Self-Notes on [Least Squares Regression]

[Unat Tekşen] [504241592]

May 2, 2025

1 Least Squares Regression

A method used to find the best-fitting line for a given X by minimizing the sum of the squared errors between the observed and the predicted values.

The error:

$$\text{Mean Squared Error} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{1}$$

The objective function:

$$J(\theta) = \frac{1}{N}\sum_{i=1}^{N}(y_i - \theta_0 - \theta_1 x_i)^2 \quad \text{(Objective Function)} \tag{2}$$

$$\frac{\partial J(\theta)}{\partial \theta_0} = -\frac{1}{N}2\sum_{i=1}^{N}(y_i - \theta_0 - \theta_1 x_i) = 0 \tag{3}$$

$$\frac{\partial J(\theta)}{\partial \theta_1} = -\frac{1}{N}2\sum_{i=1}^{N}x_i(y_i - \theta_0 - \theta_1 x_i) = 0 \tag{4}$$

Note:
$x^{(i)} = x_1^{(i)} \quad 1 \longrightarrow$ one feature
Note:

This

$$\sum(y^{(i)} - \theta_0 - \theta_1 x^{(i)})$$

term can be written as:

$$\sum y^{(i)} - \sum \theta_0 - \sum \theta_1 x^{(i)} = 0 \tag{5}$$

$$\sum x^{(i)} y^{(i)} - \sum \theta_0 x^{(i)} - \sum \theta_1 (x^{(i)})^2 = 0 \tag{6}$$

$$N\theta_0 + \theta_1 \sum x^{(i)} = \sum y^{(i)} \tag{7}$$

$$\theta_0 \sum x^{(i)} + \theta_1 \sum (x^{(i)})^2 = \sum x^{(i)} y^{(i)} \tag{8}$$

$$\begin{bmatrix} N & \sum x^{(i)} \\ \sum x^{(i)} & \sum (x^{(i)})^2 \end{bmatrix} \begin{bmatrix} \theta_0 \\ \theta_1 \end{bmatrix} = \begin{bmatrix} \sum y^{(i)} \\ \sum x^{(i)} y^{(i)} \end{bmatrix} \tag{9}$$

$(d+1) \times (d+1)$ matrix
The main equation:

$$\hat{y} = x\theta$$

where:

$$\underbrace{\begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(N)} \end{bmatrix}}_{y} - \underbrace{\begin{bmatrix} \hat{y}^{(1)} \\ \hat{y}^{(2)} \\ \vdots \\ \hat{y}^{(N)} \end{bmatrix}}_{\hat{y}} = \underbrace{\begin{bmatrix} y^{(1)} - \hat{y}^{(1)} \\ y^{(2)} - \hat{y}^{(2)} \\ \vdots \\ y^{(N)} - \hat{y}^{(N)} \end{bmatrix}}_{y-\hat{y}} = \underbrace{\begin{bmatrix} 1 & x^{(1)} \\ 1 & x^{(2)} \\ \vdots & \vdots \\ 1 & x^{(N)} \end{bmatrix}}_{X} \underbrace{\begin{bmatrix} \theta_0 \\ \theta_1 \end{bmatrix}}_{\theta} \tag{10}$$

MSE can be written as:

$$\frac{1}{N} \left[\left(y^{(1)} - \hat{y}^{(1)} \right)^2 + \cdots + \left(y^{(N)} - \hat{y}^{(N)} \right)^2 \right] \tag{11}$$

Objective function can be written as:

$$J(\theta) = \frac{1}{N} (y - \hat{y})^T (y - \hat{y}) = \frac{1}{N} (y - X\theta)^T (y - X\theta) \tag{12}$$

$$= y^T y - \hat{y}^T x\theta - \theta^T x^T \hat{y} + \theta^T x^T x\theta \quad (\text{Skipped } \frac{1}{N}) \tag{13}$$

$$= y^T y - 2y^T x\theta + \theta^T x^T x\theta \tag{14}$$

The dimensions of each term are as follows:

- $y^T y$:
  - y^T: $1 \times N$
  - y: $N \times 1$
  - Result: 1×1 (scalar)
- $2y^T \theta$:
  - y^T: $1 \times N$
  - θ: $N \times 1$
  - Result: 1×1 (scalar)

- $\theta^T x^T x\theta$:
  - θ^T: $1 \times N$
  - x^T: $N \times N$
  - x: $N \times 1$
  - θ: $N \times 1$
  - Result: 1×1 (scalar)

