regression will include all p variables/predictors in the final model
  - The penalty term shrinks all of the coefficients towards zero, but it will not set any of them exactly to zero (unless $\lambda = \infty$)

The Lasso

- Lasso stands for "Least absolute shrinkage and selection operator" and is another penalised method for regression.

- The lasso estimates the parameters $\beta_0, \beta_1, \ldots, \beta_p$ by minimising:

$$\sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} |\beta_j| = RSS + \lambda \sum_{j=1}^{p} |\beta_j|$$

- The form of the *penalty term is different*, but everything we said for ridge regression holds for the lasso
- L1 regularisation
- See also ISLR Section 6.2.1

The Lasso

- Once again we use *glmnet*. See also ISLR Section 6.6

```
> lasso.mod = glmnet(X,y,alpha=1,lambda=grid,thresh=1e-12)
```

- alpha =1 means do the Lasso penalty
- When λ is small, the lasso gives similar answers to the least squares estimates:

```
> lasso.mod$lambda[100]
[1] 0.01
> coef(lasso.mod)[,100]
     (Intercept)           income            limit            rating
    -479.2214533       -7.8017798        0.1908155         1.1375750
           cards              age        education      genderFemale
      17.7133191       -0.6135728       -1.0954253       -10.6298778
      studentYes       marriedYes   ethnicityAsian ethnicityCaucasian
     425.7054652       -8.5132864       16.7475384        10.0585961
```

