COMP 4107-Assignment#3

Due: 18th March 2020 at 11:55 pm

Name: Kirk Zhen
Student ID: 101087006

Question 1 [50 marks]

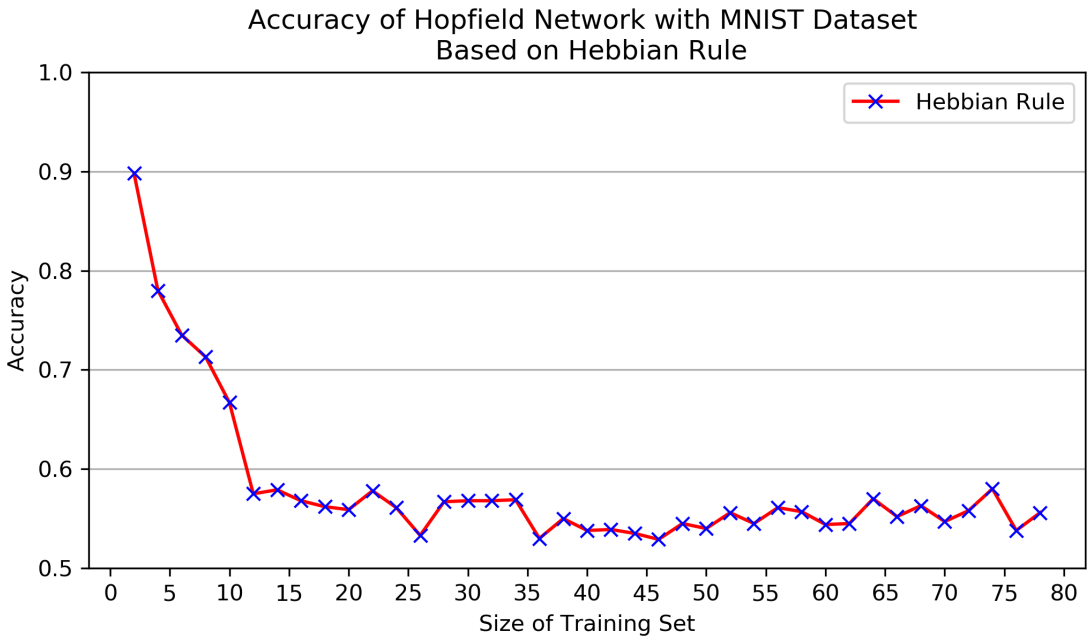
(a) Using the Scikit-learn utilities to load the MNIST data, implement a Hopfield network that can classify the image data for a subset of the handwritten digits. Subsample the data to only include images of '1' and '5'. Here, correct classification means that if we present an image of a '1' an image of a '1' will be recovered; however, it may not be the original image owing to the degenerate property of this type of network. You are expected to document classification accuracy as a function of the number of images used to train the network. Remember, a Hopfield network can only store approximately 0.15N patterns with the "one shot" learning described in the lecture (slides 58-74).

(b) **BONUS 10%** Improvements to basic Hopfield training have been proposed (Storkey 1997). Implement this improvement and contrast the accuracy with part (a).

Answer:

Part(a):

When a Hopfield network is implemented with Hebbian Rule, the model hit the highest accuracy(89.8%) at training set size 2 (one for label '1' and one for label '5'). As the size of training set grows, the classification accuracy shows a downward trend, and remain steadily at about 56% to 57%.

