**COMP4107 – Neural Networks**

**Assignment 2 Answers**

**Members: Krystian Wojcicki, Michael Kuang**

Notes:

Extra libraries required:

minisom, https://github.com/JustGlowing/minisom, pip install minisom

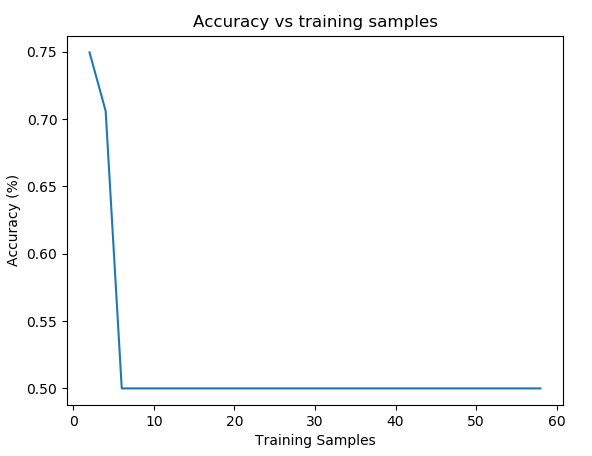
Depending on instillation pillow may also need to be required (should be downloaded when installing tensorflow however)

https://github.com/python-pillow/Pillow , pip install Pillow

1.

a)

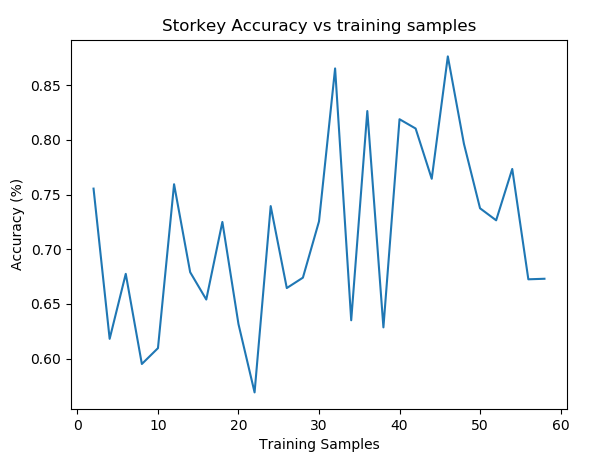
The graph below shows accuracy vs training samples for the one-shot learning as described in class. As we can see the accuracy quickly decreases as the number of training samples goes up. This seems to make sense as the Hopfield network can only learn a limited number of patterns, and as more training images are used the learned patterns become more and more degenerate until the network essentially flips a coin to decide what the output is.



*Figure 1: Comparing accuracy against training samples using regular Hopfield network*

b)

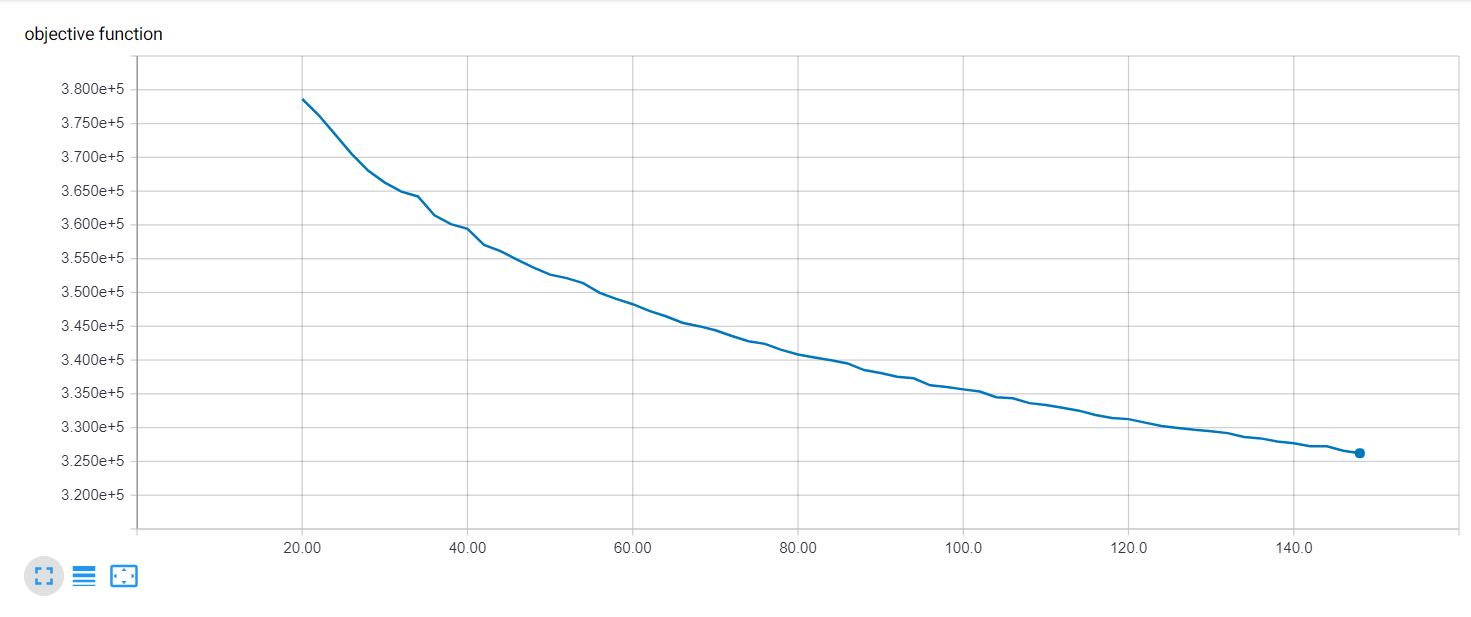
Using Storkey’s learning method we can see that the number of patterns that can be stored is much higher as indicative of the higher accuracy and that the network does not have as big of an issue with using more training images while training.



