TBMI26 – Computer Assignment Report Supervised Learning

Deadline – March 14 2021

Authors: Matilda Granqvist, Andreas Hertin
matgr197 & andhe794

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.**
   For dataset 1 - The dataset has two classes, red and green dots and two features. The classes could be described asn two clusters quite close to each other. The classes are somewhat overlapping and are therefore not completely linearly separable. The dataset would not benefit from a multi-layer though.

   The dataset 2, 3 and 4 needs non-linear classifiers since these datasets are not linearly separable. Dataset 2 has two classes, red and green dots and two features. Compared to dataset 1 these classes are far from linearly separable but can quite easily be separated with a non-linear classifier. The green dots and forming a half circle around the red cluster dots.

   Dataset 3 has 3 classes: red, green and blue dots and it has two features. The dots are forming two different half circles and one cluster. Therefore it is not possible to use a linear classifier.

   Dataset 4 has 64 features. And is no longer just dots and crosses but are instead forming numbers from 0-9, and therefore have 10 classes. This means a non-linear classifier is needed.

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 noise sensitivity and dimensionality is reduced and gives invariance to small distortions.

3. **Give a short summary of how you implemented the kNN algorithm.**
   For each sample the labels of its closest k neighbours were checked and the labels of the majority of them got us our predicted label for the sample. The euclidean distance was used to check the neighbours.

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

Classes with the most represented neighbours (if three or more classes), if two(or more) classes had same amount of neighbours:
If the sample had a neighbour with distance 0 it took its label.
Summed the lengths from each neighbour and took the average. Then we choose the label with the lowest average distance.

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 created combined bins and iterated for different variations. For each bin combination we ran the kNN algorithm with k = 1 to 30 to find the best k for that bin combination.The accuracies for each k was saved and averaged for all combinations and the highest average accuracy gave our best k. We chose 4 bins for the cross validation and the "*numSamplesPerLabelPerBin*" was set to *inf*.

