

≡ ✎ > ChatGPT 5.2 ▾

🔗 ↗ ⋮

step : 0 - Declare the algorithm

Algorithm Name: simple linear regression using OLS

Start the 12-step algorithm-mastery sequence.

Only acknowledge that you understood and wait for Step 1. Do not explain anything yet.

Example:

Algorithm Name: Logistic Regression Start the 12-step algorithm-mastery sequence. Only acknowledge you understood and wait for Step 1.

Acknowledged.

Algorithm Name: **Simple Linear Regression using OLS**

12-step algorithm-mastery sequence initialized.

Waiting for **Step 1**.

📄 🍷 🗨️ ↺ ⋮

STEP 1 — Problem Definition

Step 1: Give me the complete Problem Definition of the above algorithm using only mathematical symbols and formal statements. No intuition, no examples, no narrative.

STEP 1 — Problem Definition

Let

$$\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n, \quad x_i \in \mathbb{R}, y_i \in \mathbb{R}$$

Define the hypothesis class

$$\mathcal{H} = \{h_\theta : \mathbb{R} \rightarrow \mathbb{R} \mid h_\theta(x) = \theta_0 + \theta_1 x, \theta = (\theta_0, \theta_1) \in \mathbb{R}^2\}$$

Define the empirical risk functional

$$J(\theta) = \frac{1}{n} \sum_{i=1}^n (y_i - h_{\theta}(x_i))^2$$

Optimization problem:

$$\theta^* = \arg \min_{\theta \in \mathbb{R}^2} J(\theta)$$

Output:

$$\theta^* = (\theta_0^*, \theta_1^*)$$

Assumptions:

$$n \geq 2, \quad \exists i \neq j : x_i \neq x_j$$

Loss function:

$$\ell(y, \hat{y}) = (y - \hat{y})^2$$

Objective:

$$\min_{\theta_0, \theta_1} \frac{1}{n} \sum_{i=1}^n (y_i - (\theta_0 + \theta_1 x_i))^2$$



## STEP 2 — Variables and Parameters

Step 2: Define all variables, parameters, assumptions, and mathematical objects used in the algorithm with precise symbolic definitions.

## STEP 2 — Variables and Parameters

Dataset:

$$\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$$

Sample size:

$$n \in \mathbb{N}, \quad n \geq 2$$

Input variable:

$$x_i \in \mathbb{R}, \quad i = 1, \dots, n$$

Target variable:

$$y_i \in \mathbb{R}, \quad i = 1, \dots, n$$

Parameter vector:

$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \end{bmatrix} \in \mathbb{R}^2$$

Intercept parameter:

$$\theta_0 \in \mathbb{R}$$

Slope parameter:

$$\theta_1 \in \mathbb{R}$$

Hypothesis function:

$$h_\theta : \mathbb{R} \rightarrow \mathbb{R}, \quad h_\theta(x) = \theta_0 + \theta_1 x$$

Predicted output:

$$\hat{y}_i = h_\theta(x_i)$$

Residual (error term):

$$\varepsilon_i = y_i - \hat{y}_i$$

Loss function:

$$\ell : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}_{\geq 0}, \quad \ell(y, \hat{y}) = (y - \hat{y})^2$$

Empirical risk (objective function):

$$J(\theta) = \frac{1}{n} \sum_{i=1}^n \ell(y_i, h_\theta(x_i)) = \frac{1}{n} \sum_{i=1}^n (y_i - (\theta_0 + \theta_1 x_i))^2$$

Design matrix:

$$X = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{bmatrix} \in \mathbb{R}^{n \times 2}$$

Target vector:

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \in \mathbb{R}^n$$

Prediction vector:

$$\hat{\mathbf{y}} = X\theta$$

Assumptions:

$$\text{rank}(X) = 2$$

$$\sum_{i=1}^n (x_i - \bar{x})^2 > 0$$

$$\varepsilon_i \in \mathbb{R}, \quad \forall i$$

Optimization variable:

$$\theta \in \mathbb{R}^2$$

Solution:

$$\theta^* = \arg \min_{\theta \in \mathbb{R}^2} J(\theta)$$

Waiting for **STEP 3**.