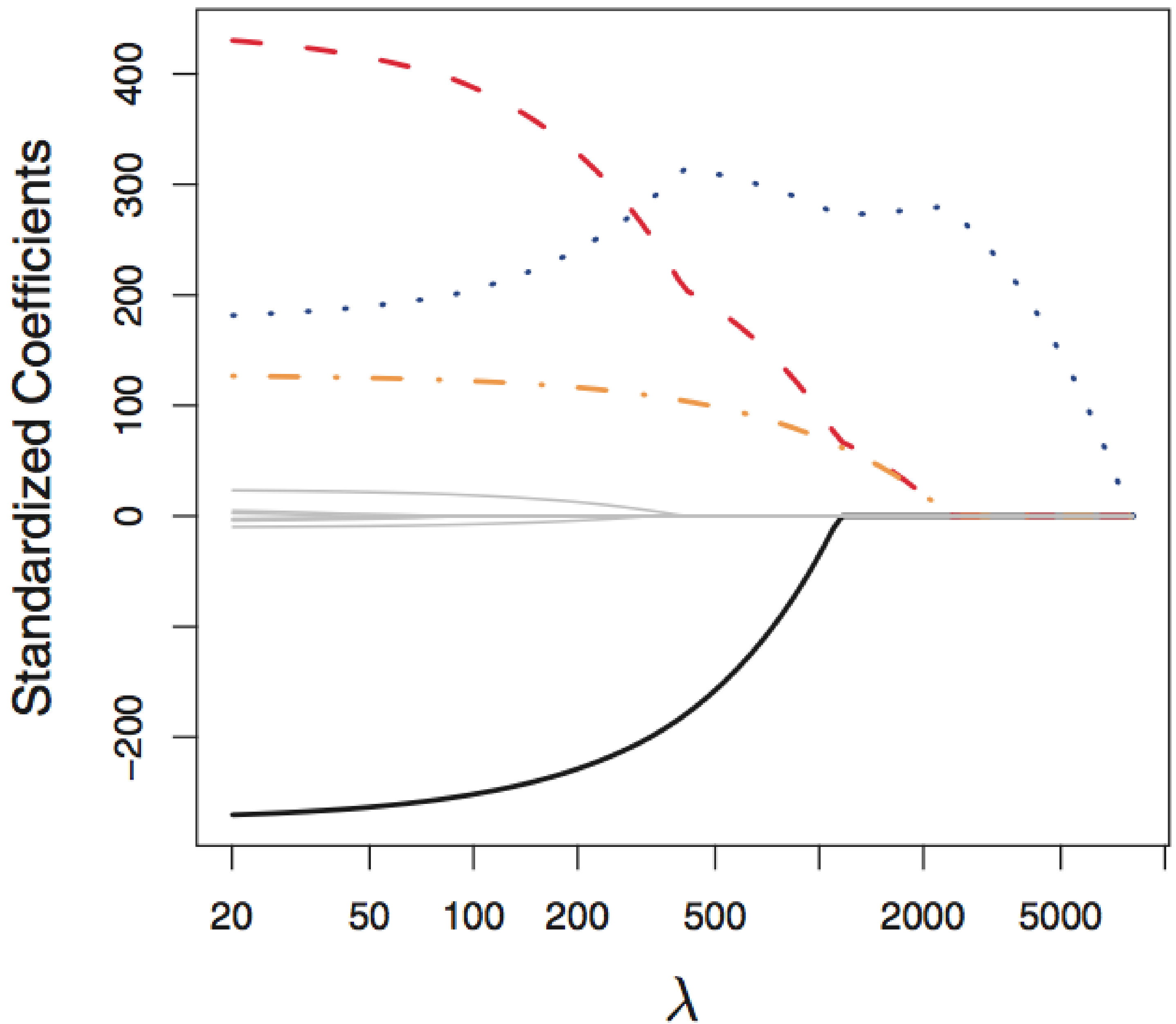
The Lasso

- But when λ gets larger, something quite remarkable happens:

```
> lasso.mod$lambda[60]
[1] 6.734151
> coef(lasso.mod)[,60]
      (Intercept)            income             limit            rating
      -474.514621         -6.916440          0.158927          1.405987
            cards               age         education      genderFemale
        12.124822         -0.362677          0.000000          0.000000
       studentYes        marriedYes   ethnicityAsian ethnicityCaucasian
       399.581608          0.000000          0.000000          0.000000
```

- Any time $\hat{\beta}_j = 0$, this means x_j is not in the model; the lasso is automatically doing feature selection
- This is sometimes referred to as l_1−magic

The Lasso



ISLR Figure 6.6: Feature selection with the lasso

Test error in the lasso

- When λ is small, we get only small improvement in the test error over linear regression:

```
> lasso.mod = glmnet(X[train,],y[train],alpha=1,lambda=grid,thresh=1e-12)
> lasso.pred = predict(lasso.mod,s=0.01,newx=X[test,])
> mean((lasso.pred-y[test])^2)

[1] 10445.01
```

If λ is a little larger, we see definite improvement:

```
> lasso.pred = predict(lasso.mod,s=5,newx=X[test,])
> mean((lasso.pred-y[test])^2)

[1] 10199.61
```

But, again, we don't want λ too big...

```
> lasso.pred = predict(lasso.mod,s=10,newx=X[test,])
> mean((lasso.pred-y[test])^2)

[1] 10525.44
```

Choosing the Tuning Parameter λ

- The obvious issue, however, with ridge regression and the lasso is that they rely on an additional parameter – the tuning parameter λ – that we don't know!

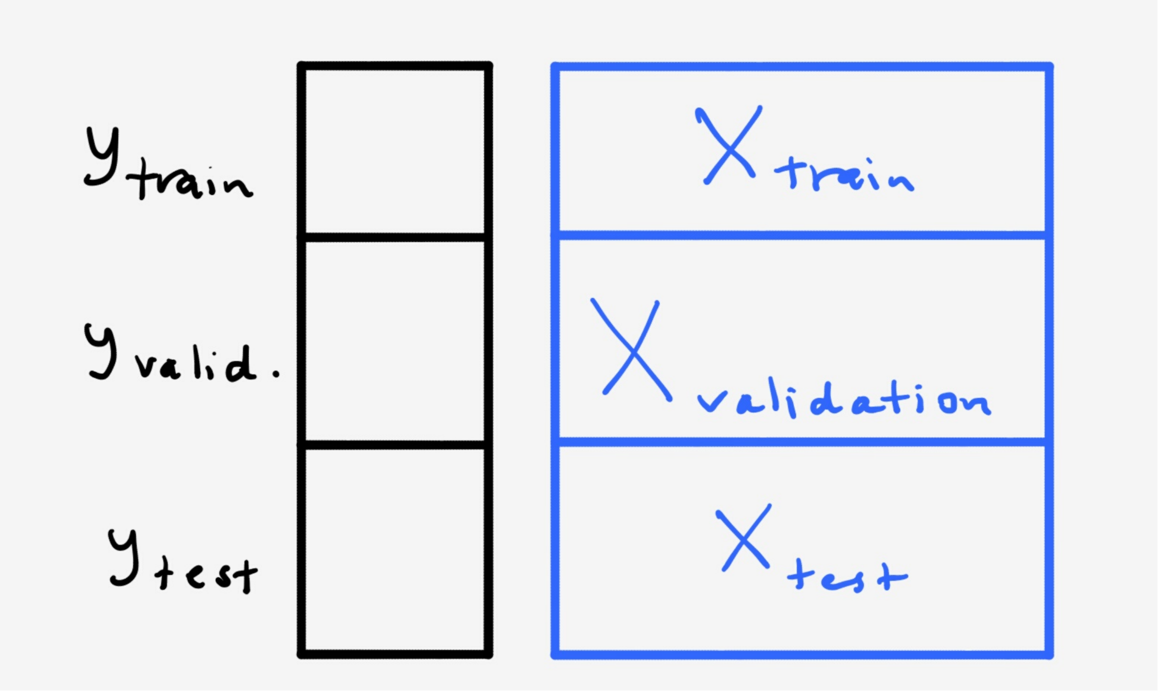
- There are two standard approaches to choosing λ:
  - The use of a validation set
  - Cross-validation

