ECON7333: Assignment 3

Thomas Doyle

Setup

```r
options(scipen=999)
attach(ISLR::Wage)
```

Exercise 1

Exercise 1.1

The function `loocv_tmse` computes the test mean square error of a regression model using a leave-one-out cross-validation approach.

```r
loocv_tmse <- function(d){
  #' @d a data.frame returned by `model.frame()`
  #
  n <- dim(d)[1]
  p <- dim(d)[2]

  MSE <- rep(0,n)

  for(i in 1:n) {
    #
    lm_i <- lm(d[-i,],y=TRUE) # leave i out
    MSE_i <- (lm_i$y[i]-predict(lm_i,d[i,]))^2
    MSE[i] <- MSE_i
  }

  return(
    #
    CV_n <- mean(MSE,na.rm = TRUE)
  )
}
```

Exercise 1.2

We use `loocv_tmse` to choose beteen the three models;

1. $logwage = \beta_0 + \beta_1 age$
2. $logwage = \beta_0 + \beta_1\ age + \beta_2\ age^2$
3. $logwage = \beta_0 + \beta_1\ age + \beta_2\ education$

Each model takes `logwage` as the dependent variable.

```r
Wage.models <- list(
  model.frame("logwage~age",ISLR::Wage),
  model.frame("logwage~age+I(age^2)",ISLR::Wage),
  model.frame("logwage~age+education",ISLR::Wage)
)
sapply(Wage.models, loocv_tmse)
```

```
## [1] 0.1306018 0.1381353 0.1531854
```

Choose the model with lowest test MSE. Choose linear regression on $logwage = \beta_0 + \beta_1 age$, which returns the smallest test MSE.

Exercise 1.3

The λ parameter is now used as tuning parameter on the ridge regression model;

