University of Dayton

School of Engineering

Department of Electrical and Computer Engineering

ECE 477/595: Artificial Neural Networks

Project 3: Radial Basis Function (RBF) Neural Network Learning

Instructor: Prof. K. Asari Vijayan

Student Name: Serge Alhalbi **Student ID:** 101682971

Date: 10/10/2022

Methodology

Loading Steps

- Using the “loadMNISTImages” and “loadMNISTLabels” functions, we load the both the training and testing images with their labels from the uploaded files respectively.
  Initially, both the training and testing datasets are made up of images with pixel values ranging from 0 to 255 (8-bit pixel). These pixel values are then normalized (divided by 255), allowing each pixel to have a value between 0 and 1.

- A subset is created from the original datasets using the “balance_MNIST_selection” function. The function is simple and clear. I have written another function that can be used to select images at random from each dataset. The user may specify the number of training and testing images any of these functions.

K-means Algorithm

1. Input $X^s; s = 1,2,\dots,P$.

2. Randomly choose K images as mean vectors. (Cluster centers)
   $\mu_1, \mu_2, \dots, \mu_k, \dots, \mu_K$ (Size of each one is $N \times 1$ and $M \leq K \leq P$)

3. Compute the Euclidean distance of point $X^s; s = 1$ with respect to μ_k for $k = 1,2,\dots,K$:
   $$Dist_k^s = \frac{1}{N}\sum_{i=1}^{N}\left(X_i^s - \mu_{K_i}\right)^2 ; k = 1,2,\dots,K$$
   ($\frac{1}{N}$ can be ignored)

4. Find the minimum distance, then include X^s to the cluster with the minimum distance $Dist_k^s$.

5. Repeat steps 3 and 4 for $s = 2,\dots,P$.

6. Recompute μ_k for $k = 1,2,\dots,K$:
   $$\mu_k = \frac{1}{P_k}\sum_{s_k=1}^{P_k} X^{s_k};\ k = 1,2,\dots,K$$
   (X^{s_k} is an $N \times 1$ vector)

7. Compute the standard deviations of each cluster from the K-means algorithm:
   $$\sigma_k = \frac{1}{P_k}\sum\nolimits_{s_k=1}^{P_k}\left[\frac{1}{N}\sum_{i=1}^{N}\left(X_i^{s_k} - \mu_{k_i}\right)^2\right]^{\frac{1}{2}};\ k = 1,2,\dots,K$$
   ($\frac{1}{N}$ can be ignored)

   This step could be excluded for less computations if the X^{s_k} are to be saved in every iteration then used to get the standard deviation in the SLP algorithm.

8. Repeat steps 3 to 7 until the centers remain the same.
   The user can specify a maximum number of iterations as well.
   $\mu_k(t) = \mu_k(t-1);\ k = 1,2,\dots,K$

   Note: To avoid having a null standard deviation and thus dividing by 0 later in the SLP, avoid having only one member in a cluster.

K-means Elbow Algorithm

1. Pick a set of K values.

2. Choose the initial K value in the created set.

3. Apply the K-means algorithm to find $\mu_k;\ k = 1,2,\dots,K$.

4. Find the mean of intra class distances, this is the average distance from the members of a cluster to their cluster center:

   $$Mdist_k = \frac{1}{P_k}\sum_{s_k=1}^{P_k}\left[\frac{1}{N}\sum_{i=1}^{N}\left(X_i^{s_k} - \mu_{k_i}\right)^2\right]^{\frac{1}{2}};\ k = 1,2,\dots,K$$

   ($\frac{1}{N}$ can be ignored)

5. Compute the mean of mean inter class distances and save it for the elbow plot:

   $$Mean = \frac{1}{K}\sum_{k=1}^{K} Mdist_k$$

6. Choose the next K value in the set and repeat steps 3 to 5 until the iteration over the set is complete.

7. Plot the elbow graph using the K values as the x-axis and the $Mean$ as the y-axis.

8. Pick the Optimum K value where the elbow point is.

SLP – RBF Algorithm

Training:

1. Input the means μ_k and standard deviations σ_k for $k = 1,2,\dots,K$ from the K-means algorithm. As well, input the corresponding members X^{s_k} of each cluster.

2. Initialize some parameters

   a. Initialize the weights:
   $w_{jk};\ j = 1,2,\dots,M;\ k = 1,2,\dots,K.\ M$: output neurons & K: hidden neurons.
   (Size: $K \times M$)
   $-0.5 \le w_{jk} \le 0.5$

   b. Initialize the learning rate η. $(0 < \eta \ll 1)$

   c. Initialize the momentum term α. $(0 < \alpha \ll 1)$

   d. Initialize the first change of weights:
   $-0.5 \le \Delta w_{jk}(t_0) \le 0.5$ (Size: $K \times M$)

   e. The maximum iteration. $(= 1000)$

3. Loop over all the patterns and apply X^s for $s = 1,2,\dots,P$:

   a. Compute the hidden layer outputs:
   $u_k = e^{\left(\frac{-\|X^s - \mu_k\|^2}{2\sigma_k{}^2}\right)};\ k = 1,2,\dots,K$
   With: $\|X^s - \mu_k\|^2 = \frac{1}{N}\sum_{i=1}^{N}\left(X_i - \mu_{k_i}\right)^2$

   b. Compute the net for $j = 1,2,\dots,M$:
   $$Net_j = \sum_{k=1}^{K} w_{jk} \cdot u_k;\ j = 1,2,\dots,M$$
   (Size: $1 \times M$)

   c. Compute the actual outputs:
   $y_j = \frac{1}{1+e^{-Net_j}}\ ;\ j = 1,2,\dots,M$ (Size: $1 \times M$)
   (Size: $K \times M$) is to be used when implementing the change of weights.

   d. Compute the errors at the output layer:
   $e_j = d_j - y_j;\ j = 1,2,\dots,M$
   d_j is the given desired output. (Size: $1 \times M$)

(Size: $K \times M$) is to be used when implementing the change of weights.

e. Find the change of weights according to the verified and proven delta rule:
$\Delta w_{jk} = \eta e_j u_k;\ j = 1,2,\dots,M;\ k = 1,2,\dots,K$ (Size: $K \times M$)
η is a learning rate that determines how fast the change is ($0 < \eta < 1$). (hyperparameter)

f. Update the weights:
$w_{jk}(t+1) = w_{jk}(t) + (1-\alpha)\cdot\Delta w_{jk}(t) + \alpha\cdot\Delta w_{jk}(t-1);\ \forall k, j$
(Momentum was implemented in the previous project)

g. Compute the squared error of pattern s: (Add this squared error to the previous pattern squared error to accumulate all the pattern squared errors)

$$(e_j)^2 = \sum_{j=1}^{M}(d_j - y_j)^2$$

4. Compute the mean squared error: (for one complete data iteration)

$$E = \frac{1}{M\cdot P}\sum\nolimits_{j=1}^{M}\sum_{s=1}^{P}\left(e_j^s\right)^2$$

5. Repeat steps 3 to 4 until the mean squared error is less than the used threshold. (tolerance)
The user may define a maximum number of iterations. As well, the user may define a condition $E(t) = E(t-t')$ where the algorithm stops at iteration t'.

6. Store the weights and use all the mean squared errors to plot the E vs the number of iterations:
The user should expect the squared errors to decrease throughout the iterations.

Testing:

1. Input the means μ_k and standard deviations σ_k for $k = 1,2,\dots,K$ from the K-means algorithm. As well, input the weights $w_{jk}\ \forall k, j$.