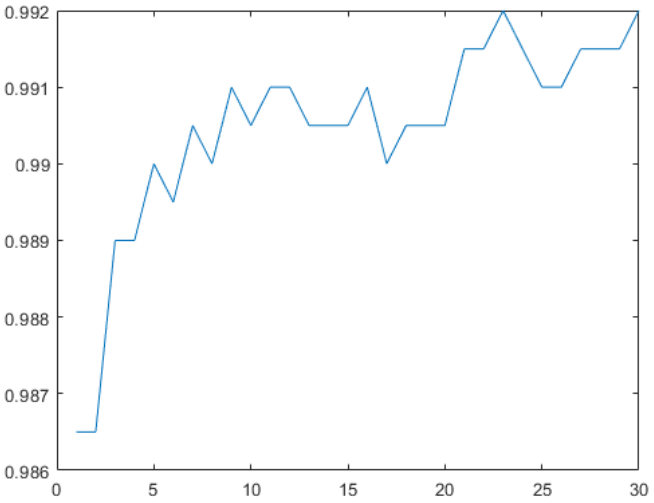

Figure 1: dataset 1, best k = 23, Accuracy = 99,20%

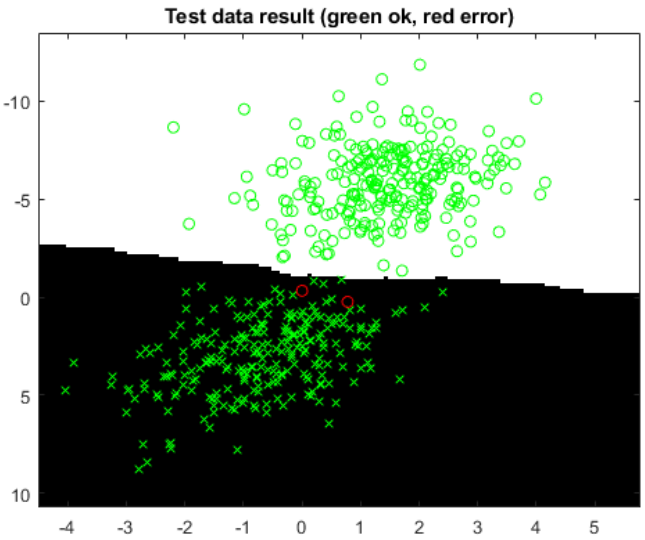

Figure 2: Test data for dataset 1

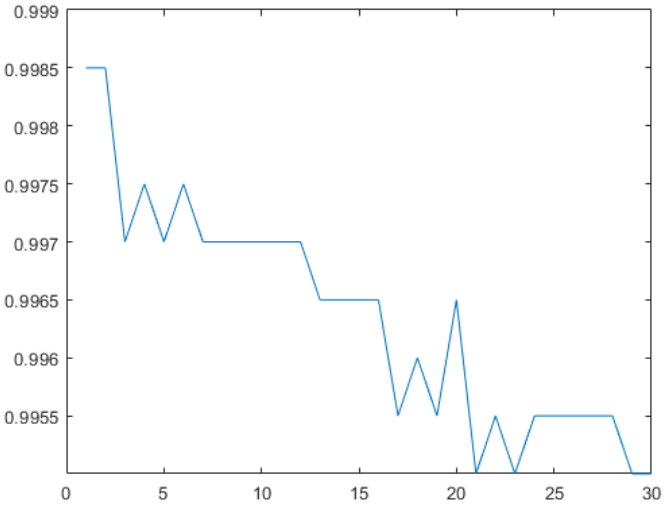

Figure 3: dataset 2, best k = 1, Accuracy = 99,85%

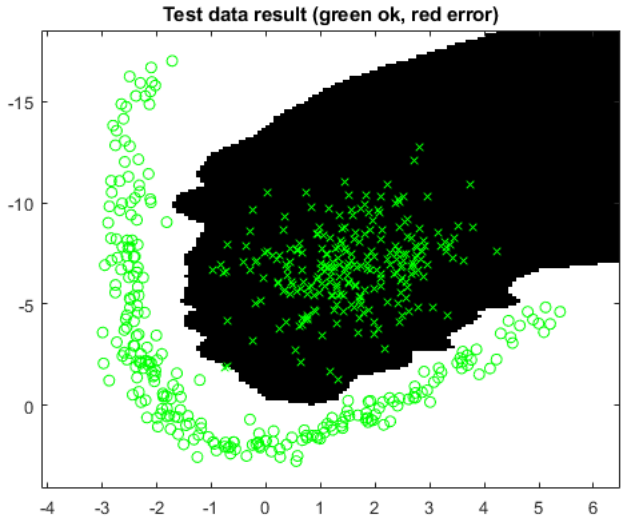

Figure 4: Test data for dataset 2

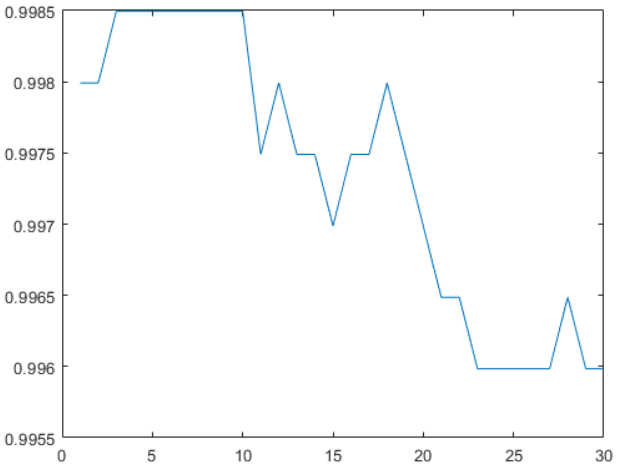

Figure 5: dataset 3, best k = 3, Accuracy = 99,85%

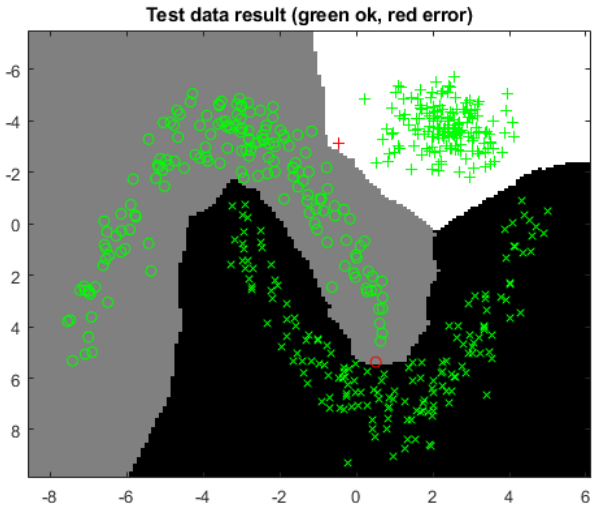

Figure 6: Test data for dataset 3

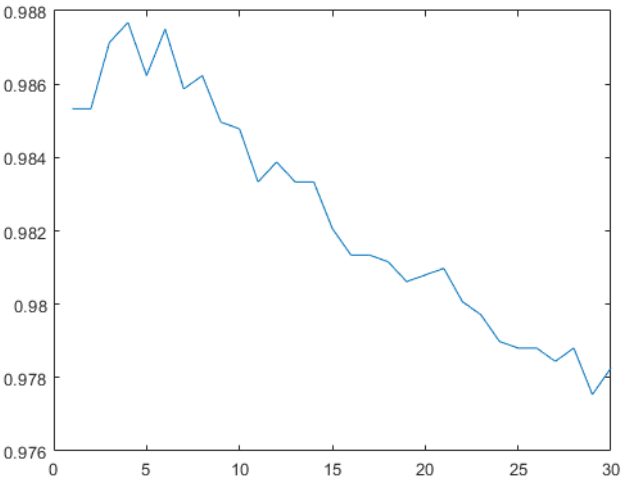

Figure 7: dataset 4, best k = 4, Accuracy = 98,77%

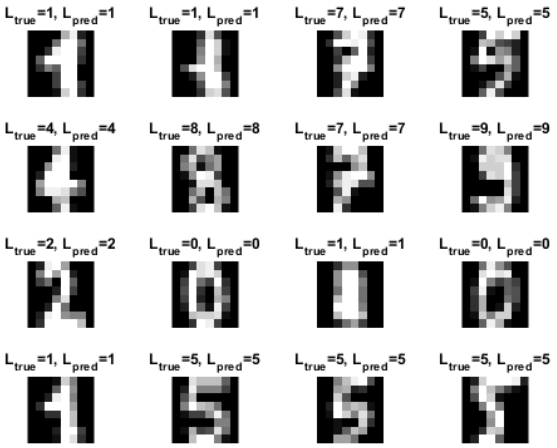

Figure 8: Test data for dataset 4

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

First we used forward propagation. The output and hidden layers have been calculated according to the class 3 with a combination between the chain rule and weighting etc. Both the signal layer and multi layer have 2 input and output nodes but the first layer has 3 inputs due to the added bias. But the multilayer has two layers in the "middle", one output from the input layer and the other one is the input to output layer. These middle layers depend on the number of hidden neurons. For example when this value was set to 10 we get: The output from the first layer has 