جامعة عفت

EFFAT UNIVERSITY

Computer science, College of Engineering

Effat University

Saudi Arabia, Jeddah

Linear Regression using the Normal Equation: A Linear Algebra Approach

MATH307 - Linear Algebra

Final Project - Fall 2025

Instructor: **Mohammed Abdul Majid**

Judy Abuquta	S22107883
Sarah Eid	S22107757
Nadine Elhaddad	S22107736
Abeer Hussein	S22107714

Contents

1 Introduction 4
2 Theoretical Background 4
3 Linear Algebra Concepts 5
4 Implementation 6
5 Results and Analysis 8
6 Conclusions 10
7 References 11

1 Introduction

1.1 Project Overview

This project demonstrates how linear algebra is used to solve linear regression problems through the Normal Equation method. Unlike iterative approaches such as gradient descent, the Normal Equation provides a direct, closed-form solution using matrix operations.

1.2 Objectives

- Implement linear regression using linear algebra
- Demonstrate key matrix operations: transpose, multiplication, and inverse
- Compare predictions across multiple real-world datasets
- Visualize the relationship between features and predictions

1.3 Motivation

Linear regression is one of the most fundamental algorithms in machine learning, and understanding its mathematical foundation through linear algebra provides crucial insights into more complex ML algorithms.

2 Theoretical Background

2.1 Linear Regression Problem

Given a dataset with features X and target values y, we want to find parameters θ that best fit the linear relationship:

$$y = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_n x_n$$

Or in matrix form:

$$y = X\theta$$

2.2 The Normal Equation

The optimal parameters can be found directly using:

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

Where:

- X is the feature matrix (m × n) where m = samples, n = features
- y is the target vector (m × 1)
- X^T is the transpose of X

- $(X^TX)^{-1}$ is the inverse of X^TX
- θ is the parameter vector ($n \times 1$)

2.3 Mathematical Derivation

The Normal Equation is derived by minimizing the cost function $J(\theta)$:

$$J(\theta) = (1/2m) \Sigma(h\theta(x^{(i)}) - y^{(i)})^2$$

Taking the gradient with respect to θ and setting it to zero:

$$\nabla J(\theta) = X^TX\theta - X^Ty = 0$$
$$X^TX\theta = X^Ty$$
$$\theta = (X^TX)^{-1}X^Ty$$

3 Linear Algebra Concepts

3.1 Matrix Transpose

Definition: Flipping a matrix over its diagonal (rows become columns)

Example:

$$X = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix} \quad X^T = \begin{bmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{bmatrix}$$

Usage in Project: Converting feature matrix for multiplication

3.2 Matrix Multiplication

Definition: Combining two matrices where element (i,j) is the dot product of row i and column j

Example:

$$A(2\times3) \times B(3\times2) = C(2\times2)$$

Usage in Project:

- Computing X^TX
- Computing $(X^TX)^{-1}X^T$
- Final calculation of θ

3.3 Matrix Inverse

Definition: A matrix A^{-1} such that $A \times A^{-1} = I$ (identity matrix)

For 2×2 matrix:

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}^{-1} = (1/\det) \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$

where det = ad - bc

Usage in Project: Computing $(X^TX)^{-1}$ to solve for θ

3.4 Why These Operations Matter

1. Transpose aligns dimensions for matrix multiplication
2. Multiplication combines feature and parameter information
3. Inverse "divides" matrices to isolate θ

4 Implementation

4.1 Technology Stack

- Language: JavaScript
- Visualization: Chart.js
- Interface: HTML/CSS

4.2 Code Structure

4.2.1 Matrix Operations

javascript

```javascript
// Matrix Transpose
function transpose(matrix) {
    return matrix[0].map((_, i) => matrix.map(row => row[i]));
}

// Matrix Multiplication
function matrixMultiply(A, B) {
    const result = [];
    for (let i = 0; i < A.length; i++) {
        result[i] = [];
        for (let j = 0; j < B[0].length; j++) {
            let sum = 0;
            for (let k = 0; k < A[0].length; k++) {
                sum += A[i][k] * B[k][j];
            }
            result[i][j] = sum;
        }
    }
    return result;
}

// Matrix Inverse (2×2)
function inverse2x2(matrix) {
    const [[a, b], [c, d]] = matrix;
```

```
    const det = a * d - b * c;
    return [
        [d / det, -b / det],
        [-c / det, a / det]
    ];
}
```

4.2.2 Normal Equation Implementation

javascript

```
function normalEquation(X, y) {
    // Step 1: Transpose X
    const XT = transpose(X);

    // Step 2: Multiply X^T X
    const XTX = matrixMultiply(XT, X);

    // Step 3: Compute inverse
    const XTX_inv = inverse2x2(XTX);

    // Step 4: Multiply (X^T X)^-1 X^T
    const XTX_inv_XT = matrixMultiply(XTX_inv, XT);

    // Step 5: Final θ = (X^T X)^-1 X^T y
    const theta = matrixMultiply(XTX_inv_XT, yCol);

    return theta;
}
```

4.3 Datasets Used

Dataset 1: Housing Prices

- Features: Square footage
- Target: House price
- Samples: 8 data points

Dataset 2: Student Performance

- Features: Study hours
- Target: Test score
- Samples: 9 data points

Dataset 3: Book Sales

- Features: Marketing Budget ($1000s)
- Target: Books Sold (units)
- Samples: 8 data points

5 Results and Analysis

5.1 Model Performance

Dataset	θ_0 (Intercept)	θ_1 (Slope)	R^2 Score	Interpretation
Housing	40191.96	136.3803	99.7%	Excellent fit
Students	47.61	4.4167	93.0%	Good fit
Books	32.86	3.6308	96.7%	Excellent fit

