Implementation of a Three Hidden Layer Neural Network for Multi-Class Classification

Md. Shafayet Hossain Rizon, 20-43599-1

Introduction

The objective of this project was to implement a neural network with three hidden layers to solve a multi-class classification problem. A synthetic dataset with five distinct classes was generated, and the existing neural network code was modified to handle multi-class classification. The neural network was designed to process 10 input features and classify them into one of five classes, using softmax activation in the output layer and categorical cross-entropy loss to train the model. The training process aimed to minimize the loss and achieve high accuracy, with the model evaluated using key metrics like accuracy, precision, recall, F1-score, and a confusion matrix. These metrics provided insights into the model's performance and its ability to correctly classify each class. This report outlines the methodology used to implement the neural network, the steps taken to optimize its architecture, and the challenges encountered, such as overfitting and class imbalance. Additionally, recommendations for further improvements are provided to enhance the model's ability to generalize and perform better on unseen data. The project emphasizes the importance of network design, hyperparameter tuning, and evaluation in building effective neural networks for multi-class classification tasks.

Dataset Generation

The synthetic dataset was generated using the make_classification function from the sklearn library. This dataset contains 1000 samples, each with 10 input features. These features were designed to be informative for multi-class classification and were distributed across five classes.

The generated dataset was split into training and testing sets to evaluate the model's performance on unseen data.

Total samples: 1000

Input features: 10

Classes: 5

Training set: 800 samples (80%)

Testing set: 200 samples (20%)

The features were designed to have multiple clusters per class, making the classification problem more realistic for a neural network. The use of 10 input features ensured that the problem was non-trivial and required a reasonably deep network to solve effectively.

Neural Network Architecture

The neural network was implemented with the following structure:

Input Layer: 10 neurons (one for each input feature).

Hidden Layers:

- First hidden layer: 64 neurons
- Second hidden layer: 32 neurons
- Third hidden layer: 16 neurons

Output Layer: 5 neurons (one for each class)

Each hidden layer uses the ReLU activation function, which helps in learning complex non-linear relationships between input features and class labels. ReLU activation introduces non-linearity into the model and ensures that neurons are only activated when they have a positive input, which helps prevent gradient saturation.

The output layer uses the Softmax activation function, which transforms the raw output values (logits) into a probability distribution over the five classes. This ensures that the network outputs class probabilities, which are suitable for multi-class classification tasks.

Optimization and Loss: The model was optimized using Stochastic Gradient Descent (SGD) with a learning rate of 0.01. Gradient descent was used to minimize the categorical cross-entropy loss. This loss function is ideal for multi-class classification because it measures the difference between the predicted class probabilities and the actual one-hot encoded class labels.

Code Modifications

Several modifications were made to the provided code to support multi-class classification:

Output Layer: The output layer was modified to have 5 neurons, one for each class. The Softmax activation function was applied to these neurons to compute the probability distribution over the 5 classes.

Activation Functions: The hidden layers were set to use the ReLU activation function, which allowed the model to capture non-linear relationships. ReLU helps prevent the vanishing gradient problem that often occurs in deeper networks using sigmoid or tanh activations.

The output layer used the Softmax activation function, which ensures that the output values represent probabilities that sum to 1.

Loss Function: The categorical cross-entropy loss was implemented. It calculates the negative log likelihood of the correct class probability, penalizing incorrect predictions more heavily.

Backpropagation: The backpropagation algorithm was modified to handle multi-class classification. It computes the gradients for each layer by calculating the partial derivatives of the loss with respect to the weights and biases. These gradients are then used to update the weights via gradient descent.

Training and Testing

The model was trained using backpropagation for 1000 epochs, with a learning rate of 0.01. Each epoch consisted of a forward pass, where the input data was passed through the network, and a backward pass, where the gradients were computed and the weights were updated. The training process was monitored by computing the loss at each epoch. The loss curve indicated that the model was learning effectively, as the loss steadily decreased over

time. After training, the model was evaluated on both the training and testing sets to measure its performance on unseen data.

Results and Performance Metrics

The following results were obtained after the training process:

Training Accuracy: 77.25%

Testing Accuracy: 64.00%

The accuracy of the training set is relatively high, indicating that the model was able to learn the patterns in the training data effectively. However, the testing accuracy is lower, suggesting that the model has some difficulty generalizing to unseen data.