2. Loop over all the patterns and apply X^{test} for $test = 1,2,\dots,P_{test}$:

   a. Compute $u_k^{test} = e^{\left(\frac{-\|X^{test}-\mu_k\|^2}{2\sigma_k{}^2}\right)}$; $k = 1,2,\dots,K$
   With: $\|X^{test} - \mu_k\|^2 = \frac{1}{N}\sum_{i=1}^{N}\left(X_i - \mu_{k_i}\right)^2$

   b. Compute the net for $j = 1,2,\dots,M$:
   $$Net_j = \sum_{k=1}^{K} w_{jk} \cdot u_k^{test};\ j = 1,2,\dots,M$$
   (Size: $1 \times M$)

   c. Compute the actual outputs:
   $y_j = \frac{1}{1+e^{-Net_j}}$; $j = 1,2,\dots,M$ (Size: $1 \times M$)
   (Size: $K \times M$) is to be used when implementing the change of weights.

   d. Find: the true positives, false negatives, false positives, and true negatives: (TP, FN, FP, and TN)

   e. Find: the true positive rate and the false positive rate: using the following formulas: (TPR and FPR)
   $$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$
   $$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}}$$

3. Plot the TPR and FPR vs the handwritten digits classes. (0 to 9)

Task 1: Results

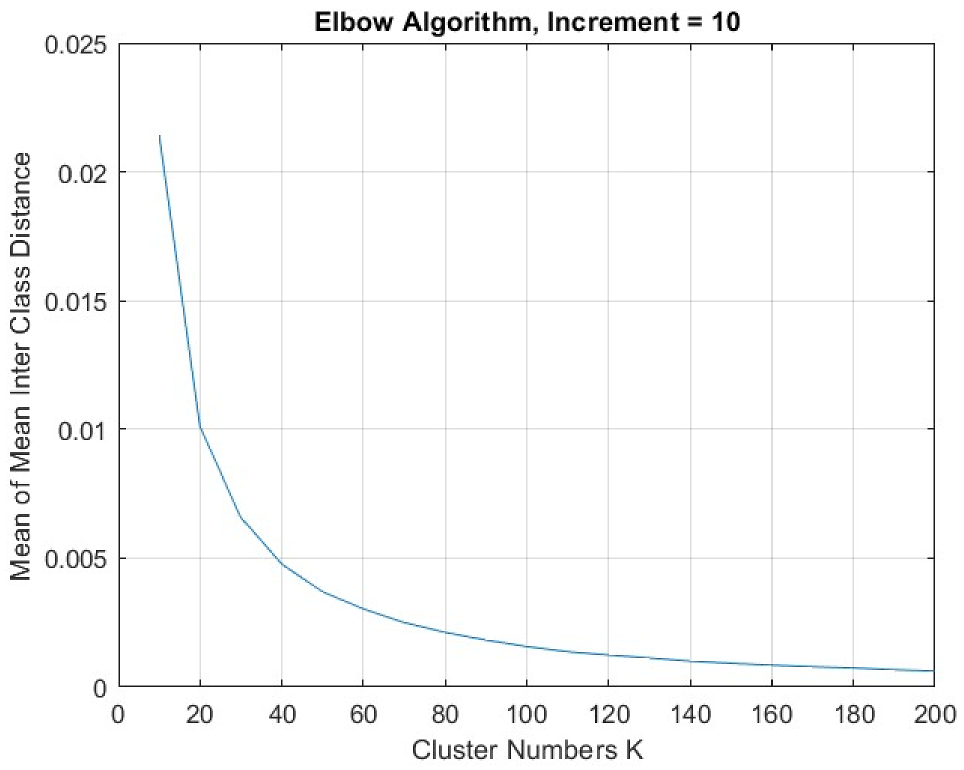

Figure 1: Clustering Score vs Clusters Number - 1000 training data, increment = 10, K optimum is around 50