*Figure 2: Comparing accuracy against training samples using Storkey Hopfield network*

2.

a)



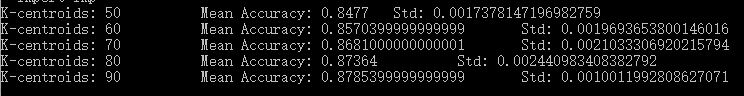
*Figure 3: Elbow finding graph*

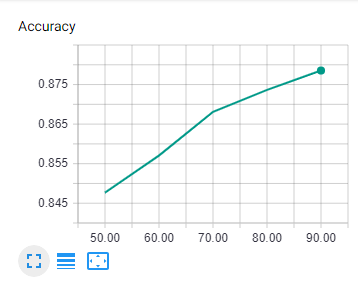
Above is a graph where the y-axis denotes the value of the objective function and the x-axis denotes k, the number of centroids. We ran the elbow finding experiment from k=20 to k=150 with increments of 2. We see that the graph is more like a curve. At the start, it is steeper and right around 60 to 70 we see that the graph slows down. So, we think it is a reasonable assumption that choosing around 70 hidden neurons for our hidden layer is an optimal choice.

b)

In our implementation of the code, you will see that we used 5-fold cross correlation when we run our experiments to investigate the performance on our neural network for different sizes of hidden layer and dropout in the hidden layer.

c) We ran a 5-fold cross validation for hidden layer sizes of 50, 60, 70, 80, and 90 for 50 epochs each due to time constraints. As such, we did not train till convergence. Another note is that we do not have a graph for standard deviation because it was not implemented at this time, however we do have a graph showing the mean accuracy over the k-fold cross validation for each hidden layer size.

