TBMI26 – Computer Assignment Report  
Supervised Learning

Deadline – February 12 2018

Author/-s:

Kristian Sikiric

Kaj Rolandsson

In order to pass the assignment you will need to answer the following questions and upload the document to LISAM. You will also need to upload all code in .m-file format. If you meet the deadline we correct the report within one week after the deadline. Otherwise we give no guarantees when we have time.

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

For dataset 1 a linear classifier is sufficient. For dataset 2 and 3 you need a non-linear classifier.

1. **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**](http://archive.ics.uci.edu/ml/datasets/Optical+Recognition+of+Handwritten+Digits)

Since the noise is reduced by this, the features become more robust. It becomes easier to distinguish between features.

1. **Give a short summery of how you implemented the kNN algorithm.**

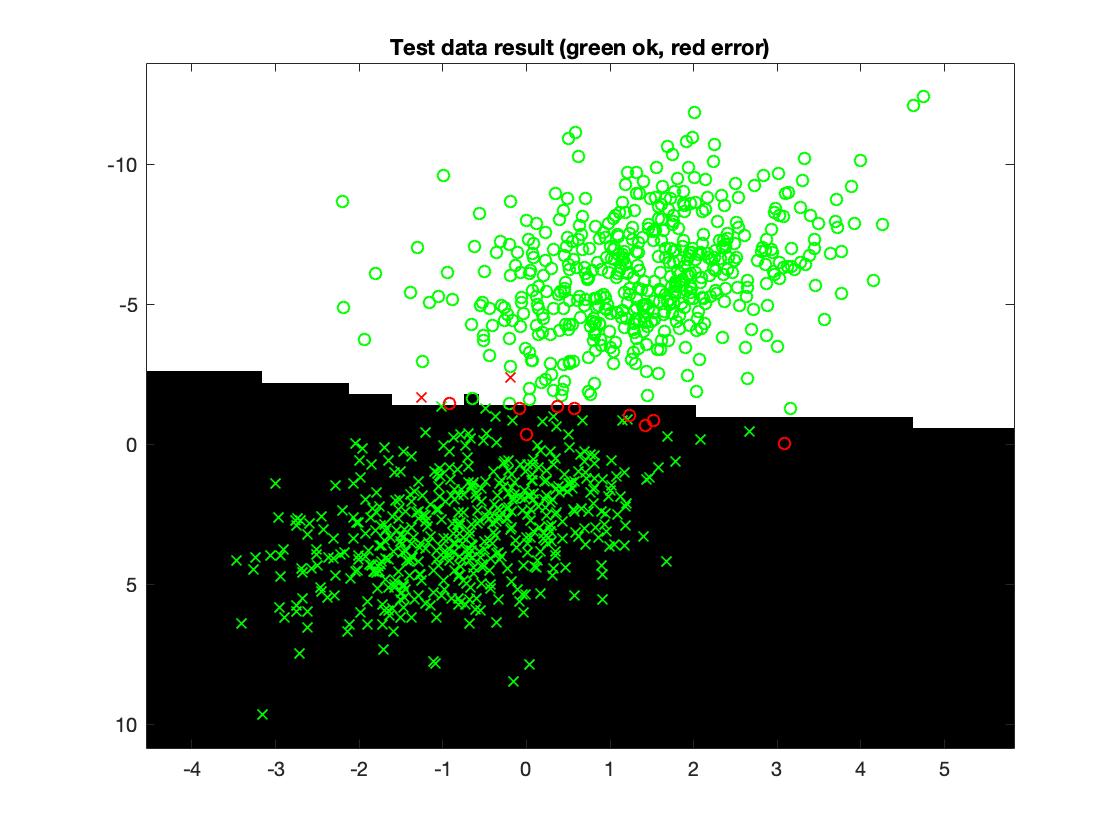
First the Euclidean distance between all features was calculated, these were then sorted. The we choses the most frequent used label in the k-nearest neighbors.

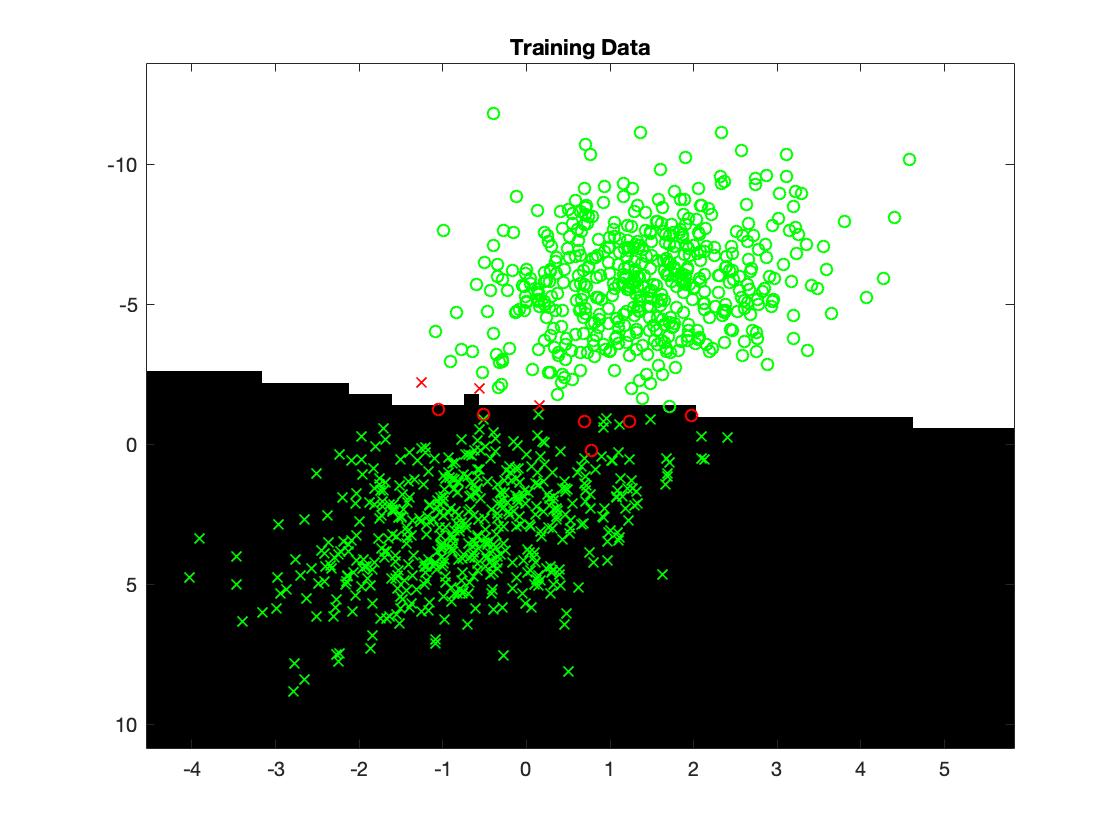
1. **Explain how you handle draws in kNN, e.g. with two classes (k = 2)?**