### STEP 3 — Model / Hypothesis Equation

Step 3: Write the exact model/hypothesis equation for the algorithm with no explanation—only equations and mathematical statements.

### STEP 3 — Model / Hypothesis Equation

$$h_{\theta}(x) = \theta_0 + \theta_1 x$$

$$\hat{y}_i = h_{\theta}(x_i) = \theta_0 + \theta_1 x_i, \quad i = 1, \dots, n$$

Vector form:

$$\hat{\mathbf{y}} = X\theta$$

$$X = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{bmatrix}, \quad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \end{bmatrix}$$

$$\hat{\mathbf{y}} \in \mathbb{R}^n$$

Waiting for STEP 4.



#### STEP 4 — Objective / Loss Function

Step 4: Write the complete objective function/loss function, including likelihood form if applicable. No text—only formal math.

#### STEP 4 — Objective / Loss Function

Empirical risk (least squares):

$$J(\theta) = \frac{1}{n} \sum_{i=1}^n (y_i - (\theta_0 + \theta_1 x_i))^2$$

Equivalent norm form:

$$J(\theta) = \frac{1}{n} \|\mathbf{y} - X\theta\|_2^2$$

Assuming Gaussian noise:

$$y_i = \theta_0 + \theta_1 x_i + \varepsilon_i, \quad \varepsilon_i \sim \mathcal{N}(0, \sigma^2)$$

Likelihood:

$$p(\mathbf{y} \mid X, \theta, \sigma^2) = \prod_{i=1}^n \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_i - (\theta_0 + \theta_1 x_i))^2}{2\sigma^2}\right)$$

Log-likelihood:

$$\log p(\mathbf{y} | \mathbf{X}, \theta, \sigma^2) = -\frac{n}{2} \log(2\pi\sigma^2) - \frac{1}{2\sigma^2} \sum_{i=1}^n (y_i - (\theta_0 + \theta_1 x_i))^2$$

Maximum likelihood objective:

$$\theta^* = \arg \min_{\theta} \sum_{i=1}^n (y_i - (\theta_0 + \theta_1 x_i))^2$$



### STEP 5 — Full Derivation

Step 5:

Give me a complete mathematical derivation of the algorithm

Follow this exact structure strictly and provide \*only mathematical and technical content\* (no intuition, no examples, no story narration, no analogies).

Structure:

#### 1. **\*\*Problem Definition\*\***

\* State the inputs, outputs, assumptions, and objective function clearly using symbols only.

#### 2. **\*\*Mathematical Setup\*\***

\* Define all variables, parameters, and functions used.  
\* Convert the algorithm into equations and formal expressions.

#### 3. **\*\*Derivation of the Core Equations\*\***

\* Derive the hypothesis/model equation.  
\* Derive the cost/loss function (if applicable).  
\* Show all partial derivatives w.r.t parameters step-by-step.

#### 4. **\*\*Solving for Optimal Parameters\*\***

\* Solve the system of equations created by setting derivatives to zero.  
\* Derive closed-form solutions (if they exist).  
\* Otherwise, derive the update rules (e.g., gradient descent, SGD, EM, dynamic programming).

5. **Final Formula Summary**

- \* Present the final compact mathematical expressions.
- \* Show vector/matrix form if appropriate.

6. **Computational Form**

- \* Provide final implementable formulas (gradient update or closed form).

Use mathematician-level notation and keep every step explicit.

No intuition, no diagrams, no storytelling — only pure derivation.