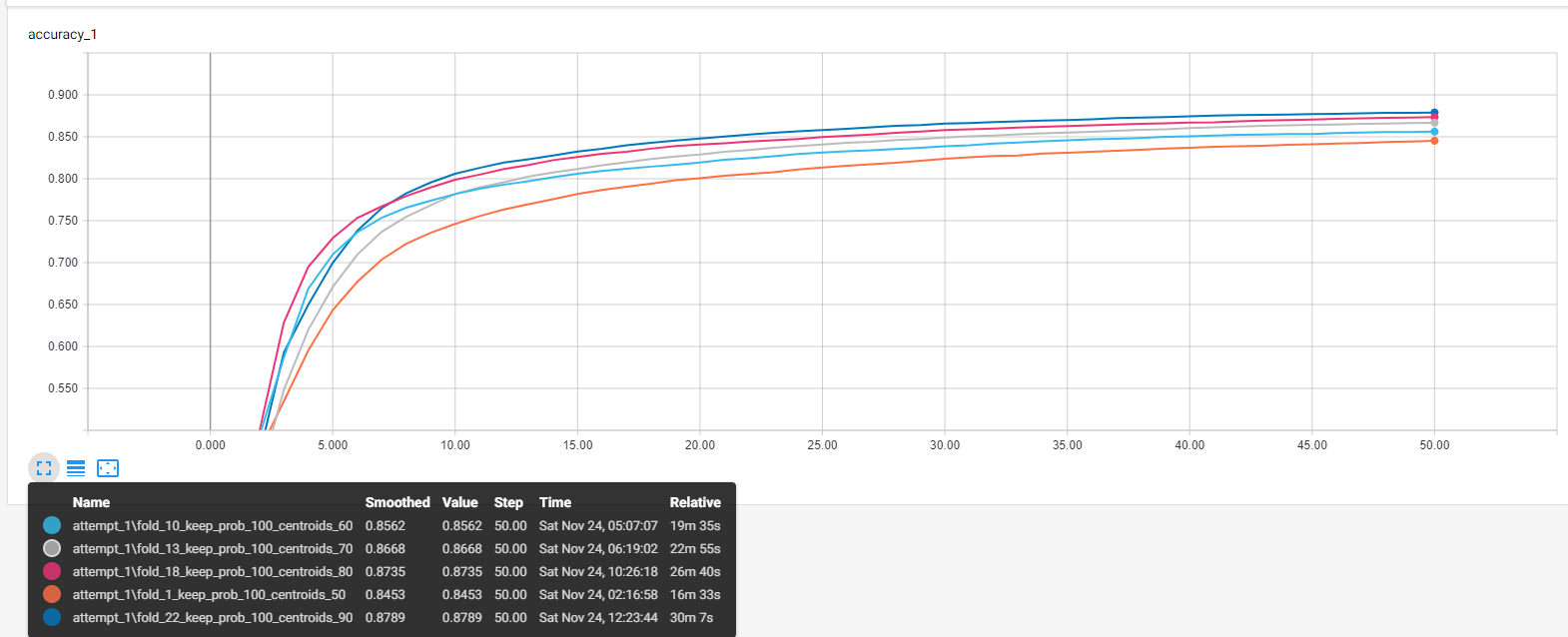




*Figure 4: Mean accuracy on 5-fold Cross Validation over Different Hidden Layer Sizes.*

*Y-axis denotes accuracy, X-axis denotes number of hidden neurons in layer*

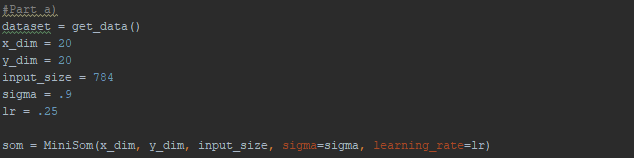
As you can see for figure 2, with each increase in the size of the hidden layer, the accuracy also increases. This makes sense as you have more centroids to differentiate the different classes.



*Figure 5: Comparing Testing Accuracy at each epoch for different hidden layer sizes*

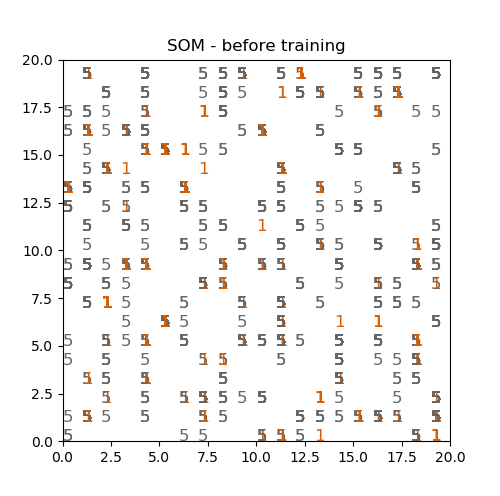
Figure 3 is just another graph visualization of the accuracy improvements when increasing hidden layer size. This graph only shows the training accuracies for a single fold for each hidden layer size of 50, 60, 70, 80 and 90.

3.

a) 

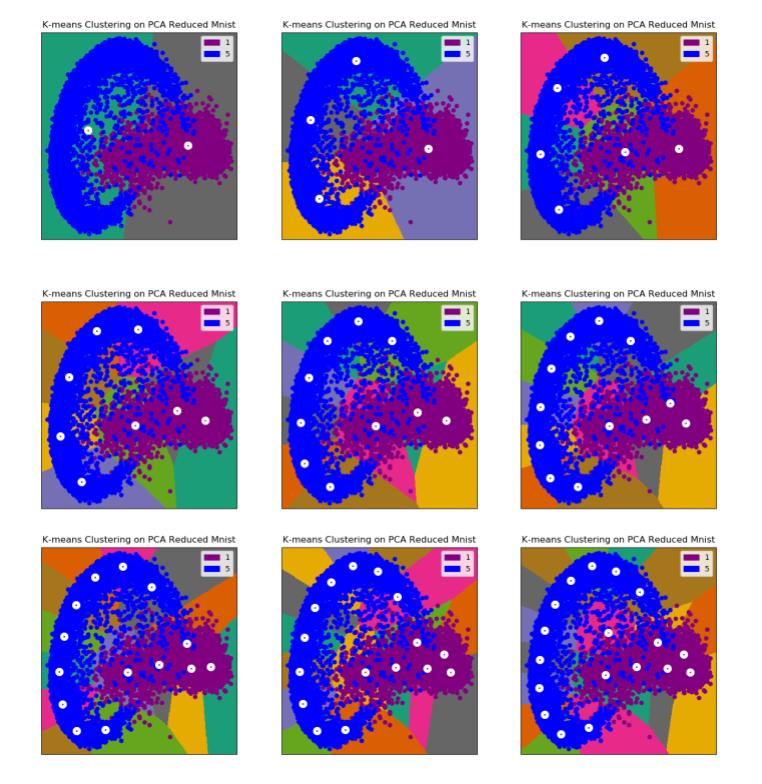
In our implementation, we used a dimension of (20,20) for our SOM, 0.9 for sigma and a learning rate of 0.25.