Tasks 2 and 3: Results

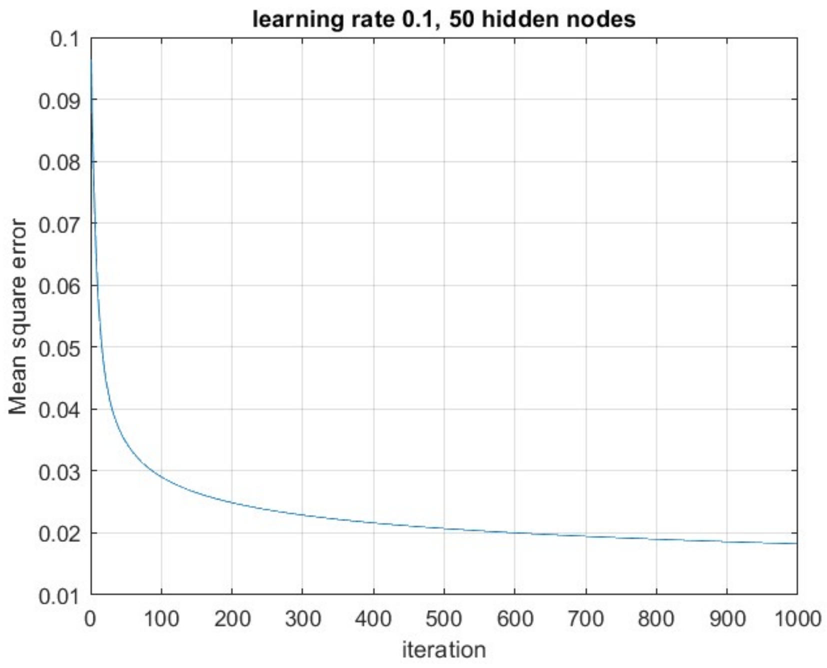

Figure 2: Error - 1000 training data, 300 testing data, learning rate = 0.1, hidden nodes = 50

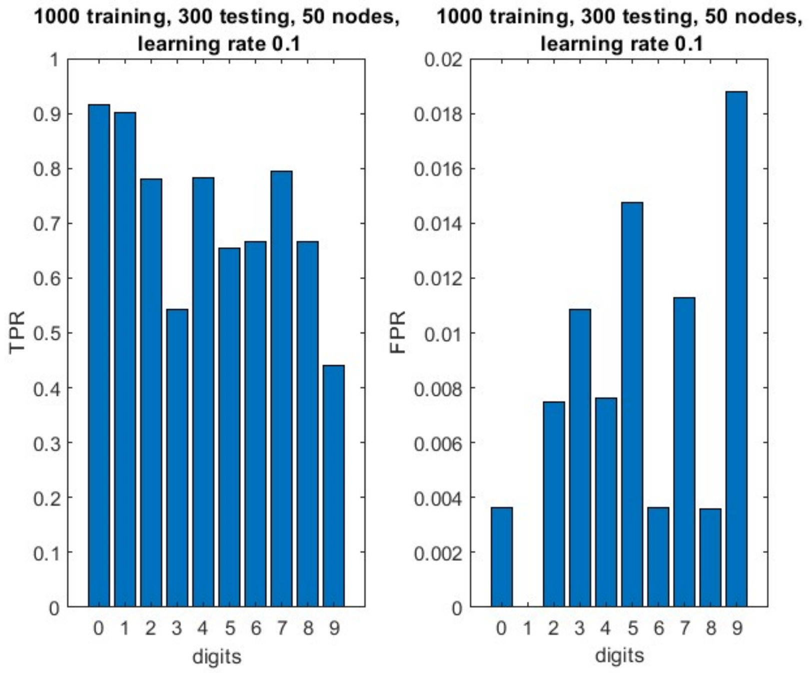

Figure 3: Plots - 1000 training data, 300 testing data, learning rate = 0.1, hidden nodes = 50