## STEP 5 — Full Derivation

---

### 1. Problem Definition

$$\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n, \quad x_i \in \mathbb{R}, \quad y_i \in \mathbb{R}$$

$$\theta = (\theta_0, \theta_1) \in \mathbb{R}^2$$

$$h_\theta(x) = \theta_0 + \theta_1 x$$

$$J(\theta) = \frac{1}{n} \sum_{i=1}^n (y_i - h_\theta(x_i))^2$$

$$\theta^* = \arg \min_{\theta \in \mathbb{R}^2} J(\theta)$$

Assumption:

$$\sum_{i=1}^n (x_i - \bar{x})^2 > 0$$


---

### 2. Mathematical Setup

$$\hat{y}_i = \theta_0 + \theta_1 x_i$$

$$\varepsilon_i = y_i - \hat{y}_i$$

$$J(\theta_0, \theta_1) = \frac{1}{n} \sum_{i=1}^n \varepsilon_i^2 = \frac{1}{n} \sum_{i=1}^n (y_i - \theta_0 - \theta_1 x_i)^2$$

Design matrix:

$$X = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{bmatrix}, \quad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}$$

$$J(\theta) = \frac{1}{n} \|\mathbf{y} - X\theta\|_2^2$$

### 3. Derivation of the Core Equations

Objective expansion:

$$J(\theta_0, \theta_1) = \frac{1}{n} \sum_{i=1}^n (y_i^2 + \theta_0^2 + \theta_1^2 x_i^2 - 2y_i \theta_0 - 2y_i \theta_1 x_i + 2\theta_0 \theta_1 x_i)$$

Partial derivative w.r.t.  $\theta_0$ :

$$\frac{\partial J}{\partial \theta_0} = \frac{1}{n} \sum_{i=1}^n 2(\theta_0 + \theta_1 x_i - y_i)$$

$$\frac{\partial J}{\partial \theta_0} = \frac{2}{n} \left( n\theta_0 + \theta_1 \sum_{i=1}^n x_i - \sum_{i=1}^n y_i \right)$$

Partial derivative w.r.t.  $\theta_1$ :

$$\frac{\partial J}{\partial \theta_1} = \frac{1}{n} \sum_{i=1}^n 2(\theta_0 + \theta_1 x_i - y_i) x_i$$

$$\frac{\partial J}{\partial \theta_1} = \frac{2}{n} \left( \theta_0 \sum_{i=1}^n x_i + \theta_1 \sum_{i=1}^n x_i^2 - \sum_{i=1}^n x_i y_i \right)$$

### 4. Solving for Optimal Parameters

Set gradients to zero:

$$n\theta_0 + \theta_1 \sum_{i=1}^n x_i = \sum_{i=1}^n y_i \quad (1)$$



$$\theta_0 \sum_{i=1}^n x_i + \theta_1 \sum_{i=1}^n x_i^2 = \sum_{i=1}^n x_i y_i \quad (2)$$

Solve (1) for  $\theta_0$ :

$$\theta_0 = \bar{y} - \theta_1 \bar{x}$$

Substitute into (2):

$$\begin{aligned} (\bar{y} - \theta_1 \bar{x}) \sum_{i=1}^n x_i + \theta_1 \sum_{i=1}^n x_i^2 &= \sum_{i=1}^n x_i y_i \\ \theta_1 \left( \sum_{i=1}^n x_i^2 - n \bar{x}^2 \right) &= \sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y}) \\ \theta_1^* &= \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^n (x_i - \bar{x})^2} \\ \theta_0^* &= \bar{y} - \theta_1^* \bar{x} \end{aligned}$$


---

## 5. Final Formula Summary

Scalar form:

$$\begin{aligned} \theta_1^* &= \frac{\text{Cov}(x, y)}{\text{Var}(x)} \\ \theta_0^* &= \mathbb{E}[y] - \theta_1^* \mathbb{E}[x] \end{aligned}$$

Vector form:

$$\theta^* = (X^T X)^{-1} X^T \mathbf{y}$$


---

## 6. Computational Form

Closed-form solution:

$$\theta^* = (X^T X)^{-1} X^T \mathbf{y}$$

Gradient descent update (optional):

$$\theta^{(t+1)} = \theta^{(t)} - \eta \frac{2}{n} X^T (X \theta^{(t)} - \mathbf{y})$$

$$\theta \in \mathbb{R}^2$$

Waiting for **STEP 6**.



