Lecture 5

Regression using the normal equation

Review the setup of the normal equation method for solving a simple linear regression problem in linear algebra. Then we'll look at solving these and more complex problems in R.

Simple Linear Regression: A Linear Algebra Approach
Simple linear regression aims to find the best-fit line through a set of sample points by minimizing the sum of the squared differences between the observed values and the values predicted by the line. Here's how this process unfolds using linear algebra.

Representation of Data Points
Consider you have a set of `n` sample data points, each consisting of an input value x_i and an output value y_i:

$$\{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$$

The goal of simple linear regression is to find the line $y = \beta_0 + \beta_1 x$ that best fits these points.

System of Linear Equations
For each data point (x_i, y_i), the predicted value from the model is $\hat{y}_i = \beta_0 + \beta_1 x_i$. The goal is to minimize the residuals (differences) between observed values and predicted values:

$$Residual = y_i - (\beta_0 + \beta_1 x_i)$$

In matrix form, you can represent this system of equations as:

$$y = X\beta + \epsilon$$

Where:
y is the vector of observed values y_i.
X is the design matrix that includes a column of ones (for the intercept) and the values of x_i:

$$X = \begin{bmatrix} 1 & x_{11} & \cdots & x_{1n} \\ \vdots & & \ddots & \vdots \\ 1 & x_{n1} & \cdots & x_{nn} \end{bmatrix}$$

$\beta = (\beta_0 \; \beta_1 \dots \beta_n)$ is the vector of coefficients we want to estimate.
ϵ is the vector of errors (residuals).

Normal Equation
To find the best-fit line, we minimize the sum of squared residuals:

$$\min_{\beta} \sum_{i=1}^{n} \big(y_i - (\beta_0 + \beta_1 x_i)\big)^2$$

This can be rewritten in matrix form and solved using the normal equation:

$$\hat{\beta} = (X^\top X)^{-1} X^\top y$$

Solution by Hand: Example
Let's say we have three data points: (1,2), (2,3), and (3,5). We want to fit a line $y = \beta_0 + \beta_1 x$.

Design Matrix X:

$$X = \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \end{bmatrix}$$

$$y = \begin{bmatrix} 2 \\ 3 \\ 5 \end{bmatrix}$$

Normal Equation:

$$\hat{\beta} = (X^\top X)^{-1} X^\top y$$

Let's calculate it step by step:

Compute $X^\top X$:

$$X^\top X = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 3 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \end{bmatrix} = \begin{bmatrix} 3 & 6 \\ 6 & 14 \end{bmatrix}$$

Compute $X^\top y$:

$$X^\top y = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 3 \end{bmatrix} \begin{bmatrix} 2 \\ 3 \\ 5 \end{bmatrix} = \begin{pmatrix} 10 \\ 22 \end{pmatrix}$$

Compute $(X^T X)^{-1}$:

$$(X^T X)^{-1} = \frac{1}{3 \times 14 - 6 \times 6} \begin{bmatrix} 14 & -6 \\ -6 & 3 \end{bmatrix} = \frac{1}{6} \begin{bmatrix} 14 & -6 \\ -6 & 3 \end{bmatrix}$$

Calculate $\hat{\beta}$:

$$\hat{\beta} = (X^T X)^{-1} X^T y = \frac{1}{6} \begin{bmatrix} 14 & -6 \\ -6 & 3 \end{bmatrix} \begin{pmatrix} 10 \\ 22 \end{pmatrix} \approx \begin{bmatrix} 4.67 \\ 4.33 \end{bmatrix}$$

Thus, the estimated coefficients are approximately $\hat{\beta}_0 = 4.67$ and $\hat{\beta}_1 = 4.33$.

The final regression line is:

$$y = 4.67 + 4.33x$$

This is the best-fit line according to the least squares criterion, derived using linear algebra and the normal equation.

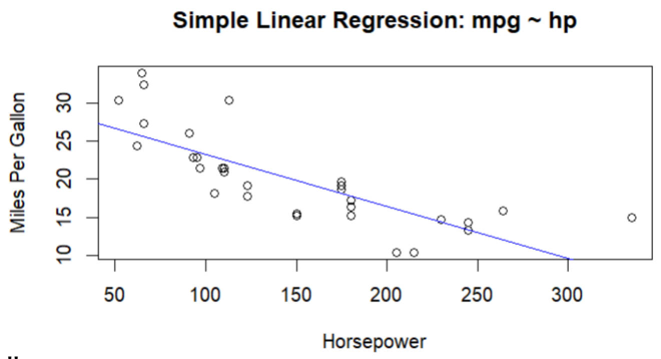
We can apply this same strategy to more complex equations, as long as the coefficient matrix is linear (the variables themselves don't need to be).