Confusion Matrix: The confusion matrix provides insights into how well the model predicted each class:

```
Confusion Matrix:
[[32  1  0  0  9]
 [12 15  1  3  2]
 [ 3  1 27  5  4]
 [15  1  2 23  0]
 [ 7  1  2  3 31]]
```

Key observations from the confusion matrix:

Class 0 was predicted correctly for 32 out of 42 samples, but some were misclassified as Class 4 (9 samples).

Class 1 had more misclassifications, with 12 samples being classified as Class 0.

Class 2 performed well, with 27 correct predictions out of 40 samples, but 5 were misclassified as Class 3.

Classification Report: The classification report provides a detailed breakdown of precision, recall, and F1-score for each class:

```
Classification Report:
              precision    recall  f1-score   support

     Class 0       0.46      0.76      0.58        42
     Class 1       0.79      0.45      0.58        33
     Class 2       0.84      0.68      0.75        40
     Class 3       0.68      0.56      0.61        41
     Class 4       0.67      0.70      0.69        44

    accuracy                           0.64       200
   macro avg       0.69      0.63      0.64       200
weighted avg       0.68      0.64      0.64       200
```

Class 2 achieved the highest F1-score (0.75), indicating the model's better performance in classifying this group.

Class 1 had high precision (0.79) but lower recall (0.45), indicating that the model often failed to predict some of the correct samples in this class.

Class 0 had a low precision (0.46), meaning that when it predicted Class 0, it often misclassified samples. However, its recall (0.76) indicates that it was able to correctly classify most Class 0 samples.

Analysis and Observations

The model showed good performance on the training set but struggled to generalize to the test set, as indicated by the lower test accuracy. This suggests the following potential issues and areas for improvement:

Class Imbalance: The confusion matrix indicates that the model struggled with certain classes, particularly Class 1 and Class 0, where there were a significant number of misclassifications. This could be due to the complexity of the decision boundaries between these classes.

Overfitting: The gap between training and testing accuracy (77.25% vs. 64.00%) suggests that the model may be overfitting to the training data. This can be addressed by implementing regularization techniques such as dropout or L2 regularization.

Vanishing Gradients: The use of ReLU activation mitigates the vanishing gradient problem to some extent, but adding more layers or using a more complex dataset could cause this issue to resurface. The gradient flow should be monitored carefully when experimenting with deeper architectures.

Challenges Faced

Several challenges were encountered during the implementation:

- Model Generalization: While the training accuracy was satisfactory, the model's generalization to the test set was weaker. This highlights the importance of tuning the network's architecture and hyperparameters.
- Misclassifications: Some classes were consistently misclassified, particularly Class 1, where 12 samples were predicted as Class 0. This suggests that the decision boundaries for these classes overlap significantly in the feature space.
- Training Time: Due to computational limitations, the training process was capped at 1000 epochs. A higher number of epochs may have allowed the model to converge to a better solution, but it also increases the time needed for training.

Discussion

The neural network achieved a training accuracy of 77.25% and a testing accuracy of 64.00%, indicating that while it learned the training data well, it struggled to generalize to unseen data. The gap between training and test accuracy suggests overfitting, which could be mitigated by regularization techniques like dropout or L2 regularization. Misclassifications were most pronounced in Class 1 and Class 0, pointing to overlapping decision boundaries and potential class imbalance issues. While Class 2 performed best, with a high F1-score, Class 0 had low precision, indicating frequent misclassification. This suggests the current model architecture might not fully capture the dataset's complexity, particularly in distinguishing between similar classes. Experimenting with deeper layers, advanced optimizers like Adam, and improving feature scaling or balancing techniques (e.g., SMOTE) could address these issues. Overall, while the neural network demonstrated solid

performance, adjustments in hyperparameters, model architecture, and data handling will be essential to improve accuracy and generalization across all classes.

Conclusion

The neural network implemented in this assignment demonstrated its capability to perform multi-class classification with reasonable accuracy. The architecture, with three hidden layers and the use of categorical cross-entropy loss, proved effective in classifying a dataset with five distinct classes. While the model achieved a 77.25% training accuracy and a 64.00% test accuracy, there is room for improvement, particularly in terms of generalization to unseen data. By applying techniques such as hyperparameter tuning, regularization, advanced optimization algorithms, and experimenting with different neural network architectures, the performance of the model can be enhanced. Additionally, addressing class imbalances and improving data preprocessing will further refine the model's ability to classify complex datasets. This assignment provided valuable insights into the challenges and intricacies of multi-class classification using neural networks. The findings underscore the importance of continual experimentation and tuning when working with neural network models, particularly in tasks involving complex decision boundaries and multi-class data.