University *of* Dayton

School of Engineering

Department of Electrical and Computer Engineering

ECE 477/595: Artificial Neural Networks

Project 4: Adaptive Resonance Theory

Instructor: Prof. K. Asari Vijayan

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

Date: 10/20/2022

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.

Adaptive Resonance Theory Algorithm

Training:

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

2. Choose a vigilance parameter, a learning rate, and a choice parameter:
   $0 < \rho < 1; 0 < \beta \leq 1$.
   $\alpha \approx \varepsilon$ is a choice parameter to avoid dividing by 0.

3. Complement coding:
   $X^s := [X^s, 1 - X^s]; s = 1,2, \dots, P$.
   This solves the problem of weights dropping after being updated.

4. Initialize weights:
   $w_{ji};\ i = 1,2, \dots, N; j = 1$
   The weights could be initialized with values equal to one, or with zero values like in the implementation.

5. Loop over the training dataset: $X^s; s = 1,2, \dots, P$

   a. Compute the choice function for each node:
      $CF_j = \frac{\sum_{i=1}^{N} w_{ji} \wedge X_i^s}{\alpha + \sum_{i=1}^{N} w_{ji}}; j = 1,2, \dots, M.$

   b. Obtain the winner node:
      $J = \max_j CF_j$

   c. Compute the category choice:
      $CC_J = \frac{\sum_{i=1}^{N} w_{ji} \wedge X_i^s}{\alpha + \sum_{i=1}^{N} X_i^s}$

   d. Test the vigilance:
      i. If $CC_J \geq \rho$
         $w_{ji}^{New} = \beta \cdot w_{ji}^{Old} \wedge X_i^s + (1 - \beta) \cdot w_{ji}^{Old}; i = 1,2, \dots, N.$

      ii. If $CC_J < \rho$
         Reset node J, then find the next largest CF_j and name its winner node as J.
         Repeat step 8 until $CC_J \geq \rho$ if possible.

   e. If all existing M nodes fail:
      Create a new node $M = M + 1$.

The weights of new class will take the values of the pattern X^s.

f. Update the labels for the class chosen:
The class will have labels recorded from the training dataset.
Continue looping over the training dataset until it's done.

6. Record and save all the output labels:
Each class will have labels recorded from the training dataset.

7. Repeat steps 5 to 6 until the new output labels are the same as the old ones: (Convergence)
This way, the number of classes is fixed, and the patterns remain fixed in their classes.

Testing:

1. Testing on the training dataset:

    a. Define a "True" matrix of size 1 by 10 to capture the handwritten digit classes (0 to 9) true recognitions. Define a "False" matrix of size 1 by 10 to capture the handwritten digit classes (0 to 9) false recognitions.

    b. Loop over the training dataset: $X^s; s = 1,2, \dots, P$

        i. Compute the choice function for each node:
        $CF_j; j = 1,2, \dots, M.$

        ii. Obtain the winner node:
        $J = \max_j CF_j$

        iii. Record the recognition:
        Compare if the pattern belongs to this class or not. If true, add a one to the corresponding entry of the "True" matrix. If false, add a one to the corresponding entry of the "False" matrix.

    c. Compute the error percentage for each handwritten digit class:
    $$Error_{Class} = 1 - \frac{True_{Class}}{True_{Class} + False_{Class}}$$

    d. Calculate the average error percentage for each handwritten digit class:
    $$AverageError = \frac{1}{Number\ of\ Classes} \cdot \sum_{Class=1}^{Number\ of\ Classes=10} Error_{Class}$$

    e. Plot the $Error_{Class}$ versus the handwritten digit classes (0 to 9).

2. Testing on the testing dataset:

    a. Define a "True" matrix of size 1 by 10 to capture the handwritten digit classes (0 to 9) true recognitions. Define a "False" matrix of size 1 by 10 to capture the handwritten digit classes (0 to 9) false recognitions.

    b. Loop over the testing dataset: $X^{Test}; Test = 1,2, \dots, P_{Test}$

        i. Compute the choice function for each node:
        $CF_j; j = 1,2, \dots, M.$

        ii. Obtain the winner node:
        $J = \max_j CF_j$

        iii. Record the recognition:
        Compare if the pattern belongs to this class or not. If true, add a one to the corresponding entry of the "True" matrix. If false, add a one to the corresponding entry of the "False" matrix.

    c. Compute the error percentage for each handwritten digit class:
    $$Error_{Class} = 1 - \frac{True_{Class}}{True_{Class} + False_{Class}}$$

    d. Calculate the average error percentage for each handwritten digit class:
    $$AverageError = \frac{1}{Number\ of\ Classes} \cdot \sum_{Class=1}^{Number\ of\ Classes=10} Error_{Class}$$

    e. Plot the $Error_{Class}$ versus the handwritten digit classes (0 to 9).