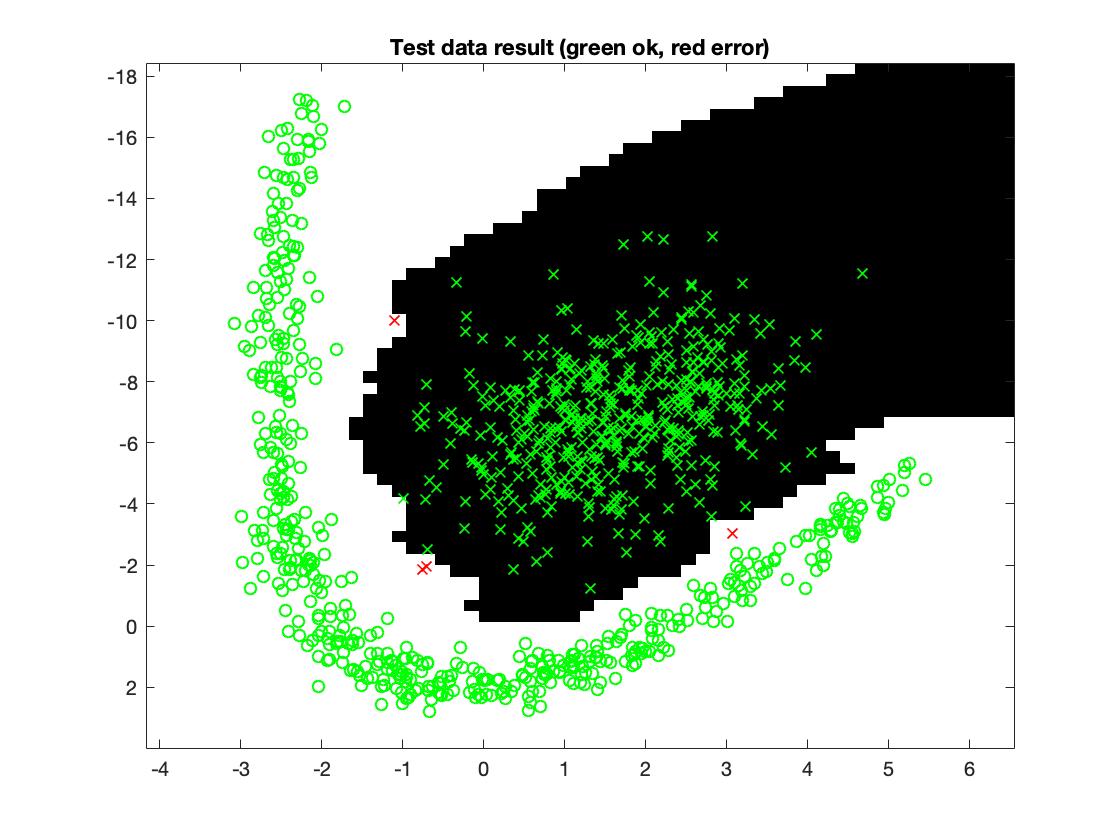
We chose the label of the closest neighbor with the smallest value. So if we had [2,1,1,2] for example, 1 was chosen.