Let's consider the implementation of simple linear regression, multiple regression, and non-linear regression using the normal equation in R, with step-by-step programming examples, visualizations, and comparisons to built-in R functions. The session will also touch on the covariance matrix and its interpretation.

Recall our framework:
For a simple linear regression: $y = \beta_0 + \beta_1 x + \epsilon$
In matrix form, $y = X\beta + \epsilon$
The normal equation is given by: $\hat{\beta} = (X^\top X)^{-1} X^\top y$

We'll predict `mpg` from `hp` in the `mtcars` dataset.

```
# Load mtcars dataset
data(mtcars)

# Extract variables
X <- as.matrix(cbind(1, mtcars$hp))  # Add a column of ones for the intercept
y <- mtcars$mpg

# Normal Equation: beta = (X^T * X)^(-1) * X^T * y
beta_hat <- solve(t(X) %*% X) %*% t(X) %*% y

# Print coefficients
print("Coefficients:")
print(beta_hat)

# Plot the data and the regression line
plot(mtcars$hp, mtcars$mpg, main = "Simple Linear Regression: mpg ~ hp", xlab =
"Horsepower", ylab = "Miles Per Gallon")
abline(beta_hat[1], beta_hat[2], col = "blue")

model_lm <- lm(mpg ~ hp, data = mtcars)
summary(model_lm)
```



```
[1] "Coefficients:"
            [,1]
[1,] 30.09886054
[2,] -0.06822828

Call:
```

```
lm(formula = mpg ~ hp, data = mtcars)

Residuals:
    Min      1Q  Median      3Q     Max 
-5.7121 -2.1122 -0.8854  1.5819  8.2360 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) 30.09886    1.63392  18.421  < 2e-16 ***
hp          -0.06823    0.01012  -6.742 1.79e-07 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 3.863 on 30 degrees of freedom
Multiple R-squared:  0.6024,	Adjusted R-squared:  0.5892 
F-statistic: 45.46 on 1 and 30 DF,  p-value: 1.788e-07
```

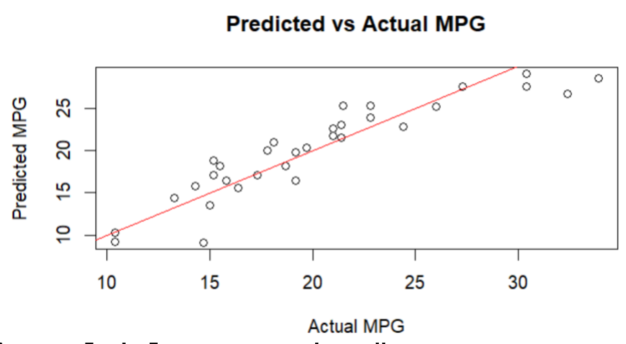
The lm() function has many more embedded operations than just finding the coefficients, but the coefficients it finds are the same, and the normal equation is how these coefficients are computed. You’ll note that in some model outputs, there will be a note about the number of iterations for convergence (such as in the logistic regression model). This is an indication that they are not using exact methods to find the model parameters, but are instead using an optimization algorithm as discussed in a prior lecture.

To help understand how this method connects to the by-hand example, it may help to display/print the matrices along the way. It’s not necessary for the computation, but it may help to connect the dots.

Let’s extend the example with multiple variables.

We'll predict `mpg` using `hp`, `wt`, and `qsec` as predictors.

```
# Extract variables for multiple regression
X_multi <- as.matrix(cbind(1, mtcars$hp, mtcars$wt, mtcars$qsec))
y_multi <- mtcars$mpg

# Normal Equation for Multiple Regression
beta_hat_multi <- solve(t(X_multi) %*% X_multi) %*% t(X_multi) %*% y_multi

# Print coefficients
print("Coefficients for Multiple Regression:")
print(beta_hat_multi)

y_pred <- X_multi %*% beta_hat_multi

plot(y_multi, y_pred, main = "Predicted vs Actual MPG", xlab = "Actual MPG", ylab = "Predicted
MPG")
abline(0, 1, col = "red")  # Line of perfect prediction

model_lm_multi <- lm(mpg ~ hp + wt + qsec, data = mtcars)
summary(model_lm_multi)
```



```
[1] "Coefficients for Multiple Regression:"
            [,1]
[1,] 27.61052686
[2,] -0.01782227
[3,] -4.35879720
[4,]  0.51083369

Call:
lm(formula = mpg ~ hp + wt + qsec, data = mtcars)

Residuals:
    Min      1Q  Median      3Q     Max 
-3.8591 -1.6418 -0.4636  1.1940  5.6092 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) 27.61053    8.41993   3.279  0.00278 ** 
hp          -0.01782    0.01498  -1.190  0.24418    
wt          -4.35880    0.75270  -5.791 3.22e-06 ***
qsec         0.51083    0.43922   1.163  0.25463    
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.578 on 28 degrees of freedom
Multiple R-squared:  0.8348,	Adjusted R-squared:  0.8171 
F-statistic: 47.15 on 3 and 28 DF,  p-value: 4.506e-11
```

