Linear Algebra Assignment

Michael J. Jones

May 11, 2015

1 Row and Column Picture of a linear model

Solve the following linear system

$$2x + y = 7 \tag{1}$$

$$2x + 3y = 1 \tag{2}$$

Answer

Given we can express this problem in the form $Ax = b$:

```
> A <- matrix(nrow = 2, data = c(2, 2, 1, 3))
> colnames(A) <- c('x', 'y')
> rownames(A) <- c('', '')
> A

 x y
 2 1
 2 3

> b <- matrix(nrow = 2, ncol = 1, data = c(7, 1))
> colnames(b) <- c('')
> rownames(b) <- c('', '')
> b

 7
 1
```

Knowing that $x = A^{-1} \cdot b$, then

```
> x = solve(A) %*% b
> x

x  5
y -3
```

Draw the row picture of the linear system

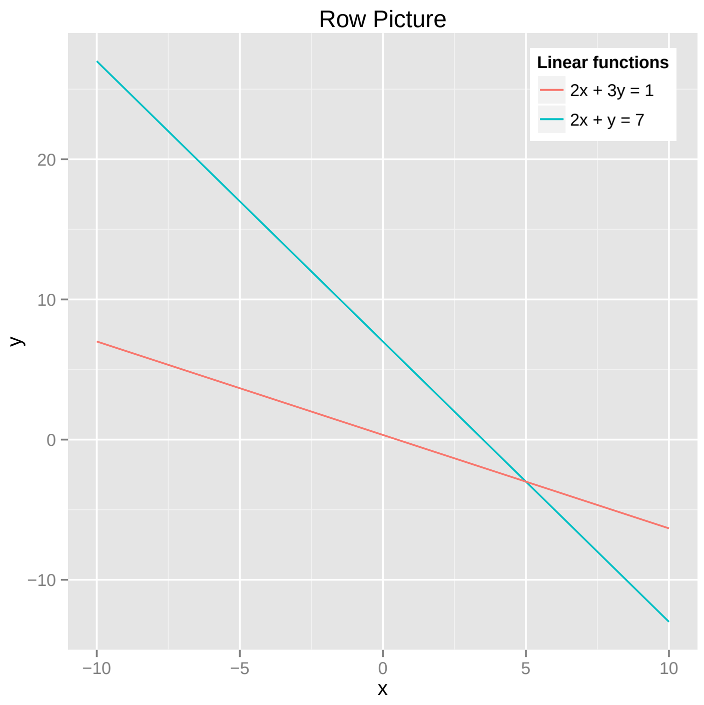
If we express y as a function of x in each of the above equations

$$f(x) = 7 - 2x \tag{3}$$

$$f(x) = \frac{1 - 2x}{3} \tag{4}$$

Then, using the *ggplot2* library, we can plot both of these functions in R.

```
> library(ggplot2)
> library(grid)
> # TODO
> f.1 <- function(x) 7 - 2*x
> f.2 <- function(x) (1 - 2*x) / 3
> ggplot(data.frame(x=c(-10, 10)), aes(x)) +
+   stat_function(fun=f.1, geom="line", aes(colour="2x + y = 7")) +
+   stat_function(fun=f.2, geom="line", aes(colour="2x + 3y = 1")) +
+   labs(title="Row Picture", colour="Linear functions") +
+   theme(legend.position=c(0.85,0.9))
```