Tasks 4a and 4b: Results

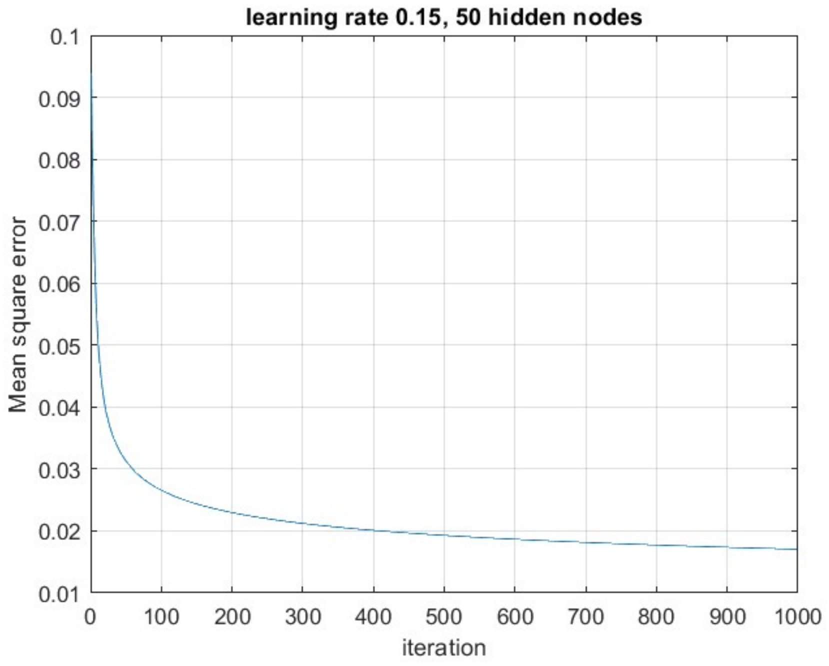

Figure 4: 1000 training data, 300 testing data, learning rate = 0.15, hidden nodes = 50

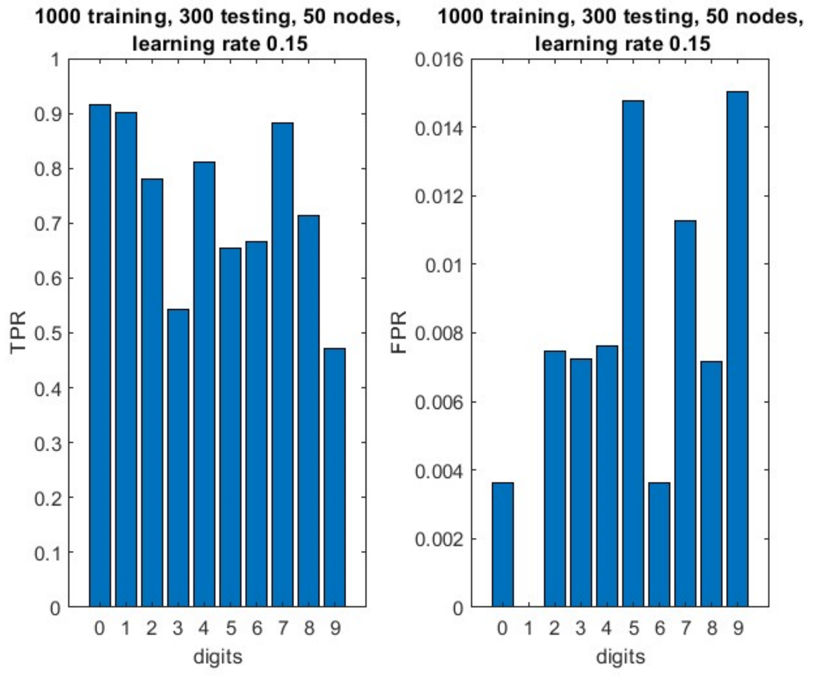

Figure 5: 1000 training data, 300 testing data, learning rate = 0.15, hidden nodes = 50