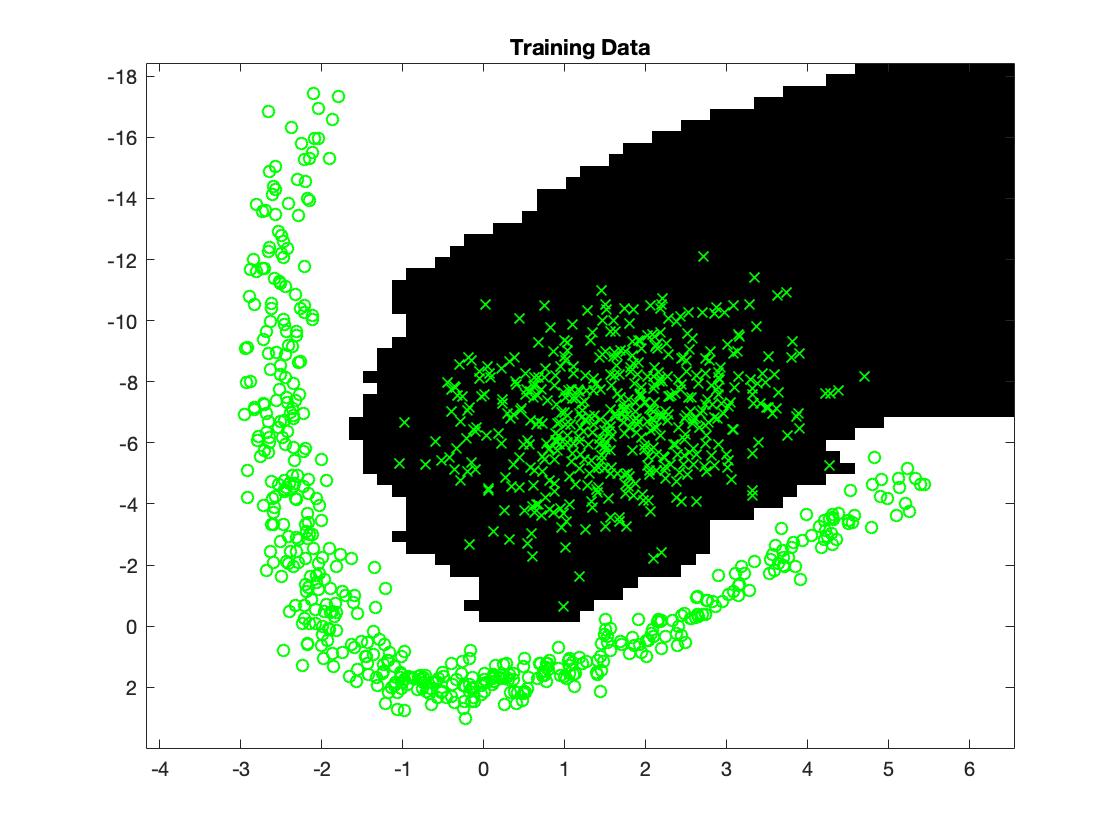
1. **Explain how you selected the best k for each dataset using cross validation. Include the accuracy and images of your results for each dataset.**

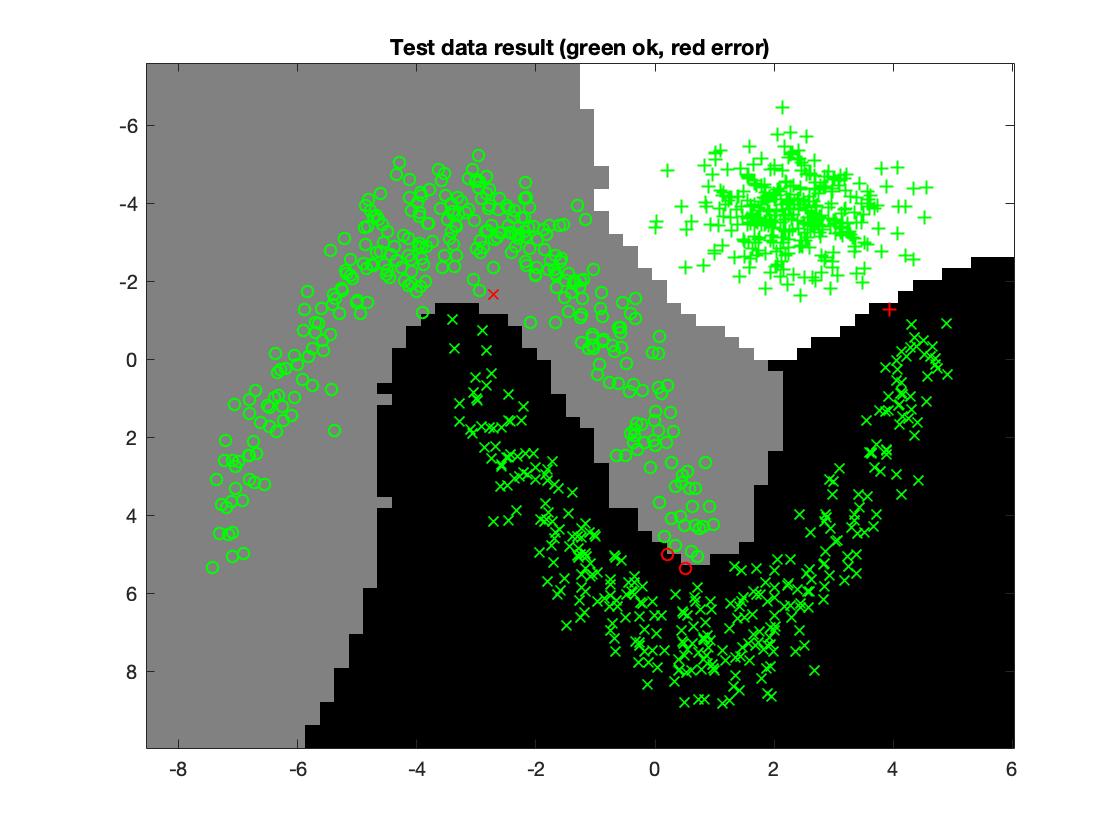
We used cross validation with 2 folds. First the distance was calculated from fold1 to fold2, and then from fold 2 to fold 1. The average of the two errors was stored. We did this for 10 different k values and plotted the average error for each one. The k with the highest accuracy was chosen.

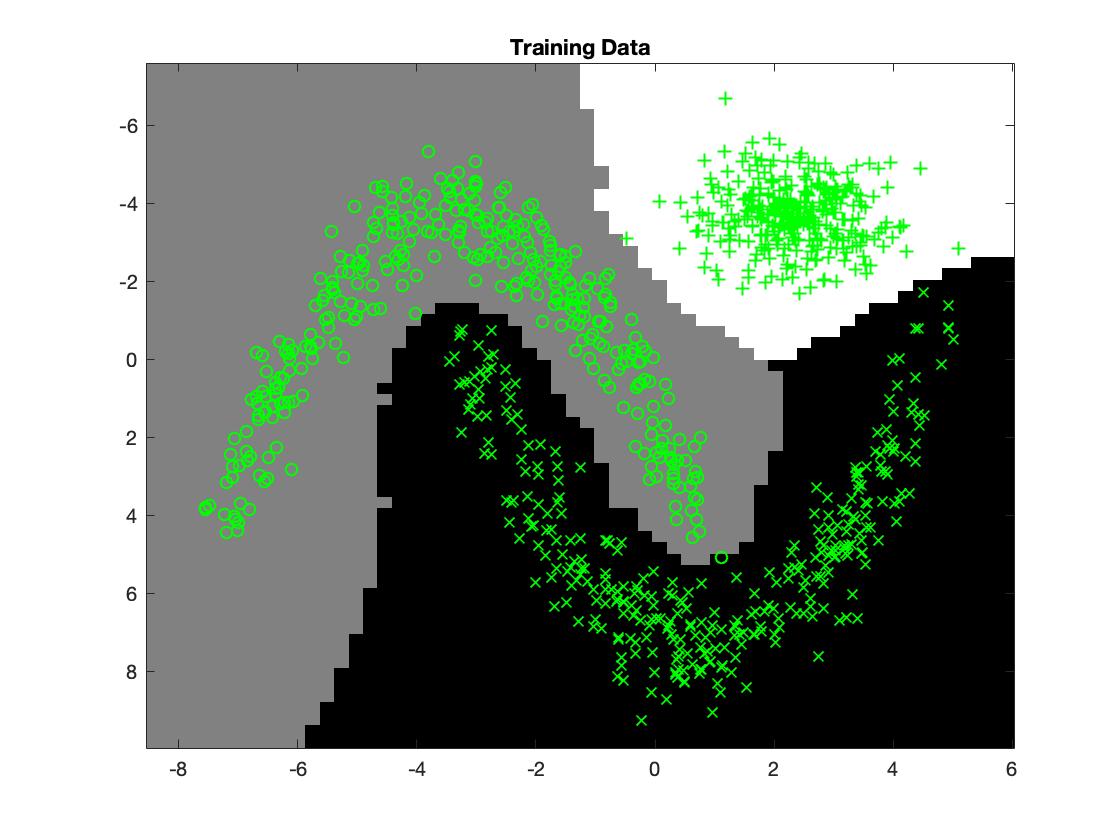
First data set: K = 9 gives an accuracy 98.8 %.

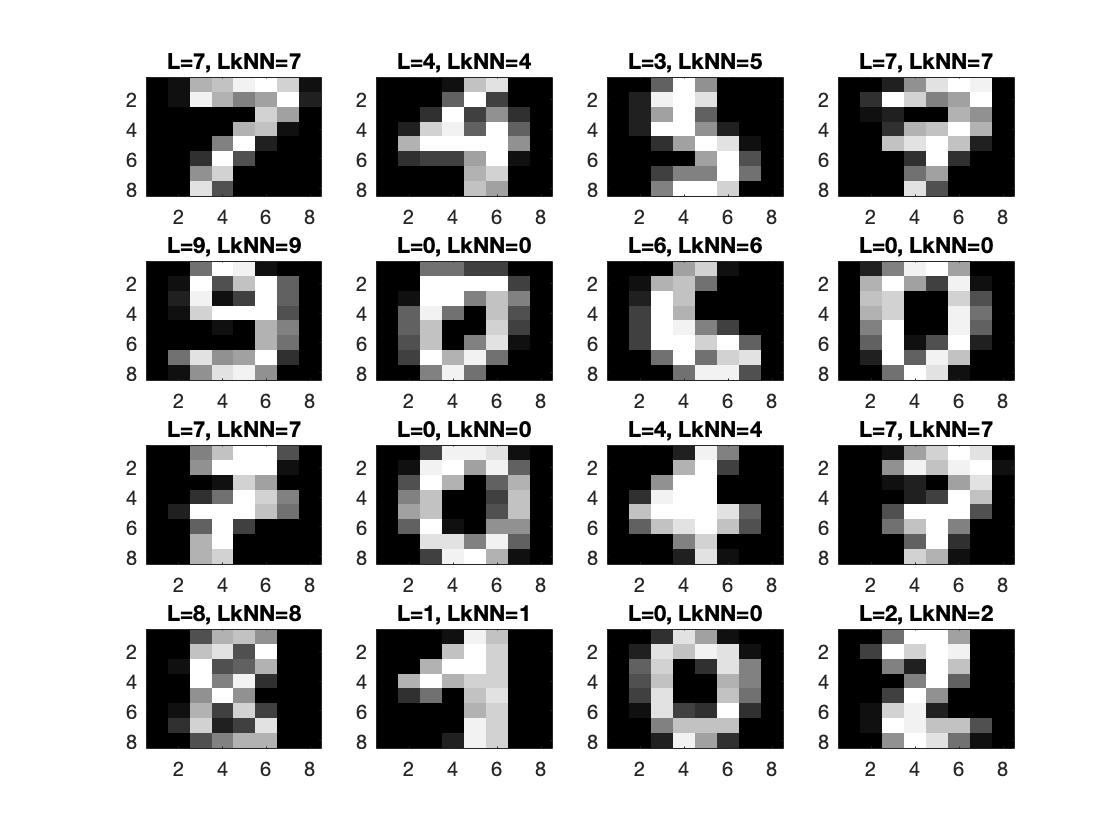


Second data set: K = 1 gives an accuracy of 99.6%.



Third data set: K = 3 gives an accuracy of 99.6 %.

****

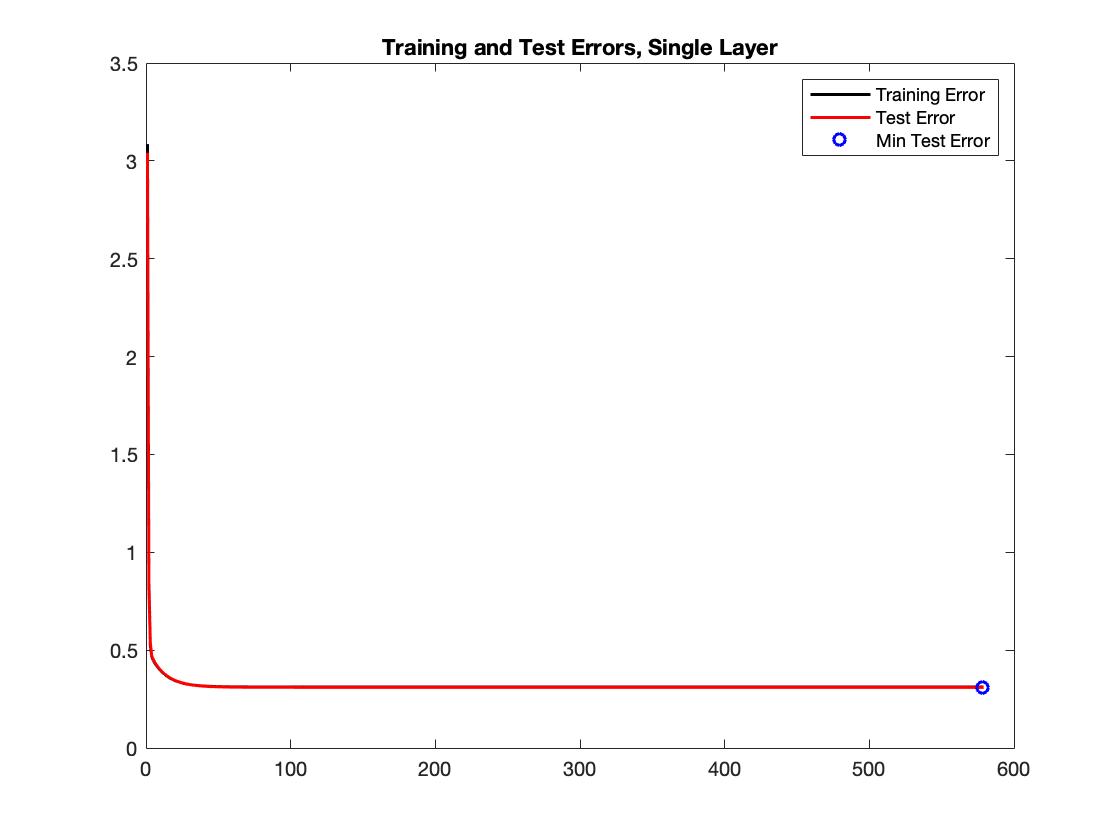
Fourth data set: K = 3 gives an accuracy of 98%.

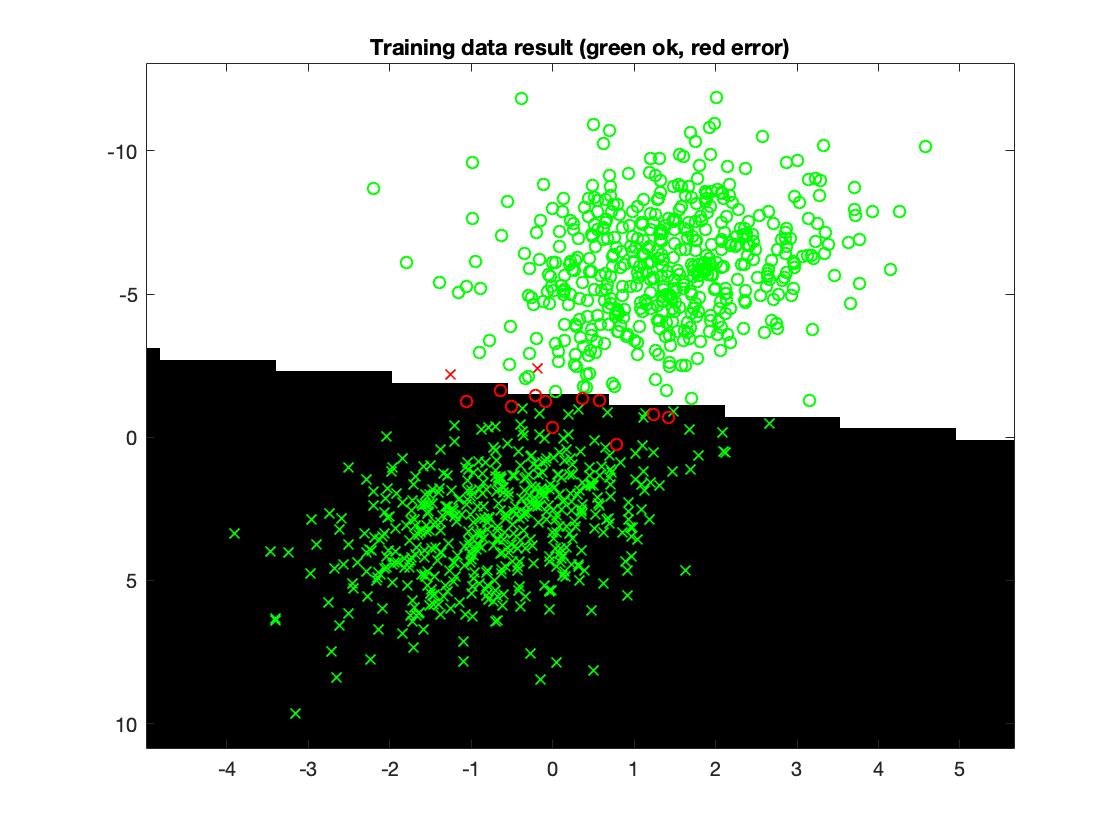
1. **Give a short summery of your backprop network implementations (single + multi). You do not need to derive the update rules.**

