Computer Vision - Homework 3

Ruchi Manikrao Dhore - W1652116

Monday 6th February, 2023

1 Introduction

This report provides an overview of backpropagation algorithm for an multilayer perceptron (MLP) neural network configuration.

2 Backpropagation algorithm

Backpropagation is a supervised learning algorithm used to train multilayer perceptron (MLP) neural networks. It calculates the error between the predicted output and the actual output, and then uses that error to adjust the weights and biases in the network in order to minimize the error and improve the accuracy of the model. The algorithm uses the chain rule of calculus to propagate the error from the output layer to the hidden layers and update the weights in a gradient descent manner. The process is repeated for multiple iterations until the error reaches a minimum or the training accuracy is high enough.

2.1 Algorithm steps

1. Feedforward: Calculate the outputs of all nodes in the network for a given input. This involves applying the activation function to the weighted sum of inputs for each node.

2. Calculate the error: The error between the actual output and the predicted output is calculated using a loss function, such as mean squared error.

3. Backpropagate the error: Using the chain rule of calculus, the error is propagated backwards through the network to calculate the error gradients for each weight and bias in the network.

4. Update weights and biases: The weights and biases are updated using gradient descent to minimize the error. This involves subtracting the gradient of the error with respect to the weights and biases from the current values of the weights and biases.

5. Repeat steps 1-4 for 100 epochs: The process is repeated for multiple iterations, with the goal of minimizing the error and improving the accuracy of the model.

6. Terminate when a stopping condition is met: The algorithm can be stopped when the error reaches a minimum, the accuracy of the model is high enough, or the maximum number of iterations has been reached.

2.2 Activation function - sigmoid

The sigmoid activation function is a type of mathematical function commonly used in artificial neural networks. It maps any input value to the range of 0 to 1, producing an "S" shaped curve. The output of the sigmoid function can be interpreted as the probability of a specific event occurring, making it useful for binary classification problems. The function is differentiable, which makes it possible to use gradient-based optimization algorithms to learn the parameters of a network. However, it also has some limitations, such as the saturation of the output for large input values, leading to vanishing gradients and slowing down the training process.

2.3 Mean squared error

Mean Squared Error (MSE) is a commonly used loss function for regression problems. It measures the average squared difference between the predicted values and the true values. MSE is defined as the average of the squared differences between the predicted and actual values for all instances in the dataset. The lower the MSE, the better the model's performance. The MSE can be used to optimize the parameters of a model, as the gradient of the MSE with respect to the model parameters can be computed and used to update the parameters through gradient descent or other optimization algorithms.

3 Output

Following are the generated graphs for different epoch values and layers.