### STEP 6 — Solve for Optimal Parameters

Step 6: Solve the system of equations obtained from Step 5. If closed-form exists, derive it. If not, derive update rules mathematically.

### STEP 6 — Solve for Optimal Parameters

Normal equations:

$$\frac{\partial J}{\partial \theta} = 0 \iff X^T X \theta = X^T y$$

Assumption:

$$\text{rank}(X) = 2 \implies X^T X \text{ invertible}$$

Closed-form solution:

$$\theta^* = (X^T X)^{-1} X^T y$$

Expanded scalar system:

$$\begin{cases} n\theta_0 + \theta_1 \sum_{i=1}^n x_i = \sum_{i=1}^n y_i \\ \theta_0 \sum_{i=1}^n x_i + \theta_1 \sum_{i=1}^n x_i^2 = \sum_{i=1}^n x_i y_i \end{cases}$$

Solution:

$$\theta_1^* = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^n (x_i - \bar{x})^2}$$

$$\theta_0^* = \bar{y} - \theta_1^* \bar{x}$$

Matrix-expanded closed form:

$$(X^T X)^{-1} = \frac{1}{n \sum x_i^2 - (\sum x_i)^2} \begin{bmatrix} \sum x_i^2 & -\sum x_i \\ -\sum x_i & n \end{bmatrix}$$

$$\theta^* = \frac{1}{n \sum x_i^2 - (\sum x_i)^2} \begin{bmatrix} \sum x_i^2 & -\sum x_i \\ -\sum x_i & n \end{bmatrix} \begin{bmatrix} \sum y_i \\ \sum x_i y_i \end{bmatrix}$$

If  $X^T X$  is singular (fallback):

Gradient descent update:

$$\theta^{(t+1)} = \theta^{(t)} - \eta \frac{2}{n} X^T (X \theta^{(t)} - \mathbf{y})$$

$$\eta > 0$$

Waiting for **STEP 7**.



#### STEP 7 — Vector / Matrix Form

Step 7: Convert all results into compact vector/matrix form. Use linear algebra notation only.

#### STEP 7 — Vector / Matrix Form

$$X = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{bmatrix} \in \mathbb{R}^{n \times 2}$$

$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \end{bmatrix} \in \mathbb{R}^2$$

$$\mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \in \mathbb{R}^n$$

Hypothesis:

$$\hat{\mathbf{y}} = X\theta$$

Residual vector:

$$\mathbf{r} = \mathbf{y} - X\theta$$

Objective:

$$J(\theta) = \frac{1}{n} \mathbf{r}^\top \mathbf{r} = \frac{1}{n} \|\mathbf{y} - X\theta\|_2^2$$

Gradient:

$$\nabla_{\theta} J(\theta) = -\frac{2}{n} X^\top (\mathbf{y} - X\theta)$$

Normal equations:

$$X^\top X \theta = X^\top \mathbf{y}$$

Closed-form solution:

$$\theta^* = (X^\top X)^{-1} X^\top \mathbf{y}$$

Hessian:

$$\nabla_{\theta}^2 J(\theta) = \frac{2}{n} X^\top X$$

Gradient descent update:

$$\theta^{(t+1)} = \theta^{(t)} - \eta \nabla_{\theta} J(\theta^{(t)})$$

$$\theta^{(t+1)} = \theta^{(t)} + \frac{2\eta}{n} X^\top (\mathbf{y} - X\theta^{(t)})$$

Waiting for **STEP 8**.



### STEP 8 — Computational Complexity

Step 8: Provide full computational and memory complexity analysis using Big-O notation. Mathematical justification only.

