TBMI26 – Computer Assignment Report Supervised Learning

Deadline – March 15 2020

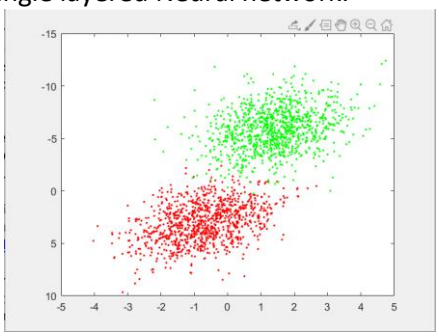
Author/-s: Vinay <vinbe289>

In order to pass the assignment you will need to answer the following questions and upload the document to LISAM. Please upload the document in PDF format. **You will also need to upload all code in .m-file format**. We will correct the reports continuously so feel free to send them as soon as possible. If you meet the deadline you will have the lab part of the course reported in LADOK together with the exam. If not, you'll get the lab part reported during the re-exam period.

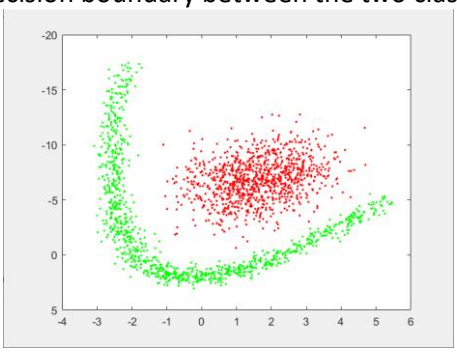
1. **Give an overview of the four datasets from a machine learning perspective. Consider if you need linear or non-linear classifiers etc.**

   Four different datasets are given here:

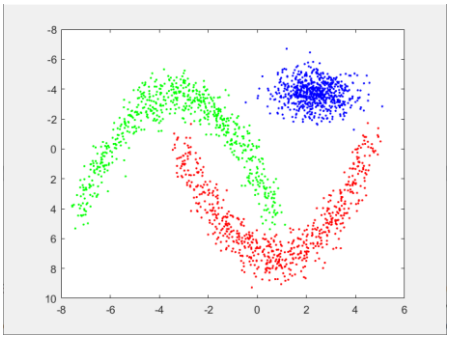
   - Dataset 1- Is a relatively simple dataset. It has two features and two classes. The data is two dimensional and is depicted in the image below. Though the two classes in the dataset is not linearly separable it is possible to gain a very good accuracy using simple models such as single layered Neural network.



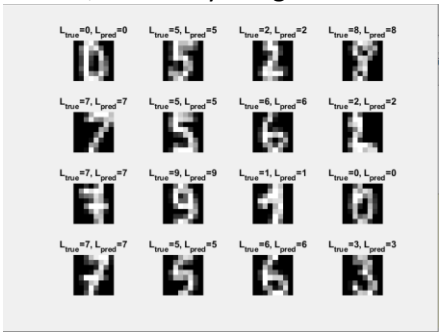
   - Dataset 2- Is a bit complex compared to the first dataset. It has two features and two classes. The data is two dimensional and is depicted in the image below. The two classes in the dataset is not linearly separable hence there is a need for a non-linear classifier to draw the decision boundary between the two classes.



- Dataset 3- Is a bit more complex compared to the second dataset. It has two features and three classes. The data is two dimensional and is depicted in the image below. The three classes in the dataset are not linearly separable hence similar to the second case, there is a need for a non-linear classifier such as a multi-layered neural network to draw the decision boundaries between the three classes.



- Dataset 4- Is OCR data which are hand written digits. This is the most complex among all the given datasets. It has 64 features and 10 classes. The data is depicted in the image below. The classes in the dataset is obviously not linearly separable, so we need a complex model such as multi-layered neural network, with more hidden layers than in the previous case, to classify the given dataset into 10 classes.