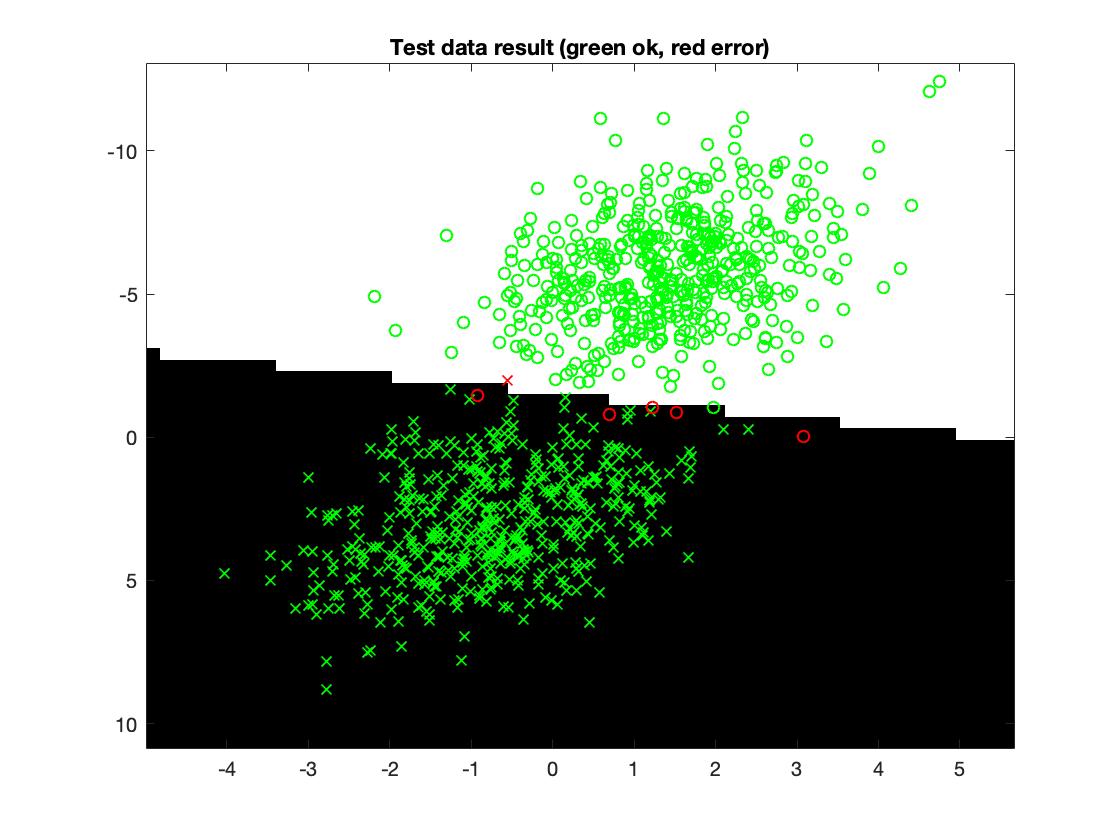
Single: In the single layer, the backpropagation is simple. We simply take the difference between the output from the network and the correct output and multiply this difference with the training features to get the gradient. The weights are then updated using the learning rate and the newly calculated gradient.