10 (number of hidden neurons) neurons, and the input of the last layer is 11 (number of hidden neurons + 1) neurons because of the bias. The output varies depending on the amounts of labels in the specific dataset.

The single layer has one weighting matrix that is created from a uniformly distributed random number in the interval (0,1) with the size of the X and Y-training data.
The samples that need to be classified are multiplied with the weighting matrix which will result in the output for each sample. The single layer uses one gradient for each sample.

The multi layer has two weighting matrices. The matrices are created from randn which results in a random normal distributed number around zero. The sizes are based on T-train and D-train and the number of hidden neurons. These needed to be divided by 100 in order to get an accurate size. These matrices represent the initial weights of the output neurons and initial weights of the hidden neurons. The multi layer uses two gradients, one for the input layer and one for the output layer. The multi layer uses tanh(s) as activation function.

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.**
   Single layer:

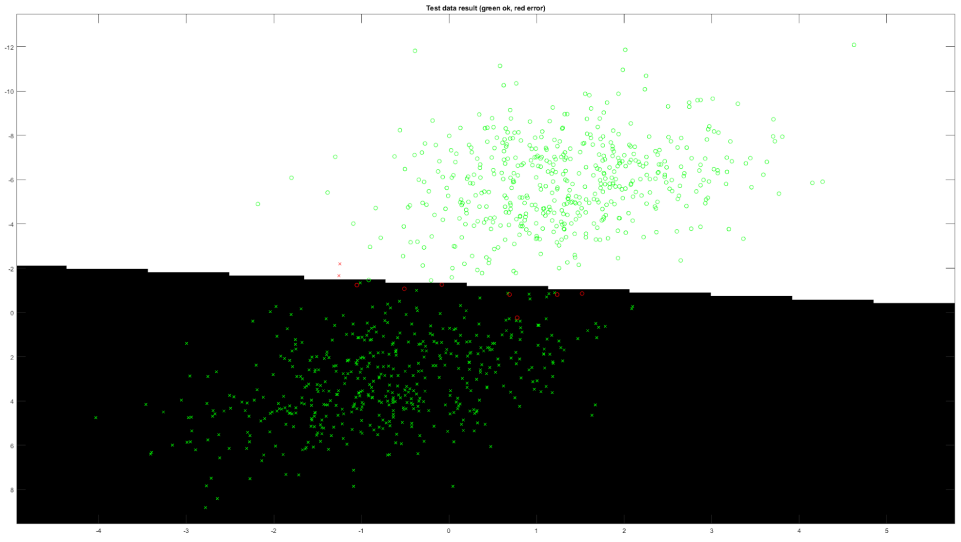

Figure 9: dataset 1, Iterations = 10000, Learning rate = 0.0001, Accuracy = 99,1%