1.1 Derivatives with respect to Vectors

$$S = \mathbf{a}^T \mathbf{v} = a_1 v_1 + a_2 v_2 + \cdots + a_n v_n \tag{15}$$

Note:

$$S \in \mathbb{R}, \quad \mathbf{a}, \mathbf{b}, \mathbf{v} \in \mathbb{R}^d \tag{16}$$

$$\frac{\partial S}{\partial \theta} = \begin{bmatrix} \frac{\partial S}{\partial \theta_1} \\ \frac{\partial S}{\partial \theta_2} \\ \vdots \\ \frac{\partial S}{\partial \theta_d} \end{bmatrix} = \begin{bmatrix} \frac{\partial(a_1\theta_1 + a_2\theta_2 + \cdots + a_d\theta_d)}{\partial \theta_1} \\ \frac{\partial(a_1\theta_1 + a_2\theta_2 + \cdots + a_d\theta_d)}{\partial \theta_2} \\ \vdots \\ \frac{\partial(a_1\theta_1 + a_2\theta_2 + \cdots + a_d\theta_d)}{\partial \theta_d} \end{bmatrix} = \underbrace{\begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_d \end{bmatrix}}_{\mathbf{a}} \tag{17}$$

$$\frac{\partial \mathbf{a}^T \theta}{\partial \theta} = \mathbf{a} \tag{18}$$

Assume that vector A is given:

$$A \in \mathbb{R}^{d \times d}, \quad A = \begin{bmatrix} x_1 & x_2 \\ x_3 & x_4 \end{bmatrix} \tag{19}$$

Expanding matrix equation:

$$S = \theta^T A \theta \tag{20}$$

$$S = \begin{bmatrix} \theta_1 & \theta_2 \end{bmatrix} \begin{bmatrix} x_1 & x_2 \\ x_3 & x_4 \end{bmatrix} \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix} \tag{21}$$

$$S = \begin{bmatrix} \theta_1 x_1 + \theta_2 x_3 & \theta_1 x_2 + \theta_2 x_4 \end{bmatrix} \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix} \tag{22}$$

$$S = \theta_1 x_1 + \theta_1 \theta_2 x_3 + \theta_1 x_2 \theta_2 + \theta_2^2 x_4 \tag{23}$$

$$\frac{\partial S}{\partial \sigma} = \begin{bmatrix} \frac{\partial S}{\partial \theta_1} \\ \frac{\partial S}{\partial \theta_2} \end{bmatrix} = \begin{bmatrix} 2\theta_1 x_1 + \theta_2 x_3 + \theta_2 x_2 \\ \theta_1 x_3 + \theta_1 x_2 + 2\theta_2 x_4 \end{bmatrix} \tag{24}$$

Expanding the terms:

$$= \begin{bmatrix} \theta_1 x_1 + \theta_2 x_2 \\ \theta_1 x_3 + \theta_2 x_4 \end{bmatrix} + \begin{bmatrix} \theta_1 x_2 + \theta_2 x_3 \\ \theta_1 x_4 + \theta_2 x_4 \end{bmatrix} \tag{25}$$

$$= \begin{bmatrix} x_1 & x_2 \\ x_3 & x_4 \end{bmatrix} \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix} + \begin{bmatrix} x_1 & x_3 \\ x_2 & x_4 \end{bmatrix} \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix} \tag{26}$$

Recall, objective function:

$$J(\theta) = y^T y - 2y^T x\theta + \theta^T x^T x\theta \tag{27}$$

$$\frac{\partial J}{\partial \theta} = \frac{\partial}{\partial \theta}(y^T y - 2y^T \theta + \theta^T x^T x\theta) \tag{28}$$

$$= -2x^T y + (x^T x + (x^T x)^T)\theta = 0 \tag{29}$$

$$= -2x^T y + 2x^T x\theta = 0 \tag{30}$$

$$= x^T x\theta = x^T y \tag{31}$$

Multiply with inverse:

$$= (x^T x)^{-1} x^T x\theta = (x^T x)^{-1} x^T y = 0 \tag{32}$$

We obtain closed form equation:

$$\theta = (x^T x)^{-1} x^T y \quad \textbf{(Closed form)} \tag{33}$$

Dimensions:

- θ: $(d+1) \times 1$
- $(x^T x)^{-1}$: $(d+1) \times (d+1)$
- x^T: $(d+1) \times N$
- y: $N \times 1$

Note:

$$X = \begin{bmatrix} 1 & x_1^{(1)} & x_2^{(1)} & \cdots & x_d^{(1)} \\ 1 & x_1^{(2)} & x_2^{(2)} & \cdots & x_d^{(2)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(N)} & x_2^{(N)} & \cdots & x_d^{(N)} \end{bmatrix}_{N \times (d+1)}, \quad \theta = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_{d+1} \end{bmatrix}_{(d+1) \times 1} \tag{34}$$

2 Gradient Descent

Gradient Descent is an optimization algorithm used to minimize a function by iteratively moving in the direction of its steepest descent, which is given by the negative of the gradient.

$$\frac{\partial J(\theta)}{\partial \theta} \longrightarrow + \tag{35}$$

$$\theta^{k+1} = \theta^{k} - \alpha \frac{\partial J(\theta)}{\partial \theta} \tag{36}$$

$$\frac{\partial J(\theta)}{\partial \theta} = -2x^T y + 2x^T x\theta \tag{37}$$

Update parameters:

$$\begin{bmatrix} \theta_0^{t+1} \\ \theta_1^{t+1} \\ \vdots \\ \theta_d^{t+1} \end{bmatrix} = \begin{bmatrix} \theta_0^{t} \\ \theta_1^{t} \\ \vdots \\ \theta_d^{t} \end{bmatrix} - \alpha \frac{\partial J(\theta)}{\partial \theta} \tag{38}$$

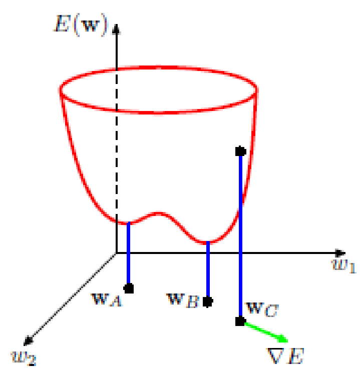

Figure 1: This illustrates the error function E(w) as a surface in weight space. It shows local minimum wA, global minimum wB, and the gradient ∇E at point wC. Retrieved from [1], page 211.

2.1 Find Potential Min/Max

There are local minimums, local maximums, saddle points in loss function plot.

$$(dimension = 1, J(\theta), 0) \tag{39}$$

$$N \text{ dimensional}, \quad \nabla J(\theta) = 0_{N\times 1} \longrightarrow \text{ gradient vector} \tag{40}$$

- if loss function is convex,
- function's 1st derivative is linear,
- function's 2nd derivative is constant and positive.

$$\begin{aligned} \text{for convex functions } J''(\theta) \geq 0 \\ \text{for concave functions } J''(\theta) \leq 0 \end{aligned} \tag{41}$$

Hessian Matrix:

$$H = \begin{bmatrix} \frac{\partial^2 J(\theta)}{\partial\theta_0 \partial\theta_0} & \frac{\partial^2 J(\theta)}{\partial\theta_1 \partial\theta_0} & \cdots & \frac{\partial^2 J(\theta)}{\partial\theta_d \partial\theta_0} \\ \frac{\partial^2 J(\theta)}{\partial\theta_0 \partial\theta_1} & \frac{\partial^2 J(\theta)}{\partial\theta_1 \partial\theta_1} & \cdots & \frac{\partial^2 J(\theta)}{\partial\theta_d \partial\theta_1} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 J(\theta)}{\partial\theta_0 \partial\theta_d} & \frac{\partial^2 J(\theta)}{\partial\theta_1 \partial\theta_d} & \cdots & \frac{\partial^2 J(\theta)}{\partial\theta_d \partial\theta_d} \end{bmatrix} \tag{42}$$

$$\begin{aligned} \text{If } H > 0, \quad & \text{P.S.D (Positive Semi-Definite)}: \text{ convex} \\ H < 0, \quad & \text{N.S.D (Negative Semi-Definite)}: \text{ concave} \end{aligned} \tag{43}$$

P.S.D (Positive Semi-Definite):

$$a \in \mathbb{R}^d, \quad H \in \mathbb{R}^{d \times d}, \quad \forall a \quad a^T H a \geq 0 \tag{44}$$

$$Hx = \lambda x \tag{45}$$

$$x^T H x = \lambda x^T x \geq 0 \tag{46}$$

- All eigenvalues are non-negative.

