Homework 6 Report

Name: Gilberto Feliú
Student ID: 801257813
Homework Number: 6
GitHub Repository: https://github.com/gfeliu-rgb/Intro-to-ML-ECGR-4105

Problem 1

For Problem 1, I built a fully connected neural network for the diabetes dataset. I kept the architecture simple and clean: three hidden layers with 16, 12, and 4 nodes, all using ReLU, and then a Sigmoid output layer. I trained the model for 300 epochs with a 0.001 learning rate and used an 80/20 train–validation split after normalizing the dataset.

Here are my final results:

- **Training time:** 0.8293 seconds
- **Final training loss:** 0.4034
- **Accuracy:** 0.7208
- **Precision:** 0.6034
- **Recall:** 0.6364
- **F1 Score:** 0.6195

The training loss dropped the way it should, but validation loss started climbing after ~100 epochs, which usually means the model is beginning to overfit. Even though the network performed reasonably well, logistic regression and SVM still had slightly better accuracy and precision. The NN did stand out in recall and F1 compared to SVM, though.

Problem 2

Problem 2 followed the same process, but this time for the cancer dataset. I used a slightly larger network: hidden layers with 32, 16, and 8 nodes. Again, ReLU everywhere except a Sigmoid at the end. Same training setup as before: 300 epochs, 0.001 learning rate, and an 80/20 split.

Results:

- **Training time:** 1.2787 seconds
- **Final training loss:** 0.0131

- **Accuracy:** 0.9825
- **Precision:** 0.9859
- **Recall:** 0.9859
- **F1 Score:** 0.9859

The model basically performed extremely well across the board, with all metrics sitting above 98%. Validation loss started increasing after about 150 epochs, but performance still stayed strong. Compared to logistic regression and SVM, the NN performed slightly better in accuracy and precision, slightly worse in recall than SVM, and much better in F1 than logistic regression.

Problem 3A

For Problem 3A, I trained a fully connected model for the CIFAR-10 dataset. The network had one hidden layer with 512 nodes and used Tanh activation, with a LogSoftmax layer at the end. Inputs were flattened 3×32×32 RGB images. I trained the model for 300 epochs with SGD (lr = 0.001) on the normalized dataset.

My results:

- **Training time:** 626.6507 seconds (about 10.44 minutes)
- **Final training loss:** 0.4972
- **Evaluation accuracy:** 0.4871

The loss decreased over time, but CIFAR-10 is much harder for a fully connected network, so the accuracy makes sense. This model simply doesn't have enough capacity for image data. One big improvement I made was moving preprocessing outside the training loop:

```
cifar10_trainX = torch.stack([img.view(-1) for img, _ in
cifar10_train])
```

Running the model on GPU also sped things up significantly.

Problem 3B

For Problem 3B, I tried a deeper model: three hidden layers with 1024, 512, and 128 nodes, all using Tanh. Same training setup as before: 300 epochs, SGD, 0.001 learning rate, and normalized CIFAR-10.

Final results:

- **Training time:** 1167.7479 seconds (19.46 minutes)
- **Final training loss:** 0.0151

- **Evaluation accuracy:** 0.4552

The model clearly learned the training set extremely well — the loss basically collapsed — but it didn't generalize. The test accuracy dropped below the simpler 3A model, which shows heavy overfitting. This is a common problem when using fully connected layers on image data.

Just like in 3A, I moved preprocessing outside the training loop: