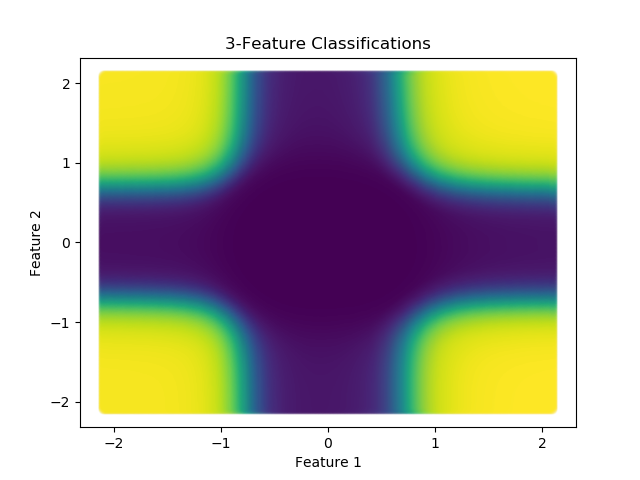
**ECE4870/7870 CS 4770/7770 F’18 Computer Assignment 1 Part B Due 10/23/2018**

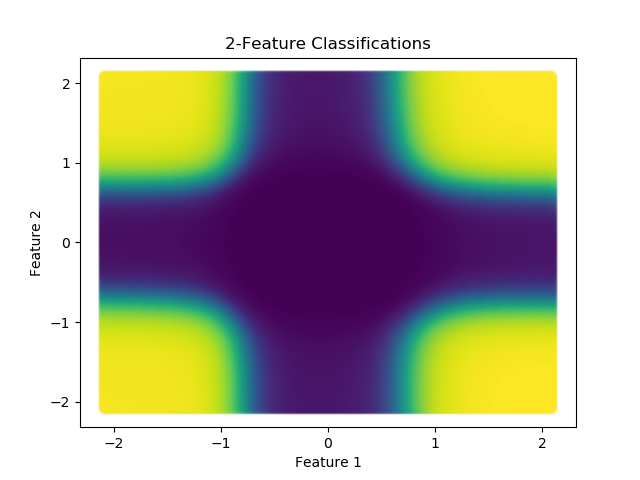
**Backpropagation Training of a MLP**

Part B:

1. Problem 1 demonstrates the decision boundaries generated by the 3:11:2 MLP (learning rate = 0.7, momentum = 0.3) on the first data set. The model was trained until the change in sum of squared errors was less than 0.001. Once the model converged, a uniform set of data points in the boundary [-2.1, 2.1] x [-2.1, 2.1] were tested. Plots of the model training error (sum of squared error) and class prediction for each sample were generated:

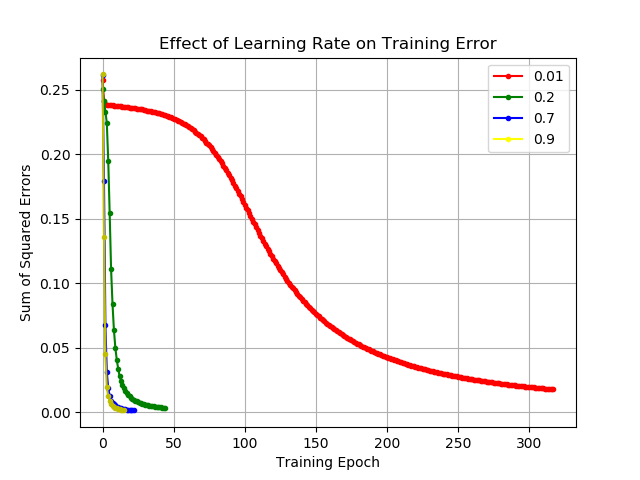
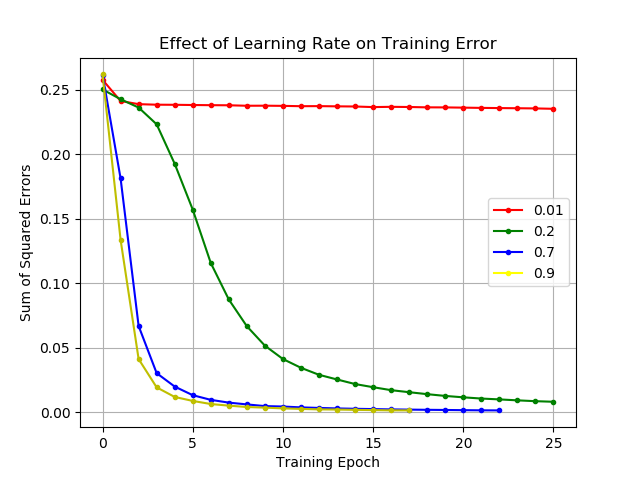