### STEP 8 — Computational Complexity

Let

$$X \in \mathbb{R}^{n \times 2}, \quad \mathbf{y} \in \mathbb{R}^n$$

### Closed-Form (Normal Equation)

Matrix multiplication:

$$X^\top X : (2 \times n)(n \times 2) \Rightarrow \mathcal{O}(n \cdot 2 \cdot 2) = \mathcal{O}(n)$$

Matrix inversion:

$$(X^\top X)^{-1} : (2 \times 2) \Rightarrow \mathcal{O}(1)$$

Matrix-vector multiplication:

$$X^\top \mathbf{y} : (2 \times n)(n \times 1) \Rightarrow \mathcal{O}(n)$$

Final multiplication:

$$(2 \times 2)(2 \times 1) \Rightarrow \mathcal{O}(1)$$

Total time complexity:

$$\boxed{\mathcal{O}(n)}$$

Memory usage:

$$X : \mathcal{O}(n), \quad \mathbf{y} : \mathcal{O}(n), \quad \theta : \mathcal{O}(1)$$

Total space complexity:

$$\boxed{\mathcal{O}(n)}$$

### Gradient Descent (per iteration)

Matrix-vector multiplication:

$$X\theta : (n \times 2)(2 \times 1) \Rightarrow \mathcal{O}(n)$$

Residual computation:

$$\mathbf{y} - X\theta : \mathcal{O}(n)$$

Gradient computation:

$$X^\top (\mathbf{y} - X\theta) : \mathcal{O}(n)$$

Parameter update:

$$\theta^{(t+1)} = \theta^{(t)} - \eta \nabla J : \mathcal{O}(1)$$

Per-iteration time complexity:

$$\boxed{\mathcal{O}(n)}$$

Total time for  $T$  iterations:

$$\boxed{\mathcal{O}(nT)}$$

Memory usage (per iteration):

$$X : \mathcal{O}(n), \quad y : \mathcal{O}(n), \quad \theta : \mathcal{O}(1)$$

Total space complexity:

$$\boxed{\mathcal{O}(n)}$$

Waiting for **STEP 9**.



#### STEP 9 — Pseudocode

Step 9: Provide clean algorithm pseudocode using ML-standard notation. No explanations. Only algorithm steps.

#### STEP 9 — Pseudocode

makefile

Copy code

Algorithm: Simple Linear Regression using OLS

Input:  $X \in \mathbb{R}^{n \times 2}$ ,  $y \in \mathbb{R}^n$

Output:  $\theta \in \mathbb{R}^2$

1: Compute  $A \leftarrow X^T X$


2: Compute  $b \leftarrow X^T y$

```

3: Compute  $\theta \leftarrow A^{-1} b$ 
4: Return  $\theta$ 

```

makefile

 Copy code

Algorithm: Simple Linear Regression using Gradient Descent

Input:  $X \in \mathbb{R}^{n \times 2}$ ,  $y \in \mathbb{R}^n$ ,  $\eta > 0$ ,  $T \in \mathbb{N}$

Output:  $\theta \in \mathbb{R}^2$

```

1: Initialize  $\theta^{(0)} \in \mathbb{R}^2$ 
2: for  $t = 0$  to  $T-1$  do
3:    $g \leftarrow (2/n) X^T (X\theta^{(t)} - y)$ 
4:    $\theta^{(t+1)} \leftarrow \theta^{(t)} - \eta g$ 
5: end for
6: Return  $\theta^{(T)}$ 

```



STEP 10 — From-Scratch Implementation (Python/Numpy)  
 Step 10: Provide a full from-scratch implementation using only Python and NumPy. No comments, no text. Code only.

python

 Copy code

```

import numpy as np

def simple_linear_regression_ols(x, y):
    x = np.asarray(x).reshape(-1)
    y = np.asarray(y).reshape(-1)
    n = x.shape[0]
    X = np.column_stack((np.ones(n), x))
    XtX = X.T @ X
    Xty = X.T @ y
    theta = np.linalg.inv(XtX) @ Xty
    return theta

def simple_linear_regression_gd(x, y, lr=0.01, epochs=1000):
    x = np.asarray(x).reshape(-1)