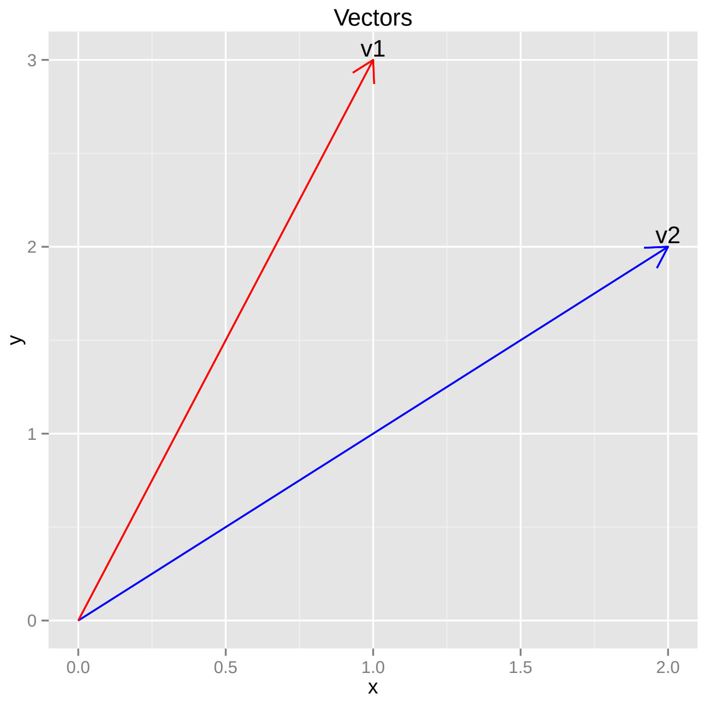
Draw the column picture of the linear system

Answer

In order to draw our column picture, it helps us to first express our problem as the summation of a set of vectors:

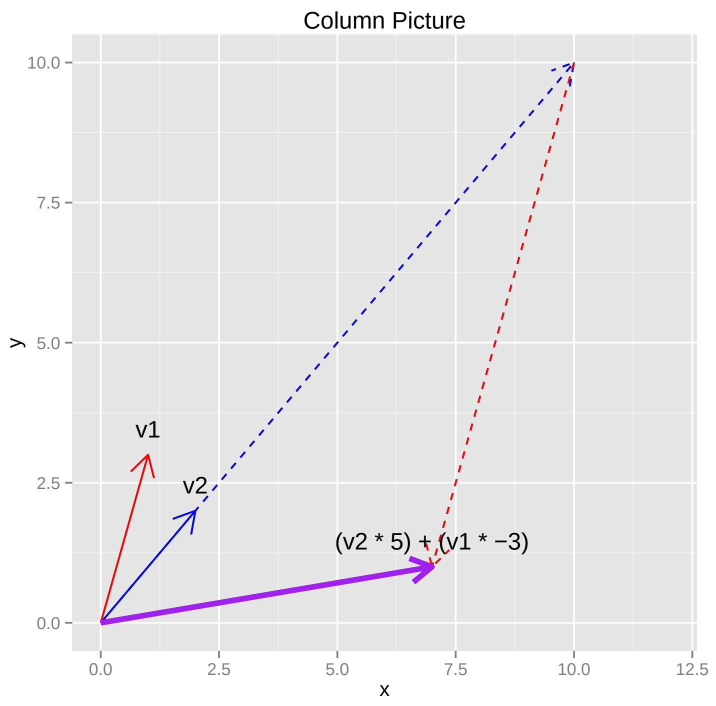
$$\begin{bmatrix}2\\2\end{bmatrix}x+\begin{bmatrix}1\\3\end{bmatrix}y=\begin{bmatrix}7\\1\end{bmatrix}x \tag{5}$$

```
> ggplot(data.frame(x=c(-5, 10)), aes(x)) +
+   geom_segment(colour="blue", aes(x = 0, y = 0, xend = 2, yend = 2), arrow = arrow(length
+   geom_segment(colour="red", aes(x = 0, y = 0, xend = 1, yend = 3), arrow = arrow(length =
+   labs(title="Vectors") +
+   xlab("x") +
+   ylab("y") +
+   geom_text(vjust=-.2, aes(label=c("v1", "v2")), x = c(1, 2), y = c(3,2))
```