- See also ISLR Section 6.2.3

Validation Set Approach

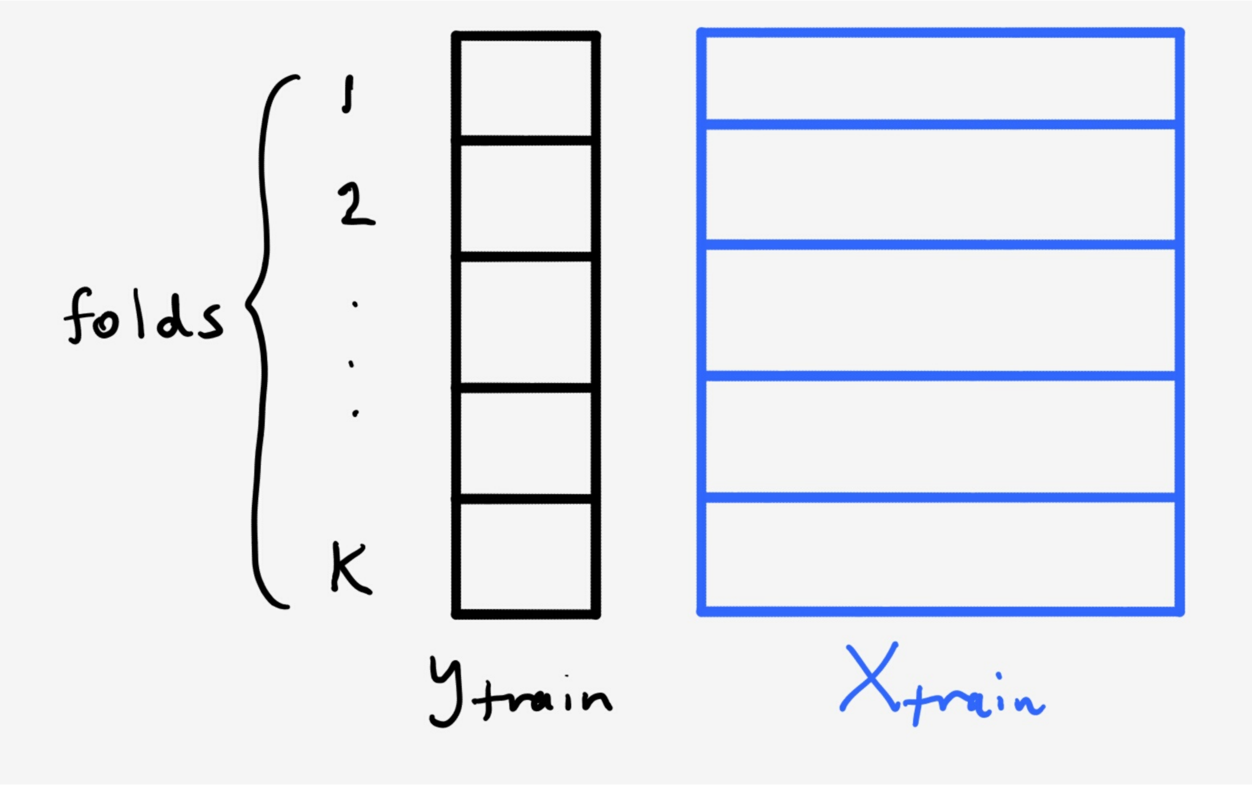
In addition to a training set, we require a randomly selected validation set

- Fix λ and use Y_{train} and X_{train} to find estimates β
- Predict the outcomes in the validation set
    - $y = X_{validation}\ \beta$
- Compute the validation set error between y and $y_{validation}$
- Find λ_{min} that minimises the validation set error
- Finally use λ_{min} to compute the test error



Cross-Validation for Model Selection

- The training set *itself* is divided randomly into K subsets, known as folds. Each fold, in turn, takes on the role of the validation set
  - λ_{min} is chosen to minimize the cross-validation error CV , which is the average of the K validation set errors
  - K = 5 or 10 is typical
- Leave-one-out cross-validation (LOOCV)



Cross-Validation for Ridge Regression

- Fortunately, the *glmnet* package can do cross-validation for us – though we need to take some care.
- Looking at ridge regression for Credit dataset

