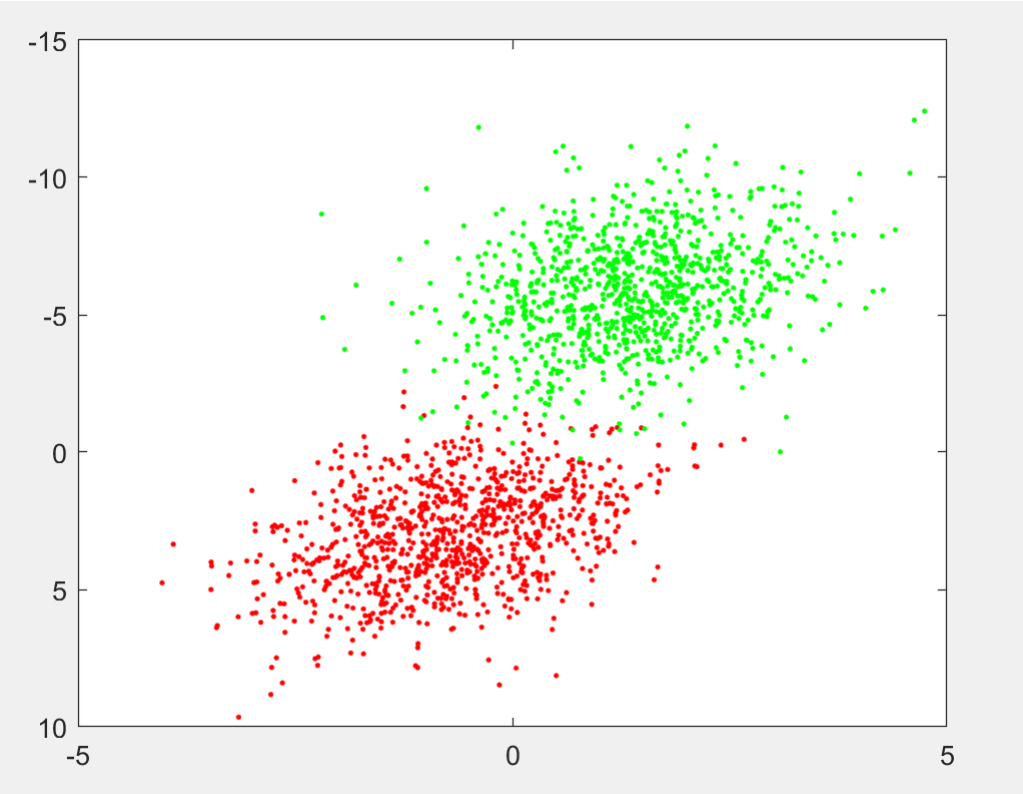
TBMI26 – Computer Assignment Report Supervised Learning

Deadline – March 15 2020

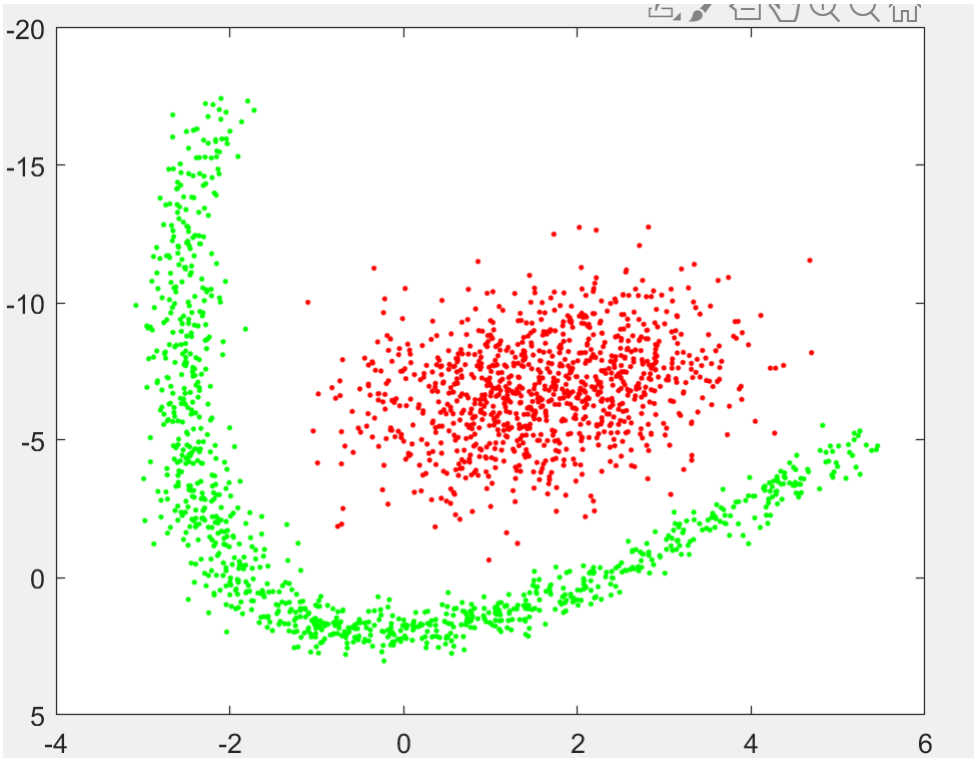
Author/-s: Gustav Wahlquist & William Stokke
Guswa204, Wilst664

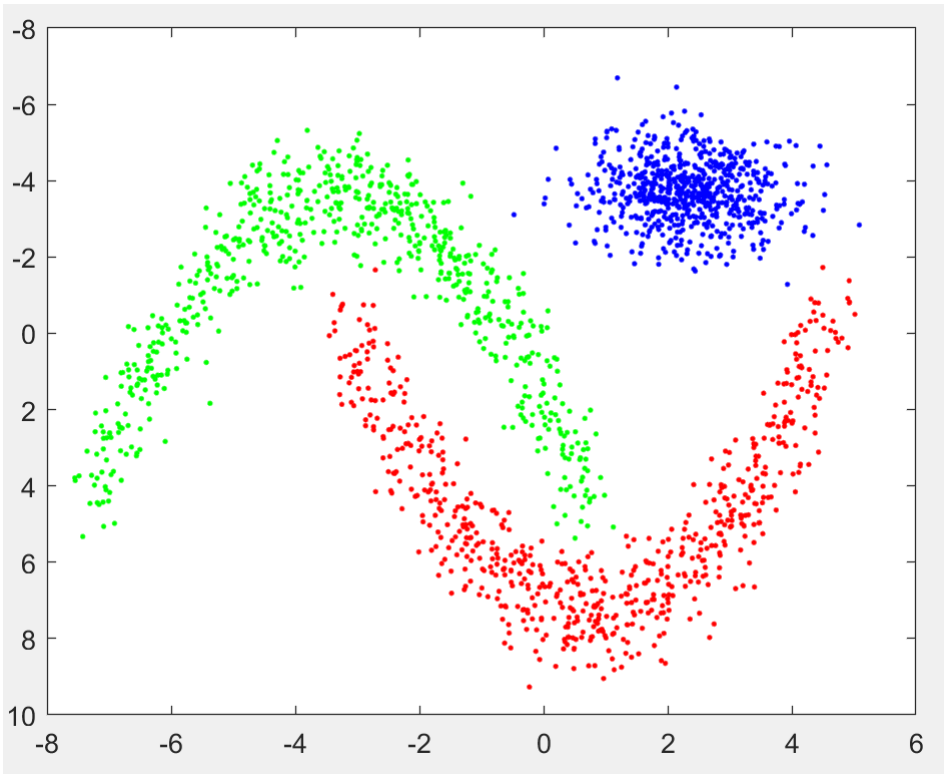
In order to pass the assignment you will need to answer the following questions and upload the document to LISAM. Please upload the document in PDF format. **You will also need to upload all code in .m-file format**. We will correct the reports continuously so feel free to send them as soon as possible. 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.**



The data seems to be linear separable since you could draw a line separating the 2 classes without to many wrongly classified training data points.





Datasets 2,3 are not linearly separable since you would need more than one or two lines to separate the classes without too many incorrect classifications.

Dataset 4 is harder to get an overview of and it is hard for us to know for sure if the data is linearly separable or not. Since we don't know we have to assume that the data could be not linearly separable.

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 variation between data from the same class will be smaller when you represent the data with a lower resolution.

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

   - For every data point that we want to classify with our algorithm, we first compute the Euclidian distance from that point to all points in the training set.
   - Then we find the K-nearest points from the training set to the data point we want to classify by finding the k points that has the smallest Euclidian distances. This is done by first identifying the smallest distance, saving the data point, setting this one to infinity and repeating the process for k times.
   - We then perform the majority vote by counting which class is most common in the set of k points we previously picked out. If the majority vote should end in a draw between multiple classes the data point is classified as the closest point that belongs to a class that is in the draw.

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

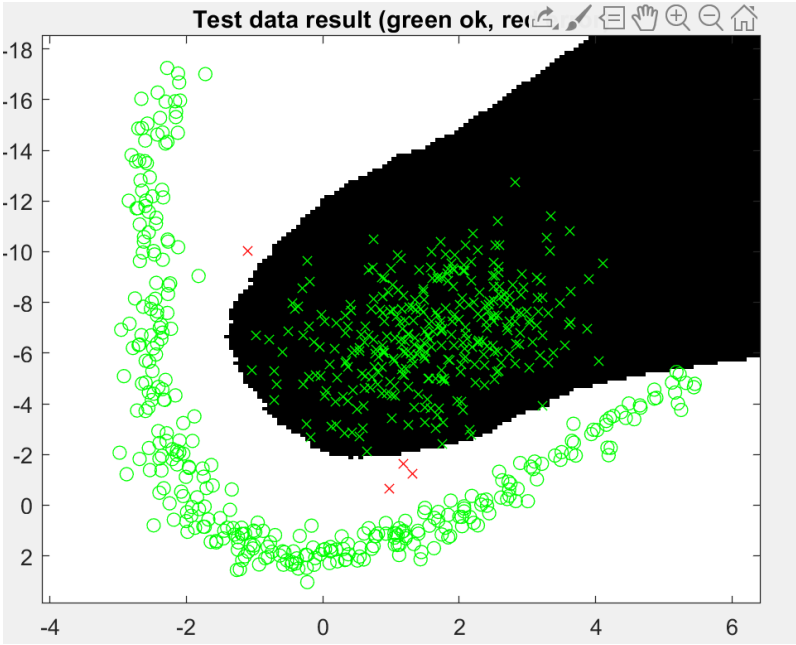
   One way to do it is to choose the closest one in the case k=2.

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.**

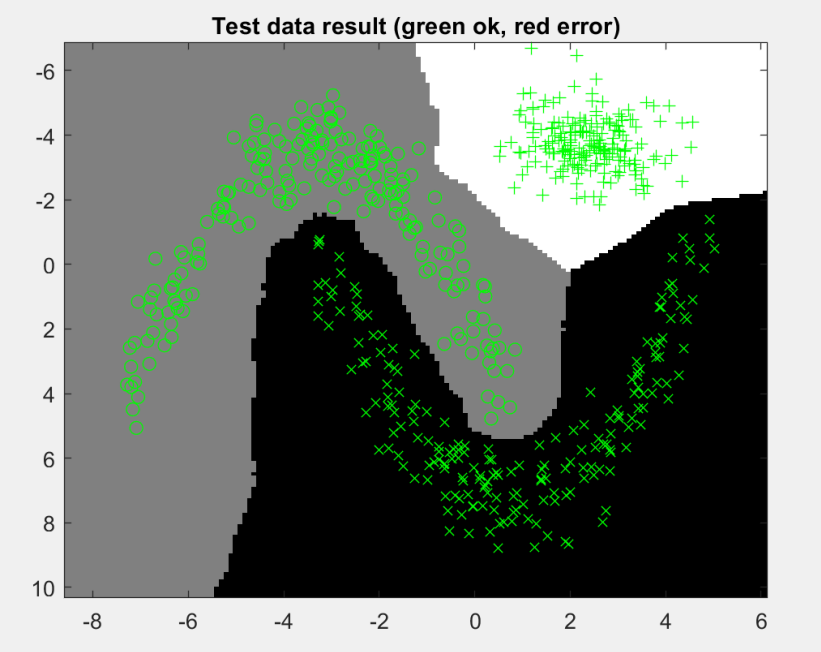
   We loop through a reasonable number of values for k (from 1 to n) and compare the accuracies for different values for k and then we choose the one with the best accuracy. When we measure the accuracy for each k, we do a 3-fold cross validation and calculate the average accuracy of these three runs as the accuracy for the respective k.