Plots for before training SOM and after training SOM.



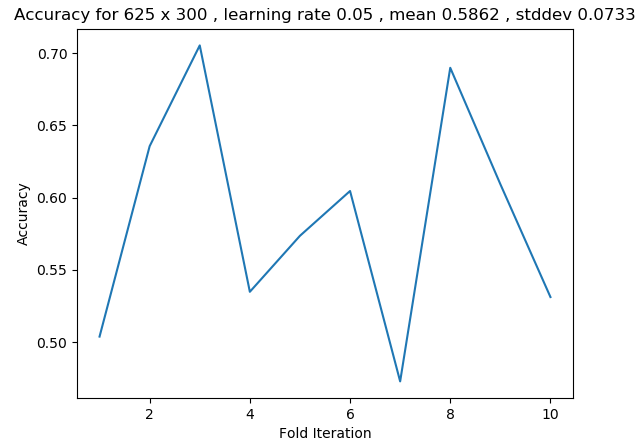
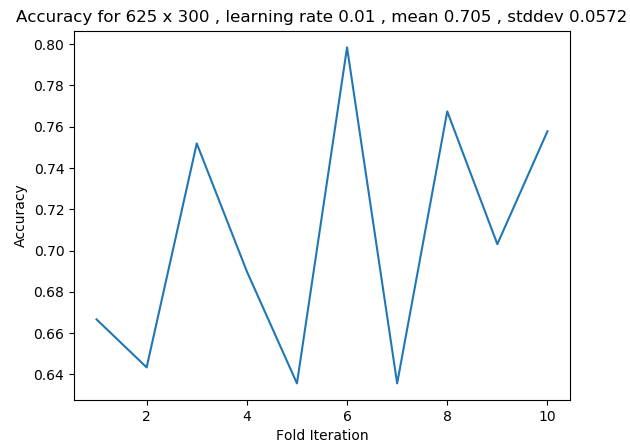
As you can see from the before and after shots of the SOM, the 1’s and 5’s are mostly organized and clustered with each other.

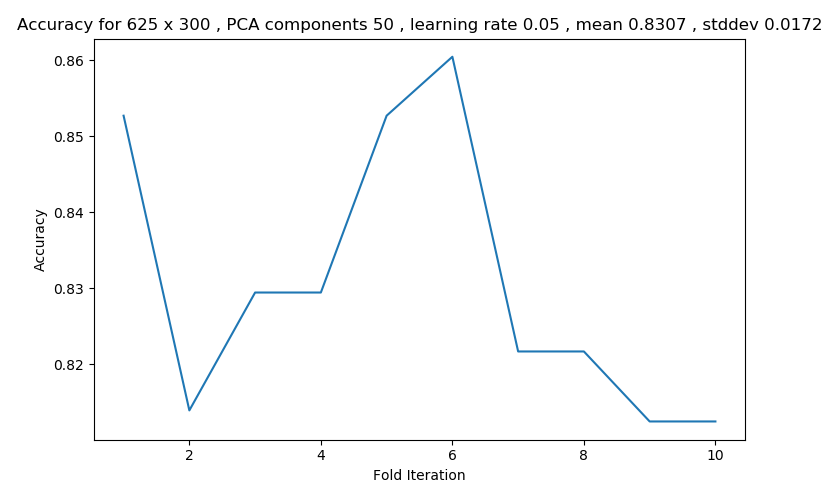
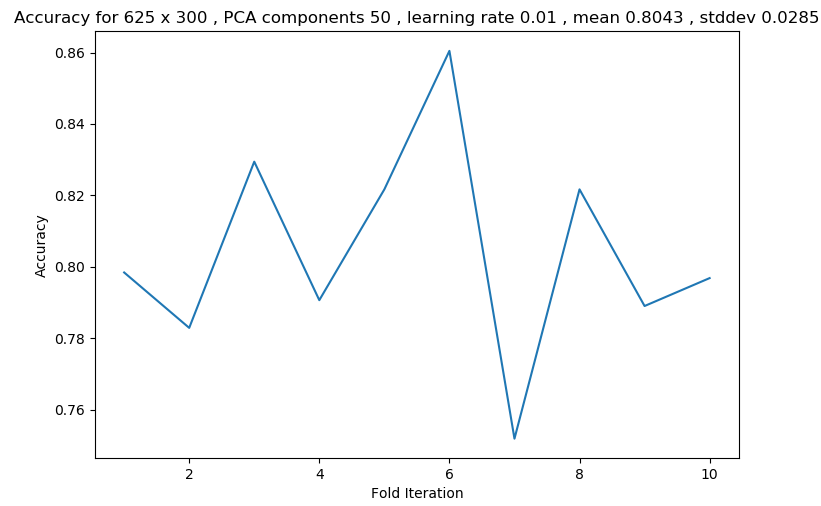
Below are Voronoi graphs for ‘1’ and ‘5’ for K-means solution. We tried from k=2 to k=20 with increments of 2. In the graph, each centroid is denoted as a white circle and the background colors surrounding each centroid show the area that is closest to that centroid. For reducing the kmeans data to a 2D representation PCA was used instead of SVD.