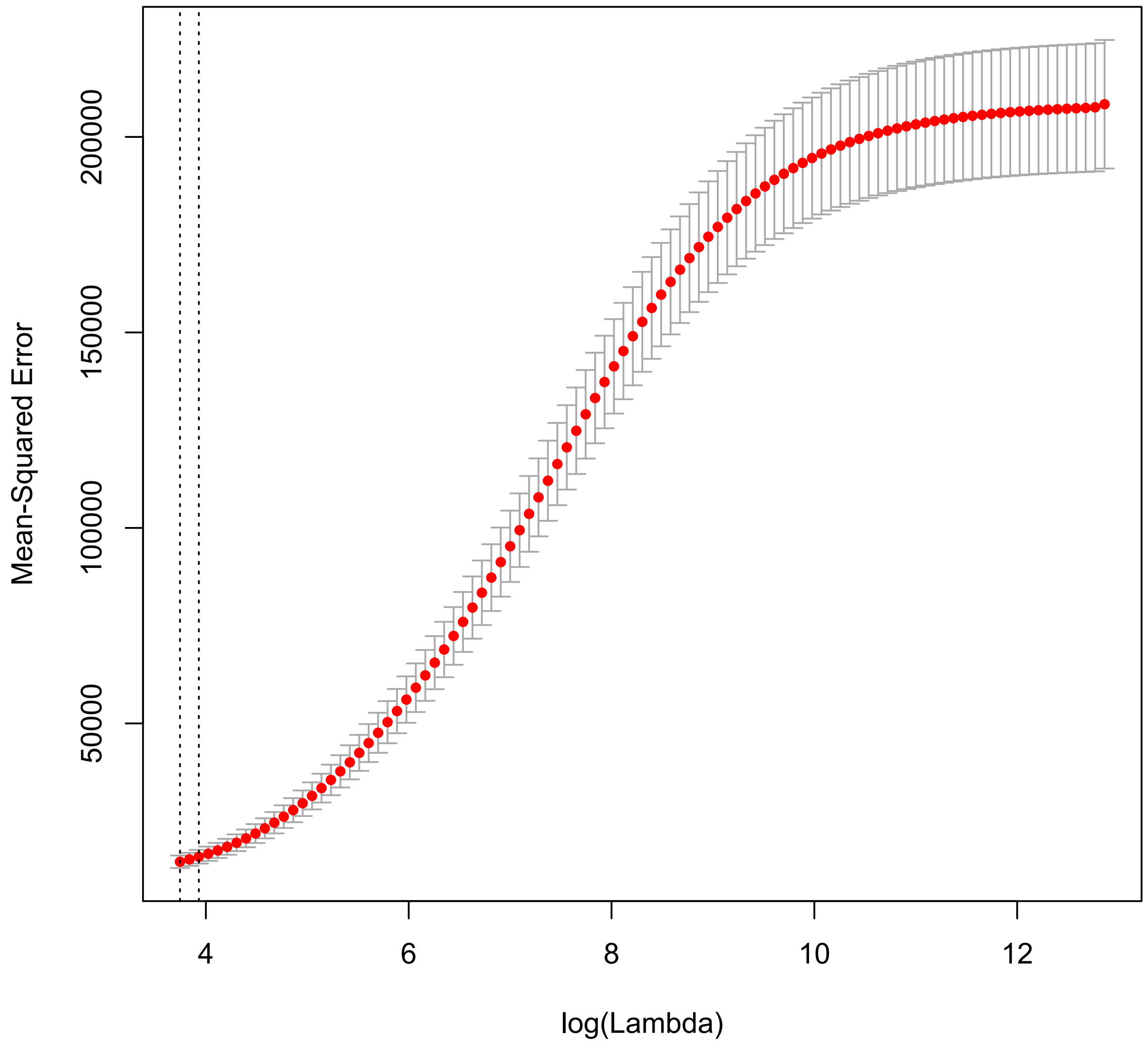
```
> set.seed(987654313)
> cv.out = cv.glmnet(X[train,],y[train],alpha=0,nfolds=10,thresh=1e-12)
> cv.out$lambda.min

[1] 42.29286
```

- The folds are chosen randomly so it pays to *set the random seed for replicability*
- cv.glmnet uses a grid-based search to find λ_{min}

