Statistical decision theory

- Let $X \in \mathbb{R}^p$ and $Y \in \mathbb{R}$ be vector valued random variables where we are trying to predict Y given the information available in X.
  - ➢ We therefore looking for a function $f: \mathbb{R}^p \to \mathbb{R}$, which has "nice" properties

- (**Loss function**) a <u>loss function</u> $L(Y, f(X))$ is a real valued function which measures the error in estimating Y with the estimate $f(X)$.
  - ➢ Typically we assume $L(y, f(x)) \geq 0$ with equality if and only if $y = f(x)$
  - ➢ Since $L(Y, f(X))$ is a random variable, we may also consider its expected value, $E_{X,Y}[L(Y, f(X))]$, often called the <u>risk</u>. Here the expectation is over the joint distribution of X, Y

- (**Quadratic loss**) a commonly used example is the squared error loss, or <u>quadratic loss</u>

$$L_{SE}(Y, f(X)) = (Y - f(X))^2$$

  - ➢ The expected value is called the <u>mean squared error</u> (MSE)

$$MSE(f) := E_{X,Y}\left(L_{SE}(Y, f(X))\right) = E_{X,Y}\left((Y - f(X))^2\right)$$

- (**Conditional expectation**) If X and Y are two random variables with $E(Y) = \mu$ and $Var(Y) < \infty$, then the function, f, which minimizes MSE(f), for X, Y, is given by the conditional expectation

$$f(X) = E_{Y|X}(Y|X)$$

Proof

- Show that the constant c that minimizes $E_Y[(Y - c)^2]$ is $c = \mu$
  - ➢ We need to differentiate and set it to zero to find the minimal value

$$\frac{d}{dc}\left[E_Y[(Y - c)^2]\right] = E_Y[2(Y - c)] = 0$$

$$\hat{c} = E(Y) = \mu$$

    - ✧ The unconditional mean
  - ➢ Now consider minimizing $E_Y\left[(Y - c(X))^2 | X = x\right]$ for any value of x
    - ✧ Here, due to the conditioning, c(x) can be traded as a constant
    - ✧ So the same argument on the previous step gives

$$\hat{c}(x) = E(Y|X = x)$$

    - ✧ The conditional mean
  - ➢ So the best forecast is $f(x) := E(Y|X = x)$

- Recall that an inequality between two random functions $Z := z(x), W := w(x)$ is $Z \geq W \ iff \ z \geq w$ for all realizations of x
  - ➢ Then the argument on the previous results says, for all realization of x, and any forecast g(.)

$$E_Y\left((Y - g(X))^2 | X = x\right) \geq E_Y\left((Y - E(Y|X = x))^2 | X = x\right)$$

$$E_Y\left(\left(Y - g(X)\right)^2 | X\right) \geq E_Y\left(\left(Y - f(x)\right)^2 | X\right)$$

$$E_X\left\{E_Y\left(\left(Y - g(X)\right)^2 | X\right)\right\} \geq E_X\left\{E_Y\left(\left(Y - f(x)\right)^2 | X\right)\right\}$$

$$E_{Y,X}\left[\left(Y - g(X)\right)^2\right] \geq E_{Y.X}\left[\left(Y - f(X)\right)^2\right]$$

$$MSE(g) \geq MSE(f)$$

- When $f(x) = E(Y|X)$

- (**Absolute Loss**) We can also define a loss function in terms of absolute values, i.e.

$$L_{abs}\left(Y, f(X)\right) := |Y - f(X)|$$

  - The function which minimizes $E\left(L_{abs}\left(Y, f(X)\right)\right)$ is

$$\hat{f}(x) = Median(Y|X = x)$$

- (**Zero-one loss**) If Y is a discrete, or categorical random variable, with sample space ***S***, then a commonly used loss function is the indicator, or the zero-one function

$$L_{01}\left(Y, f(X)\right) = \begin{cases} 0, & if\ Y = f(X) \\ 1, & if\ Y \neq f(X) \end{cases}$$

  - And then its expected value is

$$E\left(L_{01}\left(Y, f(X)\right)\right) = P\left(Y \neq f(x)\right)$$

- (Bayes classifier) The function which minimizes $E\left(L_{01}\left(Y, f(X)\right)\right)$ for X, Y where Y has sample space ***S*** is given by

$$\hat{f}(x) = \max_{y \in S} P(y|X = x)$$

  - The so-called Bayes classifier

Linear Regression Models

- (**Example**) The market value of a house is of interest to a both buyers and sellers. It should be a function of a number of features of the house and a multiple learn regression model can be used to estimate this function.
  - In order to calibrate the model, a data set has been constructed using the following variables

X_1	Current taxes
X_2	Number of Bathrooms
X_3	Lot size
X_4	Living space
X_5	Number of parking spaces
X_6	Number of rooms
X_7	Number of bedrooms
X_8	Age of house
X_9	Number of fireplaces
Y	Actual sale price

- As discussed in the previous section, the best forecast of Y given the information in X is given by $E(Y|X)$ in terms of MSE.
  - Computing a conditional expectation requires some knowledge of the joint distribution.
- The linear regression model makes some simplifying assumptions, firstly if X = (X_1, …, X_p) then we assume

$$f(X;\beta) = E(Y|X) = \beta_0 + \sum_{i=1}^{p} \beta_i X_i$$

- In detail, we have by defining the design matrix **X** and response and error vector **y**, $\boldsymbol{\epsilon}$

$$X = \begin{bmatrix} 1 & x_{11} & \cdots & x_{1p} \\ 1 & x_{21} & \cdots & x_{2p} \\ \cdots & \cdots & \cdots & \cdots \\ 1 & x_{n1} & \ldots & x_{np} \end{bmatrix}$$