Some combination of x of the first vector and y of the second vector will yield the third vector. Fortunately, we have already discovered when we solved this linear system that $x = 5$ and $y = -3$. This means that five of the first and negative three of the second vectors would add up to give us $(7, 1)'$.

```
> ggplot(data.frame(x=c(-5, 10)), aes(x)) +
+   scale_x_continuous(limits = c(0, 12)) +
+   geom_segment(colour="blue", aes(x = 0, y = 0, xend = 2, yend = 2),
+                arrow = arrow(length = unit(0.5, "cm"))) +
+   geom_segment(colour="red", aes(x = 0, y = 0, xend = 1, yend = 3),
+                arrow = arrow(length = unit(0.5, "cm"))) +
+   geom_segment(linetype="dashed", colour="blue",
+                aes(x = 0, y = 0, xend = 2*5, yend = 2*5),
+                arrow = arrow(length = unit(0.5, "cm"))) +
+   geom_segment(linetype="dashed", colour="red",
+                aes(x = 2*5, y = 2*5, xend = 2*5 - 3*1, yend = 2*5 - 3 * 3),
+                arrow = arrow(length = unit(0.5, "cm"))) +
+   geom_segment(size=1.5, colour="purple", aes(x = 0, y = 0, xend = 7, yend = 1),
+                arrow = arrow(length = unit(0.5, "cm"))) +
+   labs(title="Column Picture") +
+   xlab("x") +
+   ylab("y") +
+   geom_text(vjust=-1,
+             aes(label=c("v1", "v2", "(v2 * 5) + (v1 * -3)","")),
+             x = c(1, 2, 7,0),
+             y = c(3, 2, 1,0))
>
```



The column picture plainly shows that the combining v1 and v2 in the proportions of 5 and -3 results in observing the vector, $(7, 1)'$.

2 Data

All data is available at link `http://go.qub.ac.uk/toolkit/regularization`

3 Linear regression model

3.1 Question

Download and read the prostate cancer dataset prostate.data into a data matrix. The data set is taken from the free online book *The Elements of Statistical Learning* from Trevor Hastie, Robert Tibshirani and Jerome Friedman.
`http://statweb.stanford.edu/~tibs/ElemStatLearn/datasets/prostate.data`

```
Prostate data info
Predictors (columns 1--8)
lcavol
lweight
age
lbph
svi
lcp
gleason
pgg45

outcome (column 9)

lpsa

train/test indicator (column 10)
```

Answer

```
> require(xtable) # install the xtable package if you don't have it
> prostate.data.url <-
+   "http://statweb.stanford.edu/~tibs/ElemStatLearn/datasets/prostate.data"
> prostate.data <- read.table(prostate.data.url, head=T)
> xtable(head(prostate.data,3), )
```

	lcavol	lweight	age	lbph	svi	lcp	gleason	pgg45	lpsa	train
1	-0.58	2.77	50	-1.39	0	-1.39	6	0	-0.43	TRUE
2	-0.99	3.32	58	-1.39	0	-1.39	6	0	-0.16	TRUE
3	-0.51	2.69	74	-1.39	0	-1.39	7	20	-0.16	TRUE

3.2 Question

Estimate and define a linear model using the *lm*() function.

Answer

```
> prostate.lm <- lm(formula = lpsa ~ ., data = prostate.data, method = "qr")
```

```
> summary(prostate.lm)

Call:
lm(formula = lpsa ~ ., data = prostate.data, method = "qr")

Residuals:
     Min       1Q   Median       3Q      Max 
-1.76795 -0.35653 -0.00437  0.37978  1.55913 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept)  0.177306   1.338810   0.132  0.89495    
lcavol       0.564417   0.088387   6.386 8.08e-09 ***
lweight      0.622204   0.202179   3.077  0.00279 ** 
age         -0.021306   0.011383  -1.872  0.06460 .  
lbph         0.096833   0.058441   1.657  0.10113    
svi          0.761466   0.242697   3.138  0.00233 ** 
lcp         -0.105872   0.090661  -1.168  0.24609    
gleason      0.049967   0.158955   0.314  0.75401    
pgg45        0.004434   0.004485   0.989  0.32558    
trainTRUE    0.004104   0.162772   0.025  0.97994    
---
Signif. codes:  0 âĂŸ***âĂŹ 0.001 âĂŸ**âĂŹ 0.01 âĂŸ*âĂŹ 0.05 âĂŸ.âĂŹ 0.1 âĂŸ âĂŹ 1

Residual standard error: 0.7035 on 87 degrees of freedom
Multiple R-squared:  0.6634,	Adjusted R-squared:  0.6286 
F-statistic: 19.05 on 9 and 87 DF,  p-value: < 2.2e-16
```

3.3 Question

Split the prostate dataset into a test and training dataset (see column 10)

Answer

```
> split.data <- split(prostate.data, prostate.data$train)
> prostate.training <- split.data$T
> prostate.training <- subset(prostate.training, select=-train)
> xtable(head(prostate.training,3), )
```

	lcavol	lweight	age	lbph	svi	lcp	gleason	pgg45	lpsa
1	-0.58	2.77	50	-1.39	0	-1.39	6	0	-0.43
2	-0.99	3.32	58	-1.39	0	-1.39	6	0	-0.16
3	-0.51	2.69	74	-1.39	0	-1.39	7	20	-0.16

```
> prostate.test <- split.data$F
> prostate.test <- subset(prostate.test, select=-train)
> xtable(head(prostate.test,3), )
```

	lcavol	lweight	age	lbph	svi	lcp	gleason	pgg45	lpsa
7	0.74	3.47	64	0.62	0	-1.39	6	0	0.77
9	-0.78	3.54	47	-1.39	0	-1.39	6	0	1.05
10	0.22	3.24	63	-1.39	0	-1.39	6	0	1.05

3.4 Question

Predict *lpsa* for the test example

```
data<-read.table(file="...")
# split data
test.data<- ...
train.data<- ...
```

Answer

First, we will define the linear model again using only the training data.

```
> prostate.lm <- lm(formula = lpsa ~ ., data = prostate.training, method = "qr")
```

We now need to use `predict` to predict the the lpsa level using our linear model on our test data. Note, we need to exclude the `lpsa` column from our test data in our prediction.

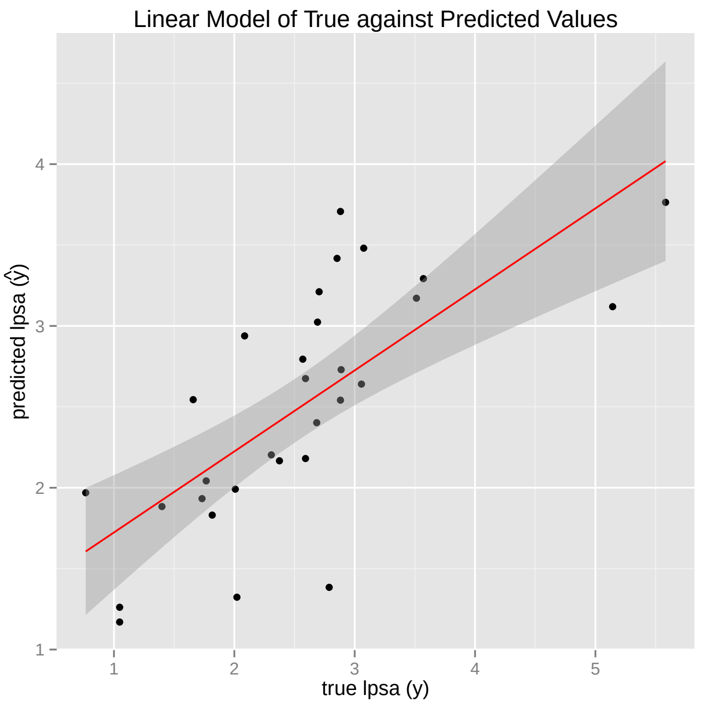
```
> prostate.test.no.label <- subset(prostate.test, select=-lpsa)
> lm.predicted.lpsa <- predict(prostate.lm, prostate.test.no.label)
```

3.5 Question

Plot your results (true *lpsa* value y_i and predicted $\hat{y}_i$).