Next, the same procedure was completed using only the first two features from the dataset. The structure of the MLP was changed to 2:11:2. The same plots were generated for the modified data set:



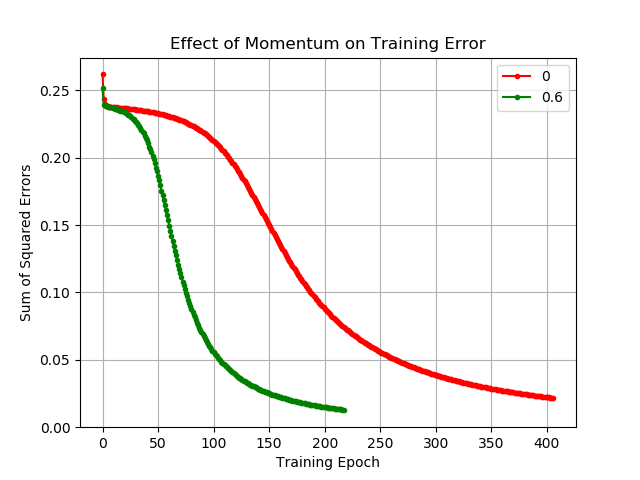
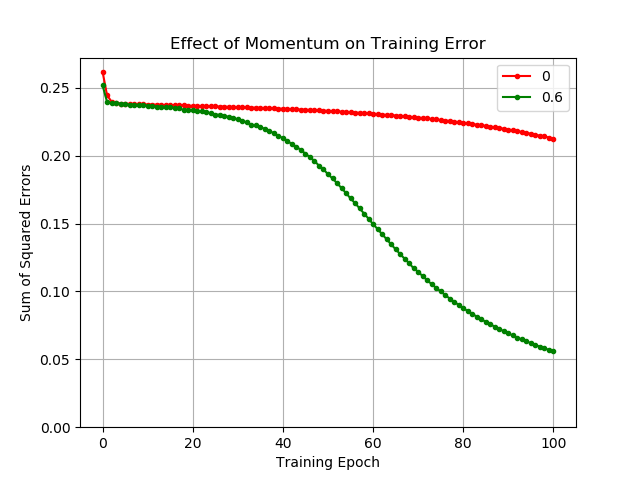
It is apparent that the decision boundaries generated by the model are nearly identical. This shows that the third feature in the dataset is not indicative of sample class, and therefore does not aid in training. In this scenario, it would be ideal to train the model without the third feature since including this feature does not impact the model’s accuracy but will increase training time and could lead to overfitting.

1. Problem 2 demonstrates the effect of learning rate on MLP training. An identical 3:11:2 MLP (momentum = 0.3) was trained on the same data set at learning rates of 0.01, 0.2, 0.7, and 0.9. For each learning rate, the training error (sum of squared errors) was plotted after 25 epochs and again after all models had converged (change in sum of squared errors < 0.0001):



