Network Report

Jim Slone
Data 622

To prepare for setting up the neural network, I implemented one-hot encoding to encode the different target values in the data into different classes and used a standard scaler to standardize the features within the input data. I split the dataset into training and testing portions, with proportions of 20% for testing and 80% for training. My network is a simple feed-forward neural network, with an input layer, two hidden layers, and an output layer. The best hyperparameters for the model shape that I found were 32 nodes for the first hidden layer and 16 nodes for the second. I have tried adding a third layer to increase performance, but the effect was not noticeable. I initialized the biases for the network to be zeros and the weights to be small random numbers to help start the networks learning process and defined the forward propagation function, with the first activation function being SELU, the second being RELU, and the final being SOFTMAX. I then computed the loss of the network using categorical cross-entropy loss as our target has multiple classes. I then defined the back propagation function using the derivatives of the various weights, biases, and activation functions, which was used to adjust the weights and biases of the network in the direction of decreasing loss.. This process was completed for each epoch, gradually teaching the network the relationships present in the data. Finally, the network's accuracy was evaluated by running inference on the testing data and comparing the predicted values of the target to the real ones.

```
#v01: lr=0.05, epochs=1000, hidden_size=16, Test Accuracy = 0.5594
#v02: lr=0.1, epochs=1000, hidden_size=16, Test Accuracy = 0.5813
#v03: lr=0.2, epochs=1000, hidden_size=16, Test Accuracy = 0.5750
#v04: lr=0.1, epochs=1000, hidden_size=64, Test Accuracy = 0.5781
#v05: lr=0.1, epochs=1000, hidden_size1=16, hidden_size2=8, Test Accuracy = 0.4062...
#v06: lr=0.1, epochs=1000, hidden_size1=32, hidden_size2=16, Test Accuracy = 0.5094
#v07: lr=0.02, epochs=10000, hidden_size1=32, hidden_size2=16, Test Accuracy = 0.5656
#v08: lr=0.02, epochs=10000, hidden_size1=32, hidden_size2=16, Test Accuracy = 0.5625, replaced second activation with selu
#v09: lr=0.03, epochs=10000, hidden_size1=32, hidden_size2=16, Test Accuracy = 0.5813, selu kept and all below unless mentioned
#v10: lr=0.05, epochs=10000, hidden_size1=32, hidden_size2=16, Test Accuracy = 0.5938
#v11: lr=0.05, epochs=10000, hidden_size1=64, hidden_size2=16, Test Accuracy = 0.5875
#v12: lr=0.08, epochs=10000, hidden_size1=32, hidden_size2=16, Test Accuracy = 0.6188
#Gonna run a sweep for learning rates between 0.05 and 0.2 at 0.01 increments below
#lr=0.05; Test Accuracy = 0.5938
#lr=0.06; Test Accuracy = 0.6094
#lr=0.07; Test Accuracy = 0.5969
#lr=0.08; Test Accuracy = 0.6188
#lr=0.09; Test Accuracy = 0.6219
#lr=0.10; Test Accuracy = 0.5875
#lr=0.11; Test Accuracy = 0.6094
#lr=0.12; Test Accuracy = 0.6125
#lr=0.13; Test Accuracy = 0.5781
#lr=0.14; Test Accuracy = 0.6031
#lr=0.15; Test Accuracy = 0.6000
#lr=0.16; Test Accuracy = 0.6031
#lr=0.17; Test Accuracy = 0.5625
#lr=0.18; Test Accuracy = 0.5719
#lr=0.19; Test Accuracy = 0.5875
#lr=0.20; Test Accuracy = 0.5594
#v13: lr=0.05, epochs=20000, hidden_size1=32, hidden_size2=16, Test Accuracy = 0.6188
#v14: lr=0.09, epochs=20000, hidden_size1=32, hidden_size2=16, Test Accuracy = 0.6188
#!v15: lr=0.09, epochs=10000, hidden_size1=32, hidden_size2=16, Test Accuracy = 0.6219
#v16: lr=0.09, epochs=10000, hidden_size1=32, hidden_size2=16, Test Accuracy = 0.6062, swapped selu for leaky relu
```

Many combinations of hyperparameters were tested for this network(See Above), and the information on these and their success can be found in the notes at the bottom of the *nn.py* file. The loss curve for the final version can be seen below, with a test accuracy of 0.6219 being achieved. The hyperparameters used were as follows: epochs=10000 and learning rate = 0.09.