MAST09104: Introduction to Statistical Learning

Week 4 Workshop and Lab

Workshop questions

Note that some of these involve R.

1. David asked a question about the green plane picture in Figure 1 in Module 4: namely, whether the data vector y could be orthogonal to the plane of the column space of X.

   To answer this, let $H = X(X^TX)^{-1}X^T$ be the hat matrix and suppose $P = \left[\ P_1 \mid P_2 \mid \cdots \mid P_n\ \right]$ has columns which are the eigenvectors of H with the first being those which are non-zero. Let $\mathbf{x} = P^T\mathbf{y}$.

   (a) Give expressions for $\mathbf{y}$ and $H\mathbf{y}$ in terms of the eigenvectors, $P_1, \cdots, P_n$ and $\mathbf{x}$, assuming that the number of predictors is $k+1$ and X is full rank $k+1$.

   (b) Hence show that the data vector y is orthogonal to the least-squares fitted values $\hat{\mathbf{y}} = X\mathbf{b}$ if, and only if, the fitted values are 0.

   (c) Suppose
   $$X = \begin{bmatrix} 1 & 2 \\ 1 & -1 \\ 1 & -1 \end{bmatrix}.$$
   Find all data vectors $\mathbf{y}$ for which the fitted values are 0.

2. Michael asked some questions about leverage which motivated this question.

   (a) Suppose that H is an idempotent and symmetric matrix. Show that the diagonal entries of H are in $[0, 1]$ and that their sum is the rank of H.

   (b) Interpret part (a) in terms of leverage.

   (c) The formula for the variance of the ith residual is $\text{var}\,(\varepsilon) = \sigma^2(1 - H_{ii})$. If observation i has a leverage close to 1, how does this show that the response variable, y, for a row of the X matrix with large leverage *can* have a significant effect on the model parameters? When does such an observation have a significant effect?

3. Consider the dataset from the Week 3 lab. As before, do the following using both matrix calculations and R's `lm` commands.

   (a) Calculate 95% confidence intervals for the model parameters.

   (b) Calculate a 95% confidence interval for the average income of a person who has had 18 years of formal education.

   (c) Calculate a 95% prediction interval for the income of a single person who has had 18 years of formal education.

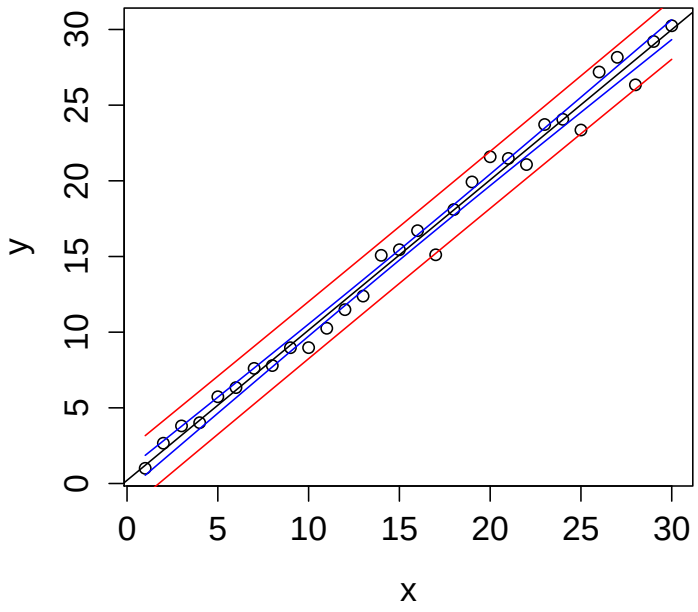
4. For simple linear regression, $y = \beta_0 + \beta_1 x + \varepsilon$, show that a $100(1-\alpha)\%$ confidence interval for the mean response when $x = x^*$ can be written as
   $$b_0 + b_1 x^* \pm t_{\alpha/2} s \sqrt{\frac{1}{n} + \frac{(x^* - \bar{x})^2}{s_{xx}}}$$
   where $s_{xx} = \sum_{i=1}^n x_i^2 - n\bar{x}^2$.

   Similarly, show that a $100(1-\alpha)\%$ prediction interval for a new response when $x = x^*$ can be written as
   $$b_0 + b_1 x^* \pm t_{\alpha/2} s \sqrt{1 + \frac{1}{n} + \frac{(x^* - \bar{x})^2}{s_{xx}}}.$$

5. We can generate some data for a simple linear regression as follows:

```
n <- 30
x <- 1:n
y <- x + rnorm(n)
```

Construct 95% CI's for $\mathbb{E}y$ and 95% PI's for y, when $x = 1, 2, \ldots, n$. Join them up and plot them on a graph of y against x. Your plot should look something like this:



What proportion of the y's should you expect to lie beyond the outer lines?