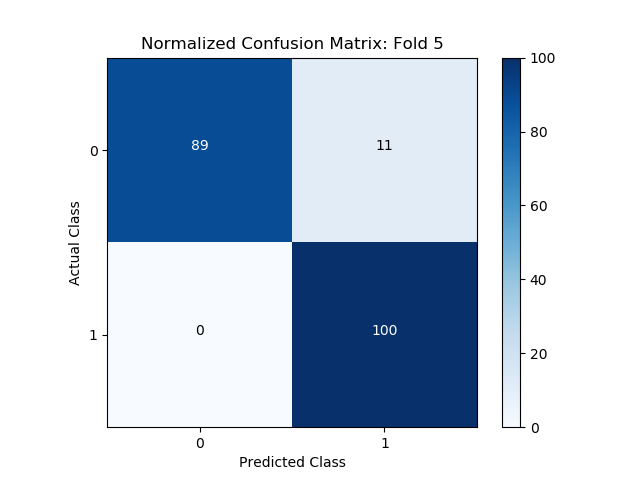
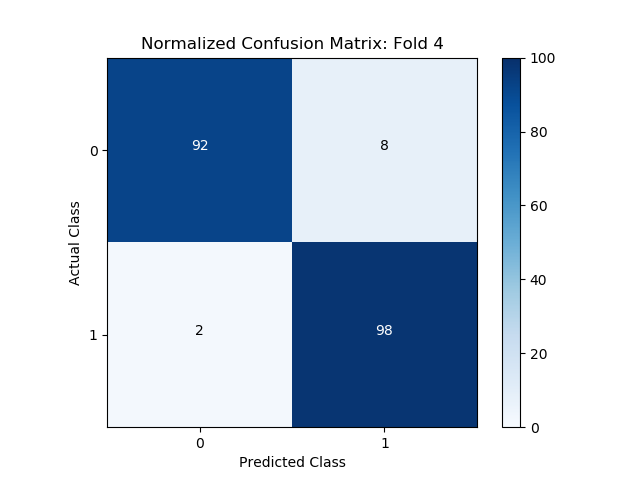
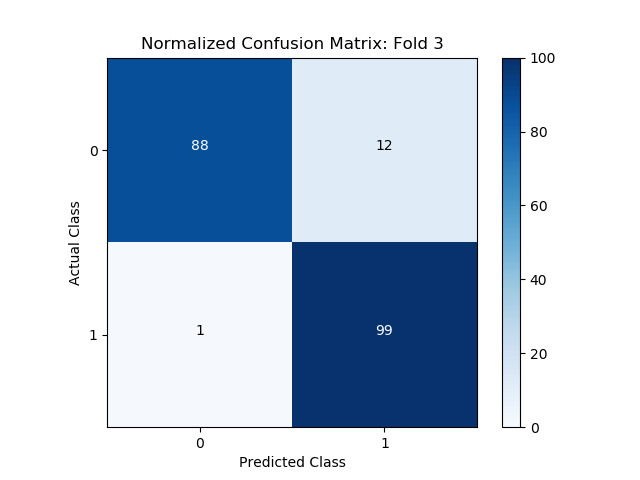
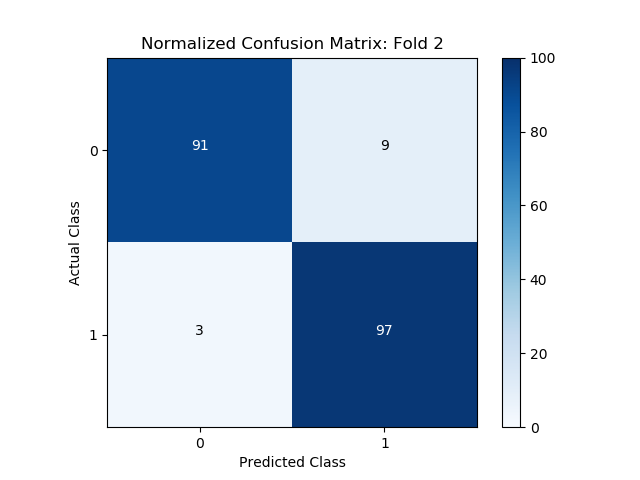
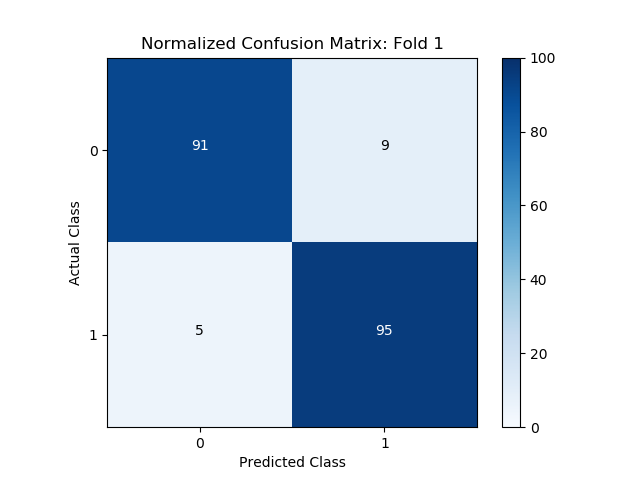
The results demonstrate that increasing the learning rate of the MLP increases the rate at which the error decreases. The model with training rate of 0.01 was significantly slower than the other models. However, given a sufficient number of epochs, the model will eventually converge to a similar error regardless of the learning rate.