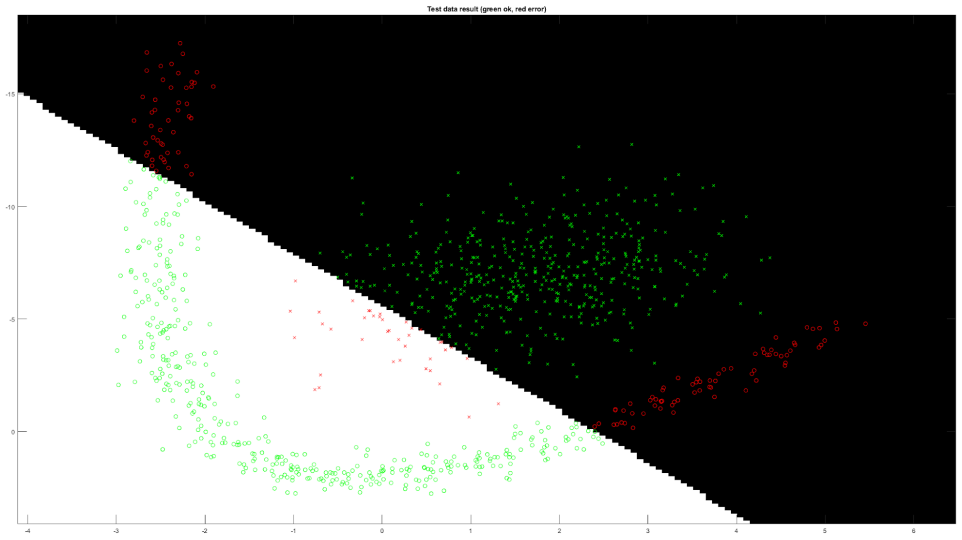

Figure 10: dataset 2, Iterations = 10000, Learning rate = 0.0001, Accuracy = 84,4%

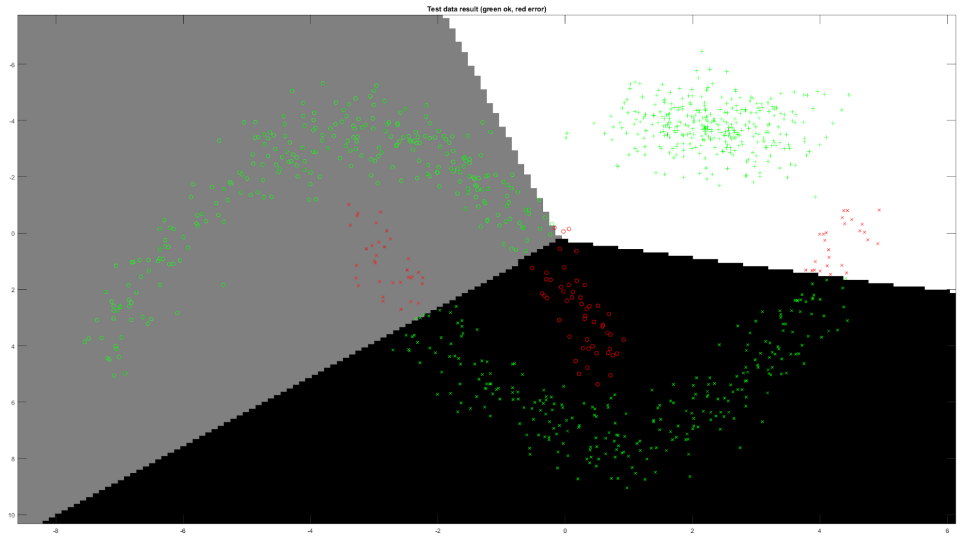

Figure 11: dataset 3, Iterations = 10000, Learning rate = 0.0001, Accuracy = 88,89%

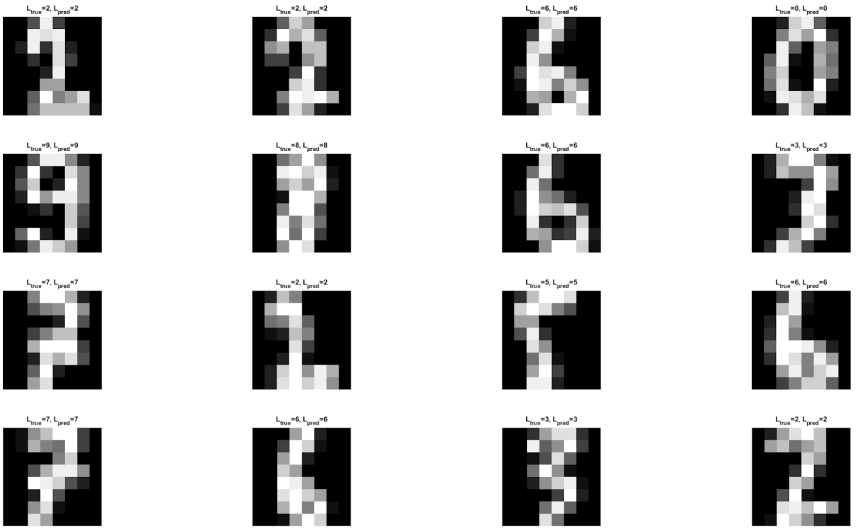

Figure 12: dataset 4, Iterations = 10000, Learning rate = 0.0001, Accuracy = 89,82%

Since dataset 2-3 are not linearly separable these results are significantly worse than for dataset 1. The reason that the single-layer works for the first is because it is linearly separable.