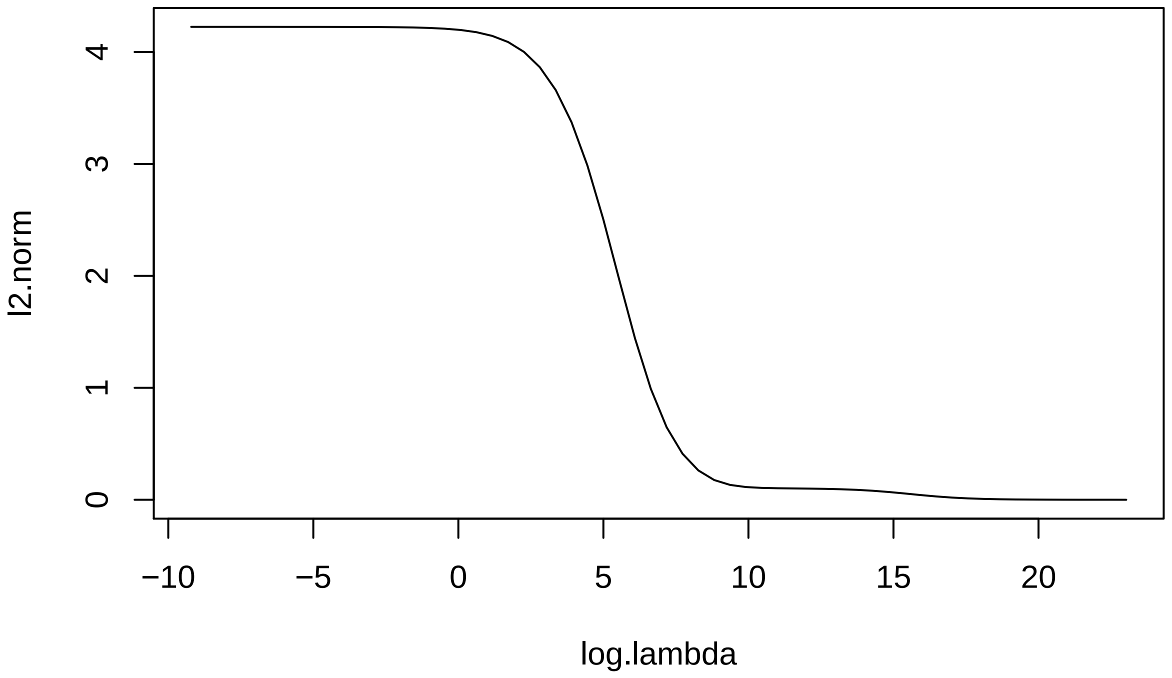
$$logwage = \beta_0 + \beta_1\ age + \beta_2\ education$$

```r
lm.ridge <- function(lambda=0) {

  m <- model.frame(logwage~age+education,ISLR::Wage)
  Terms <- attr(eval.parent(m),"terms")
  Y <- model.response(m)
  X <- model.matrix(object = Terms,data = m,contrasts.arg = list(education="contr.treatment"))
  n <- nrow(X)
  p <- ncol(X)

  # https://arxiv.org/pdf/1509.09169.pdf
  # Hoerl, A. E. and Kennard, R. W. (1970).
  ridge.betas <- solve( t(X) %*% X + lambda*diag(p) ) %*% t(X) %*% Y


  l2.norm <- sqrt(sum(ridge.betas^2)) # drop intercept

  return(
    list(
      log.lambda=log(lambda),
      l2.norm=l2.norm,
      coefficients=ridge.betas,
      lambda=lambda
    )
  )
}
```

```r
# MLE
# showing off
lm.ridge.loocv <- function(limits) {
  #' @limits
  #

  m <- model.frame(logwage~age+education,ISLR::Wage)
  Terms <- attr(eval.parent(m),"terms")
  Y <- model.response(m)
  X <- model.matrix(object = Terms,data = m,contrasts.arg = list(education="contr.treatment"))
  loocv <- function(lambda, X, Y, Delta){
    n <- nrow(X)
    p <- ncol(X)
    loss <- 0

    for (i in 1:n) {
      loo_beta <- solve(t(X[-i,])%*%X[-i,]+lambda*diag(p))%*%t(X[-i,])%*%Y[-i]
      loss <- loss+(Y[i]-X[i,1]*loo_beta[1]-X[i,-1]%*%loo_beta[-1])^2
    }

    return(loss)
  }

  # optimize penalty parameter
  # minimize RSS
  opt <- optimize(loocv, limits, X=X, Y=Y)
  l2.norm <- lm.ridge(opt$minimum)$l2.norm

  return(
    data.frame(opt,l2.norm)
  )
}
```

Plot $\ln(\lambda)$ against ℓ_2 norm.

```
lambdas <- 100^seq(-2, 5, length = 60)
plot(t(sapply(lambdas,lm.ridge))[,1:2],type="l")
```



```
# mle
# lm.ridge.loocv(c(10^-10, 10^10))
```

Exercise 2

Exercise 2.1

The function `dgf` generates data following the data generating process set out by question 2 of ECON 7333 Assignment 3 text.

```r
dgf <- function(n,p) {
  #' @n `scalar`
  #' @p `scalar`

  e <- rnorm(n,0,1)
  X <- matrix(runif(n*p),n,p)
  y <- 2*X[,1]+4*X[,2]+e

  output <- list(y=y,X=X)
  return(list(y,X))
}

p.5 <- dgf(100,5)
y <- p.5[[1]]
X <- p.5[[2]]
```

