# Lec 3: Linear Regression

Ailin Zhang

2022-09-19

# Agenda

- Multiple Regression and Least Square
- Maximum Likelihood Estimation
- Gauss Jordan Elimination
- The Sweep Operator

# Linear Regression

The dataset of linear regression consists of an $n \times p$ matrix $\mathbf{X} = (x_{ij})$, and a $n \times 1$ vector $\mathbf{Y} = (y_i)$.

$$y_i = \sum_{j=1}^{p} x_{ij}\beta_j + \epsilon_i,$$

for $i = 1, ..., n$, where $\epsilon_i \sim \mathrm{N}(0, \sigma^2)$ independently for $i = 1, ..., n$.
Ambiguous about the intercept term.

- $[\mathbf{X}, \mathbf{Y}]$ is called the training data
- $y_i$ is called response variable, outcome, dependent variable.
- $x_{ij}$ is called predictor, regressor, covariate, independent variable, or simple variable.
- In the experimental design setting, $\mathbf{X}$ is called the design matrix.

# Linear Regression

| obs | $\mathbf{X}_{n\times p}$ | $\mathbf{Y}_{n\times 1}$ |
|---|---|---|
| 1 | $x_{11}, x_{12}, ..., x_{1p}$ | $y_1$ |
| 2 | $x_{21}, x_{22}, ..., x_{2p}$ | $y_2$ |
| ... | | |
| $n$ | $x_{n1}, x_{n2}, ..., x_{np}$ | $y_n$ |

1. Explanation: understanding the relationship between $y_i$ and $(x_{ij}, j = 1, ..., p)$.
2. Prediction: learn to predict $y_i$ based on $(x_{ij}, j = 1, ..., p)$, so that in the testing stage, if we are given the predictor variables, we should be able to predict the outcome.

# Row Vector treatment

| obs | $\mathbf{X}_{n \times p}$ | $\mathbf{Y}_{n \times 1}$ |
|---|---|---|
| 1 | $X_1^\top$ | $y_1$ |
| 2 | $X_2^\top$ | $y_2$ |
| ... | | |
| $n$ | $X_n^\top$ | $y_n$ |

$X_i^\top = (x_{ij}, j = 1, ..., p)$

where $X_i^\top$ is the $i$-th row of $\mathbf{X}$ .

Here $X_i$ is not in bold font.

We can write the model as $y_i =< X_i, \beta > +\epsilon_i = X_i^\top \beta + \epsilon_i$, where $\beta = (\beta_j, j = 1, ..., p)^\top$.

# Least Square Method

Least square loss function: $Loss(\beta) = \frac{1}{2}\sum_{i=1}^{n} \epsilon_i^2$

$\epsilon_i = y_i - s_i$, where $s_i = \sum_{j=1}^{p} x_{ij}\beta_j$

$$\frac{\partial Loss(\beta)}{\partial \beta_k} = \sum_{i=1}^{n} \epsilon_i \frac{\partial \epsilon_i}{\partial s_i}\frac{\partial s_i}{\partial \beta_k} = -\sum_{i=1}^{n} \epsilon_i x_{ik}$$

$\frac{\partial Loss(\beta)}{\partial \beta_k} = -\sum_{i=1}^{n} X_i(y_i - X_i^\top \beta) = 0$

Question: what is the dimensionality of $X_i y_i$ and $X_i X_i^T$?

# Maximum Likelihood

More general than least square $\epsilon_i \sim N(0, \sigma^2)$

$$likelihood(\beta) = \prod_{i=1}^{n} p(y_i|s_i)$$

Since $y_i = s_i + \epsilon_i$, $[y_i|s_i] \sim N(s_i, \sigma^2)$

# Column Vector Treatment

| obs | $\mathbf{X}_{n\times p}$ | $\mathbf{Y}_{n\times 1}$ |
|---|---|---|
| 1<br>2<br>...<br>$n$ | $\mathbf{X}_1, \mathbf{X}_2, ..., \mathbf{X}_p$ | $\mathbf{Y}$ |