Multi-layer:

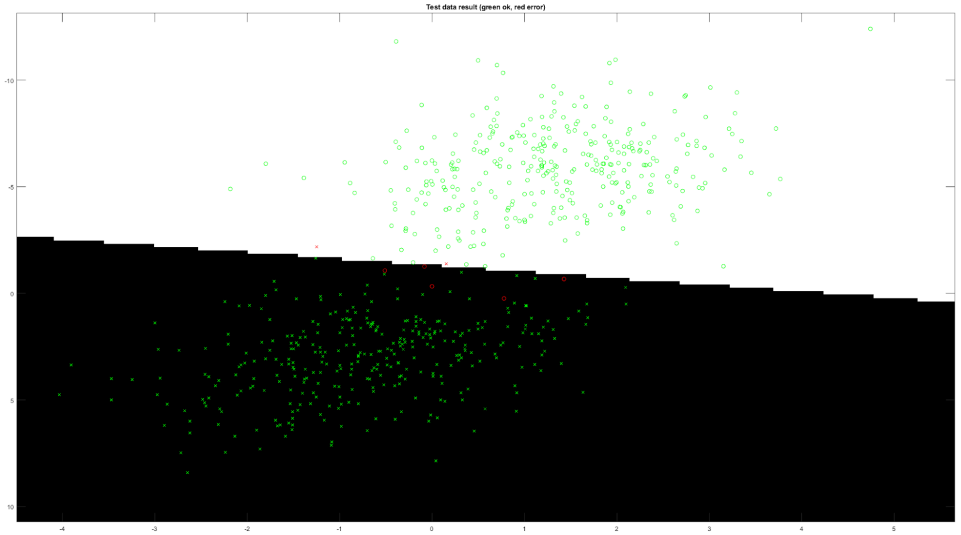

Figure 13: dataset 1, Iterations = 10000, Hidden = 10, Learning rate = 0.0075, Accuracy = 98,95%