  - And we write $\boldsymbol{y} = \boldsymbol{X}\beta + \boldsymbol{\epsilon}$

- (**Residual sum of squares**) To estimate β given a training set $\{(y_i, \boldsymbol{x_i})|i = 1, \ldots, N\}$ where $\boldsymbol{x_i} = (x_{i1}, \ldots, x_{ip})$. It is common to minimize the residual sum of squares

$$RSS(\beta) \coloneqq \sum_{i=1}^{N}\left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_j\right)^2 = (\boldsymbol{y} - \boldsymbol{X}\beta)^T(\boldsymbol{y} - \boldsymbol{X}\beta)$$

- The ordinary least squares estimate $\hat{\beta}$ is defined as $\arg\min_{\beta}\left(RSS(\beta)\right)$

- (**OLS estimation**) The ordinary least squares estimate is given by

$$\hat{\beta}_{OLS} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}$$

- Where $(\boldsymbol{X}^T\boldsymbol{X})^{-1}$ exists

- In R we can fit the full model using OLS and get the following output

```
              Estimate Std. Error t value Pr(>|t|)
(Intercept) 15.31044    5.96093   2.568   0.0223 *
X1           1.95413    1.03833   1.882   0.0808 .
X2           6.84552    4.33529   1.579   0.1367
X3           0.13761    0.49436   0.278   0.7848
X4           2.78143    4.39482   0.633   0.5370
X5           2.05076    1.38457   1.481   0.1607
X6          -0.55590    2.39791  -0.232   0.8200
X7          -1.24516    3.42293  -0.364   0.7215
X8          -0.03800    0.06726  -0.565   0.5810
X9           1.70446    1.95317   0.873   0.3976
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1
```

- We see that most of the term individually are "not significant" but the whole model has an $R^2 = 0.8512$, and an overall p-value of 0.0002015 so appears to explain a good deal of the variation of Y and hence might be a useful predictive model

- (**Multicollinearity**) If two or more of the columns of **X** are highly correlated we say that we have multicollinearity
- In terms of prediction we have that the correlated explanatory variables are essentially are sharing the same information about Y.
  - Having highly correlated explanatory variables does not add much to predictive power but is statistically expensive since there are more parameters to be estimated.
  - We will see later that the bias is not reduced by much but the variance is increased.

- We can see some correlations through the pairwise scatterplot of the 9 explanatory variables.

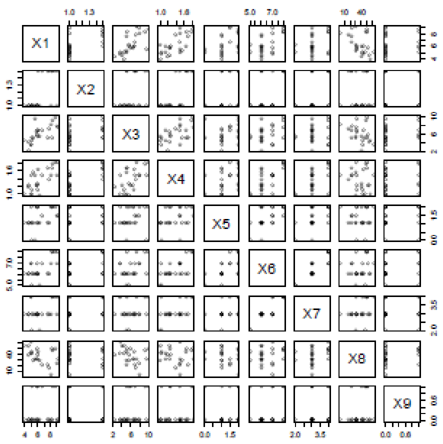


Figure 2.1: Pairwise scatterplot for covariate in house example

-

- (**prediction interval in regression**) if we assume that $(Y_0, \boldsymbol{X_0})$ is independent of the training set $T = \{(Y_1, \boldsymbol{X_1}), \dots, (Y_N, \boldsymbol{X_N})\}$ and assume we have a correctly specified model
- Define the point forecast as

$$\hat{\mu} = \boldsymbol{x_0}^T \hat{\beta}$$

- The $\alpha\%$ - prediction interval is

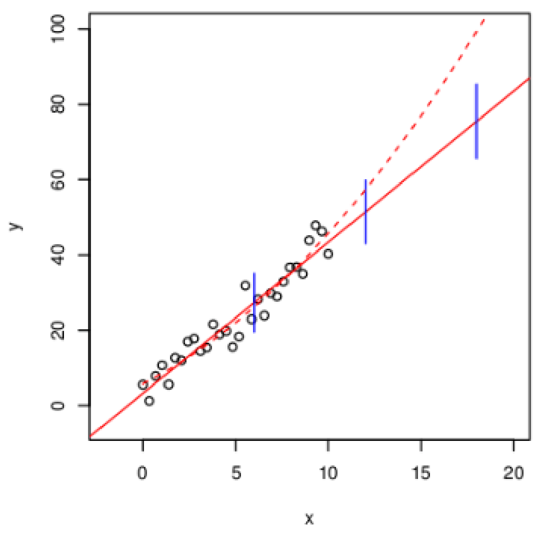
$$\hat{\mu} \pm c_\alpha \hat{\sigma} \sqrt{1 + \boldsymbol{x_0}^T (\boldsymbol{X}^T \boldsymbol{X})^{-1} \boldsymbol{x_0}}$$

  - Where c_α is the $1 - \frac{\alpha}{2}$ quantile from the t_{N-p-1} – distribution

