Bülent Emin Üstün

27822

27/04/2024

Homework Report

1. Logistic Regression Implementation

Logistic Regression is an important algorithm used in machine learning for binary classification tasks. It models the probability that a given input belongs to a particular class. I used NumPy and Python to implement Logistic Regression from scratch.

1.1. Random Weight Initialization

I started by randomly assigning weights using NumPy's np.random.rand() function. These weights are highly important because they will be adjusted during the training to minimize the loss/cost function.

1.2. Sigmoid Function

The sigmoid is the heart of the logistic regression. It takes any real number and maps it to a probability between 0 and 1. (Educative, n.d.) I used the following formula to write the function.

$$f(x) = \frac{1}{1+e^{-x}}$$

1.3. Cost Function

The cost/loss function evaluates how well one's algorithm models the dataset. In other words, it evaluates the error between actual labels and predicted values. I wrote the cost function by the given formula below:

$$J(w) = -\frac{1}{m}\sum_{i=1}^{m}[y^{(i)}\log(\sigma(w^T x^{(i)})) + (1 - y^{(i)})\log(1 - \sigma(w^T x^{(i)}))].$$

1.4. Gradient Descent

Gradient Descent is an optimization algorithm for finding a local minimum of a differentiable function. Gradient descent in machine learning is simply used to find the values of a function's parameters (coefficients) that minimize a cost function as far as possible. (Donges, 2023)

2. Training The Model

In this part, I implemented the functions that I explained above. Without any gradient descent, my loss function was 0.6413501717530481. Then, implementing gradient descent using step size to 0.1 and iteration to 100, loss function was 0.5534924737658858. This result was important because it shows us that with gradient descent, we can minimize the loss function. The graph shows the correlation between iteration number and loss function value:

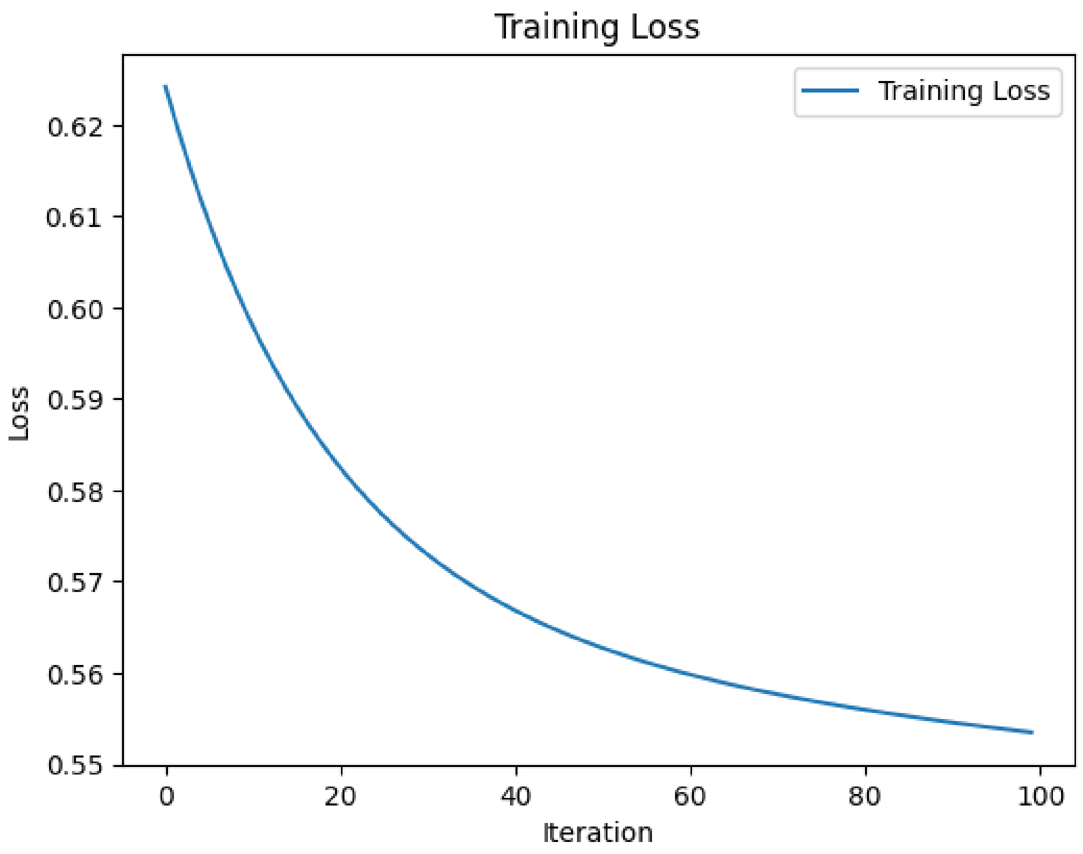

Figure 1: Training Loss

As you can see from the graph, loss function decreases which is what we are aiming for. Also, we can infer that the model is learning and improving its predictions over time. In the first part, model learns speedily, but then it is converging to a number. Moreover, the loss on the validation data can be below:

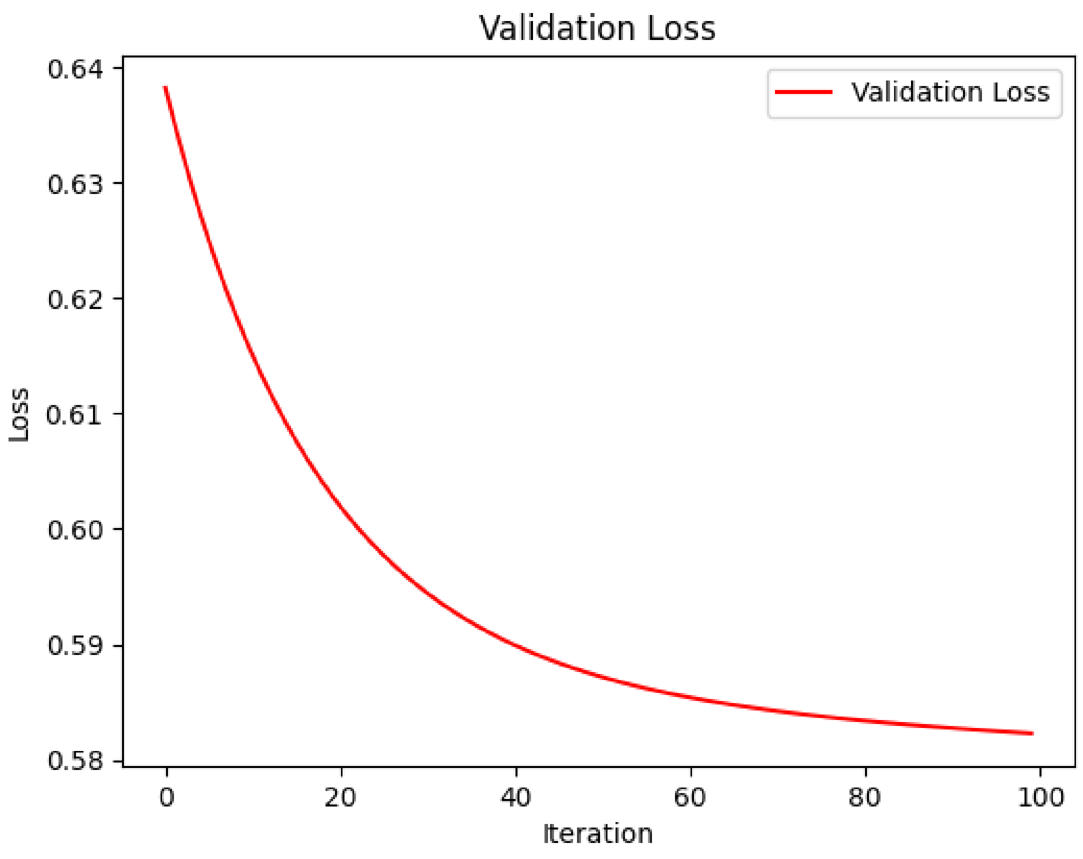

Figure 2: Validation Loss

After that, I vary the step sizes and number of iterations like this:

```
    step_size_array = [0.01, 0.1, 1, 10, 100, 1000]

    iteration_array = [100, 500, 1000, 10000]
```