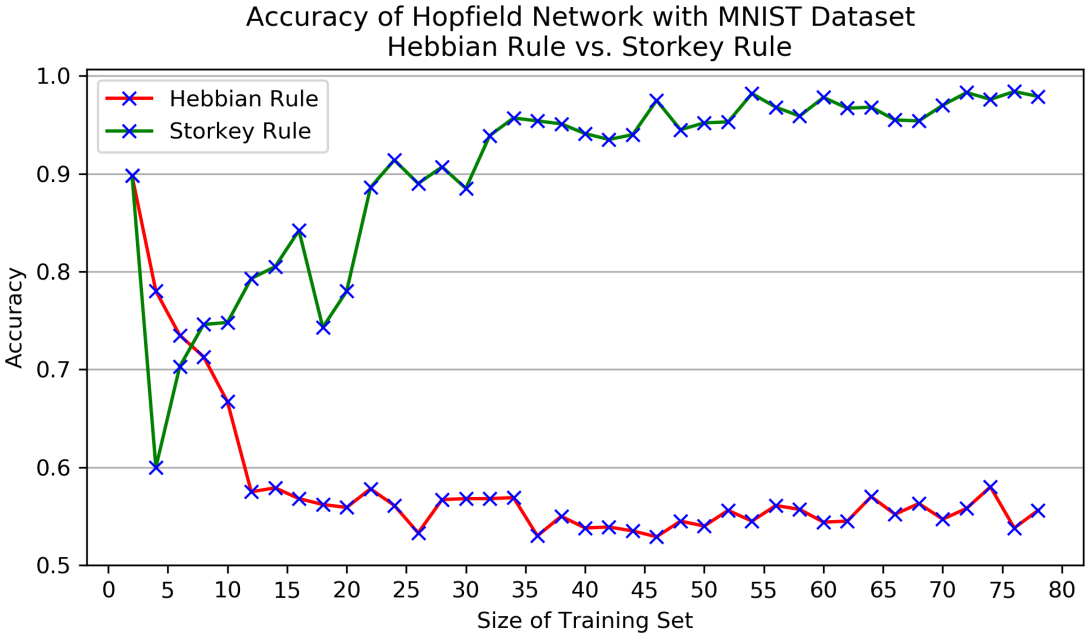


Part(b):

When a Hopfield network is implemented with Storkey Rule, the classification accuracy begin at a relatively same level(89.8%) as Hebbian Rule, and drop rapidly as the training size slightly increase. And then, the classification accuracy shows a rising trend as the size of training set grows, and finally remain steadily at about 95% to 97%.

Compare Hebbian Rule and Storkey Rule:
The classification accuracy of Hebbian Rule hits the highest accuracy when the training size is small, then the accuracy decrease as the size grows. But for Storkey Rule, the classification accuracy can reach a higher level as the training set size grows.

We can conclude that, when a Hopfield network is implemented with Storkey Rule, it's capable to store more patterns in compare of Hebbian Rule, and the performance of the model is much more better as the size grows.



Question 2 [50 marks]

Develop a feed forward RBF neural network in python that classifies the images found in the MNIST dataset. You are to train your neural network using backpropagation (simple gradient descent). You should use gaussian functions as your radial basis functions. You must show that you have:

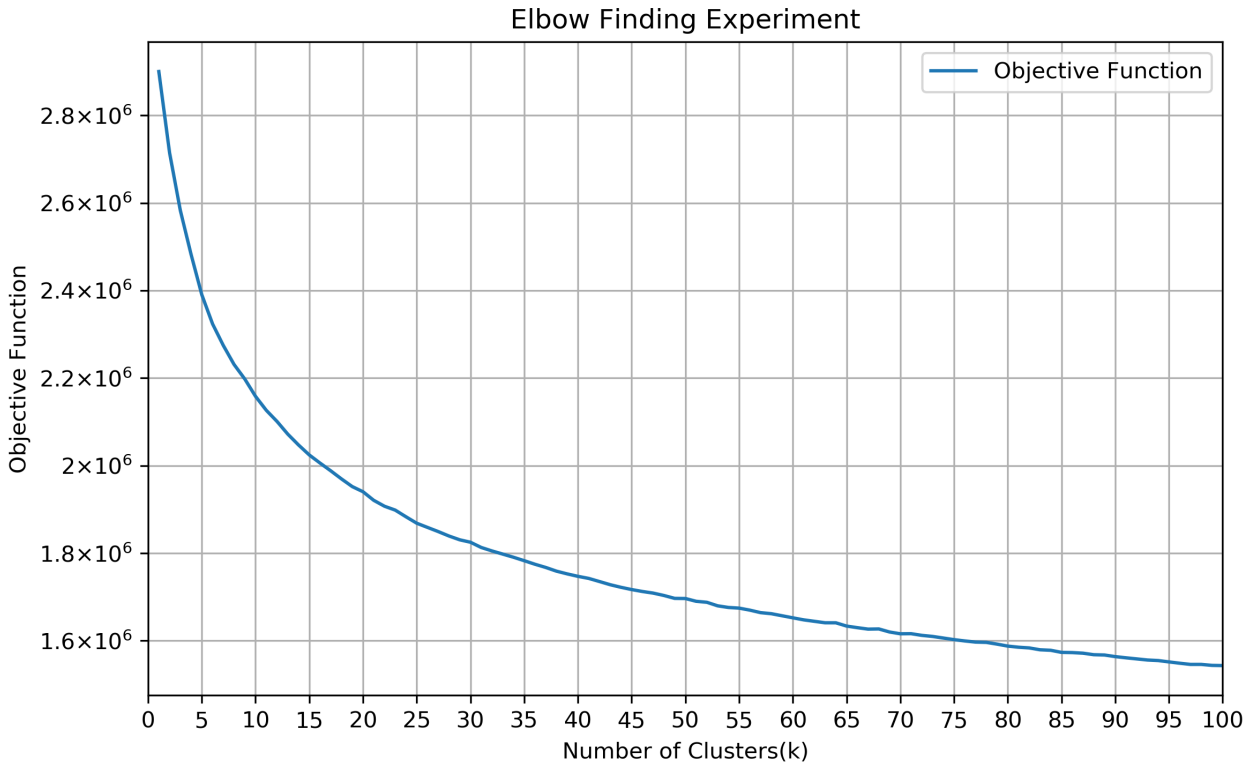
1. Used K-means to design the hidden layer in your network.
2. Performed K-fold cross correlation.
3. Investigated the performance of your neural network for different sizes of hidden layer.