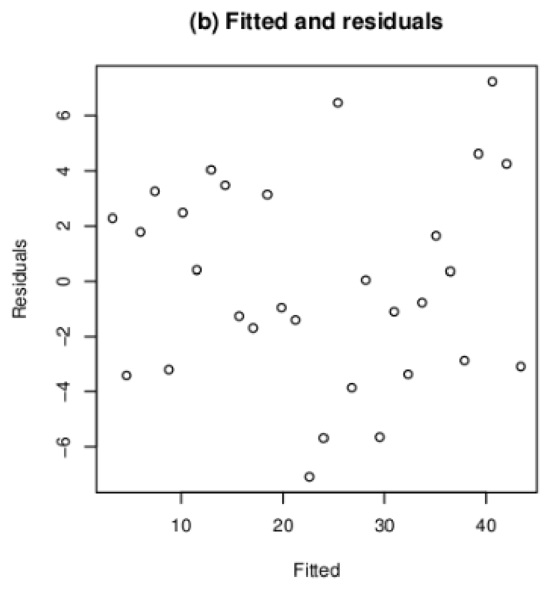
- (**Prediction in regression**) The following simple simulated example shows both how forecasting in prediction works and illustrates a major weakness of this, and other forecasting methods.

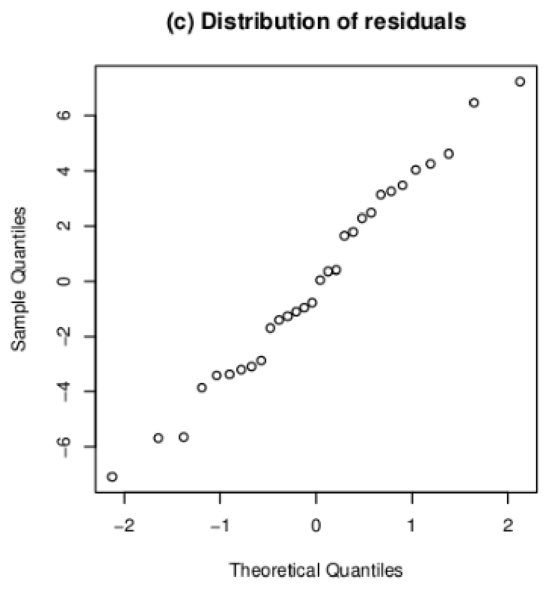
- Consider the following panel, it shows the data (black dots) a fitted linear model (solid red line) and prediction intervals (solid blue) for "new" values of the covariate of 6, 12 and 18

(a) Simple prediction



- The following panels show information about the residuals and it looks like the linear model fits very well.





- In fact, the true model here was non-linear (dashed red line) and the prediction interval for x = 12 and 16 would be very poor

- The lesson here is that making a prediction where the new covariate values are far from the values used to fit the data is a very risky thing to do.
- The goodness-of-fit statistics only tells us that the model fits the training data and very little about how the model behaves when far from the training data

Computing in R

In R the key function for regression is the `lm( )`. For example in Example 2.3.1 we fit the model using

```
>  house.fit <- lm(Y ~X1+X2+X3+X4+X5+X6+X7+X8+X9, data=house.price)
```

the `ls`-object then contains all the output needed for analysis. Functions for which it is an argument includes the following:

```
> summary(house.fit)

> plot(house.fit)

> house.fit$fitted

> house.fit$residuals
```

The last two of these are very helpful for looking at good-of-fit residual analysis plots. When we want to use the `lm( )` function for prediction we can use the `predict( )` or `predict.lm( )` function. For example, suppose we look a the simple model which regresses house price Y against $X1$ and $X2$. We fit the model using

```
 house.fit1 <- lm(Y ~ X1+X2, data=house.price)


> new <- data.frame(X1=c(6.00), X2= c(1.5) )
> predict.lm(house.fit1, new, interval="prediction")
       fit      lwr      upr
1 35.56195 29.05033 42.07357
```

Here `fit` is our point estimate of the price and `lwr` and `upr` define a 95%-prediction interval thought the lower and upper values respectively.

- **Model Complexity**

- Adding covariates which are strong correlated with ones already included typically reduces the model's prediction power
- There are even stronger reasons why over-complex models should be avoided. This is called over-fitting.
  - ➢ The fitted model is a very good description of the training data, but not such a good description of the underlying population

- (**Bias-Variance decomposition**): let T be the training set used to estimate β, we now want to estimate the forecasting error for a new set of covariates, x_0. Let the forecast be

$$\hat{y}_o := x_0^T \hat{\beta}$$

- If y_0 is the actual value we are trying to forecast, and using quadratic loss, we evaluate the forecast using

$$MSE(x_0) := E_T(\{y_0 - \hat{y}_0\}^2)$$
$$= E_T(\{y_0 - E_T(\hat{y}_0) + E_T(\hat{y}_0) - \hat{y}_0\}^2)$$
$$= \left(y_0 - E(\hat{y}_0)\right)^2 + E_T\left(\left(\hat{y}_0 - E_T(\hat{y}_0)\right)^2\right)$$
$$= biase^2(\hat{y}_0) + Var_T(\hat{y}_0)$$