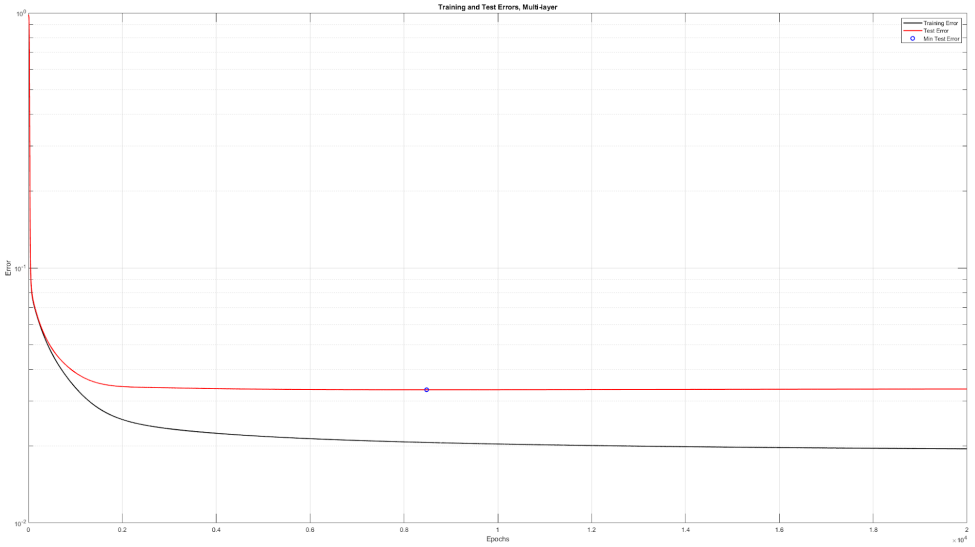

Figure 14: Training and test errors multilayer

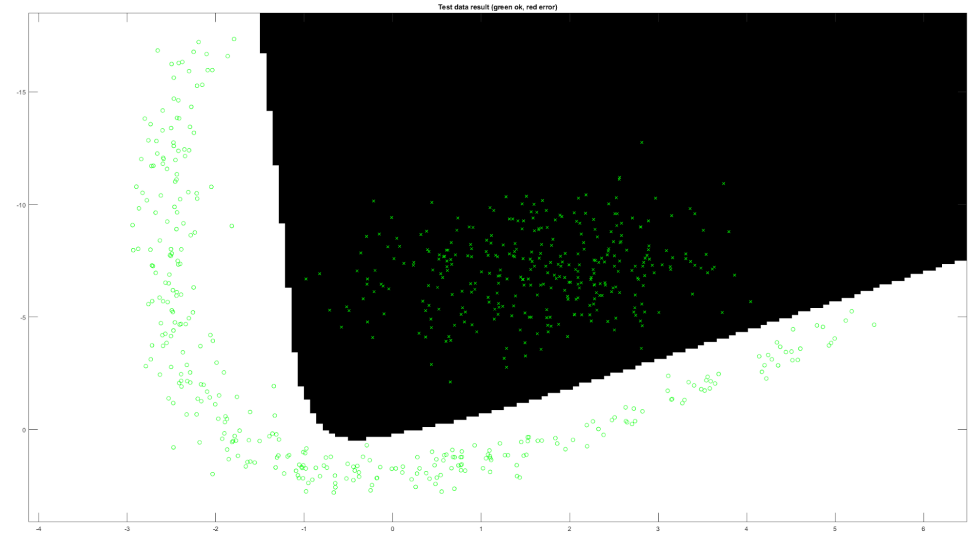

Figure 15: dataset 2, Iterations = 20000, Hidden = 10, Learning rate = 0.00135, Accuracy = 100.0%

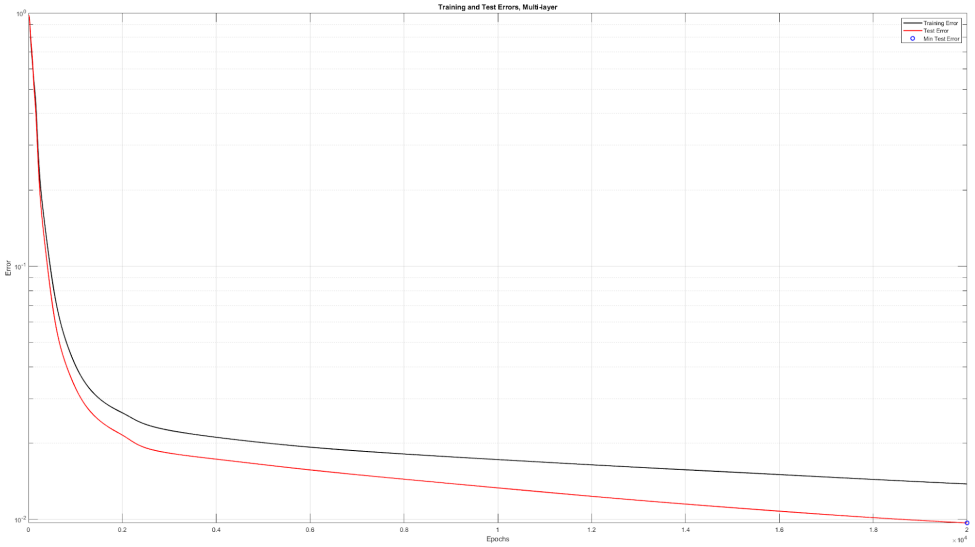

Figure 16: Training and test errors multilayer

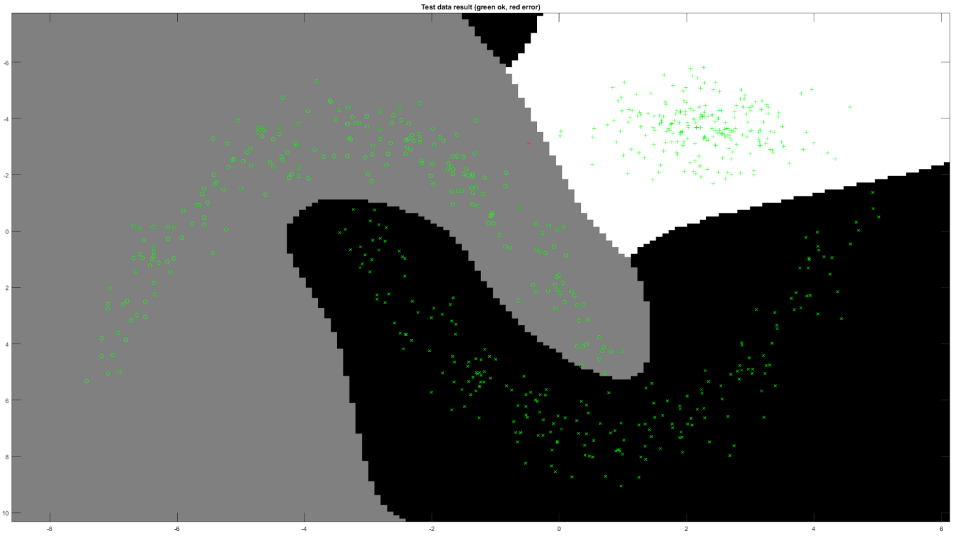

Figure 17: dataset 3, Iterations = 20000, Hidden = 10, Learning rate = 0.00135, Accuracy = 99,85%