I collected the loss function results which I found that the minimum loss function was given by the step size = 0.01, iteration = 10000. The validation loss on these hyperparameters is 0.5767. The corresponding graph can be found below:

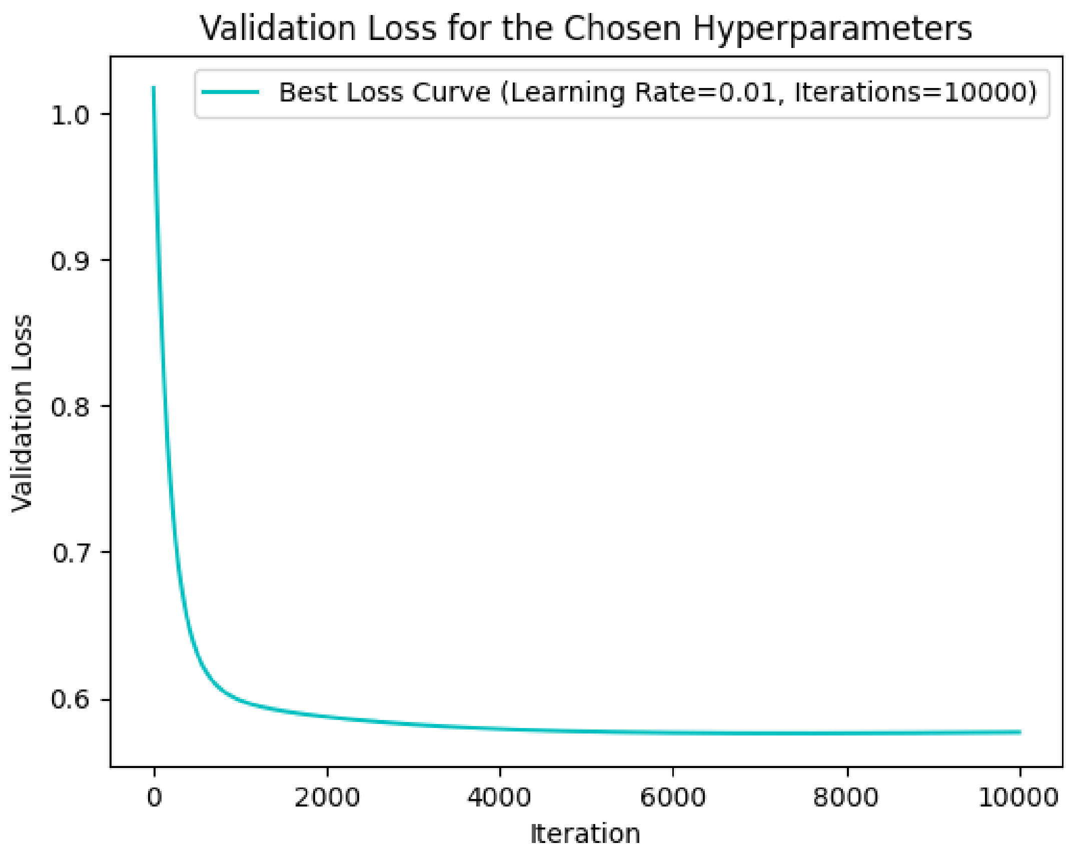

Figure 2: Validation Loss on the Best Hyperparameters

This is a quite good graph which shows that model is learning rapidly, its errors between predicted and actual labels are decreasing which is our goal.

After I selected the best hyperparameters, I combined the training and validation data and trained the model using these hyperparameters. The accuracy of the model is 0.7877.

References:

Donges, N. (2023, March 27). *Gradient Descent in Machine Learning: A Basic Introduction*. Built In. https://builtin.com/data-science/gradient-descent

Educative. (n.d.). *What is sigmoid and its role in logistic regression?* https://www.educative.io/answers/what-is-sigmoid-and-its-role-in-logistic-regression