6. In this exercise, we look at the dangers of overfitting. Generate some observations from a simple linear regression:

```
set.seed(3)
X <- cbind(rep(1,100), 1:100)
beta <- c(0, 1)
y <- X %*% beta + rnorm(100)
```

Put aside some of the data for testing and some for fitting:

```
Xfit <- X[1:50,]
yfit <- y[1:50]
Xtest <- X[51:100,]
ytest <- y[51:100]
```

(a) Using only the fitting data, estimate $\boldsymbol{\beta}$ and hence the residual sum of squares. Also calculate the residual sum of squares for the test data, that is, $\sum_{i=51}^{100}(y_i - b_0 - b_1 x_i)^2$.
Now add 10 extra predictor variables which we know have nothing to do with the response:

```
X <- cbind(X, matrix(runif(1000), 100, 10))
Xtest <- X[51:100,]
Xfit <- X[1:50,]
```

Again using only the fitting data, fit the linear model $\mathbf{y} = X\boldsymbol{\beta} + \boldsymbol{\varepsilon}$, and show that the residual sum of squares has reduced (this has to happen). Then show that the residual sum of squares for the test data has gone up (this happens most of the time).

Explain what is going on.

(b) Repeat the above, but this time add x^2, x^3 and x^4 terms:

```
X <- cbind(X[, 1:2], (1:100)^2, (1:100)^3, (1:100)^4)
```

Lab questions

1. What will be the output of the following code? Try to answer this without typing it up.

```
fb <- function(n) {
    if (n == 1 || n == 2) {
        return(1)
    } else {
        return(fb(n - 1) + fb(n - 2))
    }
}
fb(8)

## [1] 21
```

2. This question writes a function to calculate leverages and generate histograms for them, and applies it to three different datasets in order to better understand leverage.

   (a) In R, generate 20 observations from a standard bivariate normal distribution with correlation 0.8. (You can do this using the definition and pairs of standard independent normal random variables or using the function `rmvnorm` in the package `mvtnorm`).

      i. The 20 bivariate observations, labelling the first of each pair as `x1` in R, and the second as `y1`.

      ii. New variables, `x2`, `y2`, which are the pairs in (`x1`,`y1`) together with an additional point $(10, 10)$.

      iii. New variables, `x3`, `y3`, which are the pairs in (`x1`,`y1`) together with an additional point $(10, -0.3)$.

   (b) Fit simple linear regression models of the response $y$to the predictor x for each of the three data sets in part (a), storing the results in `model1`, `model2`, `model3`, including the options `x=TRUE` in order that it is possible to access the X matrix of the models subsequently.

   (c) Write a function, `histhat`, with argument, `model`, that uses matrix operations to find the leverage values in model and prints them out with the model formula as heading, as well as generates a histogram of the leverage values. To label the printout and the histogram, the following R commands may be helpful:

      i. `writeLines`

      ii. `noquote` to get text strings without quotes printed out

      iii. `paste` including the text string `"`
`n"` to get a new line before a subsequent `print`

   (d) Generate the printouts of leverages and histogram of leverages for the three models.

   (e) Comment on the leverage outputs.

(f) Plot the data in `y1` versus `x1` using the options `xlim` and `ylim` to make sure that the axes go from -2.1 to 11. Add the point $(10, 10)$ with an upper triangle and point $(10, -0.3)$ with a lower triangle. Add the lines from `model2` and `model3` using dashed and dotted lines. Add the line from `model1` in a solid line using the intercept and slope inputs.

(g) Comment on the plots.

3. Suppose you needed all 2^n binary sequences of length n. One way of generating them is with nested for loops. For example, the following code generates a matrix `binseq`, where each row is a different binary sequence of length three.

```
binseq <- matrix(nrow = 8, ncol = 3)
r <- 0 # current row of binseq
for (i in 0:1) {
  for (j in 0:1) {
    for (k in 0:1) {
      r <- r + 1
      binseq[r,] <- c(i, j, k)
    }
  }
}
binseq

##      [,1] [,2] [,3]
## [1,]    0    0    0
## [2,]    0    0    1
## [3,]    0    1    0
## [4,]    0    1    1
## [5,]    1    0    0
## [6,]    1    0    1
## [7,]    1    1    0
## [8,]    1    1    1
```

Clearly this approach will get a little tedious for large n. An alternative is to use recursion. Suppose that A is a matrix of size $2^n \times n$, where each row is a different binary sequence of length n. Then the following matrix contains all binary sequences of length $n + 1$:

$$\left(\begin{array}{c|c} \mathbf{0} & A \\ \hline \mathbf{1} & A \end{array}\right).$$

Here $\mathbf{0}$ is a vector of zeros and $\mathbf{1}$ is a vector of ones.

Use this idea to write a recursive function `binseq`, which takes as input an integer n and returns a matrix containing all binary sequences of length n, as rows of the matrix. You should find the functions `cbind` and `rbind` particularly useful.

4. Let $A = (a_{i,j})_{i,j=1}^n$ be a square matrix, and denote by $A_{(-i,-j)}$ the matrix with row i and column j removed. If A is a 1×1 matrix then $det(A)$, the determinant of A, is just $a_{1,1}$. For $n \times n$ matrices we have, for any i,

$$det(A) = \sum_{j=1}^{n} (-1)^{i+j} \, a_{i,j} \, det(A_{(-i,-j)}).$$

Use this to write a *recursive* function to calculate $det(A)$.