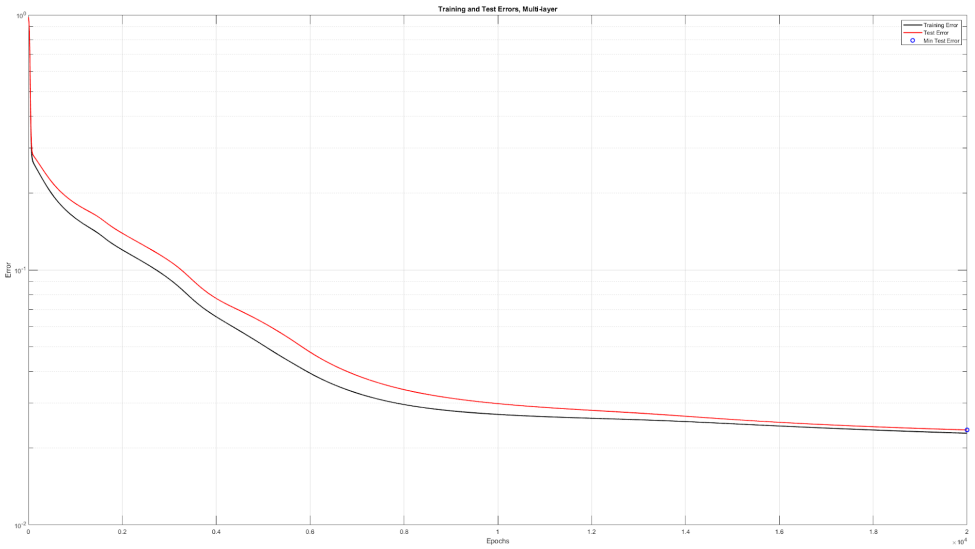

Figure 18: Training and test errors multilayer

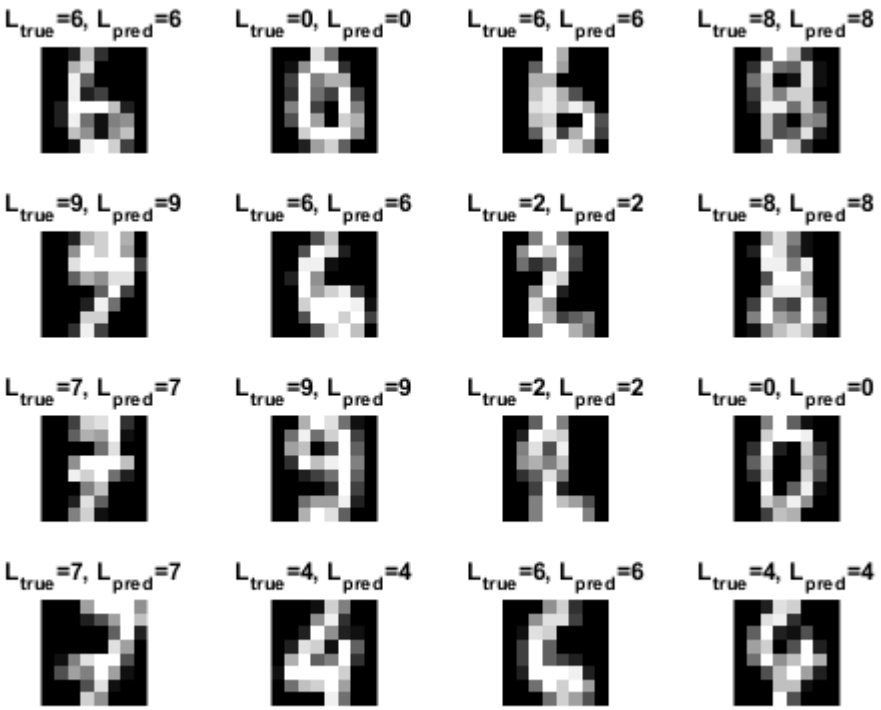

Figure 19: dataset 4, Iterations = 2000, Hidden = 40, Learning rate = 0.00135, Accuracy = 96,3%