```

```

y = np.asarray(y).reshape(-1)
n = x.shape[0]
X = np.column_stack((np.ones(n), x))
theta = np.zeros(2)
for _ in range(epochs):
    grad = (2/n) * X.T @ (X @ theta - y)
    theta = theta - lr * grad
return theta

```



### STEP 11 — Experimental Setup

Step 11: Describe the experimental protocol for testing the algorithm. Only mathematical/statistical description—no English explanations.

### STEP 11 — Experimental Setup

$$\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$$

Random partition:

$$\mathcal{D}_{\text{train}} \cup \mathcal{D}_{\text{test}} = \mathcal{D}, \quad \mathcal{D}_{\text{train}} \cap \mathcal{D}_{\text{test}} = \emptyset$$

$$|\mathcal{D}_{\text{train}}| = \lfloor \alpha n \rfloor, \quad |\mathcal{D}_{\text{test}}| = n - \lfloor \alpha n \rfloor, \quad \alpha \in (0, 1)$$

Parameter estimation:

$$\theta^* = \arg \min_{\theta} \frac{1}{|\mathcal{D}_{\text{train}}|} \sum_{(x_i, y_i) \in \mathcal{D}_{\text{train}}} (y_i - \theta_0 - \theta_1 x_i)^2$$

Prediction:

$$\hat{y}_j = \theta_0^* + \theta_1^* x_j, \quad (x_j, y_j) \in \mathcal{D}_{\text{test}}$$

Test loss:

$$\mathcal{L}_{\text{test}} = \frac{1}{|\mathcal{D}_{\text{test}}|} \sum_{(x_j, y_j) \in \mathcal{D}_{\text{test}}} (y_j - \hat{y}_j)^2$$

Cross-validation:



$$\mathcal{D} = \bigcup_{k=1}^K \mathcal{D}_k, \quad \mathcal{D}_k \cap \mathcal{D}_l = \emptyset$$

$$\theta_k^* = \arg \min_{\theta} \sum_{i \notin \mathcal{D}_k} (y_i - \theta_0 - \theta_1 x_i)^2$$

$$\mathcal{L}_{\text{CV}} = \frac{1}{K} \sum_{k=1}^K \frac{1}{|\mathcal{D}_k|} \sum_{(x_i, y_i) \in \mathcal{D}_k} (y_i - \theta_{0,k}^* - \theta_{1,k}^* x_i)^2$$

Statistical estimator:

$$\hat{\sigma}^2 = \frac{1}{n-2} \sum_{i=1}^n (y_i - \theta_0^* - \theta_1^* x_i)^2$$

Confidence interval:

$$\theta_1^* \pm t_{n-2, 1-\frac{\alpha}{2}} \sqrt{\frac{\hat{\sigma}^2}{\sum_{i=1}^n (x_i - \bar{x})^2}}$$

Waiting for STEP 12.



### STEP 12 — Connections to Other Algorithms

Step 12: Provide purely mathematical relationships between this algorithm and closely related algorithms. No intuition or narrative.

### STEP 12 — Connections to Other Algorithms

#### Linear Regression ↔ Maximum Likelihood Estimation (MLE)

Assumption:

$$\varepsilon_i \sim \mathcal{N}(0, \sigma^2)$$

Likelihood:

$$p(\mathbf{y} \mid X, \theta, \sigma^2) = \prod_{i=1}^n \mathcal{N}(y_i \mid x_i^\top \theta, \sigma^2)$$