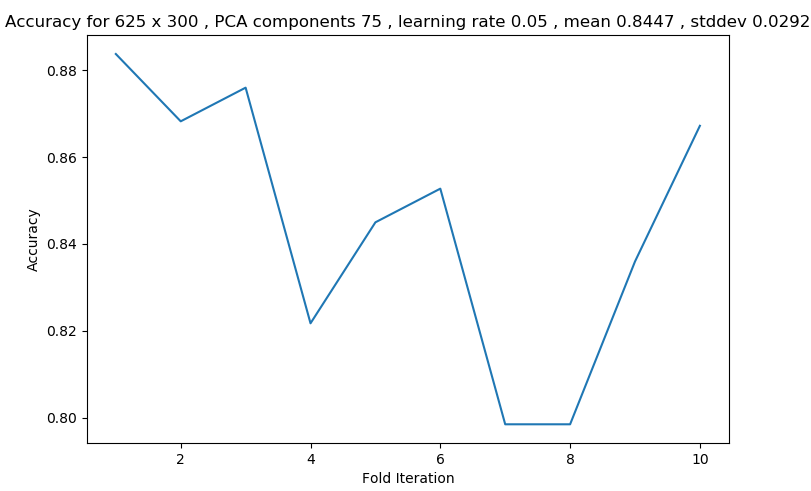
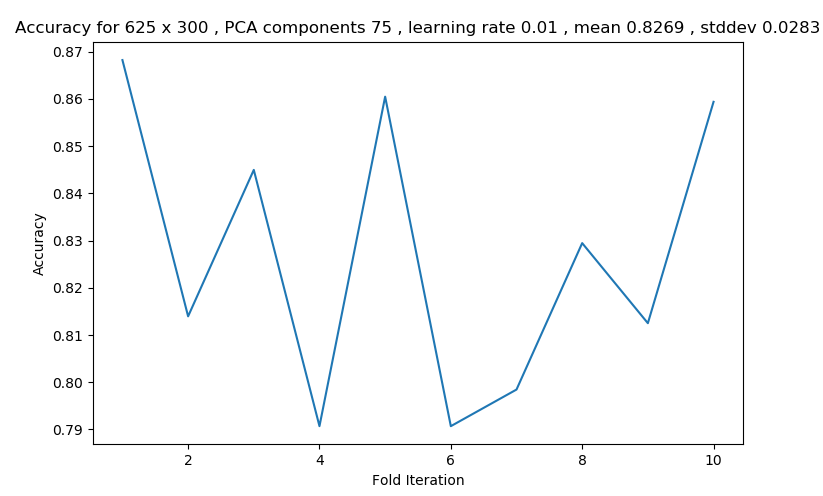
From a qualitative perspective, we see that the network will obviously have a low accuracy if k=2 as seen in the first graph, but with k=10 the graph is well separated and should have a much higher accuracy.