Answer

```
> lm.true.vs.predicted <-
+   data.frame(true=prostate.test$lpsa,predicted=lm.predicted.lpsa)
> ggplot(data = lm.true.vs.predicted, aes(true, predicted)) +
+   geom_point() +
+   stat_smooth(method = "lm", col = "red") +
+   ylab(expression(paste("predicted lpsa (", hat(y),')'))) +
+   xlab("true lpsa (y)") +
+   labs(title="Linear Model of True against Predicted Values")
```



3.6 Question

Repeat the analysis using a random forest regression and plot your results (true *lpsa* value y_i and predicted $\hat{y}_i$).

Answer

We have already partitioned our data into both training and testing data above. Just as we did with linear regression, we need to perform our regression analysis using Random Forests with the training data.

```
> library(randomForest)
> num.predictors <- 8
> prostate.rf <- randomForest(lpsa ~ ., data=prostate.training,
+                               mtry=num.predictors / 3, importance=TRUE,
+                               na.action=na.omit)
> summary(prostate.rf)
```

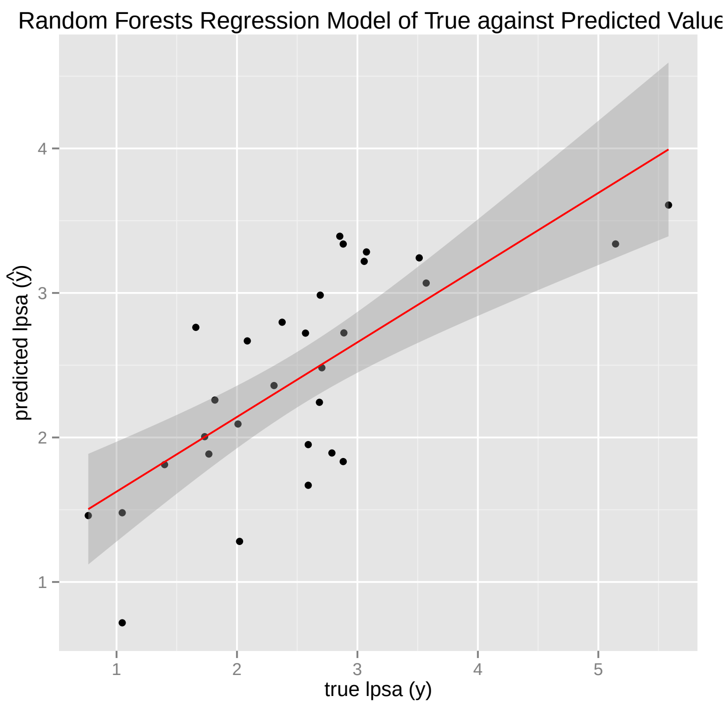
```
                Length Class  Mode
call              6    -none- call
type              1    -none- character
predicted        67    -none- numeric
mse             500    -none- numeric
rsq             500    -none- numeric
oob.times        67    -none- numeric
importance       16    -none- numeric
importanceSD      8    -none- numeric
localImportance   0    -none- NULL
proximity         0    -none- NULL
ntree             1    -none- numeric
mtry              1    -none- numeric
forest           11    -none- list
coefs             0    -none- NULL
y                67    -none- numeric
test              0    -none- NULL
inbag             0    -none- NULL
terms             3    terms  call
```

And as performed above, we can predict the values of our test data for *lpsa*

```
> set.seed(131)
> rf.predicted.lpsa <- predict(prostate.rf, prostate.test.no.label)
```

and plot these results accordingly.

```
> rf.true.vs.predicted <-
+   data.frame(true=prostate.test$lpsa,predicted=rf.predicted.lpsa)
> ggplot(data = rf.true.vs.predicted, aes(true, predicted)) +
+   geom_point() +
+   stat_smooth(method = "lm", col = "red") +
+   ylab(expression(paste("predicted lpsa (", hat(y),')'))) +
+   xlab("true lpsa (y)") +
+   labs(title="Random Forests Regression Model of True against Predicted Values")
```