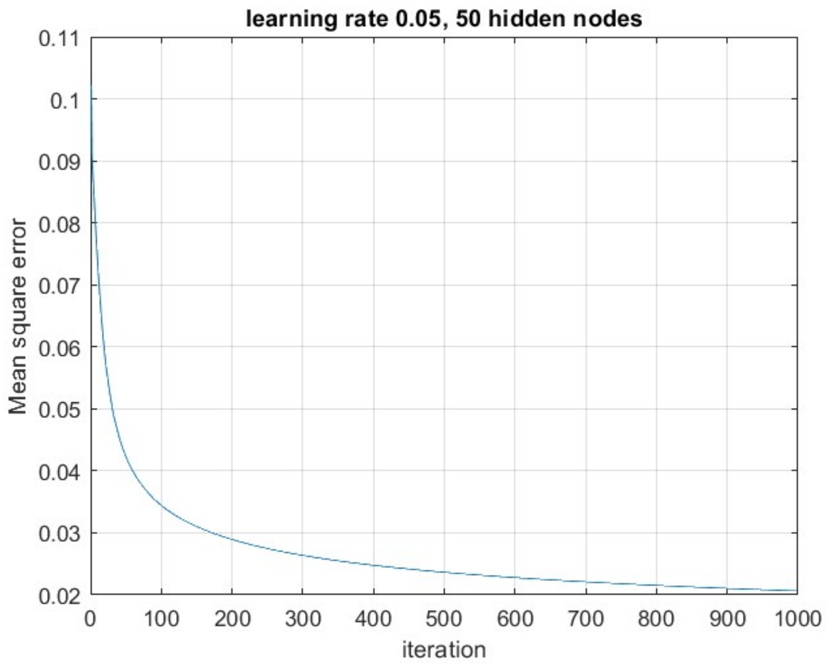

Figure 6: Error - 1000 training data, 300 testing data, learning rate = 0.05, hidden nodes = 50

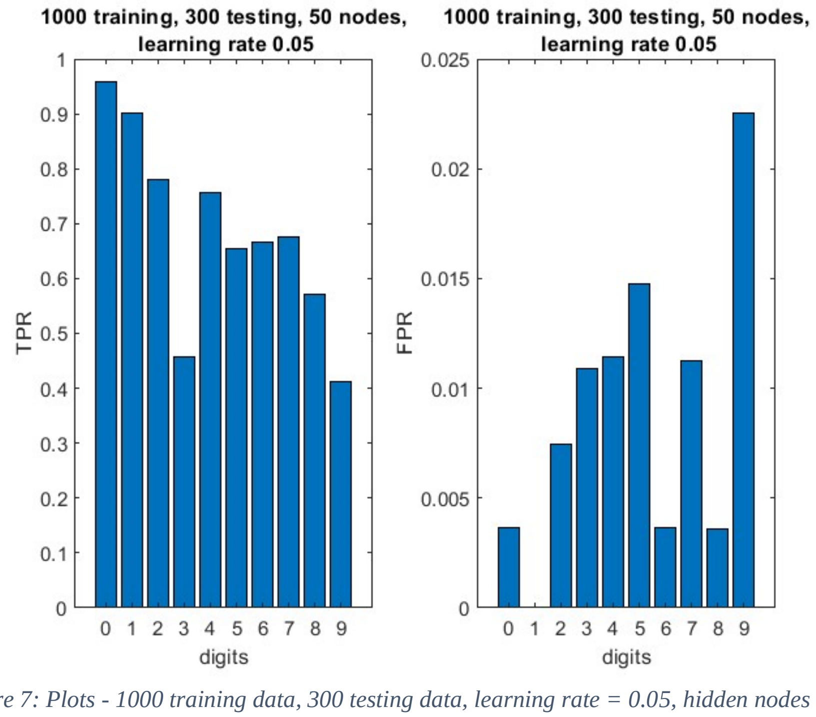

Figure 7: Plots - 1000 training data, 300 testing data, learning rate = 0.05, hidden nodes = 50