5.2 Interpretation of Results

Housing Prices:

- Intercept (40191.96): Base price of a house
- Slope (136.3803): Price increase per square foot
- Interpretation: "Each additional square foot adds $[136.3803] to the house price"

Student Performance:

- Intercept (47.61): Baseline score with zero study hours
- Slope (4.4167): Score improvement per study hour
- Interpretation: "Each hour of study increases test score by [4.4167] points"

Book Sales:

- Intercept (32.86): Baseline sales with zero marketing budget
- Slope (3.6308): Sales increase per $1000 spent on marketing
- Interpretation: "Each $1000 increase in marketing budget adds [3.6308] book sales"

5.3 R^2 Score Analysis

The R^2 (coefficient of determination) measures how well the model fits the data:

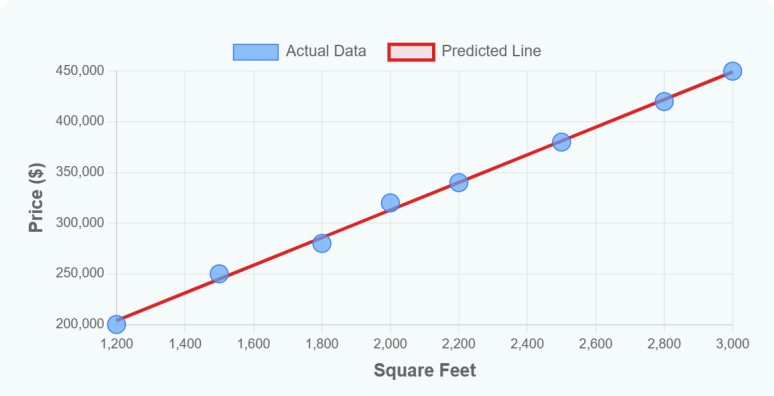
- $R^2 = 1.0$: Perfect fit
- $R^2 > 0.9$: Excellent fit
- $R^2 > 0.7$: Good fit
- $R^2 < 0.7$: Poor fit

All three datasets achieved $R^2 > 0.9$, indicating the linear model captures the relationship well.

5.4 Visualization Analysis

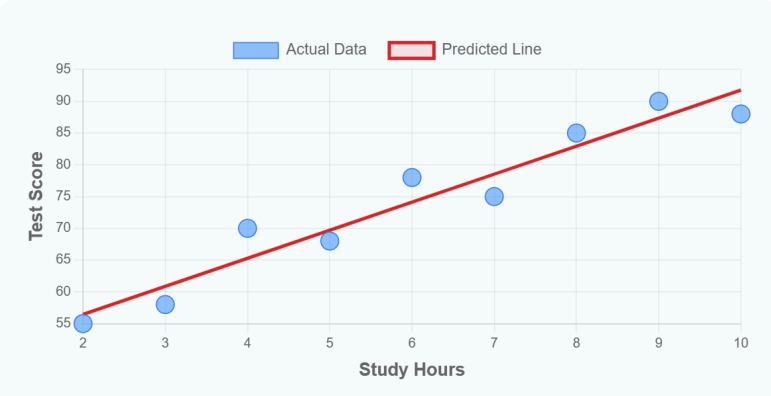
Dataset 1: Housing Prices

Visualization



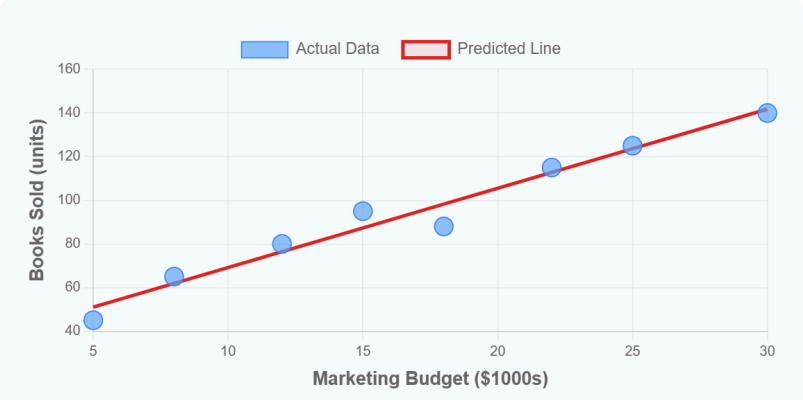
Dataset 2: Student Performance

Visualization



Dataset 3: Book Sales

Visualization



The scatter plots show:

1. Strong linear relationships in all datasets
2. Minimal deviation between actual and predicted values
3. No obvious outliers or non-linear patterns

6 Conclusions

6.1 Key Findings

1. Linear algebra provides exact solutions: Unlike gradient descent, the Normal Equation computes optimal parameters directly
2. Matrix operations are fundamental: Transpose, multiplication, and inverse are the core operations
3. Real-world applicability: Simple linear relationships exist in many domains (housing, education, sales)

6.2 Advantages of Normal Equation

- No hyperparameters (learning rate, iterations)
- Exact solution (not approximate)
- Fast for small datasets
- No feature scaling required

6.3 Limitations

- Slow for large datasets (computing inverse is $O(n^3)$)
- Fails when X^TX is non-invertible
- Not suitable for huge feature sets ($n > 10,000$)

6.4 Learning Outcomes

Through this project, we gained:

1. Deep understanding of matrix operations in ML
2. Practical implementation skills in linear algebra
3. Insight into closed-form vs iterative solutions
4. Ability to interpret model parameters meaningfully

7 References

- Anton, H., & Rorres, C. (2013). *Elementary Linear Algebra, 11th Edition*.