# Geometric Explaination

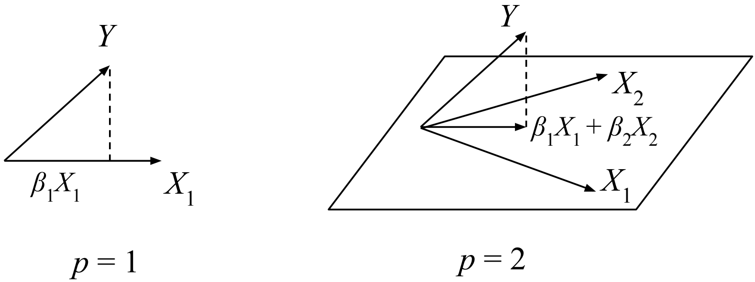


**Figure 1:** Least squares projection

Summary: Solve $\beta$ from $(\mathbf{X}^\top \mathbf{X})\beta = \mathbf{X}^\top \mathbf{Y}$

$$\beta = (\mathbf{X}^\top \mathbf{X})^{-1}(\mathbf{X}^\top \mathbf{Y})$$

# Gauss Jordan Elimination - Example

$$\begin{cases} x_1 + x_2 + x_3 = 5 \\ 2x_1 + 3x_2 + 5x_3 = 9 \\ 4x_1 + 5x_3 = 2 \end{cases}$$

# Gauss Jordan Elimination

$$\mathrm{GJ}[1:n][A|b] = [I|A^{-1}b] = A^{-1}[A|b],$$
$$\mathrm{GJ}[1:n][A|I] = [I|A^{-1}] = A^{-1}[A|I].$$

# Gauss Jordan Elimination

For a system of linear equations $Ax = b$

$A = (a_{ij})$ is $n \times n$, $x = (x_i)$ is $n \times 1$, and $b = (b_i)$ is $n \times 1$

we can solve $x = A^{-1}b$ by Gauss-Jordan elimination.

Specifically, for any matrix $A$ (any $n \times N$ matrix)

let $\tilde{A} = \mathrm{GJ}[k]A$, then

$$\begin{aligned} \tilde{A}_k &= A_k / a_{kk}, \\ \tilde{A}_i &= A_i - a_{ik}\tilde{A}_k,\ i \neq k, \end{aligned}$$

$A_k$ is the $k$-th row of $A$. $\tilde{a}_{kk} = 1$, and $\tilde{a}_{ik} = 0$ for $i \neq k$.

- Apply Gauss-Jordan sequentially: $\mathrm{GJ}[1:m]$ means we apply Gauss-Jordan for $k = 1:m$.
- Gauss-Jordan is linear: $\tilde{A} = \mathrm{GJ}[k]A \rightarrow \tilde{A} = G_k A$ for a matrix $G_k$.

# R code for Gauss Jordan

```r
myGaussJordan <- function(A, m)
{
n <- dim(A)[1]
B <- cbind(A, diag(rep(1, n)))
for (k in 1:m)
{
  a <- B[k, k]
  for (j in 1:(n*2))
     B[k, j] <- B[k, j]/a
  for (i in 1:n)
     if (i != k)
     {
        a <- B[i, k]
        for (j in 1:(n*2))
           B[i, j] <- B[i, j] - B[k, j]*a;
    }
 }
return(B)
}

A = matrix(c(1,2,3,7,11,13,17,21,23), 3,3)
myGaussJordan(A,3)
```

```
##      [,1] [,2] [,3]  [,4] [,5]  [,6]
## [1,]    1    0    0  1.00 -3.0  2.00
## [2,]    0    1    0 -0.85  1.4 -0.65
## [3,]    0    0    1  0.35 -0.4  0.15
```

# Computing Efficiency

What is the time complexity?

Can we get rid of any loop?

```
myGaussJordan <- function(A, m)
{
n <- dim(A)[1]
B <- cbind(A, diag(rep(1, n)))
for (k in 1:m)
{
  a <- B[k, k]
  for (j in 1:(n*2))
     B[k, j] <- B[k, j]/a
  for (i in 1:n)
     if (i != k)
     {
        a <- B[i, k]
        for (j in 1:(n*2))
           B[i, j] <- B[i, j] - B[k, j]*a;
    }
 }
return(B)
}
```