N.S.D (Negative Semi-Definite):

- All eigenvalues are non-positive.

$$\nabla J(\theta) = -2x^T y + 2x^T x\theta \tag{47}$$

Check P.S.D

$$H = 2x^T x \tag{48}$$

$$2 \underbrace{a^T}_{y^T} \underbrace{x^T x}_{y} a \geq 0 \quad \} \text{ (convex, single minimum point)} \tag{49}$$

Example:

$$X = \begin{bmatrix} 1 & 50 & (50)^2 & (50)^3 \\ 1 & 60 & (60)^2 & (60)^3 \\ \vdots & \vdots & \vdots & \vdots \\ 1 & 55 & (55)^2 & (55)^3 \end{bmatrix} \tag{50}$$

General equation:

$$= \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 \tag{51}$$

$$\theta_0+\theta_1 x+\theta_2 x^2+\cdots+\theta_{15}x^{15} = \theta_0+\theta_1(x^{(1)})+\theta_2(x^{(1)})^2+\cdots+\theta_d(x^{(1)})^d = y^{(i)} \tag{52}$$

$$= y^{(i)} \tag{53}$$

$$\begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix} = (x^T x)^{-1} x^T y \tag{54}$$

2.1.1 Regularization

Regularization in is used to prevent overfitting by adding a penalty term to the loss function. It helps improve the generalization of a model by discouraging it from fitting noise in the training data.

Adding Ridge Regularization:

$$J(\theta) = \frac{1}{N}(y - x\theta)^T(y - x\theta) + \lambda\theta^T\theta \tag{55}$$

Trade off:

$$\lambda \to \infty, \quad \theta \to 0 \tag{56}$$

2.2 Cross-Validation

CV procedure:

1. **Prepare Data** – Clean, preprocess, and normalize if needed.
2. **Choose Cross-Validation Method** – Common types: K-Fold, LOOCV, Stratified K-Fold.
3. **Split Data** – Divide into training and validation sets based on the chosen method.
4. **Train & Validate** – Train the model on $k-1$ folds, test on the remaining fold, and repeat.

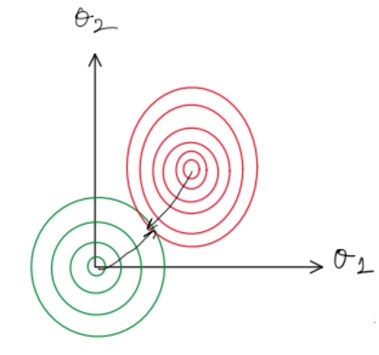

Figure 2: λ changes shape of $\theta^T\theta$. Solution $\longrightarrow$ intersection points of shapes. [My drawing from course note.]

5. **Evaluate Performance** – Compute the average validation error across all folds.
6. **Tune Hyperparameters** – Adjust model parameters based on validation results.
7. **Final Training** – Train on the full dataset before making predictions.

Example:

$$X = \begin{bmatrix} 50 \\ 60 \\ 55 \\ 75 \\ \vdots \end{bmatrix}_{N\times 1}, \quad y = \begin{bmatrix} 5K \\ 6K \\ \vdots \end{bmatrix} \tag{57}$$

p	Avr. Training	Avr. Validation
1	18	17
2	15	18
3	10	12
4	8	15
5	0	20

Table 1: Data from Avr. Training and Avr. Validation

What is the best p?

$$\theta_0 + \theta_1 x + \theta_2 x^2 + \cdots + \theta_p x^p \tag{58}$$

Training	Validation
Err = 20	5th MSE = 20
15	4th MSE = 18
⋮	⋮
Err = 18	1st MSE = 19

Table 2: Training and Validation Results

For p=1, each training and validation result is calculated by taking average of all.

If we have 100 samples, 5 folds, we can use first 4 folds for training samples:

$$X = \begin{bmatrix} 1 & x^{(1)} \\ 1 & \vdots \\ \vdots & \vdots \\ 1 & x^{(80)} \end{bmatrix} \quad (training) \tag{59}$$

For validation:

$$= \frac{1}{20} \sum_{j=1}^{20} (y^j - \hat{y}^j) \quad (test) \tag{60}$$

References

[1] Christopher M. Bishop and Hugh Bishop, *Deep learning: Foundations and concepts*. Springer Nature, 2023.