1. Problem 3 demonstrates the effect of momentum on MLP training. An identical 3:11:2 MLP (learning rate = 0.01) was trained on the same data set at momentums of 0 and 0.6. For each momentum, the training error (sum of squared errors) was plotted after 100 epochs and again after both models had converged (change in sum of squared errors < 0.0001):

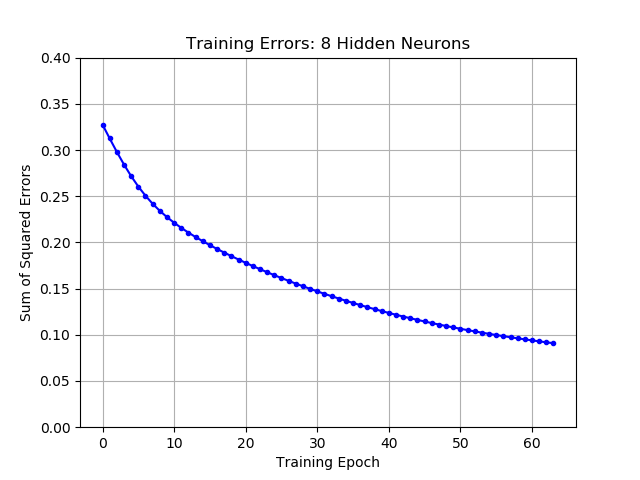
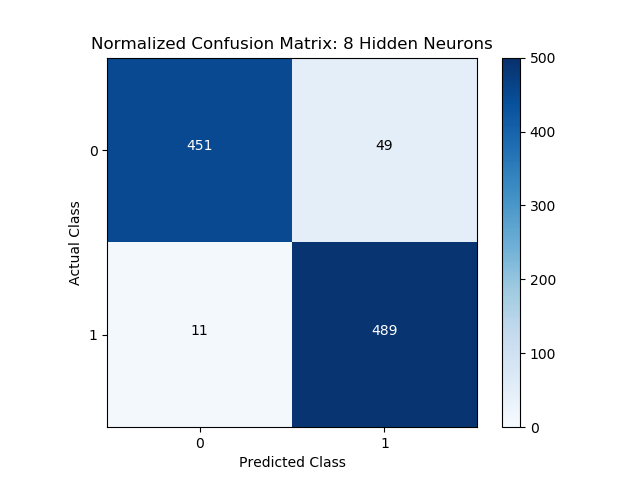


The results demonstrate that increasing the momentum of the MLP increases the rate at which the error decreases. Although both models behave similarly for the first ~10 epochs, the model with momentum reaches convergence much more quickly.