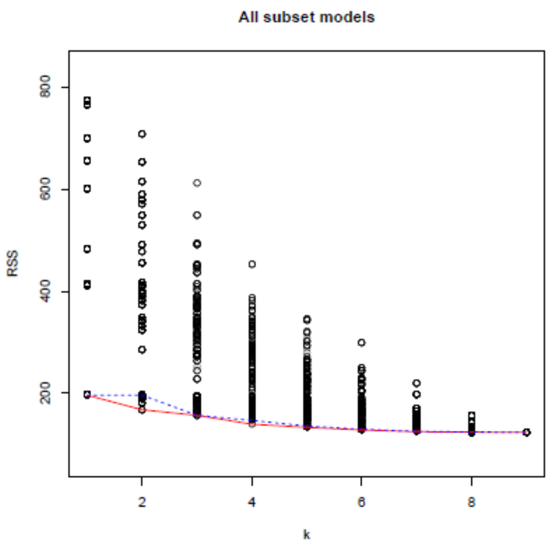
- When evaluating the MSE of a prediction, we typically trying to balance two different objectives
  - ➢ The bias of the forecast and
  - ➢ The variance
- E.g. the OLS estimate of β is unbiased for a correctly specified model.
  - ➢ However, Hastie criticizes the OLS method for models with many covariance for prediction since although the predictions have small bias they can have large variance
- Prediction accuracy can be improved by shrinking or setting some of the coefficients of $\hat{\beta}$ to zero.
  - ➢ This increases bias, but reduces variability

- One approach to improving the predictive performance is to use only a subset of the explanatory variables. This may reduce the multi-collinearity and reduce the variance of the estimator, at the cost of some bias.
  - ➢ The question is how to select the subset?

- (**Best subset regression**) best subset regression finds for each $k \in \{0, 1, \ldots, p\}$, the subset which has the smallest residual sum of squares.

- (**housing example**) here we have a maximum of 9 explanatory variates and the intercept term. The following figure shows the RSS for all possible subsets of $X_1, \ldots, X_9$ – the intercept was always included.

- The red curved is the best subset curve and defines the models which are eligible for selection in the best subset method. Since this curve must be a decreasing one – adding any variable always gives a "better" fit, hence reduce RSS – it cannot be used for finding the "optimal" value for k



- How to select k involves the tradeoff between bias and variance. Typically the smallest model is selected which minimizes an estimate of expected prediction error

- Looking at all possible subsets and fitting a model to each choice can be computationally very expensive. In our example, there are $2^9 = 512$ possible models so it is just about possible to do in a reasonable time.
- In general we use non-exhaustive search methods which are computational much faster but may not find the "best" model however commonly find ones close to the best

- Most method we look at will produce a set of possible models indexed by a single parameter – in this case k. The job is then find the "right" value of this parameter

- **Forward and backward step selection**

- (**forward step selection**) This is a greedy algorithm which starts with the intercepts then adds the covariate which most decreases the RSS at each stage.
- Rather than explore the whole of the space of possible models it looks for a "good" yet easy to compute path through them
- (**Housing example**) in the above figure, the blue dashed line shows the path, in terms of RSS

and k, of the forward selection algorithm.

  - This is close to the best subset curve for most values of k, but there is no guarantee that this is always going to happen

- (**Backward step selection**) Starts with the full model and deletes the covariate with the smallest z-statistic

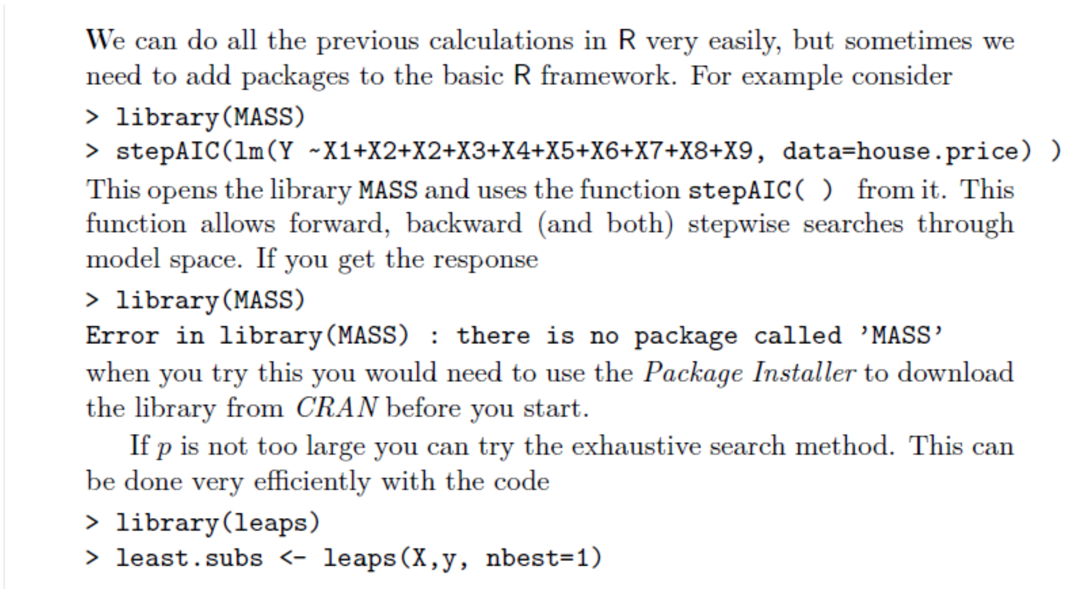
- (**Akaike information criterion**) The Akaike information criterion (AIC) tries to balance goodness of fit of a model and the models complexity.
- It measures the complexity of the model by counting the number of parameters used.
- It is defined as

$$-2l(\hat{\theta}) + 2N_p$$

  - Where l is log-likelihood, $\hat{\theta}$ is the MLE and N_p is the number of parameters in model
- In general, a smaller value of AIC is preferred, as you add parameters – making the model more complex – the first term gets smaller but the penalty term, the second term, counts against adding parameters which don't improve the fit enough

- (The *step()* function) in R we have this function whose default is to use both a forward and backward one step search at each stage – add any one variable not currently included and delete any one variable that has been included.
- The decision to add or drop is not done by looking at the RSS but using the AIC criterion which takes into account the complexity of the model

- **Computing in R**



We can do all the previous calculations in R very easily, but sometimes we need to add packages to the basic R framework. For example consider

```
> library(MASS)
> stepAIC(lm(Y ~X1+X2+X2+X3+X4+X5+X6+X7+X8+X9, data=house.price) )
```

This opens the library `MASS` and uses the function `stepAIC( )` from it. This function allows forward, backward (and both) stepwise searches through model space. If you get the response

```
> library(MASS)
Error in library(MASS) : there is no package called 'MASS'
```

when you try this you would need to use the *Package Installer* to download the library from *CRAN* before you start.