4 Implement your own lm function

Write a R function that estimates $\hat{\beta}$ coefficients for the linear model:

$$y = \beta_0 + \beta_{pred1}x_1 + \beta_{pred2}x_2 + \beta_{pred3}x_3 \tag{6}$$

Remember to center scale the predictor variables and estimate β_0 separately. You can use the function $scale()$

The coefficients of a linear model can be estimated by

$$\hat{\beta} = (X^TX)^{-1}X^Ty \tag{7}$$

Answer

```
> my.lm <- function(resp, cova){
+   errors <- c()
+   if (!is.matrix(cova)) errors <- c(errors,  "\ncova must be a matrix")
+   if (!is.numeric(cova)) errors <- c(errors, "\ncova must be numeric")
+   if (!is.vector(resp)) errors <- c(errors,  "\nresp must be a vector")
+   if (!is.numeric(resp)) errors <- c(errors, "\nresp must be numeric")
+   if (length(errors) > 0) {
+     stop(errors)
+   }
+
+   X <- scale(cova,TRUE,TRUE)
+   X <- cbind(1,X)
+
+   cp <- t(X) %*% X
+   cp.inv <- solve(cp)
+   beta <- cp.inv %*% (t(X) %*% resp)
+
+   if (!is.null(colnames(cova))) {
+     rows <- colnames(cova)
+   } else {
+     num.coefs <- ncol(cova)
+     rows <- paste("beta", 1:num.coefs, sep=".")
+   }
+   rownames(beta) <- c("Intercept", rows)
+   colnames(beta) <- "coefs"
+
+   return(beta)
+ }
```

An example

```
> covariates <- matrix(data=c(2,1,5,4,2,3), nrow=3)
> colnames(covariates) <- c("x.1", "x.2")
> y <- c(16, 8, 19)
> my.lm.coefficients <- my.lm(y, covariates)
> xtable(my.lm.coefficients)
```

	coefs
Intercept	14.33
x.1	4.16
x.2	3.00

5 Ridge Regression

5.1 Question

Write a function to estimate the coefficients using a ridge regression model

$$\hat{\beta} = (X^T X + \lambda I)^{-1} X^T y \tag{8}$$

where I is the identity matrix and λ the penalty parameter.

Answer

```
> my.lm.ridge <- function(resp, cova, lambda){
+   errors <- c()
+   if (!is.matrix(cova)) errors <- c(errors,  "\ncova must be a matrix")
+   if (!is.numeric(cova)) errors <- c(errors, "\ncova must be numeric")
+   if (!is.vector(resp)) errors <- c(errors,  "\nresp must be a vector")
+   if (!is.numeric(resp)) errors <- c(errors, "\nresp must be numeric")
+   if (!is.numeric(lambda)) errors <- c(errors, "\nlambda must be numeric")
+
+   if (length(errors) > 0) {
+     stop(errors)
+   }
+
+   X <- scale(cova,TRUE,TRUE)
+   X <- cbind(1,X)
+
+   cp <- t(X) %*% X
+   I <- diag(nrow=nrow(cp), ncol=ncol(cp))
+
+   beta <- solve(cp + I*lambda) %*% (t(X) %*% resp)
+
+   if (!is.null(colnames(cova))) {
+     rows <- colnames(cova)
+   } else {
+     num.coefs <- ncol(cova)
+     rows <- paste("beta", 1:num.coefs, sep=".")
+   }
+   rownames(beta) <- c("Intercept", rows)
+   colnames(beta) <- "coefs"
+
+   return(beta)
+ }
```

5.2 Question

Implement a function that returns the optimal λ by a 10 fold cross validation. The function minimizes the prediction error measure (sum of squared error)

$$MSE = \sum (y - \hat{y})^2 \tag{9}$$

```
rss.error <- sum((test$lpsa-pred)^2)
```