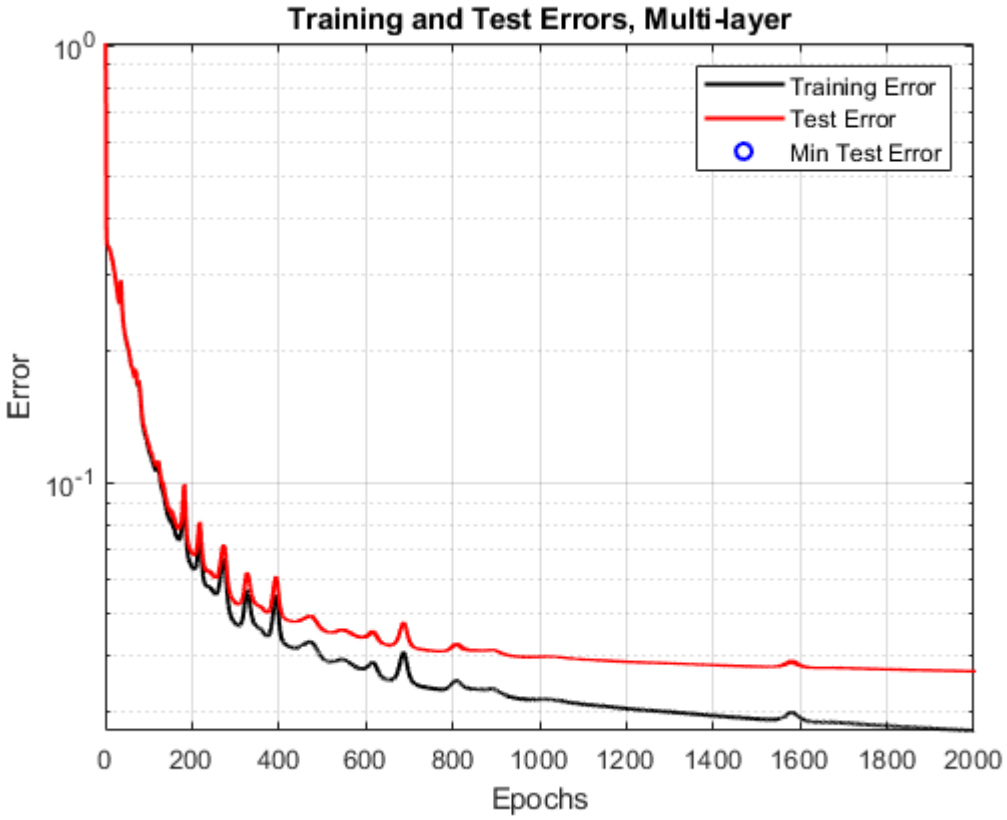

Figure 20: Training and test errors multilayer

With the multilayer we can see a large improvement compared to the single layer since the multilayers have functionalities that can handle non-linearities. The number of neurons was increased for dataset 4 because of the increased complexity of the "pattern". The values have been chosen by trial and error. One value was changed every time to see what way the accuracy turned to.

8. **Present the results, including images, of your example of a non-generalizable backprop solution. Explain why this example is non-generalizable.**
By giving the function insufficient data to train on a non-generalizable solution could be demonstrated. This solution is a non-generalizable solution because it only fits the training data. The results can be seen Figure 21.

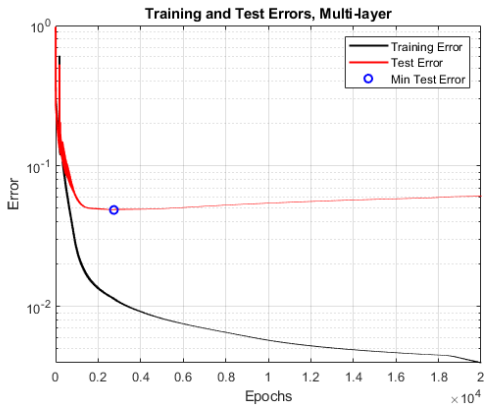

Figure 21. Training data was 2/30 bins and test data 28/30 bins.

Figure 22. Training data, a pattern working for the data has been found

Figure 23. Test data, the pattern from training misses a lot of the edge cases because of lack of data when training

9. **Give a final discussion and conclusion where you explain the differences between the performances of the different classifiers. Pros and cons etc.**
   Single layer networks work well for data that can be linearly divided(dataset 1) and is quick. Multilayer is better for the other types of data where the separating line has to be polynomial. Multilayer takes longer time. If it is possible to use Single layer network (i.e getting the wanted accuracy), one should do that because it is way faster than using a Multilayer. The kNN was good for almost all datasets but somewhat time-consuming for dataset 4. This is because dataset 4 has more dimensions and results in longer calculations for the kNN.

10. **Do you think there is something that can improve the results? Pre-processing, algorithm-wise etc.**
    Overall the results for dataset 1,2 and 3 are already very good and the results are quite stable. To improve the results for dataset number 4, on the one hand more layers would probably give a better accuracy. That would on the other hand also lead to a lot more time-consuming calculations and therefore a longer processing time.