Exercise 2.2

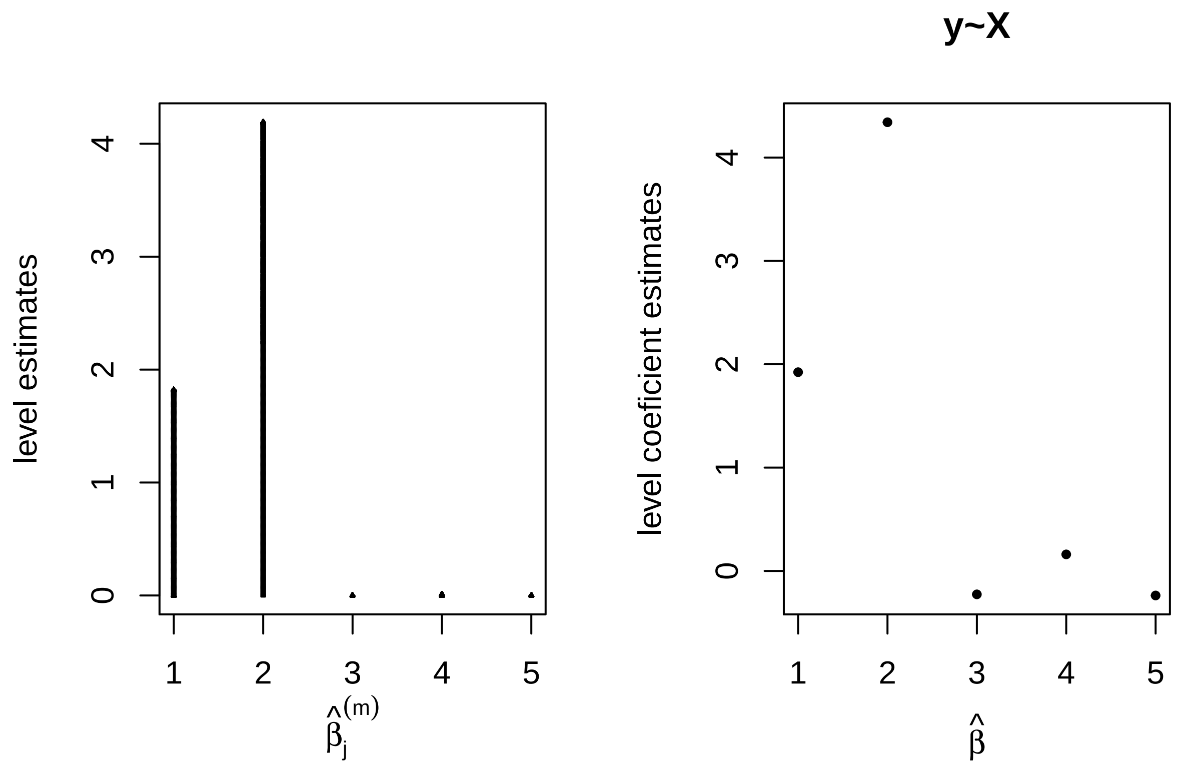
Apply the model to the sample generated by `dgf`. `model_path` implements the subset selection algorithm set by the assignment text..

```r
model_path <- function(y,X,M) {
  #' @y `vector`, response variable
  #' @X `matrix`, input variable
  #' @M `scalar`, model path

  p <- dim(X)[2]
  n <- dim(X)[1]

  # Initialise
  #
  b <- matrix(rep(NA,p*M),p,M)
  b[,1] <- 0
  r <- as.vector(y)
  # e <- as.vector(rep(0.01,n))
  e <- 0.01

  for(m in 2:M) {
    z <- b[,m-1]

    ## select x_j with highest correlation with r
    j <- which.max(cor(x=X,y=r))
    xj <- X[,j]

    a <- suppressWarnings(e*sign(t(xj)%*%r))
    z[j] <- z[j]+a
    b[,m] <- z
    r <- as.vector(r-a%*%xj)
  }

  return(b)
}
```

Exercise 2.3

Generate a sample path of length $M = 1000$ of the coeficient estimates $\hat{\beta}_j^{(m)}$, and plot the sample using `matplot`.

```
path <- model_path(y,X,1000)

par(mfrow=c(1,2))
matplot(path,pch=2, cex=.2, col="black",
        ylab="level estimates",
        xlab=expression(hat(beta)[j]^(m)),
        main="")

plot(lm(y~X-1)$coefficients, pch=20, cex=.8,
     ylab="level coeficient estimates",
     xlab=expression(hat(beta)),
     main="y~X")
```



