Cryptonite Task Phase - Module 1

Parth Agrawal

October 2025

1 Introduction

In this module of the task phase, i have analyzed, learned and implemented two foundational machine learning algorithms. Linear Regression for continuous predictions and logistic regression for binary classification.

I have implemented both the models from scratch, implementing each mathematical function and also implemented them using scikit learn.

The two datasets used were -

- Boston Housing Dataset (Regression)
- Titanic Survival Dataset (Classification)

2 Linear Regression

Dataset Used - *Boston Housing Dataset* The goal was to predict median values of owner-occupied homes (MEDV) using various features from the Boston Housing Dataset.

2.1 Scratch Implementation

In our implementation of the linear regression model, we assume a straight line fit through the data as -

$$f(x) = w_0 + w_1x_1 + w_2x_2 + \ldots + w_nx_n = w^Tx$$

2.1.1 Mean Squared Error Loss Function

We used the mean squared error as our loss function.

$$J(f(x)) = \frac{1}{2m}\sum_{i=1}^{m}(f(x^{(i)}) - y^{(i)})^2$$

Metric	Scratch Result Implementation
Final Cost on Training Set	11.81
R - squared Score on Test Set	0.719

Table 1: Scratch Implementation Results

Metric	Result
R-squared on test set	0.615

Table 2: Scikit Learn Implementation Results

2.1.2 Gradient Descent

In Gradient Descent, we update the values of w and b to slowly achieve a minimum of the cost function.

We do this by moving toward the direction where the slope (derivative) of the function is steepest.

$$w_j := w_j - \alpha \frac{\partial}{\partial w_j} J(w, b)$$

$$b_j := b_j - \alpha \frac{\partial}{\partial b_j} J(w, b)$$

2.2 Training and Evaluation (Scratch)

After training on 500 epochs with a learning rate, $\alpha = 0.1$ we obtain -

2.3 Scikit Learn Comparison

The same dataset was modeled using `sklearn.linear_model.LinearRegression`. We obtain the following result -

Observation - Implementation from Scratch performed better than the Scikit Learn Implementation, thus we can say we have a perfect implementation.

3 Logistic Regression

Dataset Used - *Titanic Survival Dataset* The goal was to predict a binary outcome (survival: 1 or no-survival: 0) of passengers of the Titanic using features like passenger class, age, and sex.

3.1 3.1 Scratch Implementation Details

3.1.1 Hypothesis Function

The Sigmoid function (or Logistic function) was implemented to transform the linear output (z) into a probability score between 0 and 1.

$\sigma(z) = \frac{1}{1+e^{-z}}$

3.1.2 Loss Function (Binary Cross-Entropy Loss)

This loss function is appropriate for binary classification tasks, penalizing models heavily for confident but incorrect probability estimates.

$J(w, b) = -\frac{1}{m} \sum_{i=1}^{m} \left[y^{(i)} \log(f_x(x^{(i)})) + (1 - y^{(i)}) \log(1 - f_x(x^{(i)})) \right]$

3.1.3 Optimization (Gradient Descent)

The weight update rule remains the same as Linear Regression, but the gradient calculation is derived from the Cross-Entropy loss function.

3.2 Training and Evaluation (Scratch)

The custom Logistic Regression model was trained on the processed Titanic data (including one-hot encoding for categorical features).

Accuracy Obtained - 0.572

3.3 Scikit Learn Comparison

The `sklearn.linear_model.LogisticRegression` was applied to the same dataset.

Accuracy Obtained - 0.829

Observations - Scikit Learn outperforms the implementation from scratch by a wide margin. Thus, it is much better in this case.

4 Conclusion

The implementation of linear regression and Logistic Regression from scratch has successfully demonstrated their applications and underlying mathematics, including the Loss Function and Gradient Descent.

Comparative analysis with scikit-learn suggests that while in some cases we may achieve better performance, for complex tasks and faster implementation, we should use the pre-built models from the library.