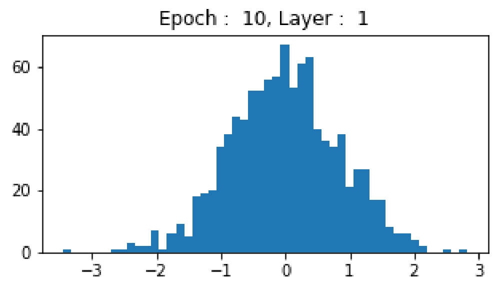

Figure 1: Epoch 10 Layer 1

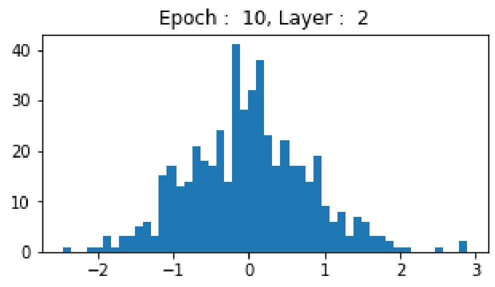

Figure 2: Epoch 10 Layer 2

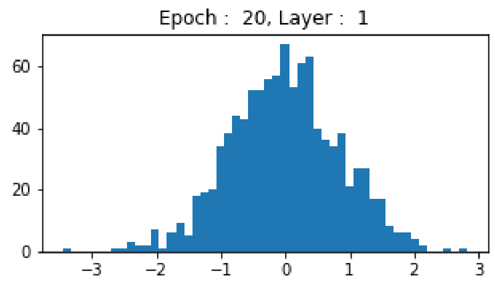

Figure 3: Epoch 20 Layer 1

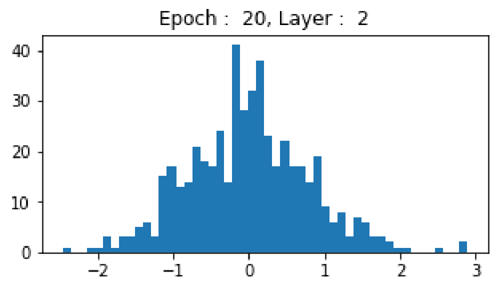

Figure 4: Epoch 20 Layer 2

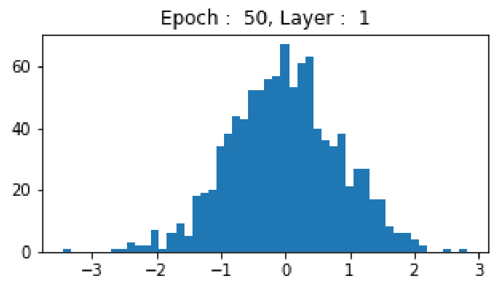

Figure 5: Epoch 50 Layer 1

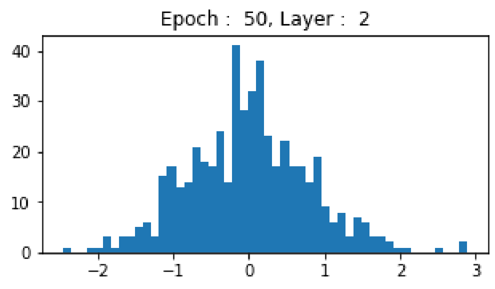

Figure 6: Epoch 50 Layer 2

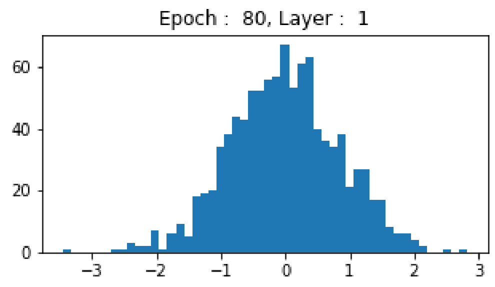

Figure 7: Epoch 80 Layer 1

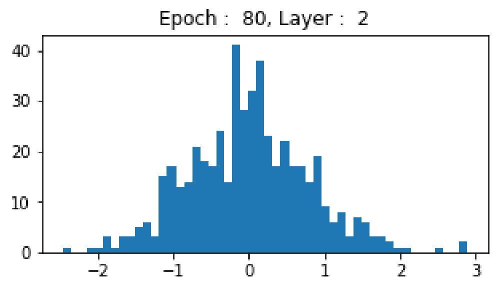

Figure 8: Epoch 80 Layer 2

4 Training loss per epoch

Training loss per epoch refers to the value of the loss function calculated after each epoch of training a machine learning model. An epoch is a complete iteration over the entire training dataset. During each epoch, the model is updated based on the gradients calculated from the loss function, and the training loss per epoch is a measure of the model's performance on the training data. It is shown as part of Figure 1.

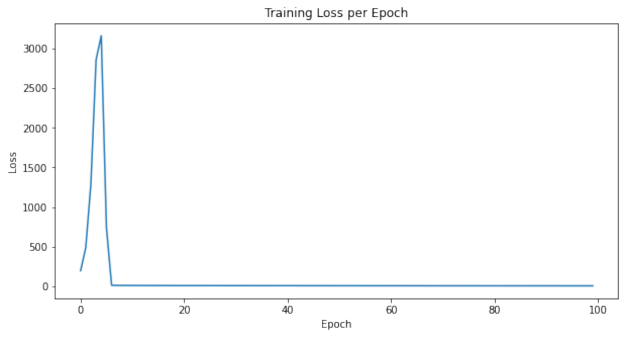

Figure 9: Training loss per epoch

5 Conclusion

By implementing the algorithm from scratch, a deeper understanding of the underlying mechanics of neural networks can be gained. By selecting the number of hidden layers and hidden units, as well as the activation functions and training data, the resulting model can be tailored to the specific regression problem at hand. By training the network for 100 epochs and plotting the training loss per epoch, the progress of the training process can be monitored. Additionally, plotting the distribution of weights for each layer at various epochs can provide insight into the behavior of the network over time.