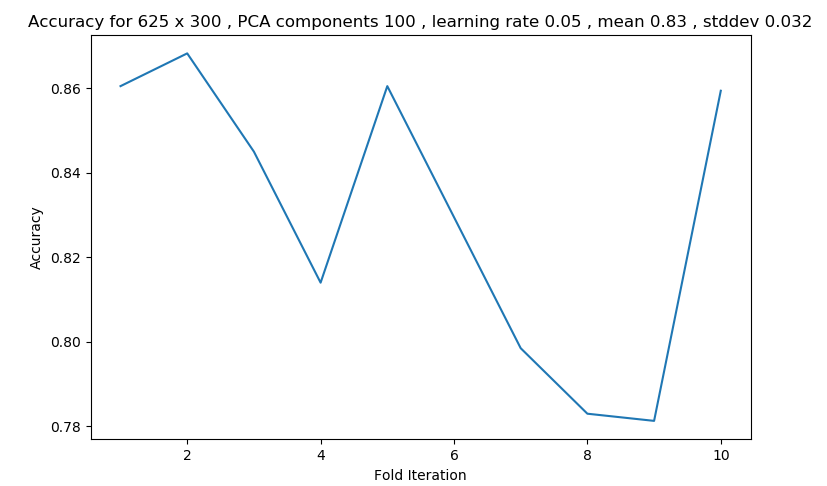
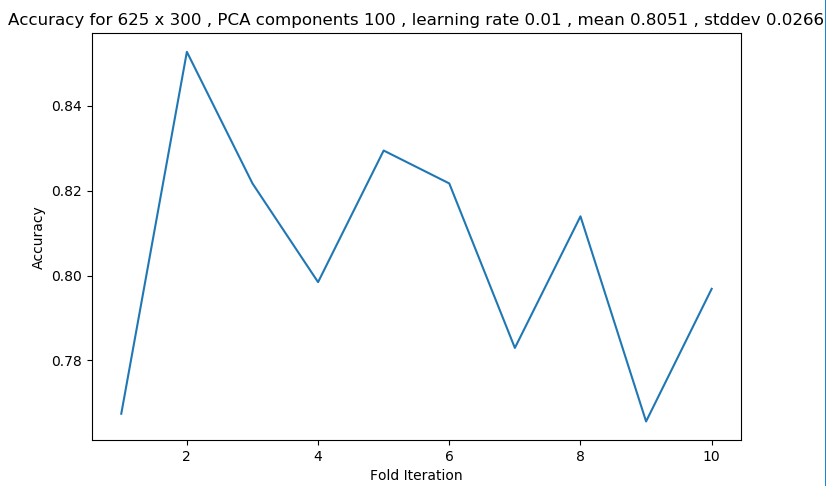
4)

For this question the network consisted of 2 hidden layers either with 625,300 neurons or 400,200 neurons, the activation functions were regular relu not leaky relu and gradient descent was used for learning. Experimentation was done around the learning rate (0.01 and 0.05) and the number of components (50, 75, 100) used in the PCA decomposition. 10-fold experimentation was done for 100 epochs.

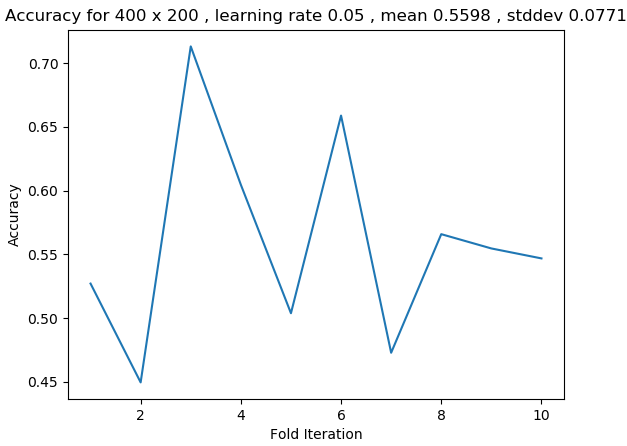
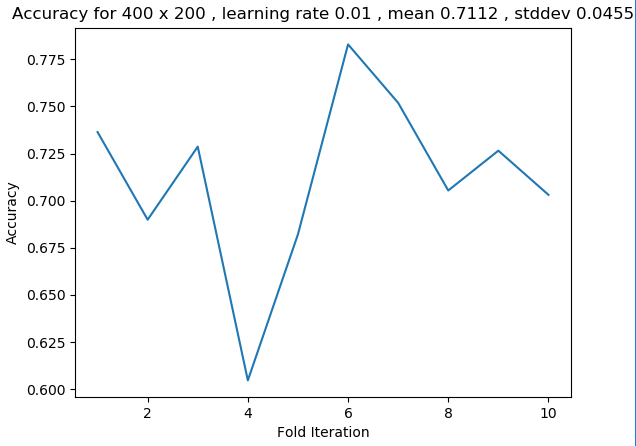