Answer:

1. Used K-means to design the hidden layer in your network.

KMeans in Scikit-Learn is used in this experiment. To reduced the time cost, the hyper-parameter n_init is set to 3, which means that the result for each number k would be determine by taking the best output among 3 runs.

When K-means is applied to determine the hidden layer size in the RBF neural network, the result of "Elbow Finding" is shown in the following graph. The abrupt change is hard to detect in the graph, but we can infer that it probably happens at some value in the range [5,15].



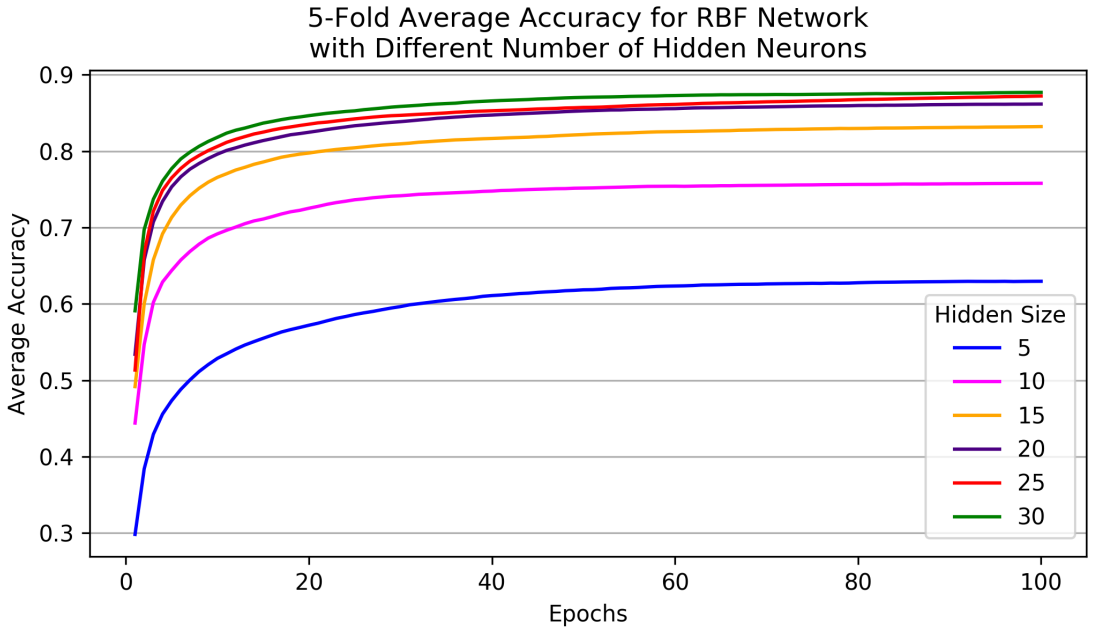
2. Performed K-fold cross correlation.
3. Investigated the performance of your neural network for different sizes of hidden layer.

These two questions would be answered together.

KFold in Scikit-Learn is applied in the experiment.

The performance of the model with hidden layer size 5, 10, 15, 20, 25, 30 would be explored via a 5-Fold cross validation. For each fold, 100 epoch would be trained. The mean accuracy for each epochs across 5 folds would be recorded to compare the performance of RBF network with hidden layer size.

The result of the experiment is shown as the blowing graph.



We can see that, with more neurons in the hidden layer, the model can converge at a more accurate performance.

Question 3 BONUS 20%

We can use self organizing maps as a substitute for K-means.
In Question 2, K-means was used to compute the number of hidden layer neurons to be used in an RBF network. Using a 2D self-organizing map compare the clusters when compared to K-means for the MNIST data. Sample the data to include only images of '1' and '5'. Use the scikit-learn utilities to load the data.
You are expected to

(a) document the dimensions of the SOM computed and the learning parameters used to generate it
(b) provide 2D plots of the regions for '1' and '5' for both the SOM and K-means solutions. You may project your K-means data using SVD to 2 dimensions for display purposes.

Answer:

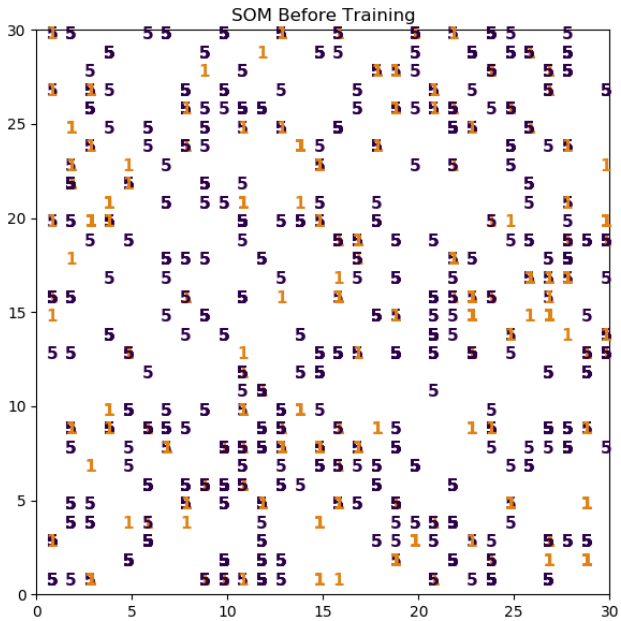
(a) document the dimensions of the SOM computed and the learning parameters used to generate it