Cross-Validation for Ridge Regression

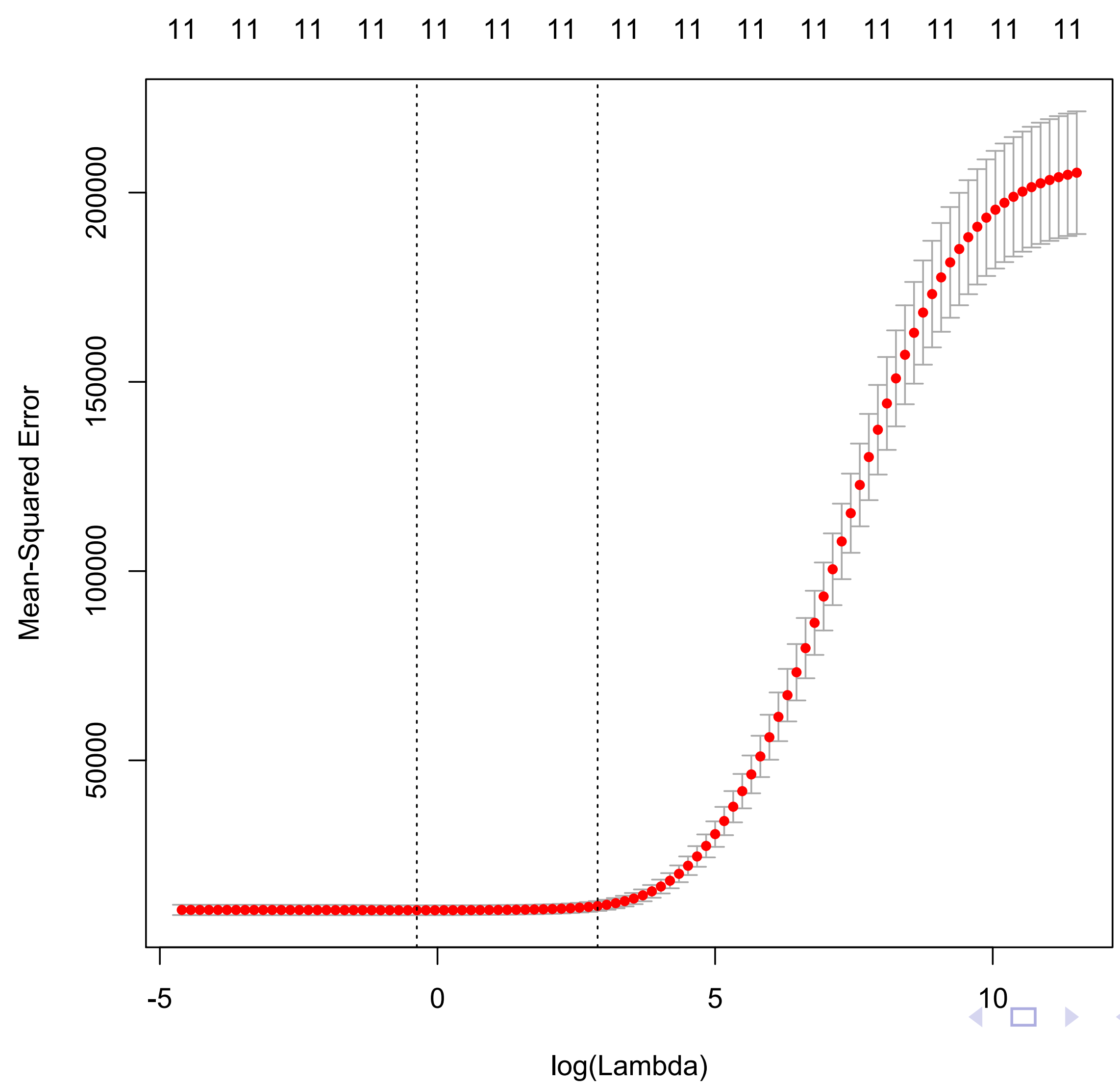
- Plotting the cross-validation output:
  - λ_{min} is suspiciously at the boundary of the search grid. This suggests we should specify our own grid...

```
> plot(cv.out)
```



Cross-Validation for Ridge Regression

```
> set.seed(987654313)
> cv.out = cv.glmnet(X[train,],y[train],alpha=0,lambda=grid,nfolds=10,thresh=1e-12)
> cv.out$lambda.min

[1] 0.6892612
```



Cross-Validation for Ridge Regression

- We can now compute the test error:

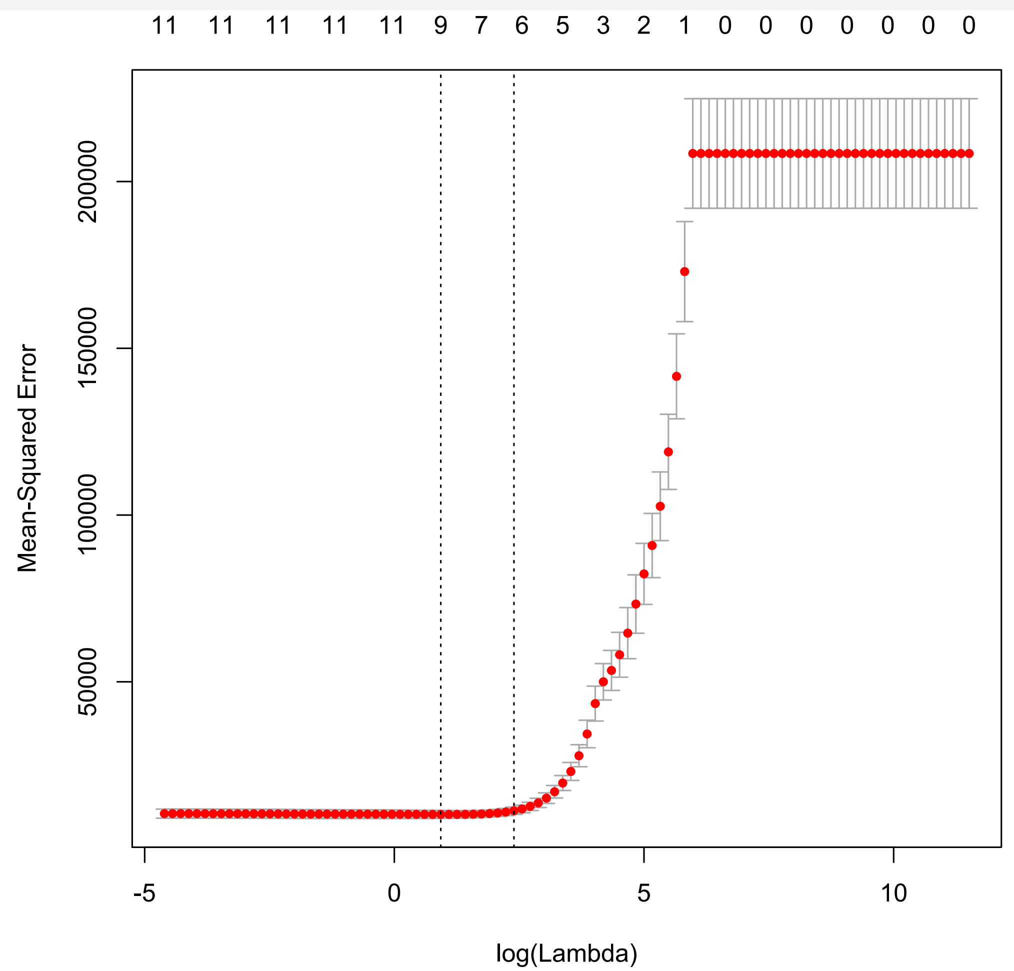
```
> bestlam = cv.out$lambda.min
> ridge.pred = predict(cv.out,s=bestlam,newx=X[test,])
> mean((ridge.pred-y[test])^2)

[1] 10204.5
```

