📌 Problem Setup

We are given:

- A dataset $D = \{(x_i, t_i)\}$, where x_i is input, and t_i is the target output (ground truth).
- Goal: Fit a polynomial function $y(x, \mathbf{w})$ such that it approximates the data.

🖍️ Polynomial Function

For a polynomial of degree M, the function is:

$$y(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \cdots + w_M x^M = \sum_{j=0}^{M} w_j x^j$$

Where:

- $\mathbf{w} = [w_0, w_1, ..., w_M]$ is the vector of coefficients we want to learn.

🔢 Short Example (Polynomial of degree 2)

Suppose we have 3 training data points:

$$D = \{(1, 2), (2, 3), (3, 5)\}$$

We want to fit a **quadratic polynomial** $y(x) = w_0 + w_1 x + w_2 x^2$

✅ Step 1: Build Design Matrix Φ

Each row corresponds to x_i, and columns are x_i^0, x_i^1, x_i^2:

$$\Phi = \begin{bmatrix} 1 & 1 & 1^2 \\ 1 & 2 & 2^2 \\ 1 & 3 & 3^2 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 4 \\ 1 & 3 & 9 \end{bmatrix}$$

Target vector $\mathbf{t} = \begin{bmatrix} 2 \\ 3 \\ 5 \end{bmatrix}$

✅ Step 2: Use Least Squares to Minimize Error

We minimize the error function:

$$E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}(y(x_n, \mathbf{w}) - t_n)^2$$

The closed-form solution (normal equation) is:

$$\mathbf{w} = (\Phi^T\Phi)^{-1}\Phi^T\mathbf{t}$$

Let's compute it:

- $\Phi^T\Phi = \begin{bmatrix} 3 & 6 & 14 \\ 6 & 14 & 36 \\ 14 & 36 & 98 \end{bmatrix}$
- $\Phi^T\mathbf{t} = \begin{bmatrix} 10 \\ 23 \\ 61 \end{bmatrix}$

Now compute:

$$\mathbf{w} = \left(\begin{bmatrix} 3 & 6 & 14 \\ 6 & 14 & 36 \\ 14 & 36 & 98 \end{bmatrix}\right)^{-1} \cdot \begin{bmatrix} 10 \\ 23 \\ 61 \end{bmatrix}$$

Solving this system (using software or calculator), we get:

$$\mathbf{w} = [1, 0.5, 0.5]$$

✅ Final Polynomial

$$y(x) = 1 + 0.5x + 0.5x^2$$

This curve approximately fits the given data points.

📌 Key Concept

We **learn the weights** $\mathbf{w}$ by minimizing the **sum of squared errors** between predicted outputs and target values.

Would you like me to show the same example in code (like Python/NumPy or sklearn)?