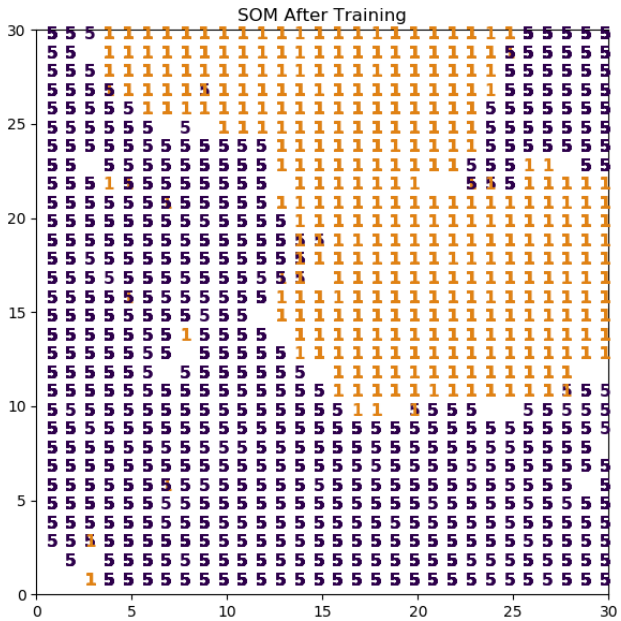
MiniSom is applied in the implementation of this question.
And https://github.com/JustGlowing/minisom/blob/master/examples/HandwrittenDigits.ipynb is referred in the part of drawing the graphs.

In my experiment, the hyper-parameters is set as below:
Dimension = 30×30, sigma = 4, learning rate = 0.5, Neighborhood Function = triangle

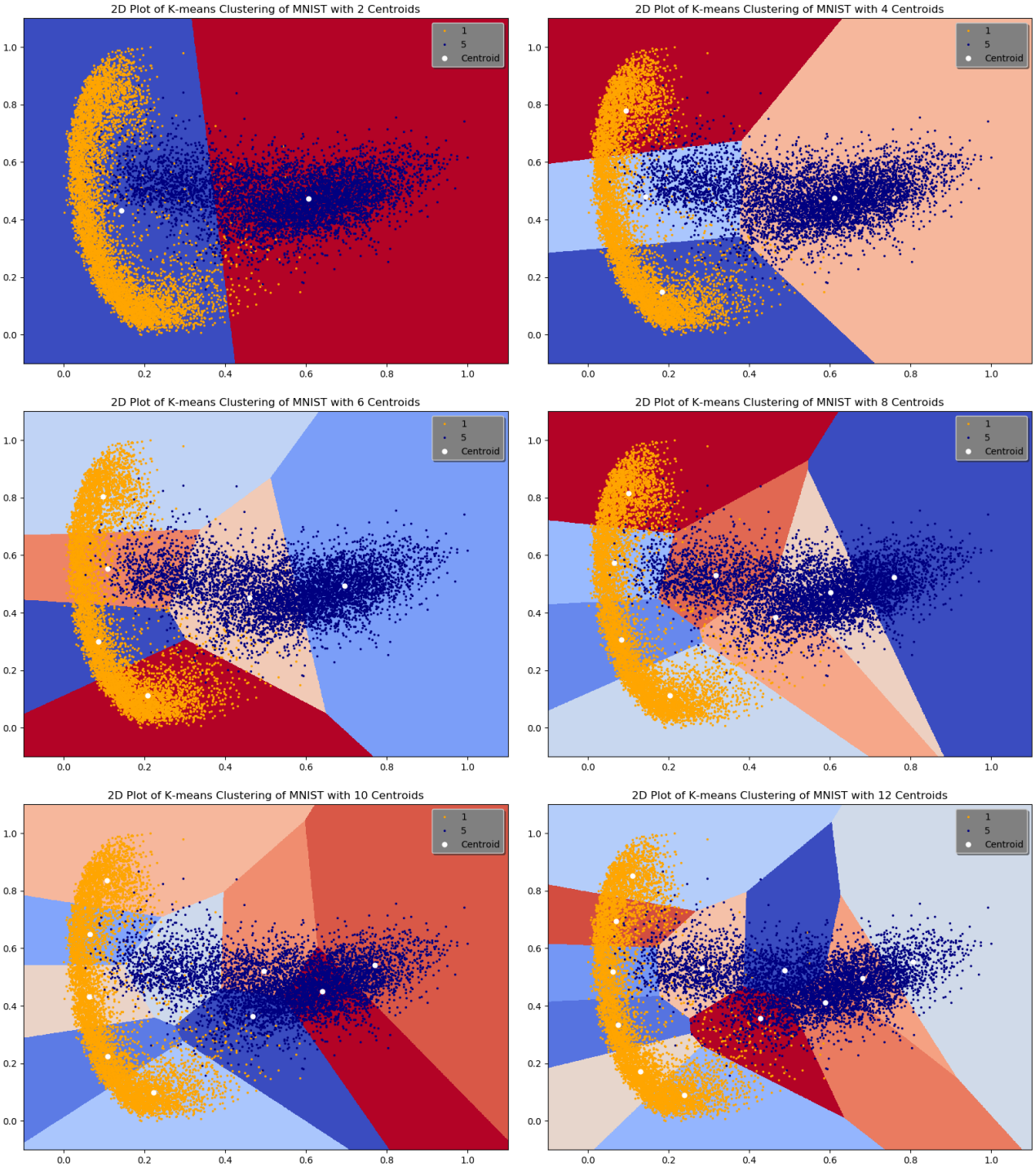
(b) provide 2D plots of the regions for '1' and '5' for both the SOM and K-means solutions. You may project your K-means data using SVD to 2 dimensions for display purposes.

