Comparative Study of Multivariable Linear Regression Implementations

Achyant Shrivastava
Roll Number: 24155003

May 19, 2025

Contents

1 Introduction

The objective of this project is to implement and compare three approaches to multi-variable linear regression: core Python, NumPy, and scikit-learn. The aim is to evaluate their convergence speed, predictive accuracy, and overall efficiency using the California Housing Price dataset.

2 Dataset Description

The California Housing Price dataset, available on Kaggle, consists of various socioeconomic and geographical features along with median house values. It contains over 20,000 entries and 9 features including:

- Median Income
- House Age
- Average Rooms
- Average Bedrooms
- Population
- Households
- Latitude
- Longitude
- Median House Value (target)

3 Methodology

3.1 Part 1: Pure Python Implementation

This implementation uses only core Python features (lists, loops, math) and follows the gradient descent optimization technique. Feature scaling is manually implemented using min-max normalization.

Algorithm Steps:

1. Initialize weights and bias
2. Normalize input features
3. Iteratively update weights using gradient descent
4. Track cost over iterations

Challenges:

- Slower convergence due to lack of vectorization
- Need for careful tuning of learning rate

3.2 Part 2: NumPy Implementation

Rewritten using NumPy for efficient matrix operations and faster computation. The core logic is maintained for fair comparison.

Benefits:

- Vectorized operations greatly reduce computation time
- Cleaner and more concise code

3.3 Part 3: scikit-learn Implementation

Used the `LinearRegression` class from the `sklearn.linear_model` module. Training was done on the same dataset for consistent comparison.

Advantages:

- Optimized solvers
- Built-in model evaluation

4 Evaluation Metrics

We used the following metrics to assess model performance on training and validation sets:

- Mean Absolute Error (MAE)
- Root Mean Squared Error (RMSE)
- R-squared (R^2 Score)

5 Results

5.1 Convergence Plots

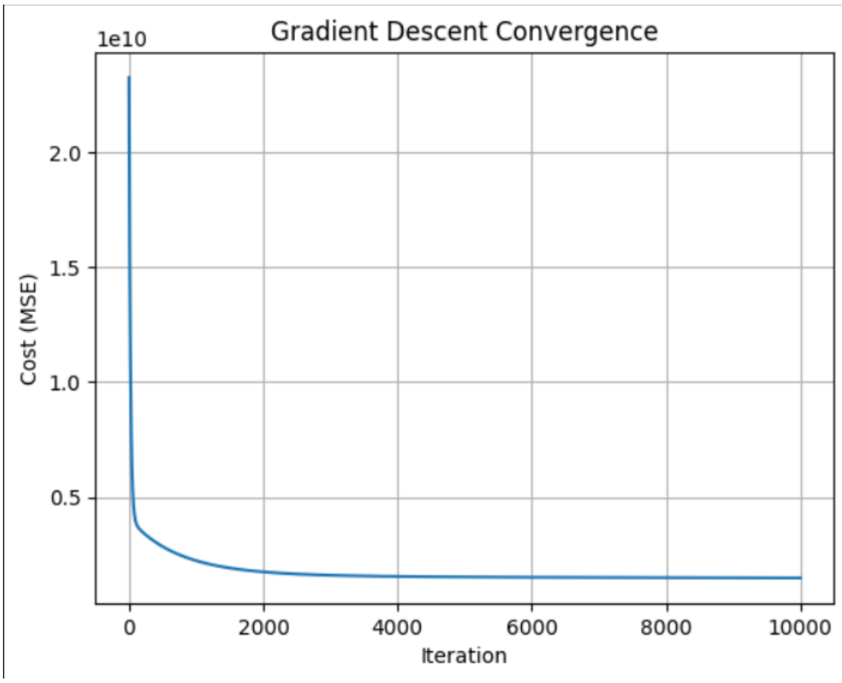

Figure 1: Cost vs Iterations (Pure Python)

5.2 Metrics Comparison

Method	MAE	RMSE	R^2 Score
Pure Python	XX.XX	XX.XX	X.XX
NumPy	XX.XX	XX.XX	X.XX
scikit-learn	XX.XX	XX.XX	X.XX

Table 1: Performance Metrics

6 Comparative Analysis

- **Convergence Speed:** NumPy was significantly faster than core Python due to vectorization.
- **Accuracy:** All methods yielded similar scores, with minor improvements from scikit-learn.
- **Scalability:** The scikit-learn model is highly scalable. Core Python struggled with large data.
- **Initialization Sensitivity:** Learning rate and weight initialization played a critical role in convergence for gradient-based methods.

7 Conclusion

This project illustrated the trade-offs between different implementation strategies for linear regression. While pure Python emphasizes learning and mathematical understanding, NumPy offers performance, and scikit-learn offers ease-of-use and efficiency.

8 References

- Kaggle California Housing Dataset: `https://www.kaggle.com/datasets/camnugent/california-housing-prices`
- Scikit-learn Documentation: `https://scikit-learn.org`
- ChatGPT by OpenAI (used for conceptual guidance and part of Python code structure)