If p is not too large you can try the exhaustive search method. This can be done very efficiently with the code

```
> library(leaps)
> least.subs <- leaps(X,y, nbest=1)
```

- **Ridge Regression**

- Subset selection methods take a very binary approach, the explanatory variable is either included or excluded. We can look at more continuous ways of adapting the model – always using a tuning parameter akin to k – of changing the explanatory information

- (**ridge regression**) The ridge regression estimate of a linear model is defined as

$$\hat{\beta}^{ridge} := \arg\overset{\min}{\beta} RSS(\lambda)$$

  - Where

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

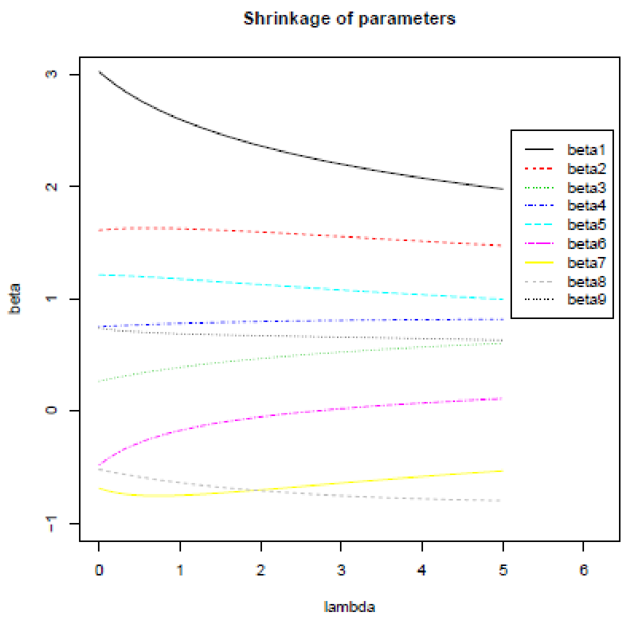
  - Where $\lambda > 0$ is a tuning parameter which determines the bias-variance trade-off

- This problem can be written in a equivalent form as

$$\hat{\beta}^{ridge} := \arg\overset{\min}{\beta} \sum_{i=1}^{N} \left(y_i - \beta_0 + \sum_{j=1}^{p} x_{ij}\beta_j \right)^2 \quad s.t. \sum_{j=1}^{p} \beta_j^2 \leq t$$

  - Where there is a one-to-one correspondence between λ and t.

- (**housing example**) We can see the effect of choosing different values of λ on the parameter values in the house example in the following figure



- As λ gets bigger, all parameters shrink towards zero, but at different rates.
  - We see for example the coefficient β_6 of the “Number of rooms” goes quickly to zero and this is the estimate that seemed to have the “wrong” sign in the standard OLS estimates

- **The Lasso**

- (**Lasso**) the lasso regression estimate is defined by

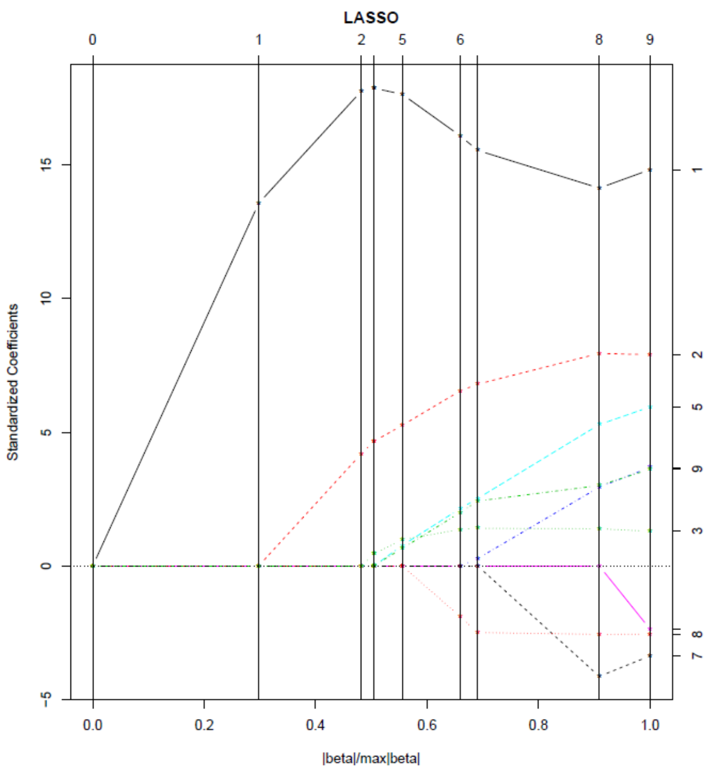
$$\hat{\beta}^{lasso} := \arg\min_{\beta} \sum_{i=1}^{N} \left(y_i - \beta_0 + \sum_{j=1}^{p} x_{ij}\beta_j \right)^2 \quad s.t. \sum_{j=1}^{p} |\beta_j| \le t$$