Tasks 5a and 5b: Results

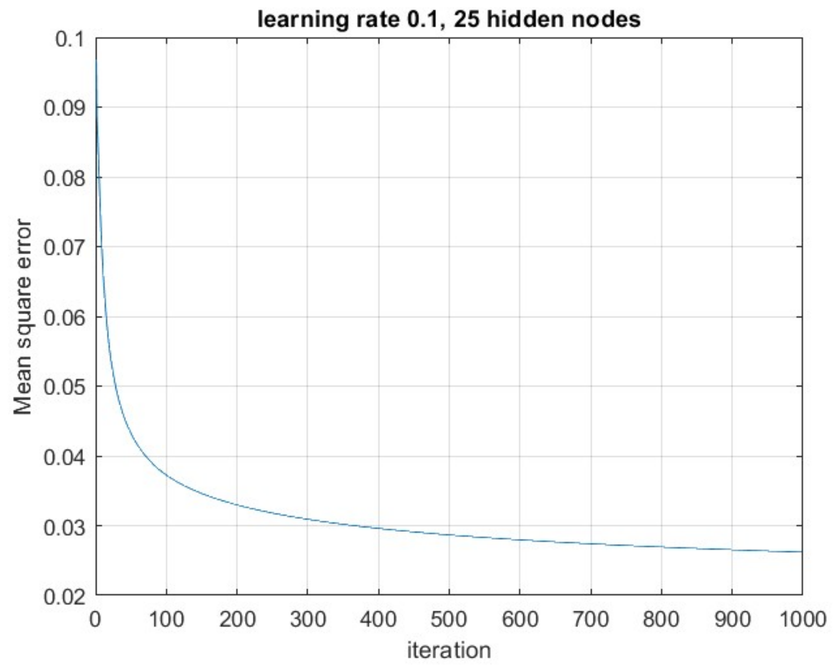

Figure 8: Error - 1000 training data, 300 testing data, learning rate = 0.1, hidden nodes = 25

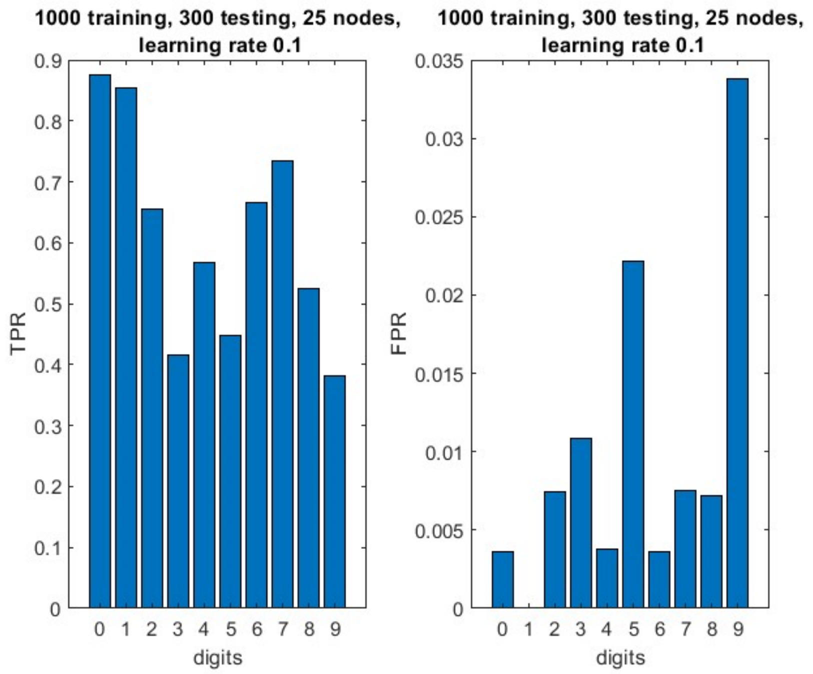

Figure 9: Plots - 1000 training data, 300 testing data, learning rate = 0.1, hidden nodes = 25