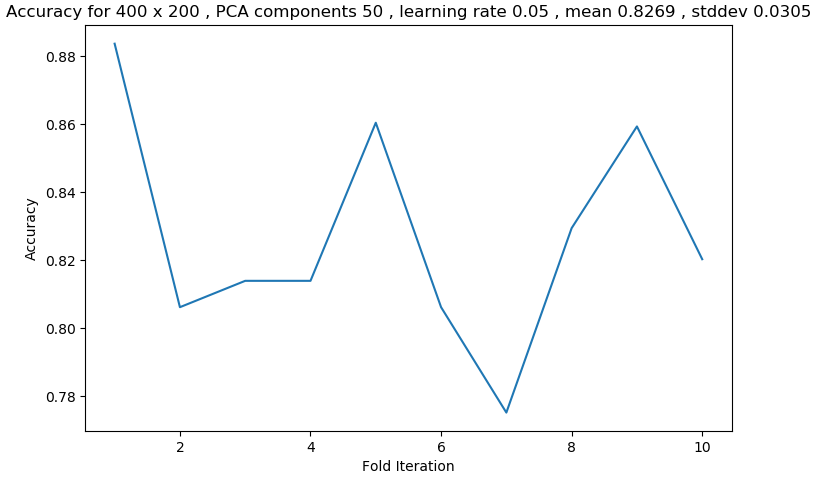
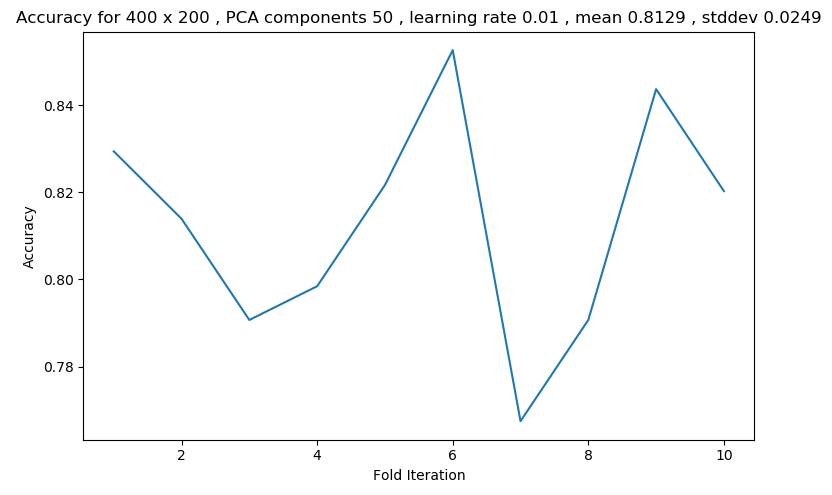


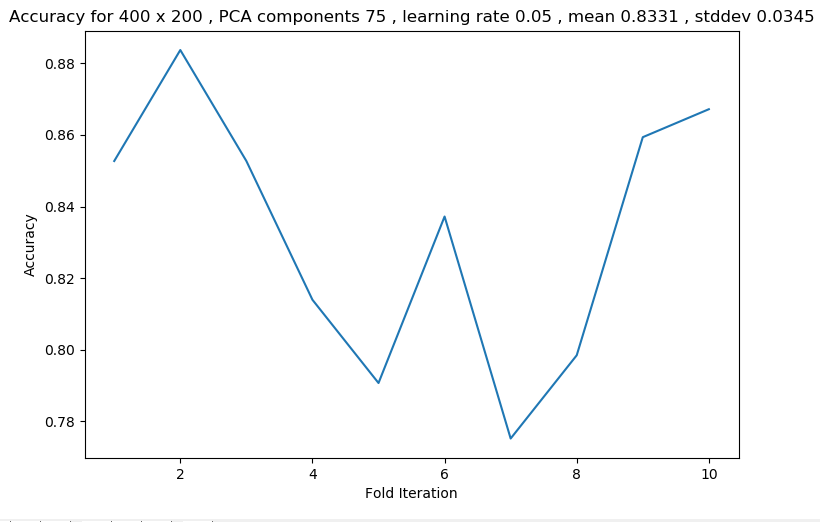
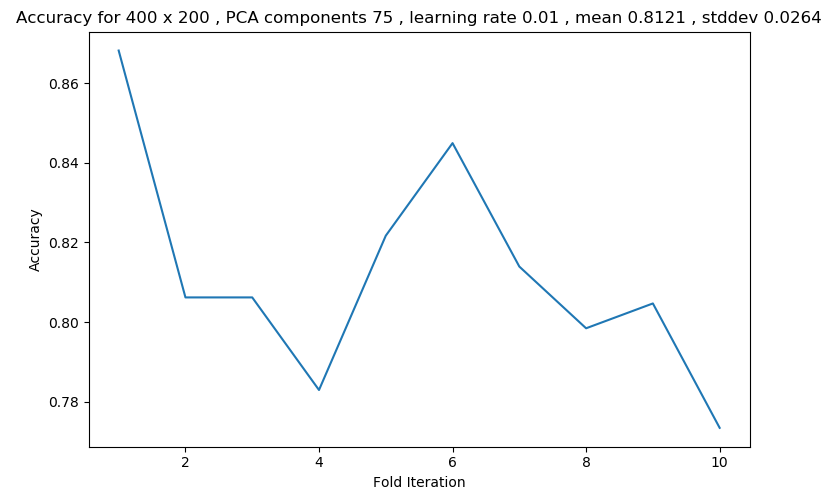