- Sometime, refit the ridge regression model on the full (training) dataset using λ_{min}

Cross-Validation for Lasso

- Cross-validation proceeds in exactly the same way for the lasso.

```
> set.seed(987654313)
> cv.out = cv.glmnet(X[train,],y[train],alpha=1,lambda=grid,nfolds=10,thresh=1e-12)
> cv.out$lambda.min

[1] 2.535364
```



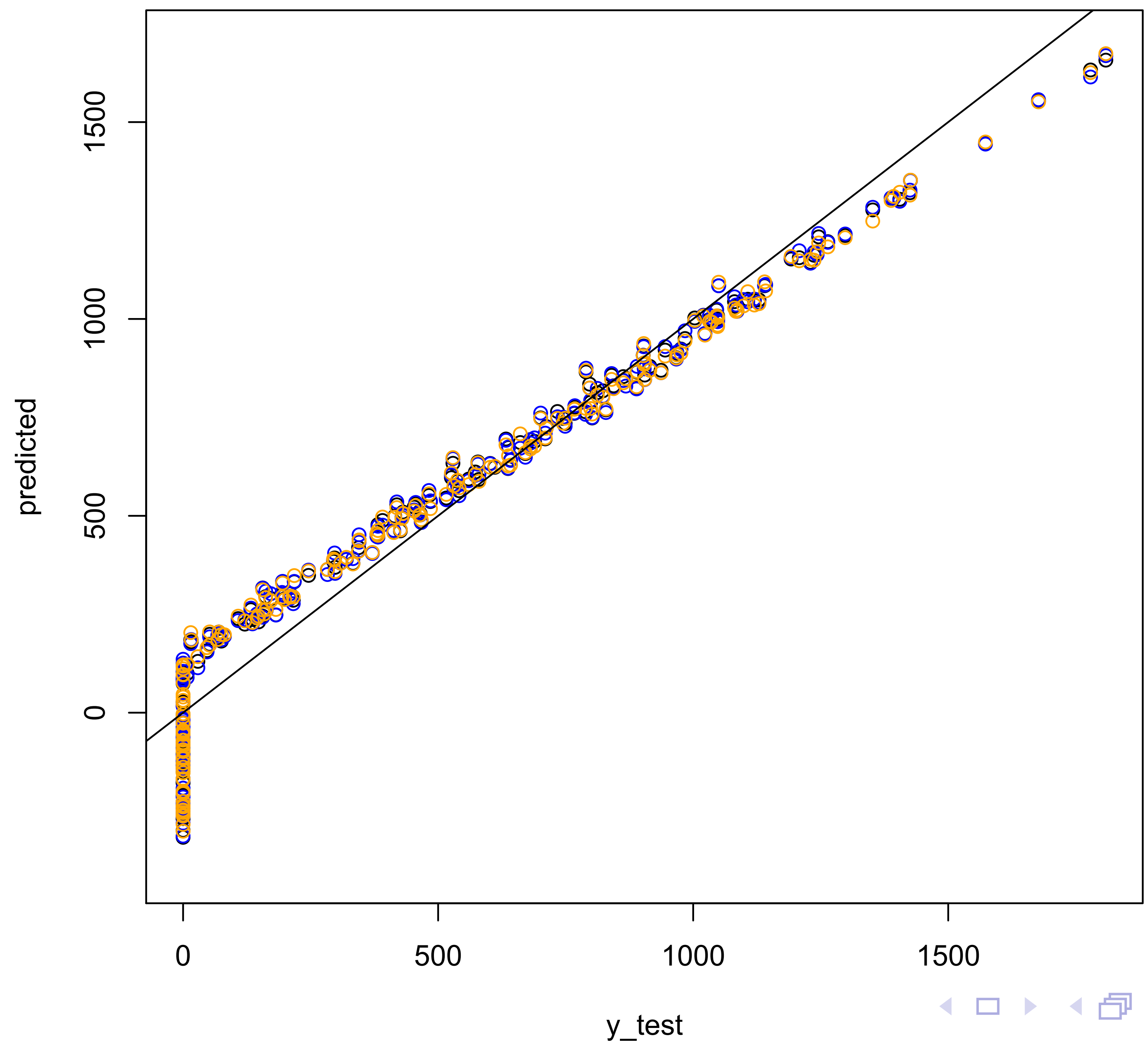
Cross-Validation for Lasso

- We can now compute the test error:

```
> bestlam = cv.out$lambda.min
> lasso.pred = predict(cv.out,s=bestlam,newx=X[test,])
> mean((lasso.pred-y[test])^2)

[1] 10258.06
```