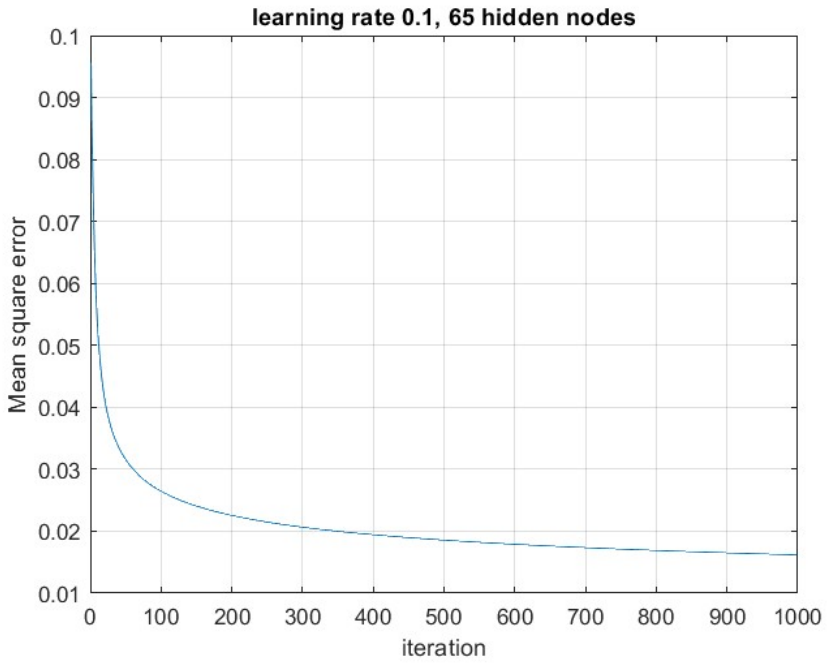

Figure 10: Error - 1000 training data, 300 testing data, learning rate = 0.1, hidden nodes = 65

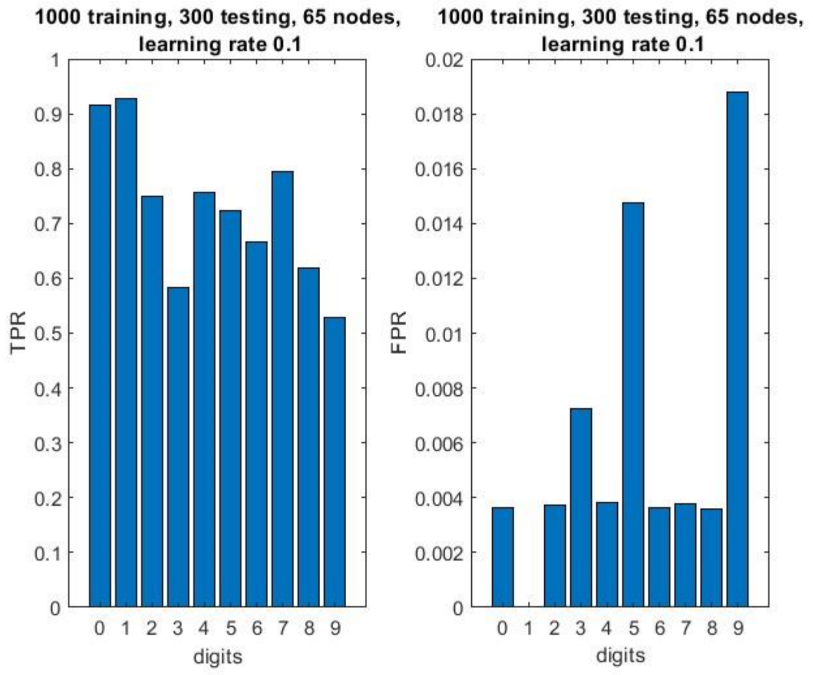

Figure 11: Plots - 1000 training data, 300 testing data, learning rate = 0.1, hidden nodes = 65

I. Conclusion/Comments

The goal of this assignment was met, and a radial basis Function (RBF) single-layer perceptron neural network learning was developed to classify handwritten images. Some points may be discussed below:

- It can be concluded from the first figure that the optimum K (elbow point) is about 50.

- All tasks and subtasks have a decreasing mean squared error as the number of iterations increases. This demonstrates that the machine only improves when the algorithm is repeated. This is evidence that the algorithm is reliable.

- Different sets of weights were computed for the same dataset with different parameters (learning rate, weights initialization, number of nodes or K in the hidden layer). This is reasonable since one parameter greatly affects the learning process.

- In all the cases mentioned above, the maximum number of iterations 1000 was reached each time.

- Tasks 1–3 demonstrate that the lower the learning rate, the slower the MSE decreases.

- When the number of hidden nodes in the hidden layer was less than the optimal $K = 50$ (25 hidden nodes), the MSE was almost 0.027. However, when K was greater than the optimum, the MSE was almost 0.017. This demonstrates that the error is minimized as the number of hidden nodes increases to an extent. This is logical because there are more nodes, and the system is more complex.

- For more complex datasets, more hidden layers could be built.