22AIE304 Deep Learning Lab Sheet 5

Fifth Semester BTech CSE(AI)

Department of Computer Science and Engineering

Amrita School of Computing

Training a Feedforward Neural Network on the MNIST Dataset Using PyTorch

Objective:

This lab is designed to help students understand how to train a feedforward neural network on the MNIST dataset using PyTorch, visualize the training process, experiment with different hyperparameters, visualize errors and explore the effects of different activation functions on model performance.

Exercise 1: Use this sample ipynb file for training and do the following:

- Part 1: Setting Up the Environment
  - Installing Required Libraries
  - Loading the MNIST Dataset
- Part 2: Defining the Feedforward Neural Network
  - Input layer (784 units)
  - 1 or 2 hidden layers (try different configurations like 128, 256, 512 units)
  - Output layer (10 units corresponding to the 10 classes in MNIST)
- Part 3: Training the Model - Use sigmoid activation

Exercise 2: Visualizing the Training Process

- Plot the Training Loss Curve: Modify your code to store the training loss at each epoch and plot a curve that shows how the loss decreases over time. How does the loss change as the model trains? Does the loss converge as the number of epochs increases?
- Plot the Accuracy Curve: Track the training accuracy after each epoch and plot a graph showing how accuracy improves over time (comparing predictions with true labels). How does training accuracy change during training? Does the accuracy saturate after a certain number of epochs?
- Visualize Weight Updates: Visualize the changes in the weights of the hidden layer during training. After each epoch, store the values of the weights and plot them. How do weights evolve throughout training? Are there any patterns in the weight changes?

Exercise 3: Visualizing Validation/Testing results

a. Plot confusion Matrix on validation/test set: After the model is trained, plot a confusion matrix to show how well the model classifies each digit (0-9). Which digits does the model classify well, and which ones are more often confused with others
b. Plot validation accuracy and validation error for each epoch. How does the validation accuracy compare to the training accuracy
c. Compare training and validation loss

Exercise 4: Explore different Activation Functions

Replace the activation functions (ReLU, Sigmoid, Tanh, Leaky ReLU) in your network and observe the effect on convergence and accuracy. Plot the performance for different activation functions.

Exercise 5: Experimenting with Hyperparameters

a. Experiment with different learning rates (e.g., 0.1, 0.01, 0.001). Plot the effect of learning rate on training loss and accuracy.
b. Experiment with different batch sizes (e.g., 16, 32, 64, 128). Plot the training time and accuracy for each batch size.
c. Experiment with different numbers of hidden units in the hidden layers (e.g., 64, 128, 256). Plot the validation accuracy as a function of the number of hidden units.
d. Try different optimizers (SGD, Adam, RMSprop) and compare the convergence speed and final performance.

Submission: Copy each question into the notebook and write the corresponding code below each sub-question. After the code, provide the inference of the results in a text box. Submit both the .ipynb (Jupyter Notebook) file and a PDF version through the AUMS assignment link by **1st November 2024.**