- Which can be compared to ridge regression where we see the constraint is on the absolute value of the parameter values not the squares.
- This can then also be expressed, using Lagrange multipliers as

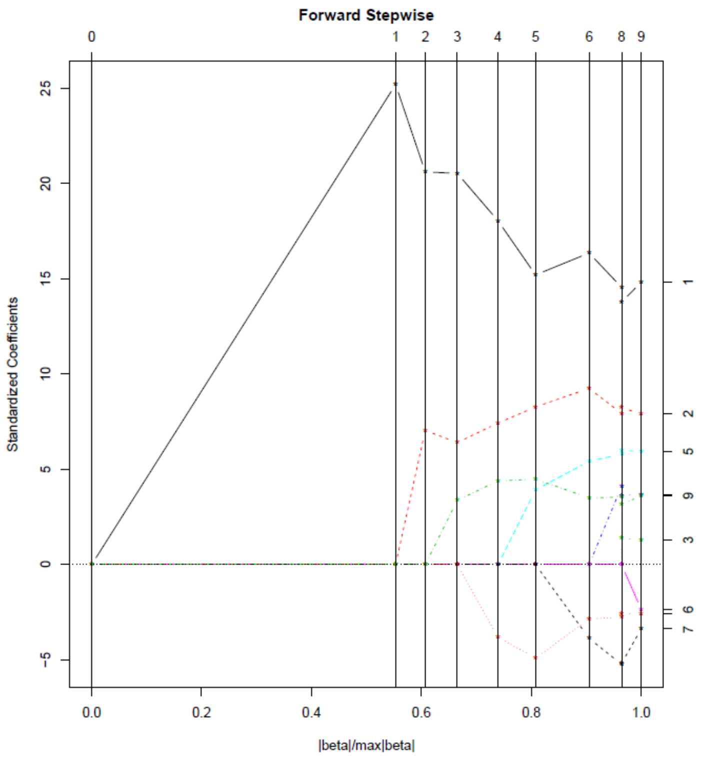
$$\hat{\beta}^{ridge} := \arg\min_{\beta} \left\{ \sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2 + \lambda \sum_{j=1}^{p} |\beta_j| \right\}$$

- The lasso not only shrinks small estimates to zero but also can make them exactly zero.
  - ➢ Thus it is able to perform subset selection as well as shrinkage

- (**housing example**) We can see the shrinkage effects of the lasso in the following figure.



- This shows the lasso as λ moves from 0 (on the right of the plot) to infinity (on the left)
- We see the values of the parameter estimates as they shrink and then go to zero.
  - ➢ Following the red dashed line, which corresponding to β_2 is the second to last to be set to zero (i.e. when k=1)
  - ➢ The path for β_6 is the first to be shrunk completely to zero.



- Here we have the forward stepwise selection, where we can see the covariates being added in the order of 1, 2, 9, 8, 5, 7, 4, 3, 6

- **R** – lasso and ridge regression

```
###########################
#Ridge regression
> library(MASS)
> fit.ridge <-  lm.ridge(Y ~X1+X2+X2+X3+X4+
   X5+X6+X7+X8+X9, lambda=1,
   data=house.price, y=T)

###################
#Lasso
> library(lars)
> house.lasso <- lars(X1, y, type="lasso")
```

where $X1$ is the matrix of centred covariates.

Model assessment and selection

- The previous section describes how we can look at the model complexity of regression models in terms of one dimensional tuning parameter: k the number of covariates in subset selection, λ a positive real number in ridge/lasso regression.
- We have not yet discussed how to select the "best" value of such a tuning parameter for the prediction problem
  - ➢ Emphasis that finding the model which best fits the training data is not going to be the best model for forecasting because of the bias-variance trade-off.
  - ➢ Very complex models can fit training data very well but have associated very high variance
  - ➢ Often a complex model will not be able to generalize away from the observed sample – we say it has over fitted, describing detailed aspects of the sample that are there just by "chance"
- Let's define how we can **evaluate a models predictive performance** in a way that has taking to account all aspects of the model fitting process

- (**Generalization error**) Let T be the training data for regression the observed set $\{(y_1, \boldsymbol{x_1}), \dots, (y_N, \boldsymbol{x_N})\}$ where N is the number of cases. Let $\hat{f}(\boldsymbol{X_0})$ be the prediction model fitted to the training data and evaluated at value of the covariate selected from the population, $\mathbf{X_0}$.
- We define the generalization error to be

$$Err_T := E\left[L\left(Y, \hat{f}(\boldsymbol{X_0})\right)|T\right]$$

  - ➢ Where L() is any loss function, but in this section will typically be squared loss. i.e.

$$Err_T := E\left[L\left(Y, \hat{f}(\boldsymbol{X_0})\right)|T\right] = E\left[\left(Y - \hat{f}(\boldsymbol{X_0})\right)^2 |T\right]$$

- We can also define the expected generalization error to be Err_T averaged over training sets

$$Err := E_T[Err_T]$$

- When we are evaluating how well a model predicts we want to compute – or at least estimate – either Err_T or, in practice, Err

- (**Training error**) The training error in general is defined as

$$\overline{err} := \frac{1}{N}\sum_{i=1}^{N} L\left(y_i, \hat{f}(\boldsymbol{x_i})\right)$$

  - ➢ Which is the residual sum of squares (RSS) typically in this section
  - ➢ In general, training error is not a good estimate of prediction error and typically underestimates it

- (**Ideal procedure**) Hastie et al. (2009) state that if we had lots of data, and a set of possible models, what we would ideally do is split the data into three different parts: training, validation and test.