The following two graphs show the 2D plots of the regions for '1' and '5' for SOM solutions:





The following graphs show the 2D plots of the regions for '1' and '5' for K-means solutions with 2, 4, 6, 8, 10, 12 centroids:



Question 4 BONUS 20%

Using Principal Component Analysis (PCA) and scikit-learn face data compare the classification accuracy of faces when using this orthonormal basis as input to a feed forward neural network when compared to using the raw data as input features. You are expected to document

(a) the size of your feed forward neural network in both cases and

(b) the prediction accuracy of your neural networks using a K-fold analysis.

Answer:

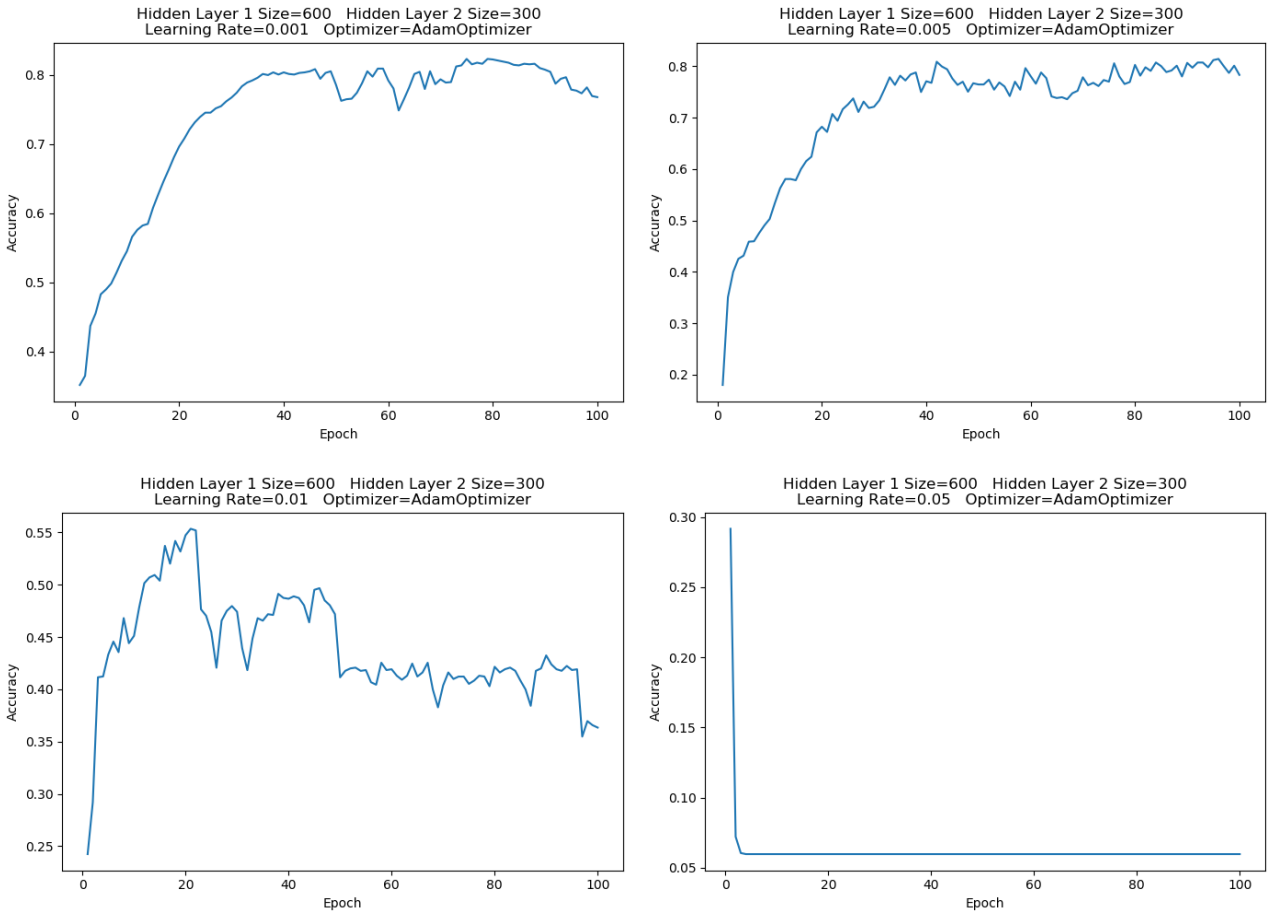
(a) the size of your feed forward neural network in both cases and