Answer

```
> find.lambda <- function(resp, cova, lambda.start = 0.1,
+                         step.size = 0.1, lambda.stop = 1) {
+   max.lambda.count <- 100
+
+   lambda.vals <- seq(from = lambda.start, to = lambda.stop, by = step.size)
+   if (length(lambda.vals) > max.lambda.count) {
+     stop(paste("Too many possible lambda values. Make lambda.start",
+                "and lambda.stop closer or make step size larger."))
+   }
+   rss.errors <- vector(mode = "numeric", length = length(lambda.vals))
+
+   for (i in 1:length(lambda.vals)) {
+     curr.lambda <- lambda.vals[i]
+     rss.errors[i] <- calc.rss(resp, cova, curr.lambda)
+   }
+
+   index.of.least.rss <- which.min(rss.errors)
+   optimal.lambda <- lambda.vals[index.of.least.rss]
+   coefficients <- my.lm.ridge(resp, cova, optimal.lambda)
+
+   result <- list(
+     optimal.lambda = optimal.lambda,
+     model.coefficients = coefficients
+   )
+   class(result) <- "find.lambda"
+
+   result
+ }
> print.find.lambda <- function(find.lambda) {
+   cat("Optimal lambda:",
+       find.lambda$optimal.lambda,
+       "",
+       "Coefficients",
+       find.lambda$model.coefficients,
+       sep="\n")
+ }
```

```
> #  calculate rss for a given lambda using 10-fold cross validation
> calc.rss <- function(resp, cova, lambda, K=10) {
+   N <- length(resp)
+   random.indices <- sample(x = 1:N, size=N, replace=F)
+
+   #list of vectors with indices of each fold
+   folds <- split(random.indices, ceiling(1:N/K))
+
+   rss.errors <- sapply(X = folds, FUN=function(test.indices) {
+     training.indices <- random.indices[which(!(random.indices %in% test.indices))]
+
+     training.cova <- cova[training.indices,]
+     training.resp <- resp[training.indices]
+
+     test.cova <- cova[test.indices,]
+     test.actual.resp <- resp[test.indices]
+
+     X <- scale(test.cova)
+     X <- cbind(1, X)
+
+     beta <- my.lm.ridge(training.resp, training.cova, lambda)
+     test.predicted.resp <- (X %*% beta)
+
+     # now calculate rss.error
+     sum((test.actual.resp - test.predicted.resp)^2)
+   })
+   mean(rss.errors)
+ }
```

5.3 Question

Apply your function to the prostate dataset and report the model coefficients and optimal λ.

Answer

```
> set.seed(131)
> total.prostate.data <- subset(prostate.data, select=-train)
> total.prostate.resp <- as.vector(subset(total.prostate.data, select=lpsa)[,1])
> total.prostate.cova <- as.matrix(subset(total.prostate.data, select=-lpsa))
> find.lambda(total.prostate.resp, total.prostate.cova,
+             lambda.start = 0.1, step.size = 0.02, lambda.stop = 2)
```

```
Optimal lambda:
0.4
```

```
Coefficients
2.468209
0.6591861
0.2661398
-0.155881
0.1392158
0.3130646
-0.1412707
0.03647961
0.1231814
```

6 Running the code in this document

This document was compiled using the *Sweave* extension to R. Sweave is an implementation of the *Literate Programming* paradigm conceived of by Donald Knuth. This allows an author to weave *R* code and text written in LaTeX together into one document.

To run the code from this document, follow these instructions:

1. Open the zipped directory containing this pdf (Alternatively, download the whole project from http://github.com/hiraethus/linear-algebra-assignment)
2. Open RStudio and load this project by choosing `File > Open Project` and navigating to the file `linear-algebra-assignment.Rproj`.
3. From the file browser in RStudio, choose `linear-algebra-assignment.Rnw`
4. Use the `chunks` menu located on the text editor opened in R to run each of the chunks in the file
5. Run all the code chunks in RStudio by typing `Ctrl + Alt + R`.