   Dataset 1:
   K = 20
   Acc = 99.25%

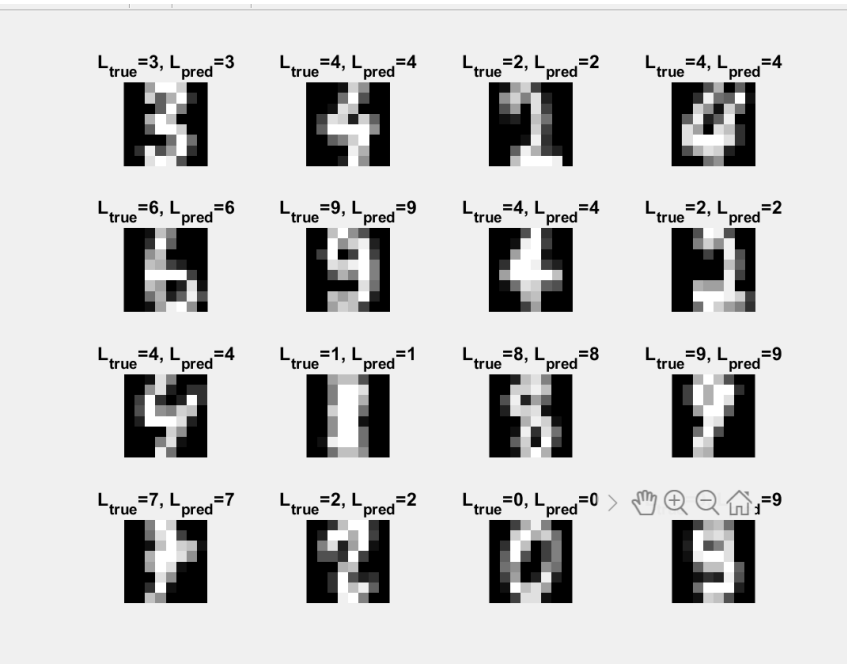
Dataset 2:
K = 2
Acc = 99.90%



Dataset 3:
K = 4
Acc = 99.83%



Dataset 4:
K = 3
Acc = 98.79%



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

Single layer

- Calculate initial error in order to be able to plot the error and number of iterations.
- Then we loop for number of iterations
    - First, we calculate the gradient of the error with respect to the weight matrix W.
    - Then we update W with the gradient scaled with the learning rate parameter (Taking a learning step).

- Then the errors are evaluated in order to be able to plot the error and number of iterations.
- This is repeated for “number of iteration” times.

Multiple layer

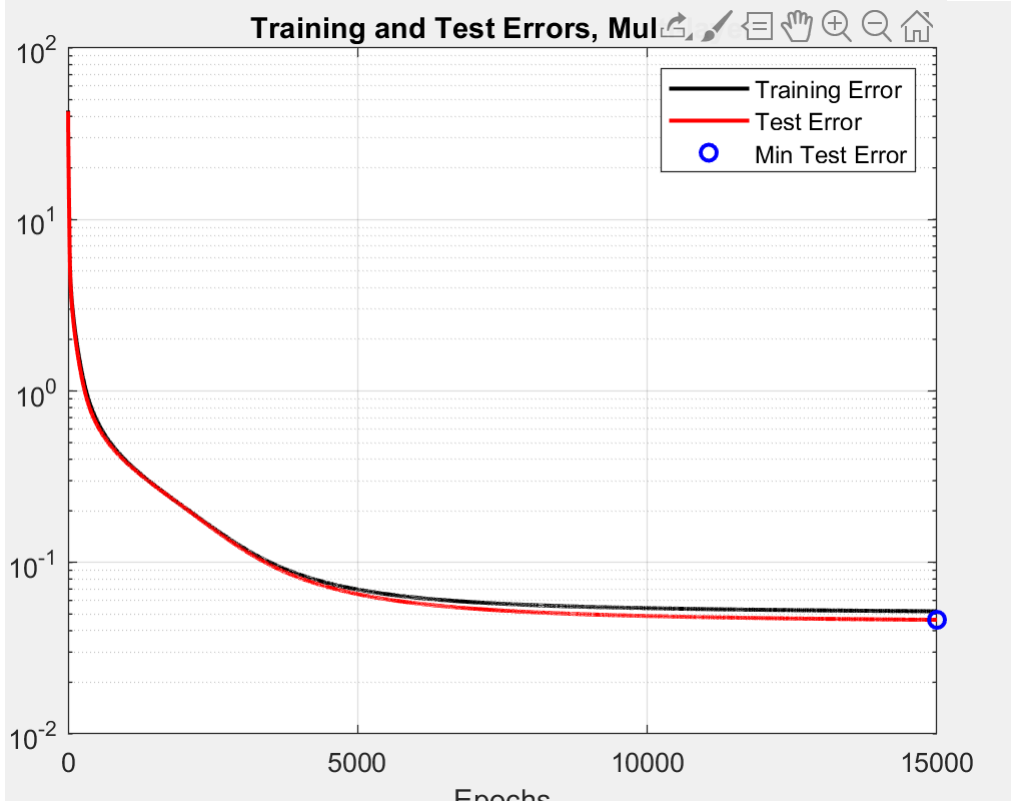
- Calculate initial error in order to be able to plot the error and number of iterations.
- Then we loop for number of iterations
  - First, we calculate the gradient of the error with respect to the weight matrix V.
  - Then we calculate the gradient for the error with respect to the matrix W. This is done by using a modified version of V, since W does not depend on the bias in V.
  - Then we update V and W with the gradients scaled with the learning rate parameter (Taking a learning step).
  - Then the errors are evaluated in order to be able to plot the error and number of iterations.
  - This is repeated for “number of iteration” times.

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.**

Dataset 1:
Nr of hidden neurons: 10
Learning rate: 0.01
Num of iter: 2000
Test accuracy: 99.5%

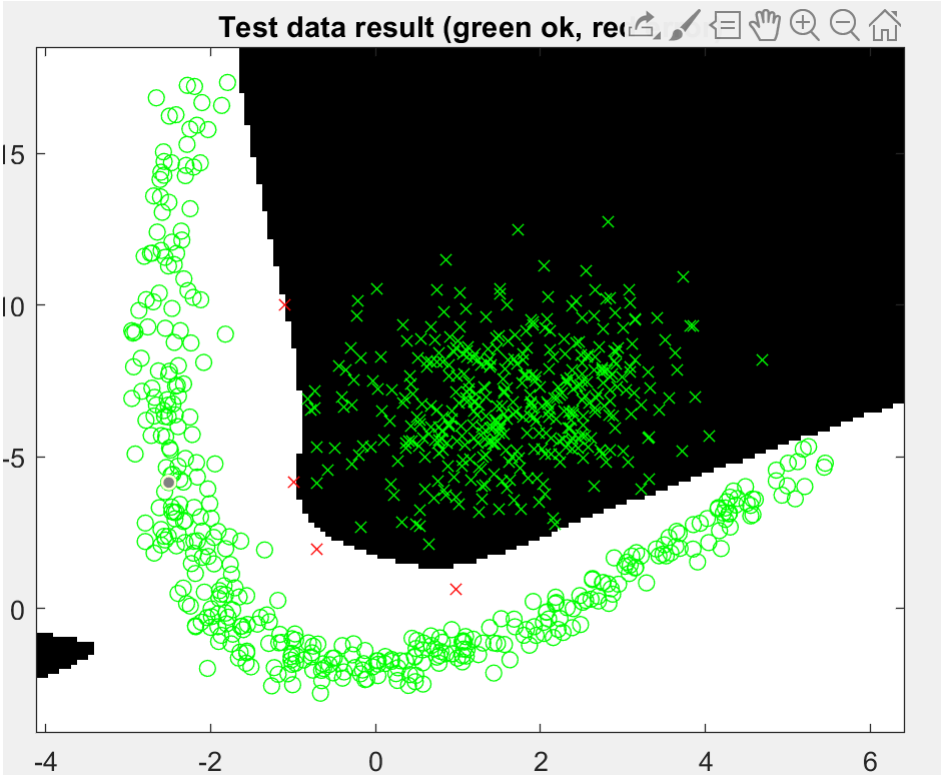
We chose the Multi layer network since it gave us better results than the single layer consistently.

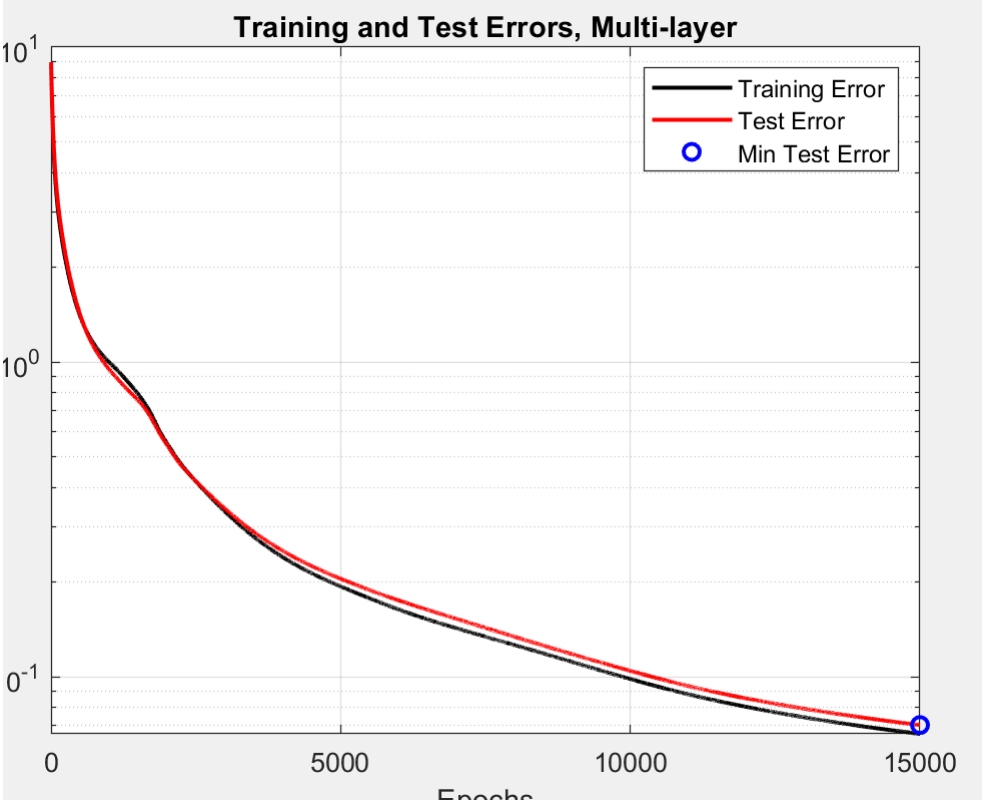

Test data result (green ok, red error)
10
-5
0
5
10
-5
0
5


Training and Test Errors, Mul
Training Error
Test Error
Min Test Error
10^2
10^1
10^0
10^-1
10^-2
0
5000
10000
15000
Epochs

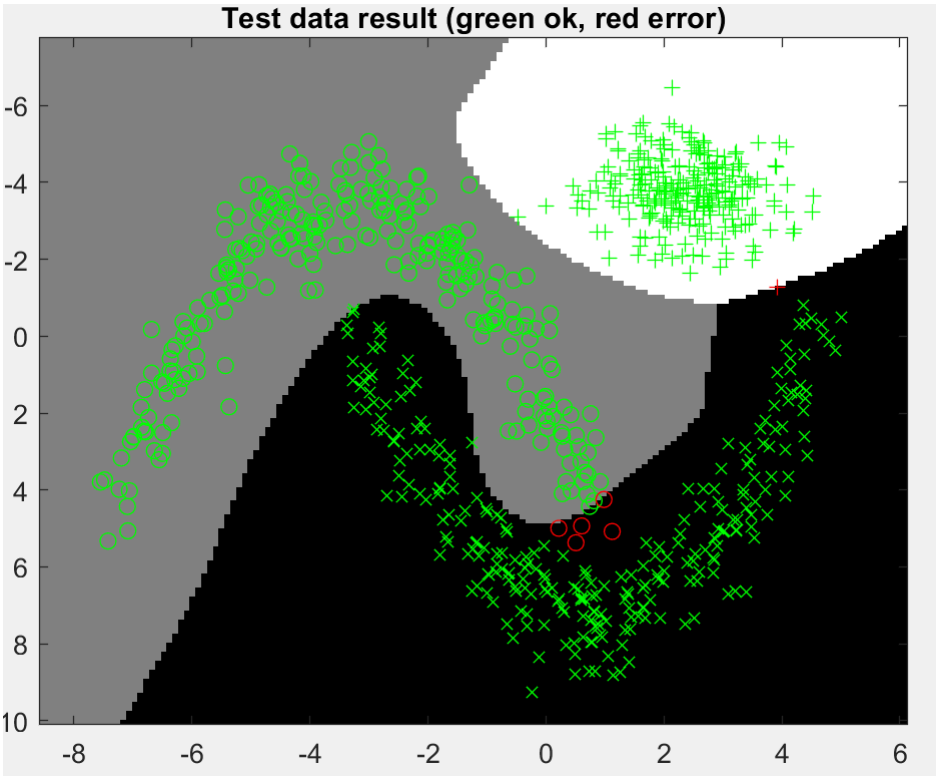
Dataset 2:
Nr of hidden neurons: 12
Learning rate: 0.01
Num of iter: 5000
Test accuracy: 99.5%

For dataset 2,3, and 4 we also chose the multi layered network since we have to be able to deal with not linearly separable data.

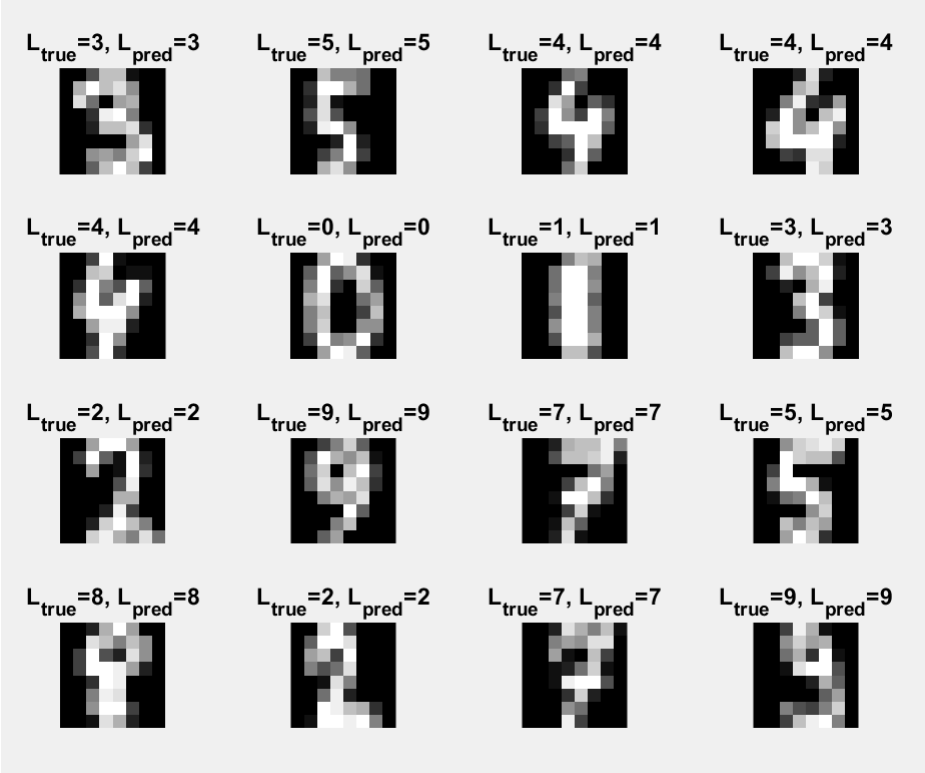


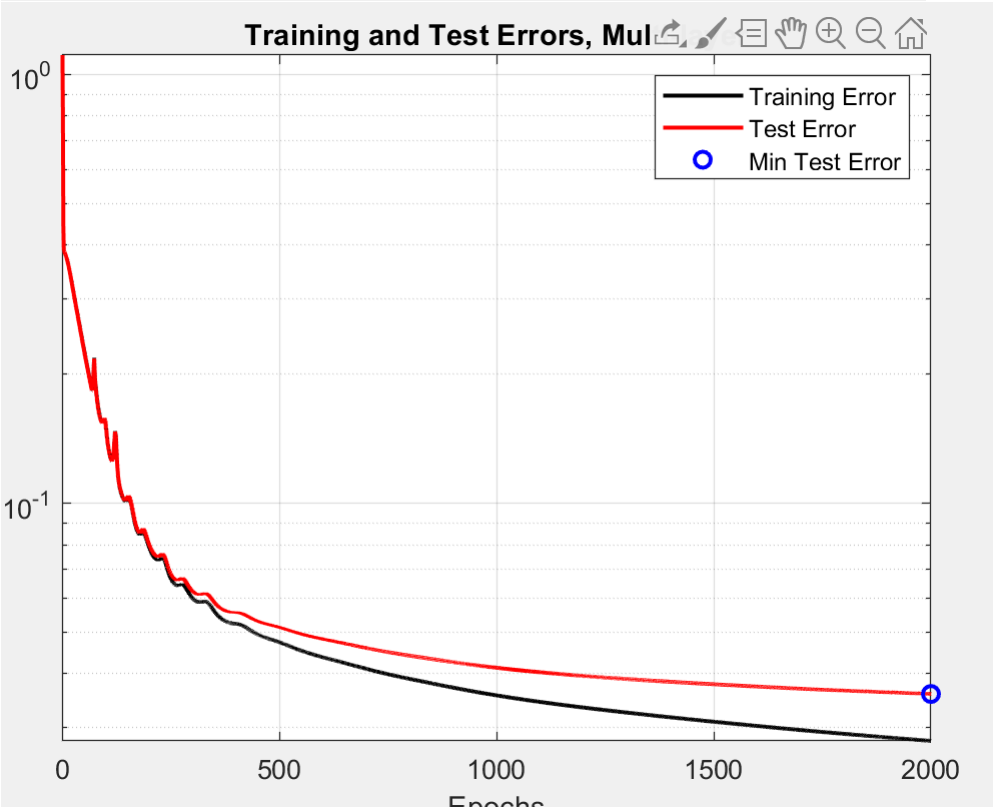


Dataset 3:
Nr of hidden neurons: 10
Learning rate: 0.005
Num of iter: 20000
Test accuracy: 99.2%



Dataset 4:
Nr of hidden neurons: 50
Learning rate: 0.01
Num of iter: 2000
Test accuracy: 97.7%





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

This is a non-generalizable solution since the training set does not represent the real distribution in a good way, therefore the gradient in the backprop will not be accurate.

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

- Single layered network can only deal with linearly separable data but it is faster than the multi layered to train.
- The kNN algorithm can deal with not linearly separable data and does not need any training, but classification takes a long time and you also need to store all the data points in your model.
- The multi layered network takes a lot of time to train but can deal with non-linearly separable data and the final model does not take up a lot of memory storage. There is a risk that you overfit your model to the training data which can lead to worse performance.

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

- Changing our activation function could maybe give us better results (RelU etc) or atleast give us a faster training time.
- Having a momentum term which could give us better direction of our gradient.
- Increase the number of hidden layers.
- Initialize our weights in a smart way could decrease the risk of ending up in a local minima.
- Stocasthic gradient decent would give us a faster learning time. This method is noisy and does not guarantee that you end up in a optima.
- Feature extraction on the OCR-data set could be useful for speeding up the network and could maybe give us better accuracy.