(b) the prediction accuracy of your neural networks using a K-fold analysis.

The two question would be answer together.
Notice: The accuracy stated in the all following part are average testing accuracy across 5-folds.

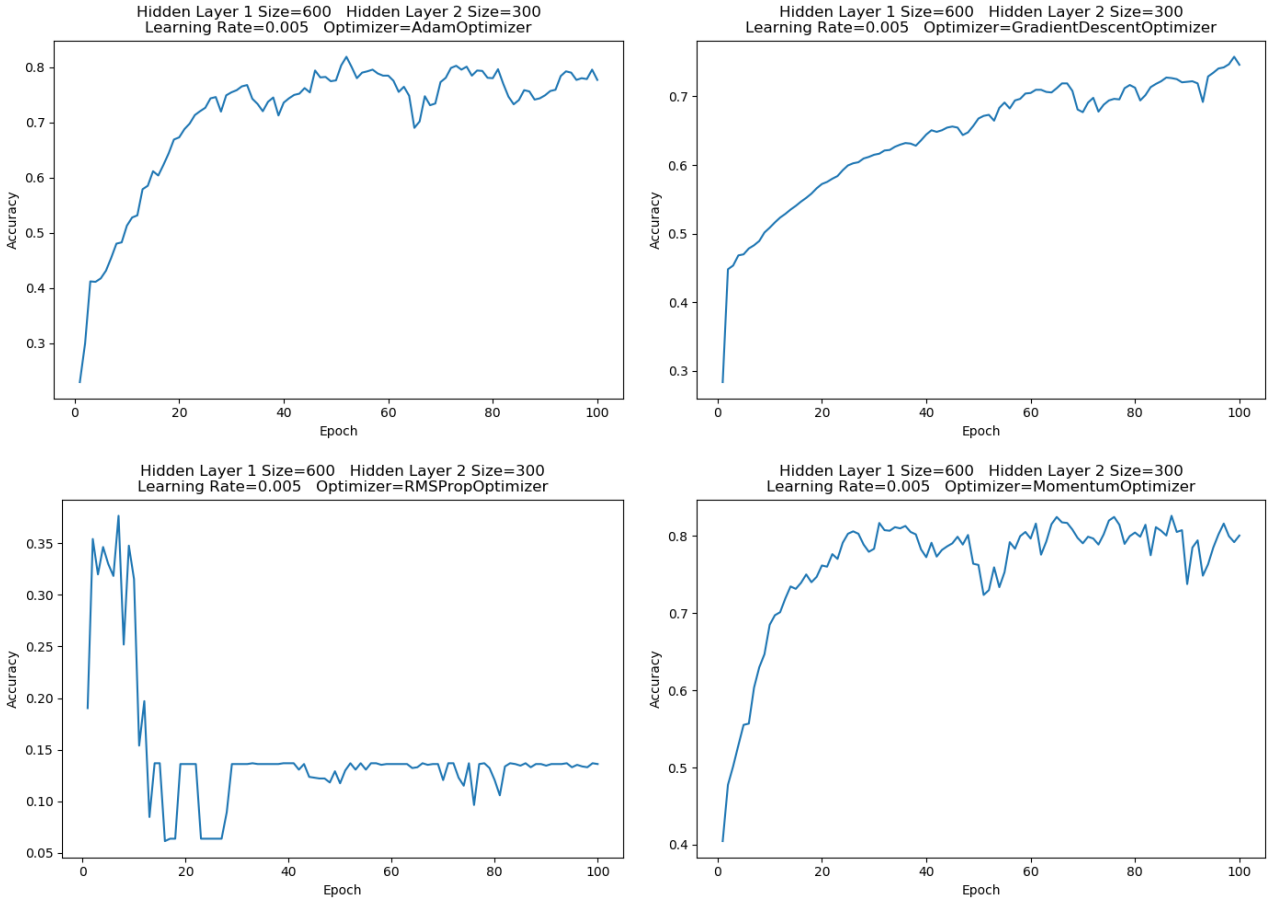
Data processed with PCA and without PCA would be tested.
Different learning rate, hidden layer size, and Optimizers would be tested.