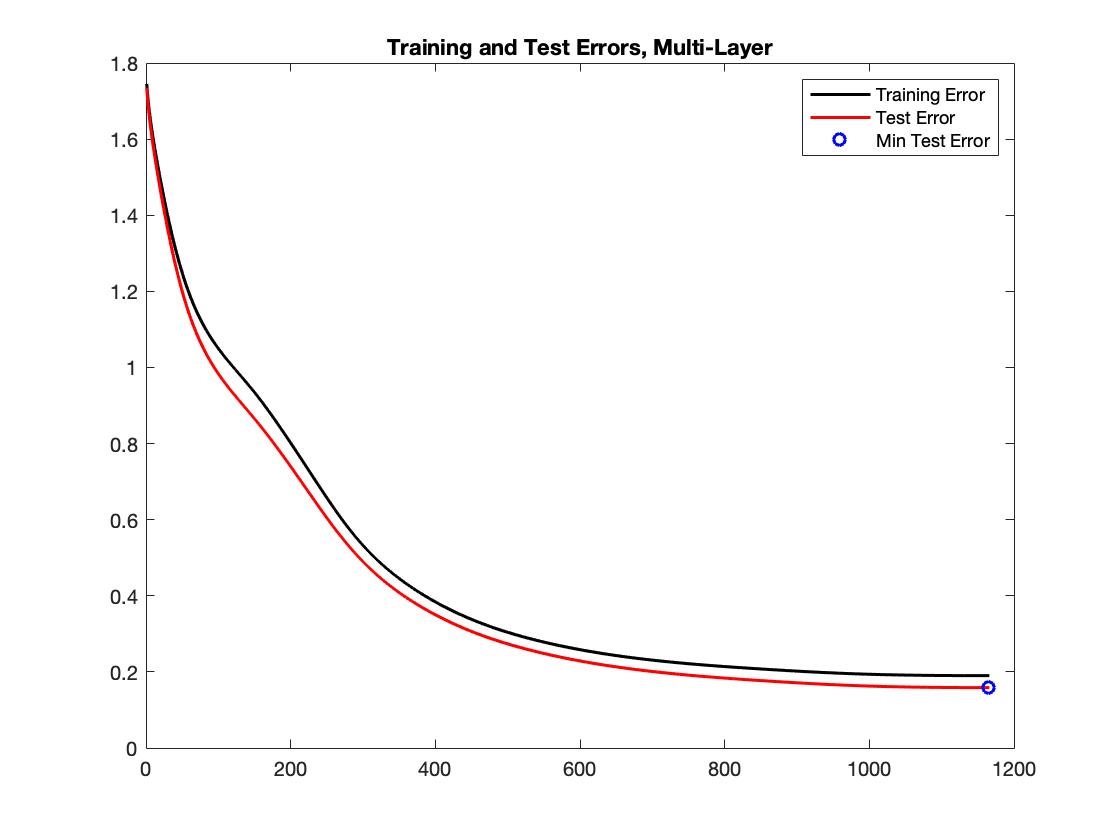
Multi: For the multilayered network we used two different type of weights. One type, w, from the input nodes to the hidden layer, and another type, v, from the hidden layer to the output nodes. The v weights were updated similarly to the weights in the single layer. The difference was taken between the output and the correct output and then multiplied with the output from the hidden nodes. The w weights were updated using the v weights multiplied with the difference between output and correct output as well with the derivative of the output from the hidden layer and lastly multiplied with the training features. These two gradients were used to update the weights.

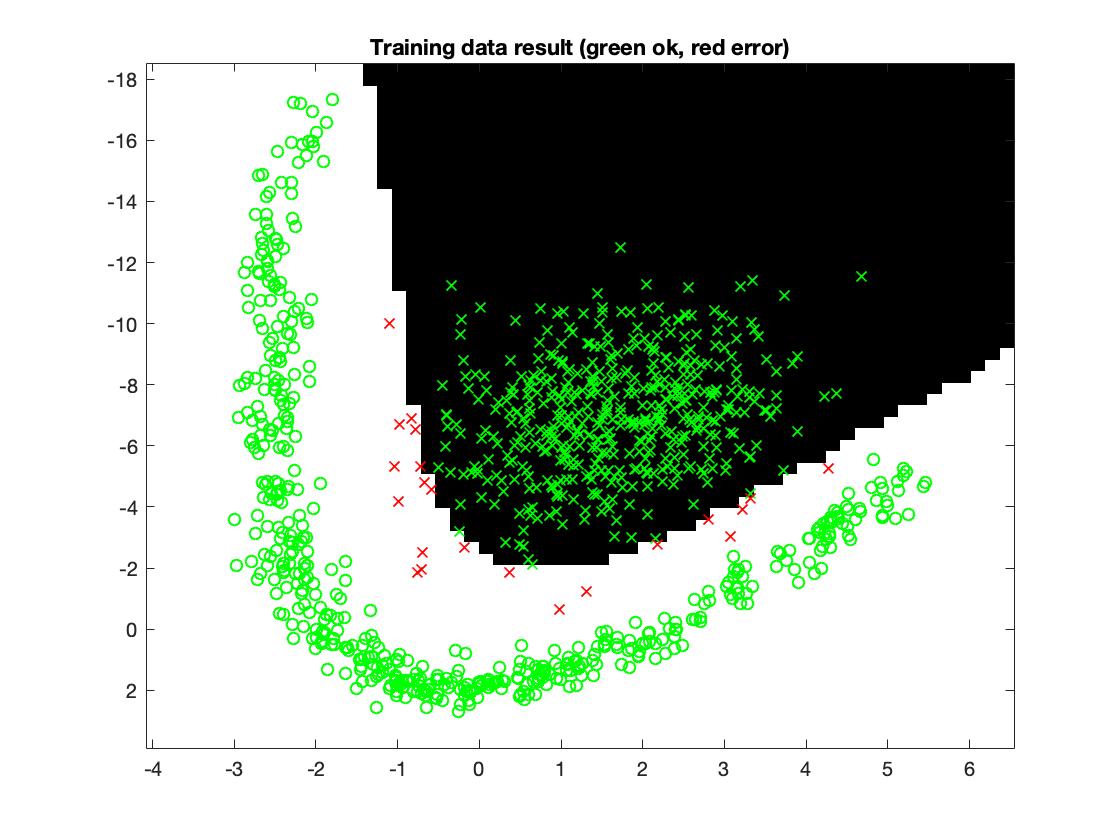
1. **Present the results from the backprop training and how you reached the accuracy criteria for each dataset. Motivate your choice of network for each dataset. Explain how you selected good values for the learning rate, iterations and number of hidden neurons. Include images of your best result for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