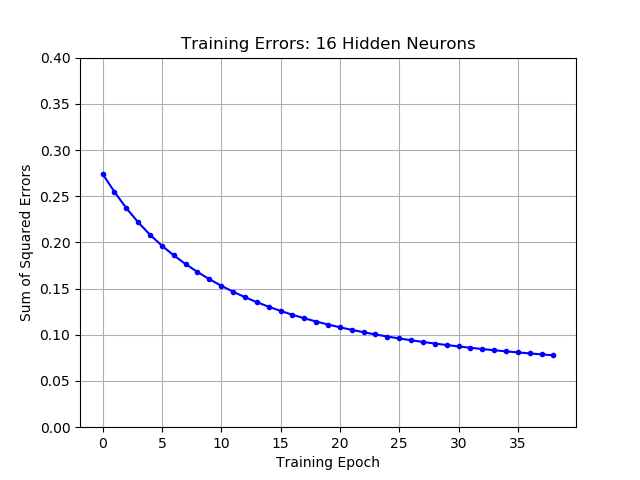
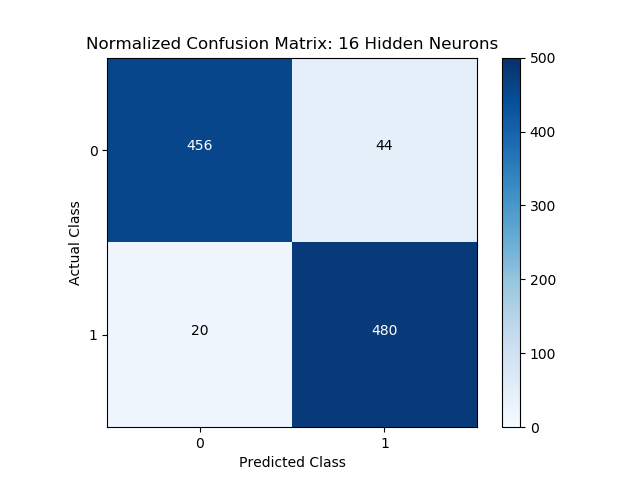
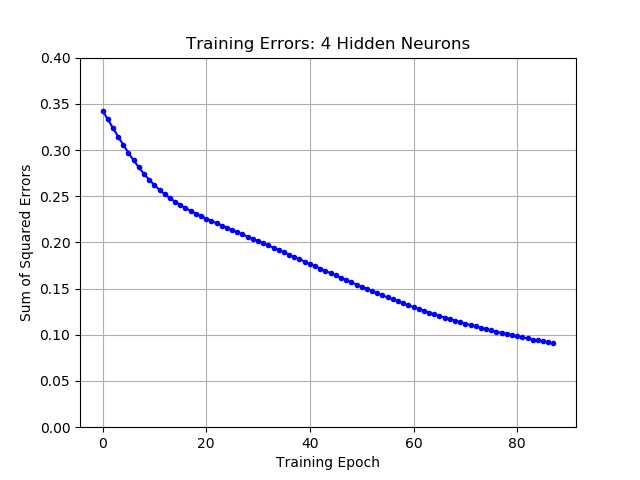
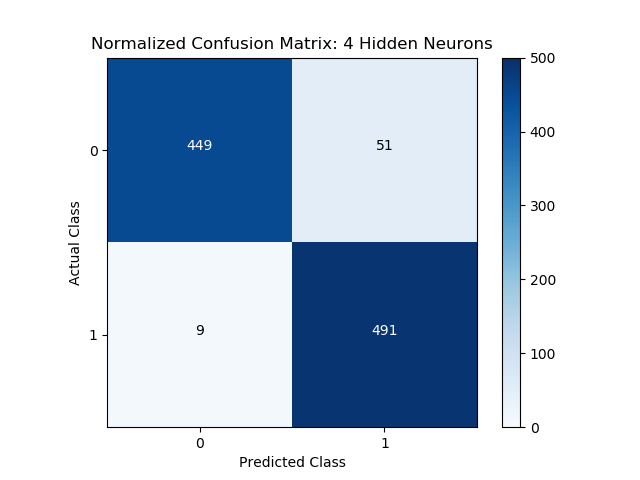
1. Problem 4 introduces a new data set: a two-class, four-dimensional set of 500 Gaussian distributions. This problem demonstrates cross-validation and hidden layer size on MLP training. First, a 4:8:2 model was constructed and trained using 5-fold cross-validation. The model has a learning rate of 0.001, and a momentum of 0.2. Each fold was trained until the change in sum of squared errors was less than 0.001. A confusion matrix was created for each of the five folds:



After all five folds were complete, the results from each fold were combined and an aggregated confusion matrix was created along with a plot of training error (sum of squared error) until convergence:



Next, the same experiment was repeated twice with half the number of hidden neurons and double the number of hidden neurons. The new 4:4:2 and 4:16:2 MLPs had the same learning rate and momentum and were also trained using 5-fold cross-validation until convergence. After all five folds were complete, the results from each fold were combined and an aggregated confusion matrix was created along with a plot of training error (sum of squared error) until convergence:



The results demonstrate the effects of different numbers of hidden neurons on MLP training. The most notable difference is the increased rate of training speed (decreasing error) as the number of neurons is increased; the models with more hidden layer neurons train and reach convergence much quicker: the 4:16:2 model converged after 38 epochs, 229% faster than the 4:4:2 model and 166% faster than the 4:8:2 model. Interestingly, the cross-validation accuracy of the model was very similar for each of the models. After training until convergence, the accuracy of the 4:4:2 model was 94%, the accuracy of the 4:8:2 model was 94%, and the accuracy of the 4:16:2 model was 93.6%.

Increasing the number of hidden nodes had a much more dramatic effect on training error than cross-validation accuracy. This is most likely because increasing the number of free parameters in the model allows the model to fit more closely to the training data but does not necessarily create a more accurate general model. Overfitting is more likely to occur with the higher-neuron models. For the given data set, the only true benefit of increasing the number of hidden neurons seems to be training speed.