Homework 6 Report
Name: Viprav Lipare
Student ID: 801288922
Homework Number: 6
Github Repository: https://github.com/vipravlipare/ECGR-5105-Intro-to-Machine-Learning

Problem 1

For Problem 1, a fully connected neural network (NN) was built for the diabetes dataset. The NN was designed with three hidden layers containing 16, 12, and 4 nodes, using ReLU activation for each hidden layer and a Sigmoid activation for the output layer. The model was trained for 300 epochs with a learning rate of 0.001. Training and validation used an 80%-20% split of the normalized dataset. The final training time, training loss, and evaluation metrics (accuracy, precision, recall, and F1 score) were recorded. The output produced the following results: Training time of 0.8293 seconds, Final training loss of 0.4034, and evaluation metrics of Accuracy = 0.7208, Precision = 0.6034, Recall = 0.6364, and F1 Score = 0.6195. The loss curves decreased as expected, but the validation loss started slightly increasing after 100 epochs. The evaluation metrics indicate that the neural network performed reasonably well on this dataset, however the logistic regression and SVM metric were slightly better. The accuracy and precision for the NN was slightly worse than the other two, the recall was much better for the NN compared to the SVM, and the F1 score is much better for the NN than the SVM.

Problem 2

For Problem 2, a fully connected neural network (NN) was built for the cancer dataset. The NN was designed with three hidden layers containing 32, 16, and 8 nodes, using ReLU activation for each hidden layer and a Sigmoid activation for the output layer. The model was trained for 300 epochs with a learning rate of 0.001. Training and validation used an 80%-20% split of the normalized dataset. The final training time, training loss, and evaluation metrics (accuracy, precision, recall, and F1 score) were recorded. The output produced the following results: Training time of 1.2787 seconds, Final training loss of 0.0131, and evaluation metrics of Accuracy = 0.9825, Precision = 0.9859, Recall = 0.9859, and F1 Score = 0.9859. The loss curves decreased steadily across the training duration, but the validation loss started slightly increasing around 150 epochs. The evaluation metrics indicate that the neural network performed extremely well on this dataset, with all metrics above 98%. The accuracy and precision for the NN was slightly better than the other two, the recall was worse for NN compared to the SVM, and the F1 score is much better for the NN than the logistic regression.

Problem 3A

For Problem 3A, a fully connected neural network (NN) was built to classify all 10 classes in the CIFAR-10 dataset. The NN was designed with one hidden layer containing 512 nodes, using a Tanh activation for the hidden layer and a LogSoftmax output layer. The input size is based on three channels (RGB) of 32 * 32. The model was trained over 300 epochs with a learning rate of 0.001 using stochastic gradient descent (SGD). Training and validation were performed using the normalized CIFAR-10 dataset. The final training time, training loss, and evaluation accuracy after 300 epochs were recorded. The output produced the following results: Training time of

626.6507 seconds (10.44 minutes), Final training loss of 0.4972, and an evaluation accuracy of 0.4871. The loss gradually decreased throughout training, and the training time took much longer than other training approaches because this model is going over images rather than numbers. The results indicate that while the network was able to learn from the images, the single-layer fully connected model was not complex enough to achieve a lower loss, or higher evaluation accuracy on the image dataset.

Unlike in the lecture code, the current code was changed so that instead of having the preprocessing code inside of the training loop it was taken outside of the loop. This was done by changing the code “out = model(img.view(-1).unsqueeze(0))” to the code “cifar10_trainX = torch.stack([img.view(-1) for img, _ in cifar10_train])”. These changes paired with running the code from the gpu made the training run much faster compared to before these changes.

Problem 3B

For Problem 3B, a fully connected neural network (NN) was built to classify all 10 classes in the CIFAR-10 dataset. The NN was designed with three hidden layers containing 1024, 512, 128 nodes, using three Tanh activation for the hidden layer and a LogSoftmax output layer. The input size is based on three channels (RGB) of 3* 32 * 32. The model was trained over 300 epochs with a learning rate of 0.001 using stochastic gradient descent (SGD). Training and validation were performed using the normalized CIFAR-10 dataset. The final training time, training loss, and evaluation accuracy after 300 epochs were recorded. The output produced the following results: Training time of 1167.7479 seconds (19.46 minutes), Final training loss of 0.0151, and an evaluation accuracy of 0.4552. The loss swiftly decreased throughout training, and the training time took much longer than other training approaches because this model is going over images rather than numbers. The results indicate that while the network was able to learn from the images, the more complex neural network was able to achieve a very low loss value, but the evaluation accuracy for the model is similar to that of the single-layer model.

These results indicate that the complex model is overfitting the training data. The final training loss is extremely low at 0.0151, and the evaluation accuracy is 0.4552, which is slightly worse than the baseline single-layer model from Problem 3A. This gap shows that while the model learns the training set extremely well, but doesn’t test the learning effectively against the test set.

Unlike in the lecture code, the current code was changed so that instead of having the preprocessing code inside of the training loop it was taken outside of the loop. This was done by changing the code “out = model(img.view(-1).unsqueeze(0))” to the code “cifar10_trainX = torch.stack([img.view(-1) for img, _ in cifar10_train])”. These changes paired with running the code from the gpu made the training run much faster compared to before these changes.