Step-by-Step Math Behind the Single Neuron Classifier

1 Introduction

We are given a binary classification problem where we predict the wine color (red or white) based on several features. A single neuron classifier, which applies logistic regression, will be used to solve the problem.

The dataset contains features $\mathbf{X}$ and binary labels y:

$$\mathbf{X} \in \mathbb{R}^{m \times n}, \quad y \in \{0, 1\}^m$$

where:

m is the number of samples and n is the number of features.

The target variable $y = 1$ represents red wine, and $y = 0$ represents white wine.

2 Model Definition

The classifier is a linear model followed by a non-linear activation function (sigmoid). The linear model is defined as:

$$z = \mathbf{w}^T \mathbf{x} + w_0$$

where:

- $\mathbf{w} \in \mathbb{R}^n$ are the weights of the model.
- w_0 is the bias term.
- $\mathbf{x} \in \mathbb{R}^n$ is a single sample from the dataset.

The sigmoid activation function is applied to the linear combination:

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

The output of the model is the predicted probability:

$$\hat{y} = \sigma(z) = \frac{1}{1 + e^{-(\mathbf{w}^T \mathbf{x} + w_0)}}$$

This value represents the probability that the sample belongs to class 1 (red wine).

3 Loss Function

The loss function used is binary cross-entropy (also called log loss). For a single sample, the loss is defined as:

$$\mathcal{L}(\hat{y}, y) = -\left(y \log(\hat{y}) + (1 - y) \log(1 - \hat{y})\right)$$

This loss function is minimized when the predicted probability $\hat{y}$ is close to the true label y. The total loss over the dataset is the sum of individual losses:

$$\text{Total Loss} = \sum_{i=1}^{m} \mathcal{L}(\hat{y}_i, y_i)$$

4 Gradient Descent

To minimize the loss function, we use gradient descent. The parameters $\mathbf{w}$ and w_0 are updated using the gradients of the loss with respect to each parameter.

4.1 Gradient with Respect to the Bias Term

The gradient of the loss with respect to the bias term w_0 is:

$$\frac{\partial \mathcal{L}}{\partial w_0} = \hat{y} - y$$

The update rule for the bias term is:

$$w_0 \leftarrow w_0 - \alpha \frac{\partial \mathcal{L}}{\partial w_0}$$

where α is the learning rate.

4.2 Gradient with Respect to the Weights

The gradient of the loss with respect to the weights $\mathbf{w}$ is:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = (\hat{y} - y)\mathbf{x}$$

The update rule for the weights is:

$$\mathbf{w} \leftarrow \mathbf{w} - \alpha \frac{\partial \mathcal{L}}{\partial \mathbf{w}} = \mathbf{w} - \alpha(\hat{y} - y)\mathbf{x}$$

5 Training Procedure

The training procedure follows these steps:

1. Initialize the weights $\mathbf{w}$ and bias w_0 to small random values.

2. For each epoch:
   (a) For each sample $(\mathbf{x}_i, y_i)$:
      - Compute the linear combination $z_i = \mathbf{w}^T \mathbf{x}_i + w_0$.
      - Apply the sigmoid function to get the predicted probability $\hat{y}_i = \sigma(z_i)$.
      - Compute the binary cross-entropy loss.
      - Compute the gradients with respect to $\mathbf{w}$ and w_0.
      - Update the weights and bias using gradient descent.

   (b) Record the total loss for the epoch.

6 Classifier Evaluation

After training, we classify a sample $\mathbf{x}$ using the following rule:

$$\text{Classify}(\mathbf{x}) = \begin{cases} 1 & \text{if } \hat{y} \geq 0.5 \\ 0 & \text{otherwise} \end{cases}$$

The accuracy of the classifier is computed by comparing the predicted labels with the true labels on the test set:

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}} \times 100$$

7 Conclusion

In this problem, we used a single neuron classifier (logistic regression) to predict wine color based on several features. The model was trained using binary cross-entropy loss and gradient descent. After training, the model's performance was evaluated on a test set using accuracy as the metric.