Binary Classification using Custom Deep Neural Network with Class Imbalance Handling

Your Name

June 3, 2025

1 Introduction

This report presents the development and training of a custom deep neural network in NumPy for a binary classification task, with a strong focus on handling class imbalance using weighted loss updates. The network is evaluated using metrics such as accuracy and F1 score, and confusion matrix is analyzed.

2 Model Architecture

The neural network architecture is defined as:

$$\texttt{layer\_dims} = [n_{\text{features}}, 3, 3, 1]$$

This includes:

- Input layer with n features
- Two hidden layers with 40 and 3 units, respectively
- Output layer with 1 neuron and sigmoid activation

3 Forward Propagation

The forward pass computes activations as:

$$Z^{[l]} = W^{[l]}A^{[l-1]} + b^{[l]}, \quad A^{[l]} = \sigma(Z^{[l]})$$

where σ is the sigmoid function for the output layer and ReLU for hidden layers.

4 Loss Function

The Binary Cross-Entropy Loss is used:

$$\mathcal{L} = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log(\hat{y}^{(i)} + \varepsilon) + (1 - y^{(i)})\log(1 - \hat{y}^{(i)} + \varepsilon)\right]$$

where $\varepsilon = 10^{-8}$ ensures numerical stability.

5 Class Imbalance Handling

To address class imbalance, a weighted loss gradient is computed:

$$dA = -w \cdot \left(\frac{y}{\hat{y}} - \frac{1-y}{1-\hat{y}} \right)$$

Here, w is a class weight assigned to the positive class to amplify the influence of the minority class during training.

6 Backpropagation and Weight Updates

The gradients for each layer are computed in reverse order:

$$\begin{aligned} dZ^{[l]} &= dA^{[l]} \cdot \sigma'(Z^{[l]}) \\ dW^{[l]} &= \frac{1}{m} A^{[l-1]} (dZ^{[l]})^T \\ db^{[l]} &= \frac{1}{m} \sum dZ^{[l]} \\ dA^{[l-1]} &= (W^{[l]})^T dZ^{[l]} \end{aligned}$$

Then, parameters are updated as:

$$W^{[l]} := W^{[l]} - \alpha \cdot dW^{[l]}, \quad b^{[l]} := b^{[l]} - \alpha \cdot db^{[l]}$$

7 Training Setup

- Epochs: 400
- Initial Learning Rate: 0.01
- Final Learning Rate: 0.00001 (linear decay)
- Batch Size: 200
- Activation: Sigmoid (output), ReLU (hidden)

8 Evaluation Metrics

After training, predictions are thresholded at 0.205 and evaluated using:

- Accuracy
- F1 Score
- Confusion Matrix

Example Output

- Train Accuracy: 0.5265378134153651
- Test Accuracy: 0.5295394915407582
- F1 Score (Test): 0.36616284739151633

Table 1: Confusion Matrix (Test)

	Predicted 0	Predicted 1
Actual 0	8702	8940
Actual 1	1460	3004

9 Threshold Analysis

The threshold of 0.18 was chosen after analyzing the cumulative distribution of predicted probabilities to favor recall of the minority class. Threshold tuning is critical in imbalanced problems.

10 Conclusion

This project demonstrates the implementation of a custom neural network using NumPy from scratch. By amplifying the minority class using class weights and tuning thresholds, the model attempts to strike a balance between recall and precision in an imbalanced dataset.