MLE:

$$\theta_{\text{MLE}} = \arg \max_{\theta} \log p(\mathbf{y} \mid X, \theta, \sigma^2) = \arg \min_{\theta} \|\mathbf{y} - X\theta\|_2^2$$

$$\Rightarrow \theta_{\text{MLE}} = \theta_{\text{OLS}}$$


---

## Linear Regression $\leftrightarrow$ MAP Estimation (Ridge Regression)

Prior:

$$\theta \sim \mathcal{N}(0, \lambda^{-1} I)$$

MAP:

$$\theta_{\text{MAP}} = \arg \min_{\theta} (\|\mathbf{y} - X\theta\|_2^2 + \lambda \|\theta\|_2^2)$$

Closed form:

$$\theta_{\text{MAP}} = (X^\top X + \lambda I)^{-1} X^\top \mathbf{y}$$

$$\lim_{\lambda \rightarrow 0} \theta_{\text{MAP}} = \theta_{\text{OLS}}$$


---

## Linear Regression $\leftrightarrow$ Weighted Least Squares (WLS)

Objective:

$$\min_{\theta} (\mathbf{y} - X\theta)^\top W (\mathbf{y} - X\theta)$$

Solution:

$$\theta_{\text{WLS}} = (X^\top W X)^{-1} X^\top W \mathbf{y}$$

Special case:

$$W = I \Rightarrow \theta_{\text{WLS}} = \theta_{\text{OLS}}$$


---

## Linear Regression $\leftrightarrow$ General Linear Model (GLM)

Model:

$$\mathbb{E}[\mathbf{y} \mid \mathbf{X}] = \mathbf{X}\theta$$

Identity link:

$$g(\mu) = \mu$$

Gaussian family:

$$\Rightarrow \text{GLM} \equiv \text{OLS}$$

---

## Linear Regression $\leftrightarrow$ Projection Theory

Column space:

$$\mathcal{C}(\mathbf{X}) \subset \mathbb{R}^n$$

Projection matrix:

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

Prediction:

$$\hat{\mathbf{y}} = \mathbf{P}\mathbf{y}$$

Residual:

$$\mathbf{r} = (\mathbf{I} - \mathbf{P})\mathbf{y}$$

Orthogonality:

$$\mathbf{X}^\top \mathbf{r} = \mathbf{0}$$

---

## Linear Regression $\leftrightarrow$ Gradient Descent

Objective:

$$J(\theta) = \frac{1}{n} \|\mathbf{y} - \mathbf{X}\theta\|_2^2$$

Gradient flow:

$$\frac{d\theta}{dt} = -\nabla_{\theta} J(\theta)$$

Fixed point:

$$\nabla_{\theta} J(\theta) = 0 \iff X^{\top} X \theta = X^{\top} \mathbf{y}$$


---

## Linear Regression $\leftrightarrow$ Principal Component Regression (PCR)

SVD:

$$X = U \Sigma V^{\top}$$

OLS:

$$\theta_{\text{OLS}} = V \Sigma^{-1} U^{\top} \mathbf{y}$$

PCR (rank- $k$ ):

$$\theta_{\text{PCR}} = V_k \Sigma_k^{-1} U_k^{\top} \mathbf{y}$$


---

## Linear Regression $\leftrightarrow$ Moore–Penrose Pseudoinverse

Pseudoinverse:

$$X^+ = (X^{\top} X)^{-1} X^{\top}$$

Solution:

$$\theta^* = X^+ \mathbf{y}$$


---

## Linear Regression $\leftrightarrow$ Polynomial Regression

Feature map:

$$\phi(x) = (1, x, x^2, \dots, x^d)$$

Design matrix:

$$\Phi = [\phi(x_1)^{\top}; \dots; \phi(x_n)^{\top}]$$

Solution:

$$\theta = (\Phi^{\top} \Phi)^{-1} \Phi^{\top} \mathbf{y}$$


---

## Linear Regression $\leftrightarrow$ Kernel Ridge Regression

Kernel:

$$K = XX^\top$$

Dual solution:

$$\alpha = (K + \lambda I)^{-1} \mathbf{y}$$

Primal relation:

$$\theta = X^\top \alpha$$

---

END OF 12-STEP ALGORITHM MASTERY SEQUENCE