Exercise 2.4

Divide the sample into training and test subsamples of size 50 each. Use cross-validation to nd the best model along the paths (that is, find the best model among the M dfferent models generated along the path). Choose between that best model and linear regression conducted by least squares.

```r
mse_i <- function(y,X,m,i) {
  mean(sum((y[i] - m%*%t(X[i,]))^2))
}

subset_selection <- function(y,X,M,k) {
  #' @y
  #' @X
  #' @M
  #' @k

  # calculate MSE by iterate over M
  MSE_i <- apply(M,2,function(m) {
    mse_i(y,X,m,k)
  })

  # Return coef for model with min MSE
  return(
    list(
      coefficients = M[,which.min(MSE_i)],
      best_subset_m = which.min(MSE_i),
      best_mse = min(MSE_i)
    )
  )
}

k_fold <- function(n,k) {

  ix <- sample(1:n,n)

  fold_i <- split(ix,cut(1:n,k,FALSE))

  folds <- data.frame(fold_i)

  colnames(folds) <- paste("fold",1:k,sep = "_")

  return(folds)
}
```

The function `k_fold` returns an $n \times k$ vector used to create k subsets of sample size n/k from the matrix of independent variables, X.

Use function `subset_selection` to select the best model from the training sample and compute the trainse MSE using `mse_1`.

Based on the minimum average MSE of k folds, choose the best model of coefficients. Compare the best subset model against the linear regression model.

```r
folds <- k_fold(100,2)

# obtain test MSE using k-fold
```

```r
#

fold_1_tmse = mse_i(y,X,subset_selection(y,X,path,folds[,1])[["coefficients"]],folds[,2])
fold_2_tmse = mse_i(y,X,subset_selection(y,X,path,folds[,2])[["coefficients"]],folds[,1])

SS_MSE <- subset_selection(y,X,path,folds[,which.min(c(fold_1_tmse,fold_2_tmse))])$best_mse
LM_MSE <- sum(lm(y~X-1)$residuals^2)

data.frame(SS_MSE,LM_MSE)
```

```
##     SS_MSE   LM_MSE
## 1 64.79127 117.5728
```

Choose best subset selection model, which returns the lowest CV k-fold MSE when compared to linear regression.

Exercise 2.5

```r
CV_k <- function(n,p) {

  # call dgf to generate new n x p data
  d <- dgf(n,p)

  y <- d[[1]]
  X <- d[[2]]

  # generate models
  path <- model_path(y,X,1000)

  folds <- k_fold(n,2)

  # obtain test MSE using k-fold
  tmse_1 <- mse_i(y,X,subset_selection(y,X,path,folds[,1])[["coefficients"]],folds[,2])
  tmse_2 <- mse_i(y,X,subset_selection(y,X,path,folds[,2])[["coefficients"]],folds[,1])

  SS_MSE <- subset_selection(y,X,path,folds[,which.min(c(fold_1_tmse,fold_2_tmse))])$best_mse
  LM_MSE <- sum(lm(y~X-1)$residuals^2)
  return(data.frame(SS_MSE,LM_MSE))
}

CV_k(100,5)
```

```
##     SS_MSE   LM_MSE
## 1 42.98299 96.15328
```

```r
CV_k(100,10)
```

```
##     SS_MSE   LM_MSE
## 1 54.17434 97.33566
```

```r
CV_k(100,20)
```

```
##     SS_MSE LM_MSE
## 1 50.23396 92.566
```

```r
CV_k(100,50)
```

```
##     SS_MSE   LM_MSE
## 1 47.56949 60.68574
```

As p increases, the linear regression MSE decresses while the MSE reported by best subset selection is relatively stable. At each level of p, the best subset selection MSE is lower than the linear regression MSE.