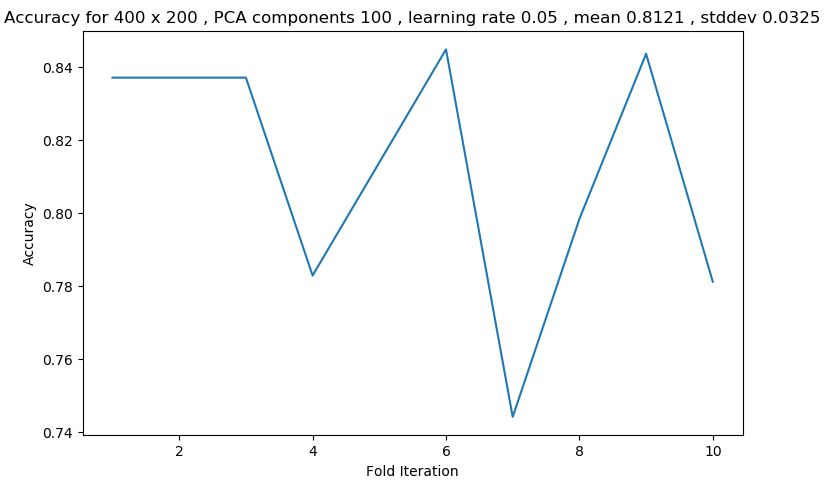
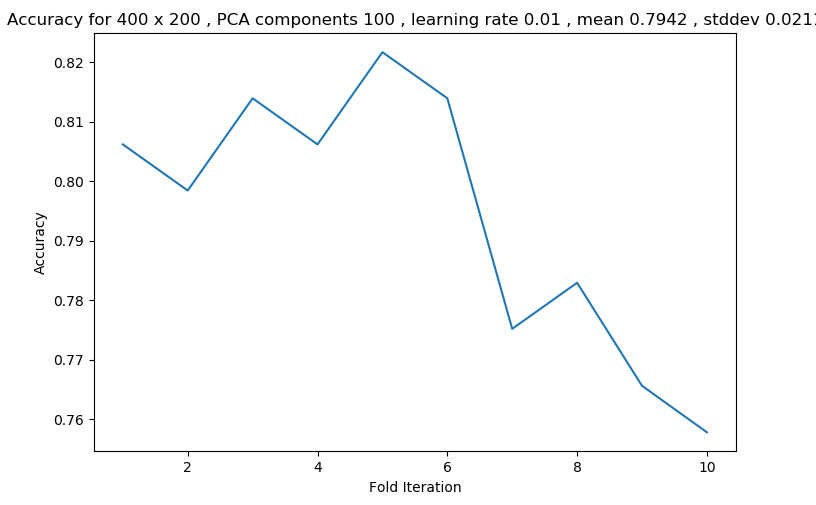


With a network of size 625 by 300 it is quite clear to see that doing PCA on the input greatly increases the accuracy of the network regardless of the number of components used. With respect to the optimal number of components that seems to be 75, meaning 50 is likely under fitting the data to some degree and 100 components over fits the data in both cases this causes a slight decrease in accuracy. Using a learning rate of 0.05 seems to be better as well.









The results with a network of size 400,200 were quite like that of the network of size 625, 300. Using PCA greatly increased the accuracy but in this case 50, 75 components offered similar accuracy.

Overall the larger network that used PCA was more successful which makes sense as similar to SVD, PCA obtains the most important hidden features allowing the network to be trained on the most important part of the data.