- (1) use the training to fit each of our set of models i.e. estimate their parameters
- (2) use the validation data to estimate the prediction error for each of our candidate fitted models in order to pick the best. Note here we are not reusing data to both fit and evaluation
- (3) finally, use the test data to estimate the prediction error for our selected single best model

- They proposed a rough rule that 50% of data be training, 25% validation and 25% being testing.
- This is however often not possible since data is scarce, so what should we do?

- (**Expected prediction error regression**) we have a regression model of the form $Y = (X) + \epsilon$, where $Var(\epsilon) \coloneqq \sigma_\epsilon^2$ and the fitted value is $\hat{f}(x_0)$ for some input $\boldsymbol{X}_0 = x_0$. Then the expected prediction error is

$$Err(x_0) = E\left[\left(Y - \hat{f}(x_0)\right)^2 | X_0 = x_0\right]$$

$$= \sigma_\epsilon^2 + \left\{E\left[\hat{f}(x_0)\right] - f(x_0)\right\}^2 + E\left[\left(\hat{f}(x_0) - E\left[\hat{f}(x_0)\right]\right)^2\right]$$

$$= \sigma_\epsilon^2 + Bias^2 + Variance$$

  - The first term is called the irreducible error associated with the "true" model, f, and will always be there, the second and third terms depend on the model class we use.

- (**Simulation Example**) In order to understand the generalization error for models selected by best subset regression, a simulation experiment was done.
  - The true model is given by

$$Y = 1.0X_1 + 0.775X_2 + 0.550X_3 + 0.325X_4 + 0.1X_5 + \epsilon$$

    - Where $\epsilon \sim N(0, 0.3^2)$.
  - The set of possible covariate X_i is p = 20 and we generate each independently from $Unif(0,1)$ distribution.
  - A training sample, T with N = 80 was generated and the best subset regression was fitted for each subset size k = 1, …, 20

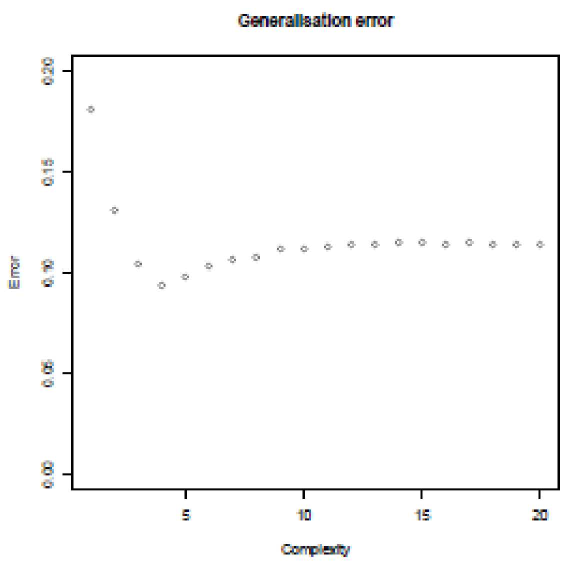
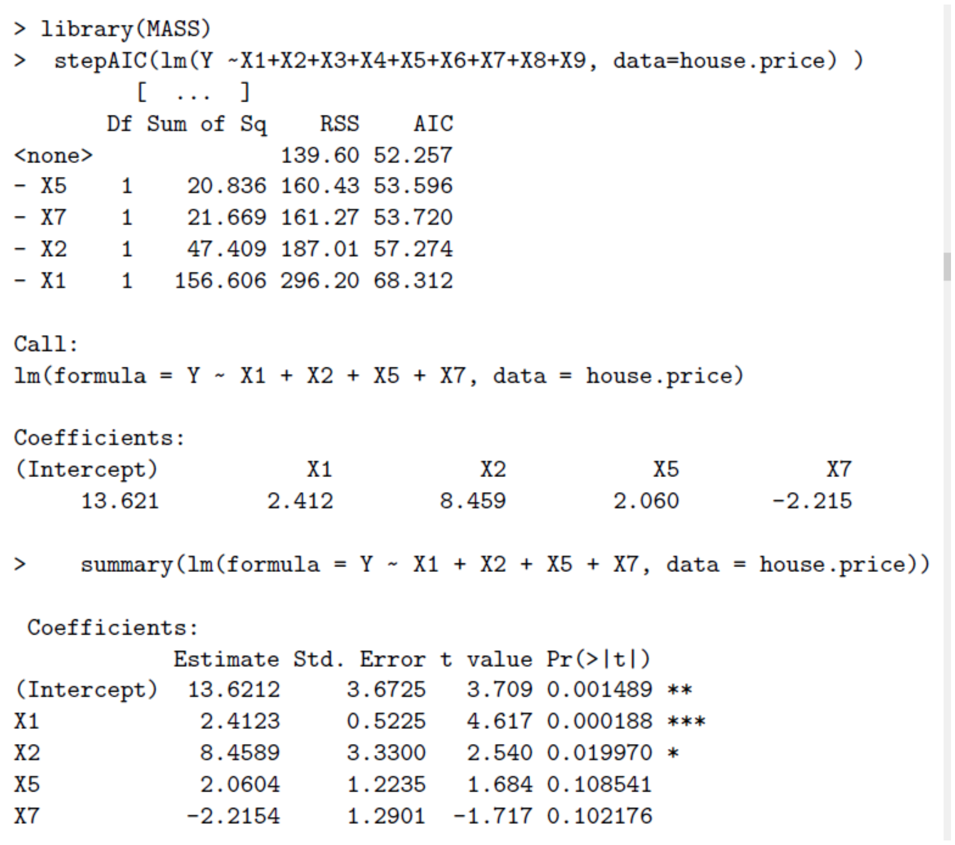