2. **Explain why the down sampling of the OCR data (done as pre-processing) result in a more robust feature representation. See http://archive.ics.uci.edu/ml/datasets/Optical+Recognition+of+Handwritten+Digits**

The increase in the number of features increases the complexity of the model, this can be taken care of by using dimensionality reduction techniques or by down sampling as in this case. Down sampling also reduces the noise in the data.

3. **Give a short summary of how you implemented the kNN algorithm.**

1. First we create a distance matrix which contains the distance from each point to all the other points
2. We extract the indices of the n nearest points for each point to be classified and use their class labels for voting
3. The point of interest is put into a class which has highest number of votes

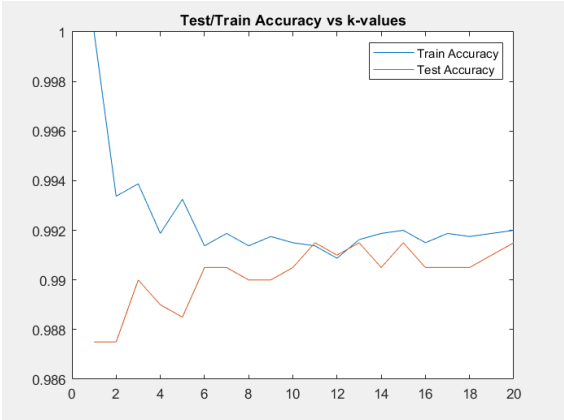
4. **Explain how you handle draws in kNN, e.g. with two classes (k = 2)?**

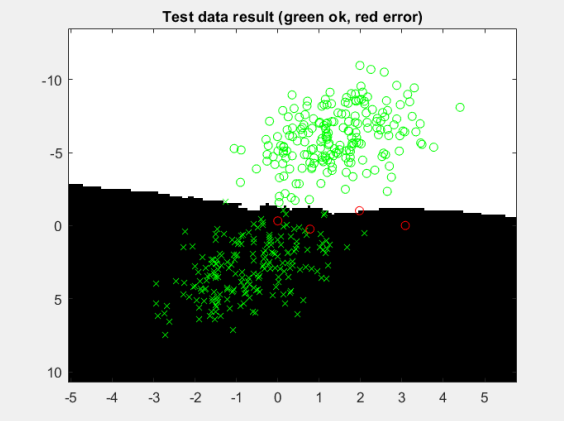
We are using mode function to find out the majority voting for the target data point. In case of a draw the class which has the lowest numeric value will be chosen and the target data point will be classified into that particular class.

5. **Explain how you selected the best k for each dataset using cross validation. Include the accuracy and images of your results for each dataset.**

The best k was selected by using a range of different values for k (from 1 to 20 in this case, it can be changed by setting a different value for maxK in the code) and checking which particular k value corresponds to the highest average accuracy. Five fold cross-validation was used here (which can be changed by setting a different value for numBins in the code).

Dataset 1:
Training data-
Best k - 1
Maximum average accuracy – 100%
Test data-
Best k - 11
Maximum average accuracy – 99.15%





Dataset 2:

Training data-

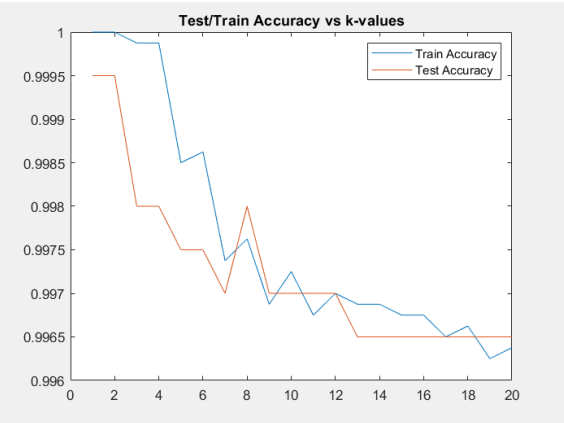
Best k - 1

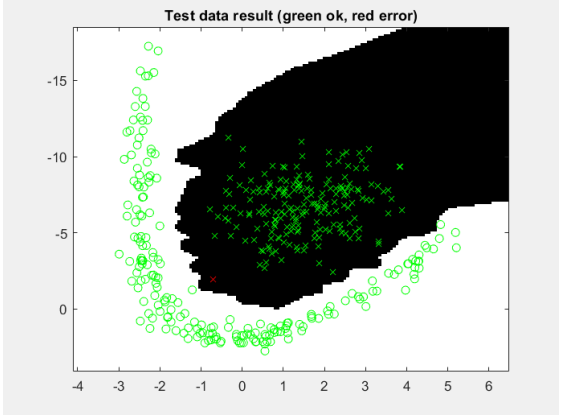
Maximum average accuracy – 100%

Test data-

Best k - 1

Maximum average accuracy – 99.95%





Dataset 3:

Training data-

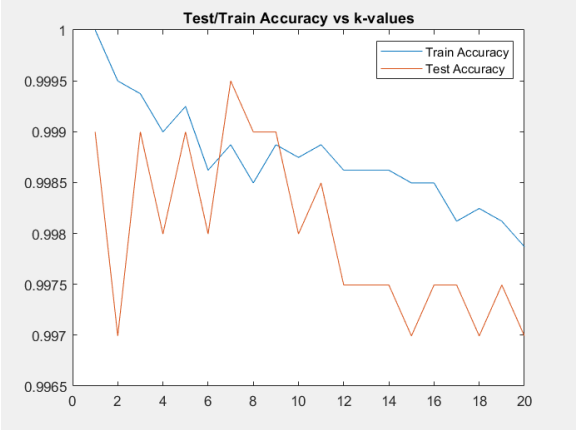
Best k - 1

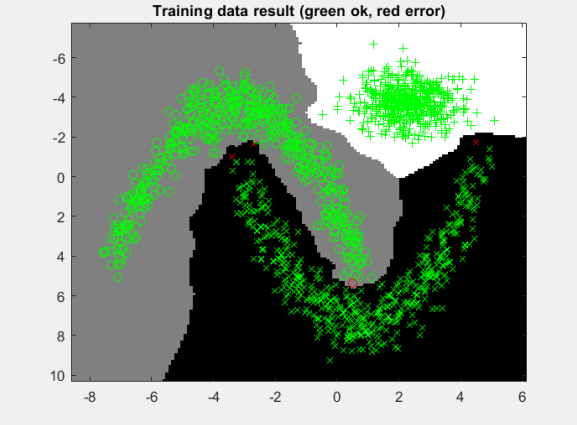
Maximum average accuracy – 100%

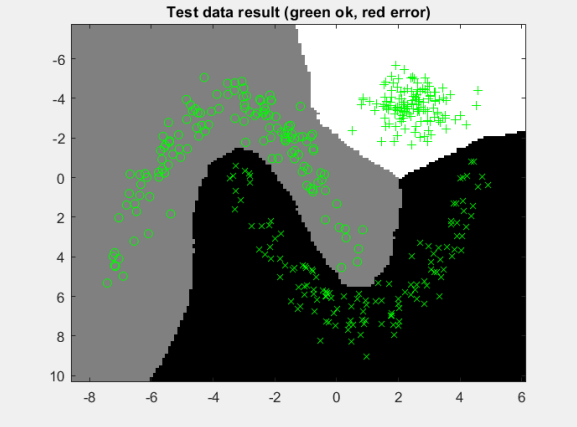
Test data-

Best k - 7

Maximum average accuracy – 99.95%







Dataset 4:

Training data-

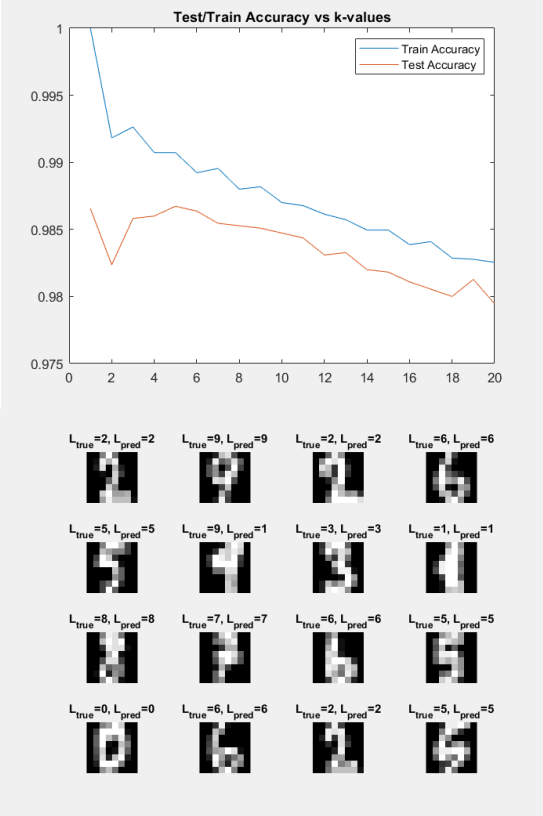
Best k - 1

Maximum average accuracy – 100%

Test data-

Best k - 5

Maximum average accuracy – 98.67%



6. **Give a short summary of your backprop implementations (single + multi). You do not need to derive the update rules.**

The number of input nodes depends on the number of features in the dataset and the number of output nodes depends on the number of classes in the dataset.

Dataset 1:

- Number of input nodes- 2 for features + 1 for bias
- Number of output nodes - 2 for unique classes

Dataset 2:

- Number of input nodes- 2 for features + 1 for bias
- Number of output nodes - 2 for unique classes

Dataset 3:

- Number of input nodes- 2 for features + 1 for bias
- Number of output nodes - 3 for unique classes

Dataset 4:

- Number of input nodes- 64 for features + 1 for bias
- Number of output nodes - 10 for unique classes

Error function: $E = \sum(Y - Exp)^2$

Single layer:

- $dE/dY = 2(Y - Exp)$
- $dE/dW = 2(Y - Exp)X$

Multi layer:

- $dE/dY = 2(Y - Exp)$
- $dE/dV = (Y - Exp)U$
- $dE/dW = (Y - Exp)V(1-U^2)X$

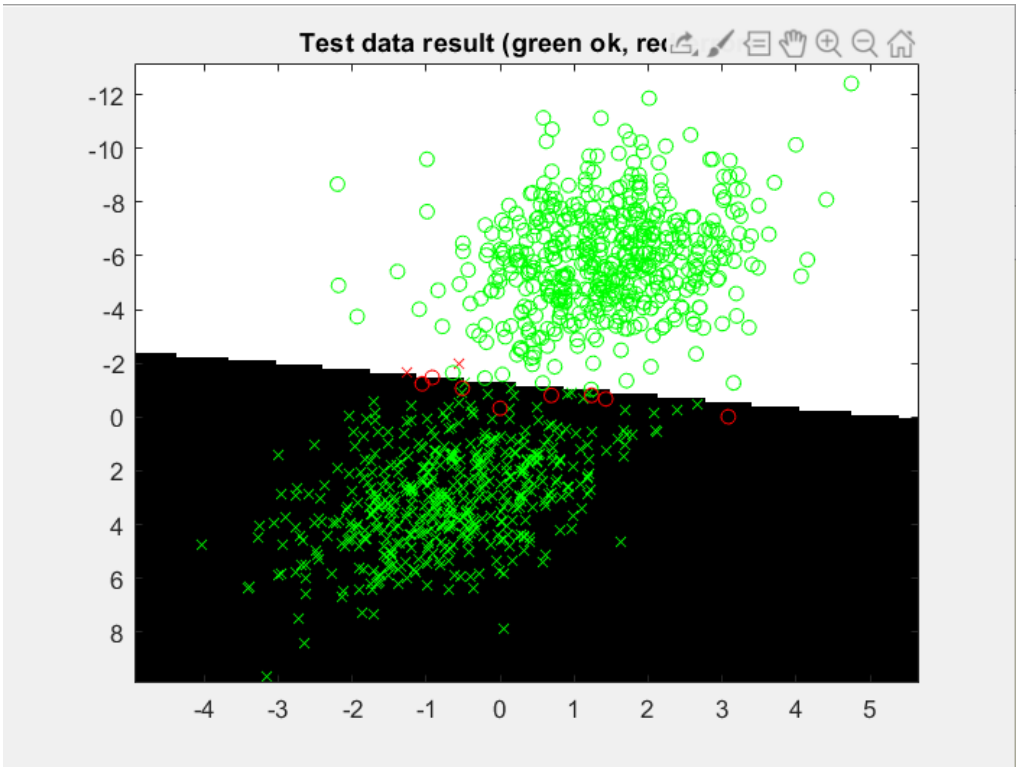
A single layer neural network will have an input layer, consisting of input nodes through which the feature vector (values) are fed into the network, and an output layer, consisting of output nodes one node for each class. There is only one set of parameter that is to be trained (1 weight matrix).
A multi layer neural network will have at least one hidden layer in between the input and the output layer with user specified number of nodes in the hidden layers. If there are N number of hidden layers between the input and the output layer then N+1 number of parameter matrices (weight matrices) are to be trained.

1. The weight matrices were initialized with Normally distributed pseudorandom numbers.
2. In both the cases forward pass was made and the error was calculated.
3. The error was back-propogated and the weight matrices were tuned accordingly to reduce the error in the proceeding iteration using gradient decent.
4. This is done recursively until a specified number of iterations is reached.

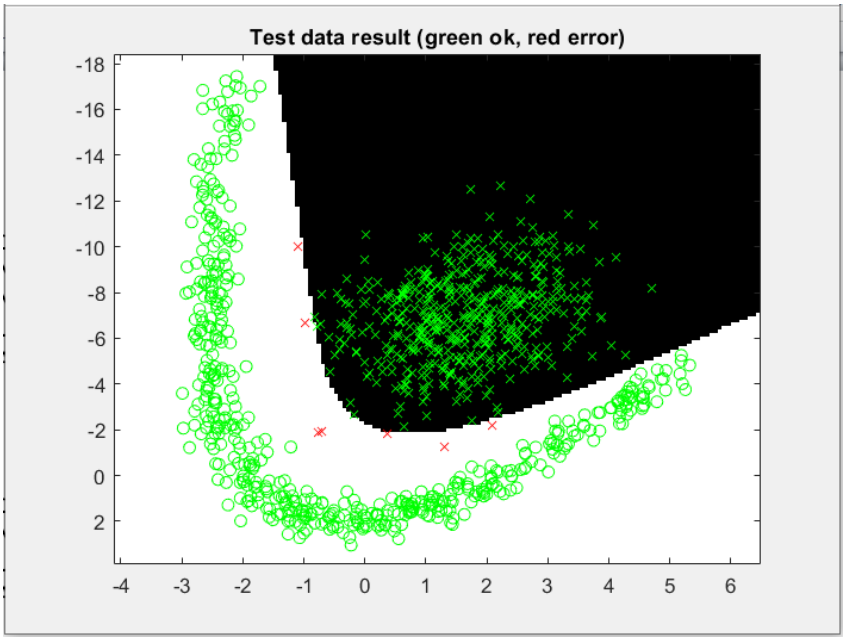
7. Present the results from the neural network training and how you reached the accuracy criteria for each dataset. Motivate your choice of network for each dataset. Explain how you selected good values for the learning rate, iterations and number of hidden neurons. Include images of your best result for each dataset, including parameters etc.

The accuracy criteria for each dataset was reached by changing the hyper-parameters like Number of iterations and learning rate.
A single layer neural network will be sufficient for dataset 1 since it is linearly separable and a multilayer model is required for datasets 2,3 and 4 since those are not linearly separable.
Good values for the learning rate, iterations and number of hidden neurons were selected by trial and error.

Dataset 1: (Single layer)
Number of iterations - 10000
Learning rate - 0.0001
Accuracy - 99%

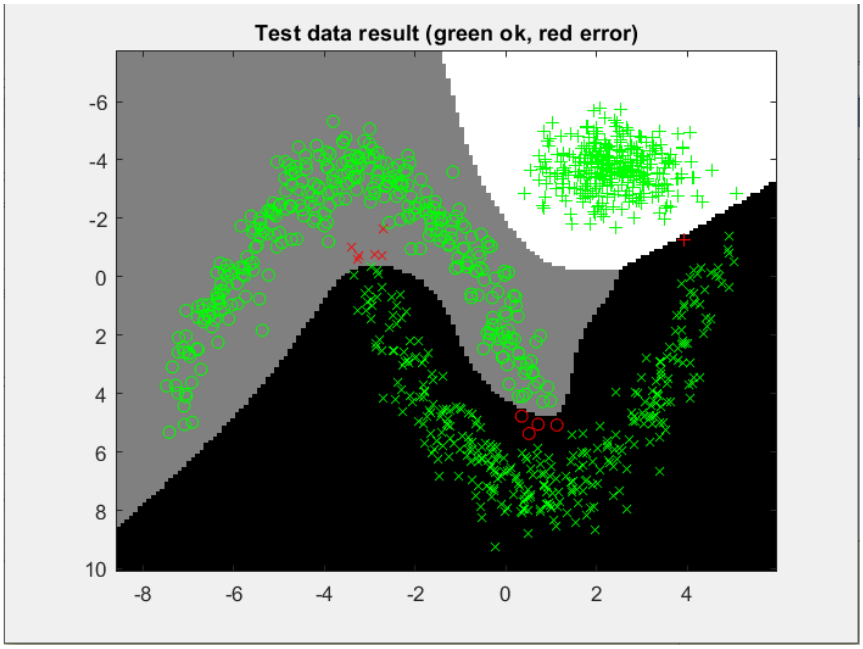


Dataset 2: (Multi-layer)
Number of neurons - 10
Number of iterations - 4000
Learning rate - 0.001
Accuracy - 99.3%

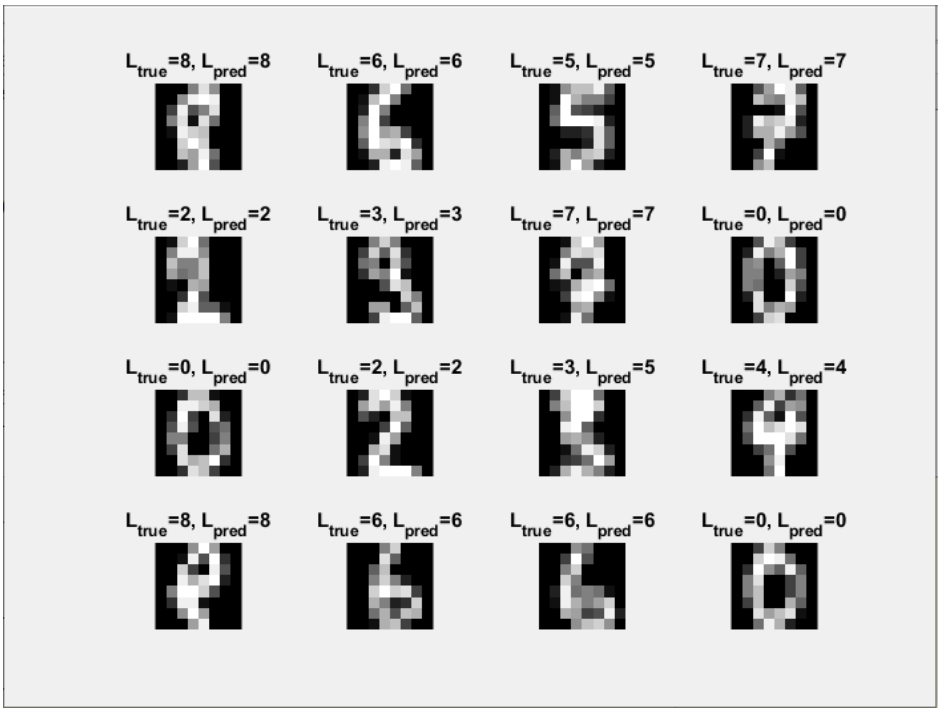


Dataset 3: (Multi-layer)
Number of neurons - 16

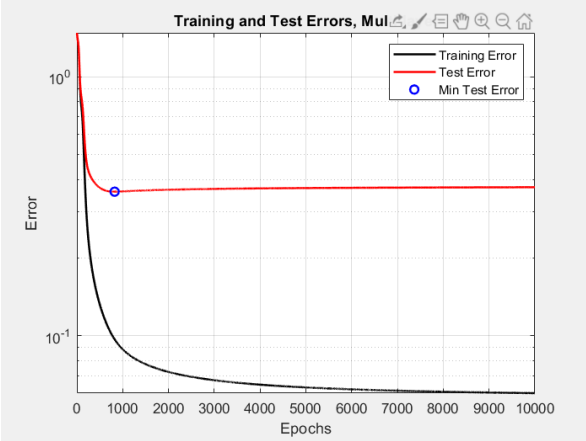
Number of iterations - 8000
Learning rate - 0.01
Accuracy- 98.9%



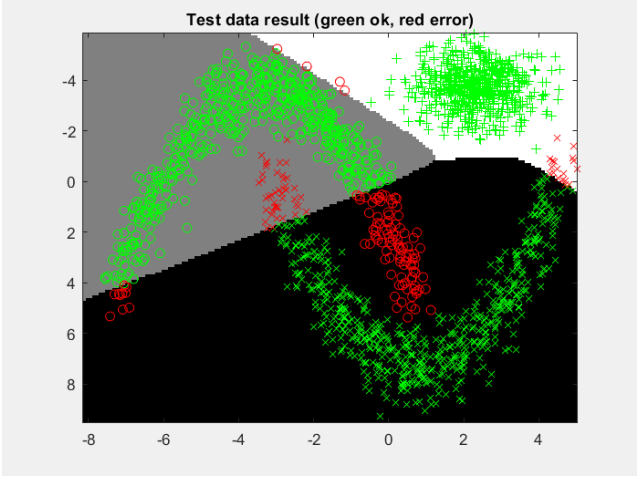
Dataset 4: (Multi-layer)
Number of neurons - 100
Number of iterations - 5000
Learning rate - 0.01
Accuracy - 97.8%



8. Present the results, including images, of your example of a non-generalizable backprop solution. Explain why this example is non-generalizable.



The number of bins are increased to 30 and only the first bin is used for training the network while the rest 29 bins are used for prediction. As seen in the figure above the Training error decreases with the increase in epochs while the Test error increases. This happens due to the fact that very few data points are used for training the network which cannot capture the pattern in the whole data fully. The Test error would further increase if the bins are divided into much more number of bins and if the network is trained only on the first bin and tested on the rest of the data.



9. **Give a final discussion and conclusion where you explain the differences between the performances of the different classifiers. Pros and cons etc.**

- KNN:
  K nearest neighbor is easy to implement and the training time is low. The classification task would be computationally demanding since the model has to remember all the training data to classify a given sample and for each test sample. K i.e. the number of nearest neighbors is a hyper-parameter that is to be specified by the user. It can be used to build model for datasets with relatively simpler patterns.

- Neural Network:
  Training a network is computationally demanding since the whole process of training a network is an iterative process. The training samples are forward propagated and the error is backward propagated through the network a number of times. There is no need for the model to remember all the training samples but just the weights. It is possible to build a model for relatively complex datasets.

10. **Do you think there is something that can improve the results? Pre-processing, algorithm-wise etc.**

    KNN:

    For draws we could check the class label of the K + 1'th nearest point or check the average distance of the target point with respect to the classes and assign the classes label of the class which is nearest to the target data point.

    Neural Network:
    We could use a deep network instead of a wide network for dataset 4. We could choose the hyper parameters by cross validation.