Results

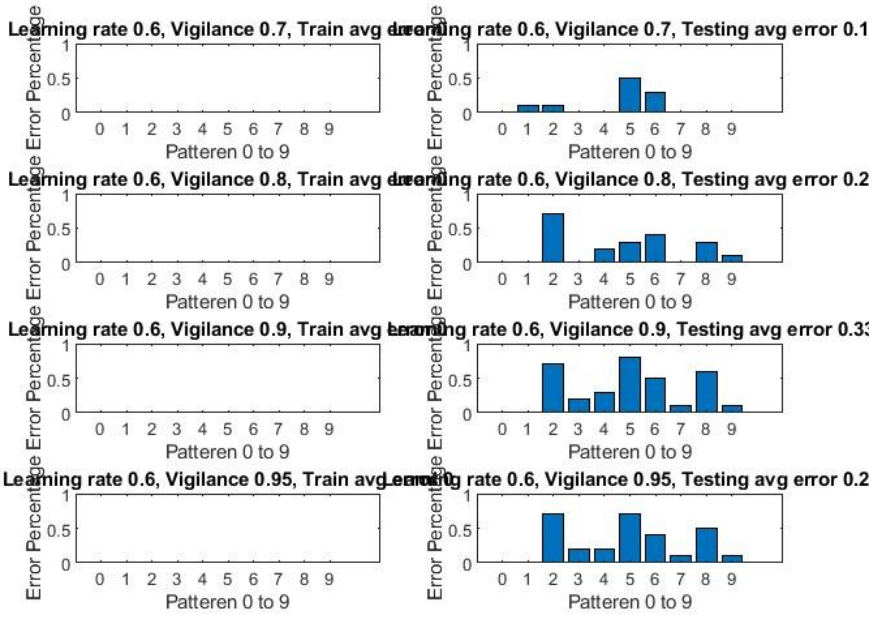

Figure 1: Percentage Error - 200 training data, 100 testing data, learning rate = 0.6, Complement coding

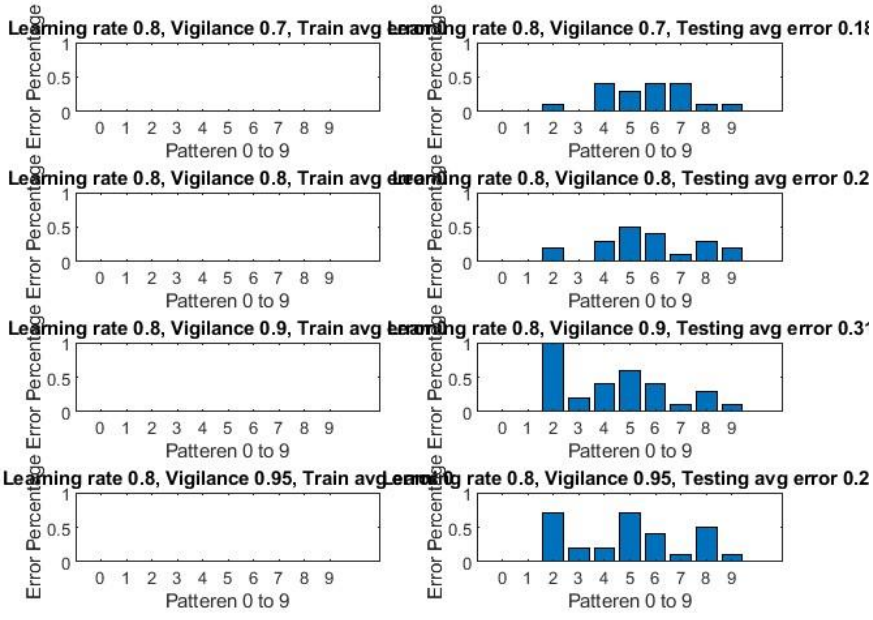

Figure 2: Percentage Error - 200 training data, 100 testing data, learning rate = 0.8, Complement coding