Here are the results of my experiments:
First Hidden Layer Size = 600, Second Hidden Layer Size = 300
Optimizer=AdamOptimizer, and train without PCA.
Different learning rate is explored via a 5-Fold cross validation.



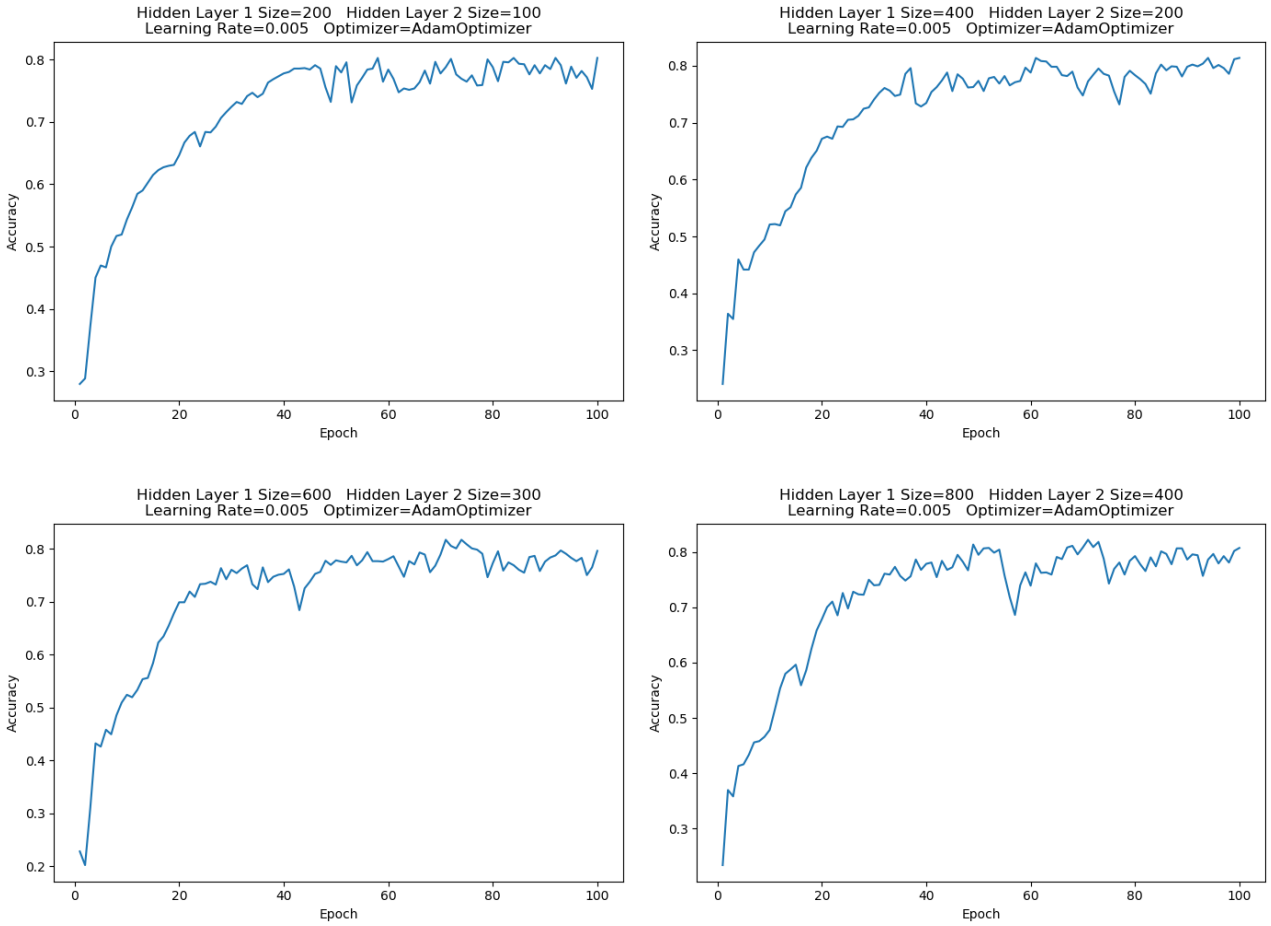
We can see that, with First Hidden Layer Size = 600, Second Hidden Layer Size = 300 and Optimizer=AdamOptimizer, 0.005 would be a good choice of learning rate

First Hidden Layer Size = 600, Second Hidden Layer Size = 300, Learning rate=0.005
Trained without PCA, performance of different Optimizers would be explored via a 5-fold cross validation:



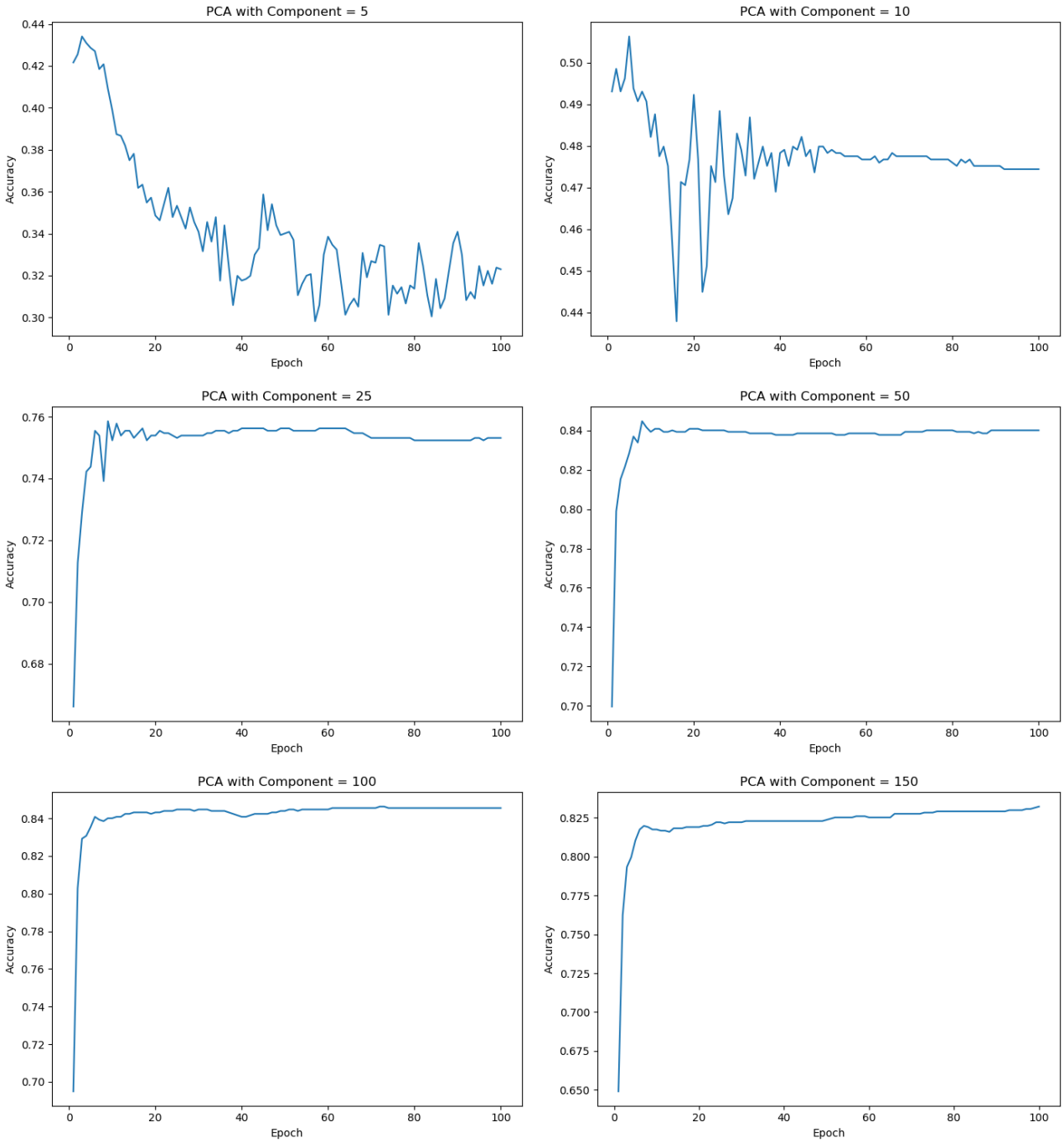
Compare the performances of different Optimizers, AdamOptimzier shows a relatively better performance.

Learning rate=0.005, trained without PCA, Optimizers=AdamOptimizer
Different Hidden Layer Size would be explored via a 5-fold cross validation:



We can see from the above graphs, with Optimizers=AdamOptimizer, Learning rate=0.005, and trained without PCA, the tested Hidden Layer Sizes gives very close predicting accuracy.

Processed data via PCA, First Hidden Layer Size = 600, Second Hidden Layer Size = 300
Optimizer=AdamOptimizer, Learning rate = 0.005
Performance of different PCA with different principle component would be explored via a 5-Fold cross validation:



From the above graphs, we can see that, PCA with a small size of component (5, 10) gives a bad testing performance, and PCA with more than 25 components gives very good testing accuracy, and the difference between size over 25 are very close.

Reference

1. https://github.com/JustGlowing/minisom/blob/master/examples/HandwrittenDigits.ipynb

2. https://matplotlib.org/3.2.0/gallery/images_contours_and_fields/image_masked.html#sphx-glr-gallery-images-contours-and-fields-image-masked-py

3. Storkey, Amos. "Increasing the capacity of a Hopfield network without sacrificing functionality." Artificial Neural Networks – ICANN'97 (1997)