Figure 2.7: Generalisation error

- From the training set we have 20 possible models to choose from.
  - We see how well each does in terms of its err_T prediction error by randomly selecting sets of explanatory variables (each independent Unif(0,1)) generating the observed value of Y from model and then seeing how well each of the 20 candidate fitted models did at predicting Y in terms of squared error.
- With fixed T this was repeated many times and the average error (err_T) is plotted above
  - We see that when the model is too simple – has less than 5 covariates in the true model – it does very badly. It is strongly biased
  - But having is not a good thing since the prediction variance starts to increase

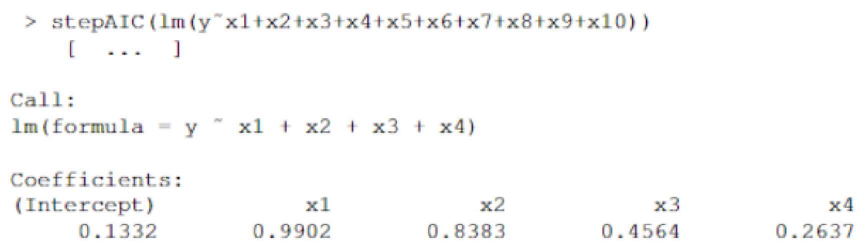
- However, the simulation can't be used in practice with a real data set since it requires knowing the true model.
- If we try to use the training error, we find that, in words of Hastie et al. (2009), it is too optimistic, it underestimates the true error

- (**Housing example**) We can use the stepAIC function in the MASS package in R to do a forward and backwards search which stops when the smallest AIC value is found



```
> library(MASS)
>  stepAIC(lm(Y ~X1+X2+X3+X4+X5+X6+X7+X8+X9, data=house.price) )
          [  ...  ]
        Df Sum of Sq    RSS    AIC
<none>              139.60 52.257
- X5     1    20.836 160.43 53.596
- X7     1    21.669 161.27 53.720
- X2     1    47.409 187.01 57.274
- X1     1   156.606 296.20 68.312

Call:
lm(formula = Y ~ X1 + X2 + X5 + X7, data = house.price)

Coefficients:
(Intercept)           X1           X2           X5           X7
     13.621        2.412        8.459        2.060       -2.215

>    summary(lm(formula = Y ~ X1 + X2 + X5 + X7, data = house.price))

 Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  13.6212     3.6725   3.709 0.001489 **
X1            2.4123     0.5225   4.617 0.000188 ***
X2            8.4589     3.3300   2.540 0.019970 *
X5            2.0604     1.2235   1.684 0.108541
X7           -2.2154     1.2901  -1.717 0.102176
```

- If we apply the function to our simulation example, we have the following result



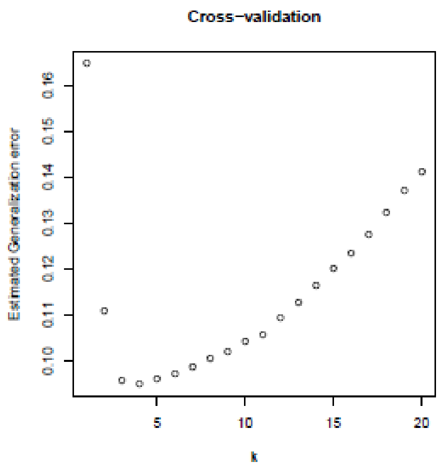
```
 > stepAIC(lm(y~x1+x2+x3+x4+x5+x6+x7+x8+x9+x10))
    [  ...  ]

Call:
lm(formula = y ~ x1 + x2 + x3 + x4)

Coefficients:
(Intercept)           x1           x2           x3           x4
     0.1332       0.9902       0.8383       0.4564       0.2637
```

- It indeed picks out the p = 4 model that proved to have the best prediction error, but of course unlike that figure it has not used any knowledge of the true model to do it

- For most examples the "ideal procedure" requires too much data but it can be thought of as a way of motivating the most popular method for estimating prediction error, this is called cross validation.
- (**K-fold cross-validation**) The idea is to split the N subjects in the training set into K (roughly) equal-sized parts, say $\{T_k\}_{k=1}^{K}$.
  - For k = 1, …, K take out the kth-part of this partition, T_k, and fit the model to the other K-1 parts of the data.
  - Calculate the prediction error of the fitted model for T_k. The prediction error is then

estimate by averaging over the K times this can be done

- (**Simulation example**) best subset regression gives a set of models, indexed by k, but does not tell you which k to use. The cross validation method can be used here to estimate k.



- In the figure, one case was left out at a time – i.e. K = N cross-validation – the 20 possible models were fitted to the remaining data and the error each makes in predicting the remaining data was recorded. We average this over all N cases.
- We see very similar results to our previous graph but the cross validation only uses the data which is available to the analyst. It does not use any knowledge of the “true model”

- One difference, though, between the two graphs is the type of generalization error being estimated.
  - ➢ In general cross-validation gives us an estimate of the expected generalization error err and not the generalization error, err_T, itself.
  - ➢ Since in a given prediction problem we have one specific training set, it would be more appropriate to have a good idea about err_T, but in general this is not an easy thing to estimate