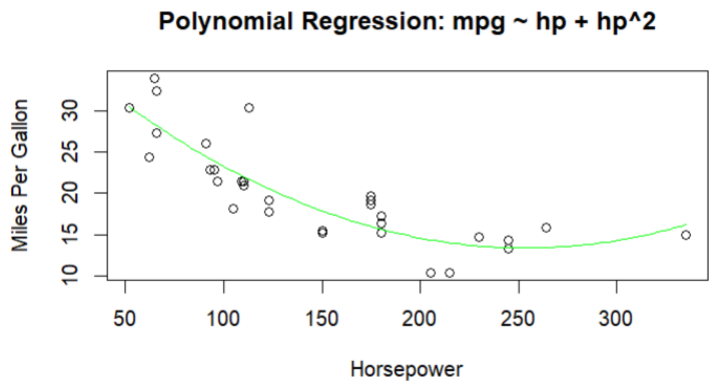
Again, we can see that we get identical outcomes of the coefficients.

When dealing with summation versions of the regression equations, we have different equations for single or multiple regression, or if we are using polynomial regression, and differing by the number of variables in the equation. But when we use the normal equation, the process is exactly the same except for the set-up of the design matrix X.

We'll extend the simple linear regression to a polynomial (quadratic) regression.

```
# Setup for quadratic regression
X_poly <- as.matrix(cbind(1, mtcars$hp, mtcars$hp^2))
y_poly <- mtcars$mpg

# Solve using the normal equation
beta_hat_poly <- solve(t(X_poly) %*% X_poly) %*% t(X_poly) %*% y_poly

# Print coefficients
print("Coefficients for Polynomial Regression:")
print(beta_hat_poly)
```

```
# Plot the data and the regression curve
plot(mtcars$hp, mtcars$mpg, main = "Polynomial Regression: mpg ~ hp + hp^2", xlab =
"Horsepower", ylab = "Miles Per Gallon")
curve(beta_hat_poly[1] + beta_hat_poly[2] * x + beta_hat_poly[3] * x^2, add = TRUE, col =
"green")

model_lm_poly <- lm(mpg ~ hp + I(hp^2), data = mtcars)
summary(model_lm_poly)
```



```
[1] "Coefficients for Polynomial Regression:"
              [,1]
[1,] 40.4091172029
[2,] -0.2133082599
[3,]  0.0004208156

Call:
lm(formula = mpg ~ hp + I(hp^2), data = mtcars)

Residuals:
    Min      1Q  Median      3Q     Max 
-4.5512 -1.6027 -0.6977  1.5509  8.7213 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)    
(Intercept)  4.041e+01  2.741e+00  14.744 5.23e-15 ***
hp          -2.133e-01  3.488e-02  -6.115 1.16e-06 ***
I(hp^2)      4.208e-04  9.844e-05   4.275 0.000189 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 3.077 on 29 degrees of freedom
Multiple R-squared:  0.7561,	Adjusted R-squared:  0.7393 
F-statistic: 44.95 on 2 and 29 DF,  p-value: 1.301e-09
```

Here, we see some differences in the output only in terms of scientific notation or standard decimal notation.

Covariance Matrix

Another important element we want to consider is the covariance matrix.

```
# Covariance matrix for the coefficients in the simple linear regression
sigma_squared <- sum((y - X %*% beta_hat)^2) / (nrow(X) - ncol(X))
```

```
cov_matrix <- sigma_squared * solve(t(X) %*% X)

print("Covariance Matrix for Simple Linear Regression:")
print(cov_matrix)
```

Interpreting the Covariance Matrix:
The diagonal elements represent the variance of each coefficient. Off-diagonal elements represent the covariance between coefficients.

Resources:

1. https://www.cuemath.com/algebra/covariance-matrix/
2. https://www.stat.cmu.edu/~larry/=stat401/lecture-13.pdf
3. https://www.geeksforgeeks.org/covariance-matrix/#