Multiple Linear Regression Algorithm

Using Acetylene Data

Author : Rose Ellison

The Intuition

Linear regression is a type of supervised learning algorithm which predicts continuous values of a given data point by generalising on the data that we have in hand. The linear part indicates that we are using a linear approach in generalising over the data. Multiple indicates we are describing the relationship between one dependent and more than one independent variable using a straight line. With simple linear regression we found the coefficients using the least squares method. This is too complicated for multiple linear regression so we will use matrices to make things more simple.

$$\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 \ + ...\beta_n x_n$$

Step One : Split the data into test and training sets.

Step Two: Define the matrices.

Step Three : Estimate coefficients.

Step Four : Make Predictions based on the coefficients.

Step Five : Validate the model.

Exploration

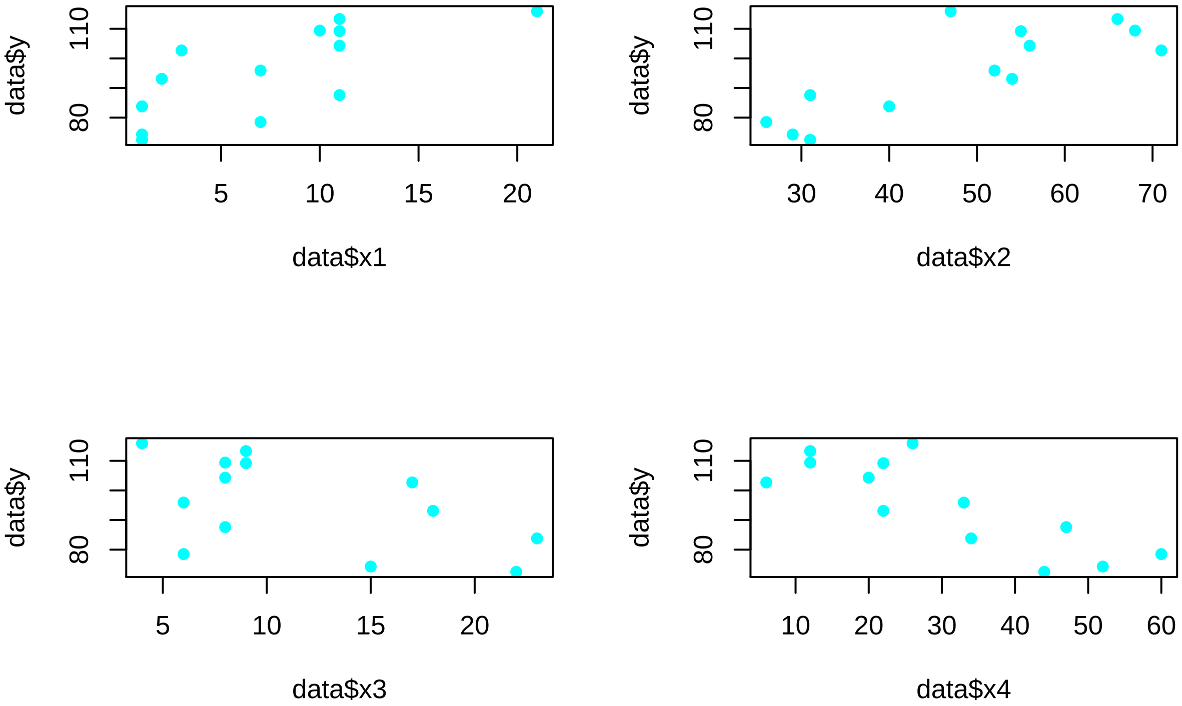
Exploration process includes preprocessing and visualizing plots. This algorithm will be implemented on the aceytyne data. There are five columns (four independent variables, and one dependent). We will use multiple linear regression to predict if the conversion of n-Heptane to Acetylene (%) on future data points.

```r
# Read the data
data = read.csv('../../../_resources/data/Acetyne.csv')
head(data)
```

```
##       y x1 x2 x3 x4
## 1  78.5  7 26  6 60
## 2  74.3  1 29 15 52
## 3 104.3 11 56  8 20
## 4  87.6 11 31  8 47
## 5  95.9  7 52  6 33
## 6 109.2 11 55  9 22
```

Plot

```r
par(mfrow = c(2,2))
plot(data$x1, data$y, col = 'cyan', pch = 16)
plot(data$x2, data$y, col = 'cyan', pch = 16)
plot(data$x3, data$y, col = 'cyan', pch = 16)
plot(data$x4, data$y, col = 'cyan', pch = 16)
```



We can see that there is a linear relationship between each of these predictors and the dependent variable. Some relationships are stronger than others. This makes this data a good candidate for multiple linear regression.

Implementation

Step One :

Split the data into test and training sets.

```
# Splitting test and training sets
split <- sample.split(data[ , 1], SplitRatio = 0.6)
training <- subset(data, split == TRUE)
test <- subset(data, split == FALSE)
```

Step Two :

Define the matrices.

```
y_matrix <- as.matrix(training$y)
y_matrix
```

```
##       [,1]
## [1,]  78.5
## [2,] 104.3
## [3,]  87.6
## [4,]  95.9
## [5,] 109.2
## [6,] 115.9
## [7,]  83.8
```

```
x_matrix <- as.matrix(cbind(1, training$x1, training$x2, training$x3, training$x4))
x_matrix
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    1    7   26    6   60
## [2,]    1   11   56    8   20
## [3,]    1   11   31    8   47
## [4,]    1    7   52    6   33
## [5,]    1   11   55    9   22
## [6,]    1   21   47    4   26
## [7,]    1    1   40   23   34
```

Step Three :

Estimate the beta hat matrix

```
betahat_matrix <- solve(t(x_matrix)%*%x_matrix)%*%t(x_matrix)%*%y_matrix
betahat_matrix
```

```
##             [,1]
## [1,] -54.4189463
## [2,]   2.7146109
## [3,]   1.7628194
## [4,]   1.3869294
## [5,]   0.9870886
```

Thus,

$$\beta_0 = 99.98944464$$
$$\beta_1 = 1.23448939$$
$$\beta_2 = 0.08963962$$
$$\beta_3 = -0.14305410$$
$$\beta_4 = -0.52219985$$

Step Four : Make predictions

$$\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 \ + \beta_3 x_3 + \beta_4 x_4$$

```
multiple_lr <- function(betas, test)
{
  B0H <- betas[1]
  B1H <- betas[2]
  B2H <- betas[3]
  B3H <- betas[4]
  B4H <- betas[5]

  x_test <- test[2:5]

  for (i in 1:nrow(test))
  {
    yhat <- B0H + B1H * x_test[i,1] + B2H * x_test[i,2] + B3H * x_test[i,3] + B4H * x_test[i,4]
    test$y_pred[i] <- yhat
  }

  return(test)
}

test <- multiple_lr(betahat_matrix, test)
```

Results

Compare predicted results(y-pred column) to the actual y values(y column) on our test set.

```
test[, c(1,6,2,3,4,5)]
```

```
##        y    y_pred x1 x2 x3 x4
## 2   74.3  71.54997  1 29 15 52
## 7  102.7 108.38539  3 71 17  6
## 8   72.5  76.88741  1 31 22 44
## 9   93.1  92.88320  2 54 18 22
## 12 113.3 116.11528 11 66  9 12
## 13 109.4 115.53938 10 68  8 12
```