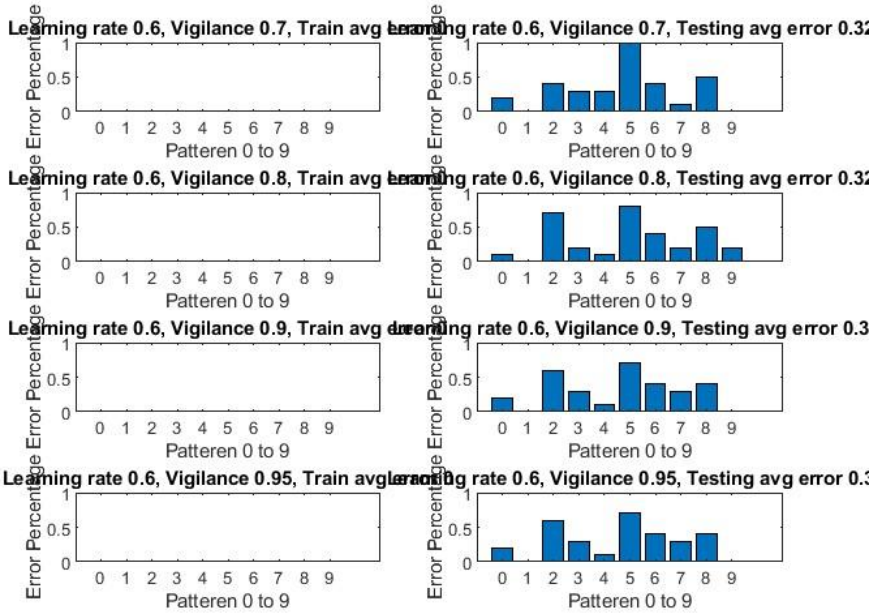

Figure 3: Percentage Error - 200 training data, 100 testing data, learning rate = 0.6, No complement coding

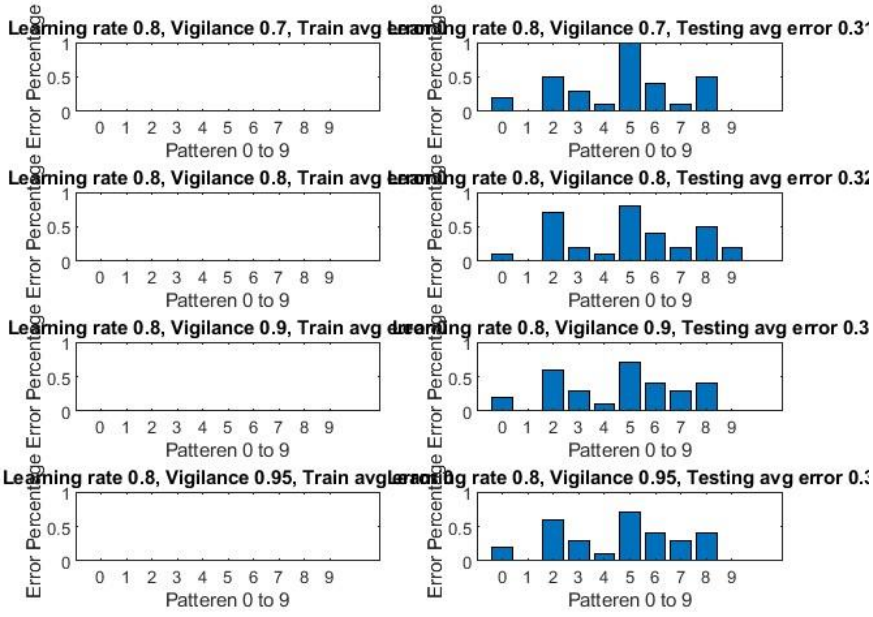

Figure 4: Percentage Error - 200 training data, 100 testing data, learning rate = 0.8, No complement coding

Conclusion/Comments

The goal of this assignment was met, and an adaptive resonance theory neural network learning was developed to classify handwritten images. Some points may be discussed below:

- In all the results, the training percentage error for all classes is always zero. This is very logical and necessary because otherwise, there would be patterns changing classes or creating new classes, and the system would not be stable, implying that no convergence was achieved.

- The optimal vigilance parameter appears to be 0.7, as the percentage error for each class is minimized with this value in all results.

- A new pattern introduced into the model only changes the weights of the class to which it belongs. In traditional neural networks, this is not the case.

- The handwritten digit class of number 5 has the highest percentage error in all results. This is because the number 5 shares features with the numbers 3, 6, and 8. This means that the model is more likely to misclassify patterns of this type.

- The vigilance parameter is critical for monitoring the number of classes targeted. The calibration of this parameter might be a way to supervise the unsupervised ART algorithm.

- When complement coding is used, the results are better, and the error is generally lower. This is reasonable as complement coding solves the problem of weights dropping fast reaching zeros after being updated.

- The weights drop faster without complement coding.

- It takes less time to run the experiments without complement coding. This is one disadvantage of using complement coding.