Comparison of Model Predictions

```
> plot(y[test],linear.pred,ylim=c(-400,1700),xlab="y_test",ylab="predicted")
> points(y[test],ridge.pred,col="blue")
> points(y[test],lasso.pred,col="orange")
> abline(0,1)
```



Comments

- For the Credit dataset
  - The ridge regression model has the smallest test error, but only a 2% improvement over the linear model
  - The large number of zero credit card balances probably affects the predictions and might need to be modelled separately
- More generally
  - Penalised methods typically improve over ordinary linear regression by trading off a small increase in bias for a large decrease in variance
  - Ridge regression will tend to perform better when there are a large number of informative features; lasso does better when there are only a few
  - Penalised methods can also work when $p > n$. Feature selection via the lasso is particularly useful in this case
  - When the plot of the cross-validation error is very flat near its minimum, λ_{min} may vary a lot between different choices of the folds. In this case, it might be worth averaging over multiple cross-validation scenarios

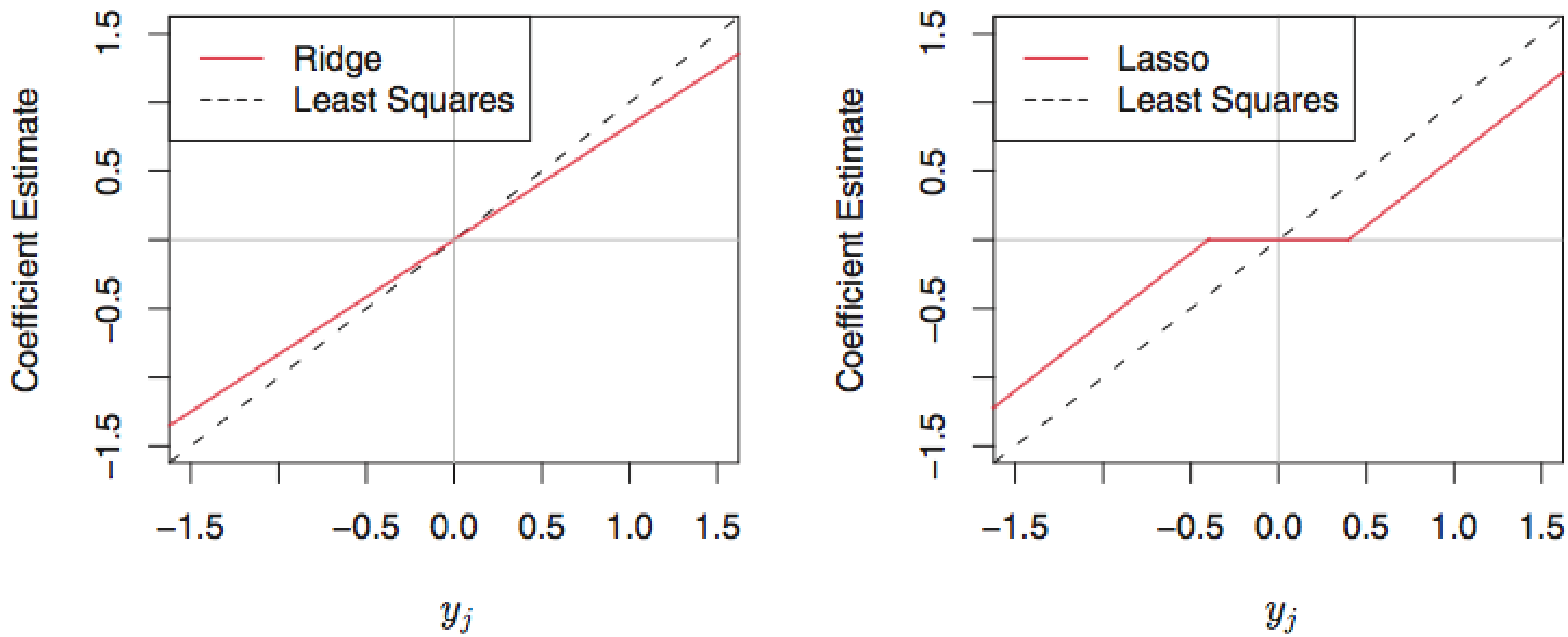
How Shrinkage Methods Work: Intuition

Assume $n = p$, $\beta_0 = 0$ and X is a diagonal matrix with 1's on the diagonal and 0's in all off-diagonal elements:

Least squares is simplified as minimising: $\sum_{j=1}^{p}(y_j - \beta_j)^2$

- Least squares estimation gives $\beta_j = y_j$
- Ridge regression shrinks every estimate by the same proportion: $\widehat{\beta_j^R} = \frac{y_j}{1+\lambda}$
- The lasso essentially shrinks every estimate to zero by the same amount:
  - $\widehat{\beta_j^L} = \begin{cases} y_j - \frac{\lambda}{2}, y_j > \frac{\lambda}{2} \\ y_j + \frac{\lambda}{2}, y_j < -\frac{\lambda}{2} \\ 0, |y_j| \leq \frac{\lambda}{2} \end{cases}$
  - This is known as **soft thresholding**

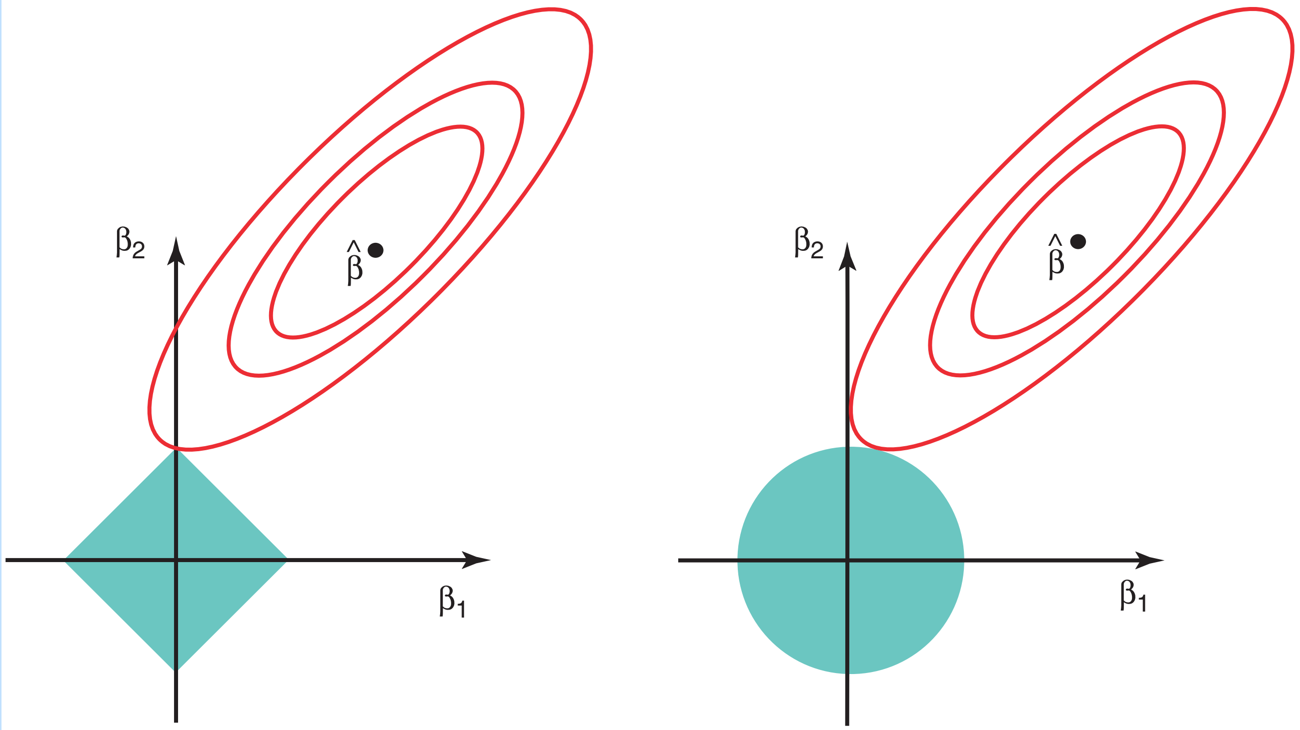
How Shrinkage Methods Work: Intuition



ISLR Figure 6.10: Different types of shrinkage exhibited by ridge regression and the lasso

How Shrinkage Methods Work

- Minimise the RSS subject to a constraint
- Ridge regression, constraint is
  - $\sum_{j=1}^{p} \beta_j^2 \leq s$
  - $\widehat{\beta_j^R}$ is where the RSS contours meet the constraint surface.
- Lasso, constraint is
  - $\sum_{j=1}^{p} |\beta| \leq s$
  - $\widehat{\beta_j^L}$ is typically a "corner" where some of the parameters are 0, which explain the $l_1 - magic$
- Note that large s is equivalent to small λ, and vice versa
- constraint, regularise



ISLR Figure 6.7: Contours of the error and constraint functions for the Lasso (left) and Ridge regression (right)

The solid blue areas are the constraint regions
The red ellipses are the contours of the RSS

Summary

- Linear Regression
- Model selection is guided by minimizing test error; test error is a combination of bias and variance
- Two shrinkage methods: ridge regression and the lasso:
  - also called shrinkage methods since they tend to shrink parameter estimates towards zero
  - sometimes referred to as regularization procedures
  - achieve a bias-variance trade-off via a tuning parameter λ
- The tuning parameter is typically chosen by cross-validation, i.e. λ is chosen to minimize the cross-validation error
- Ridge regression and the lasso retain the interpretability of linear regression
- The lasso can do automatic feature selection by setting some parameter estimates to zero