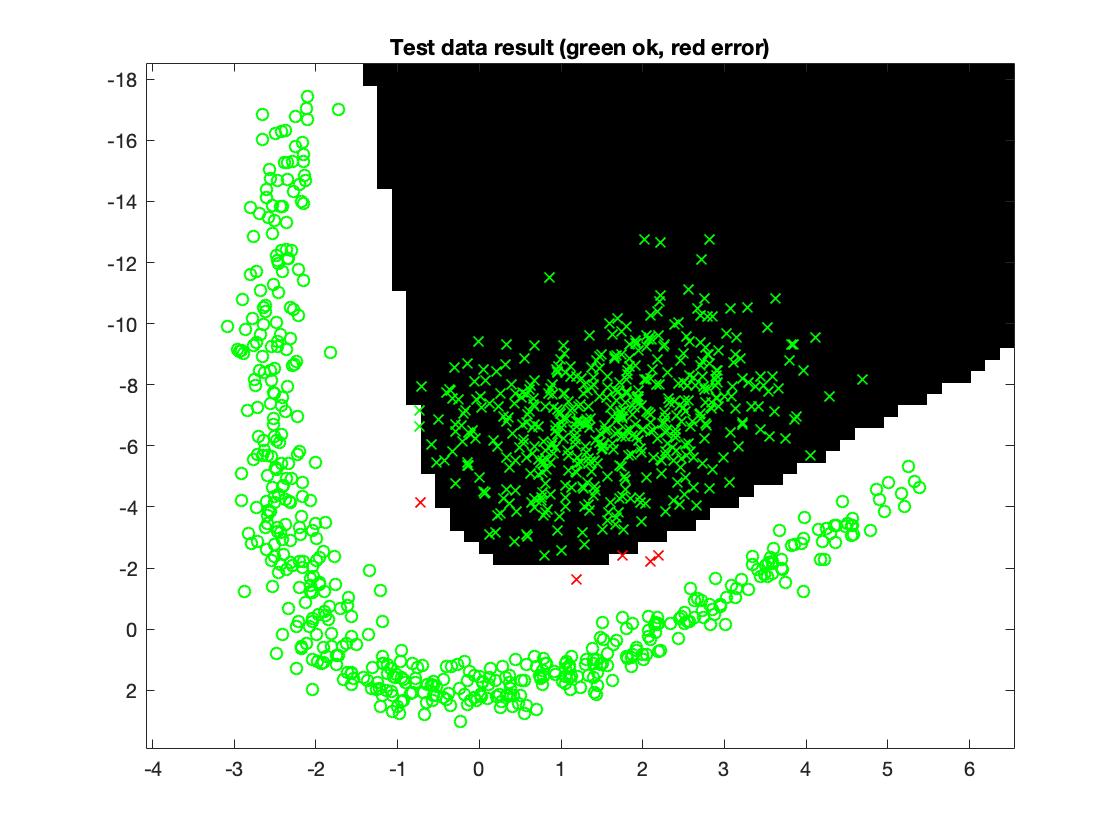
Since a single layer is a linear classifier, this network could only be used on the first data set. It has an accuracy of above 99%. No hidden layers were needed, 40000 iterations were set, but the training usually stopped before 1000 iterations to avoid overfitting. Learning rate was 0.00005. 

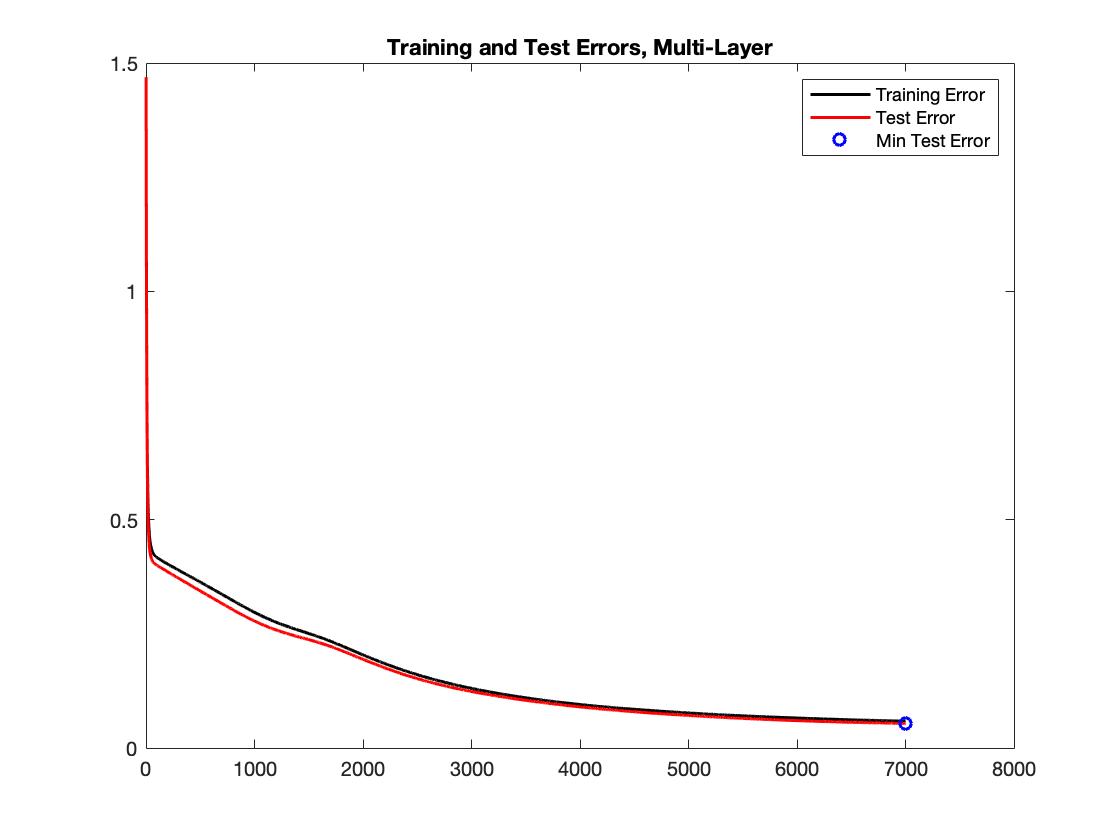
****

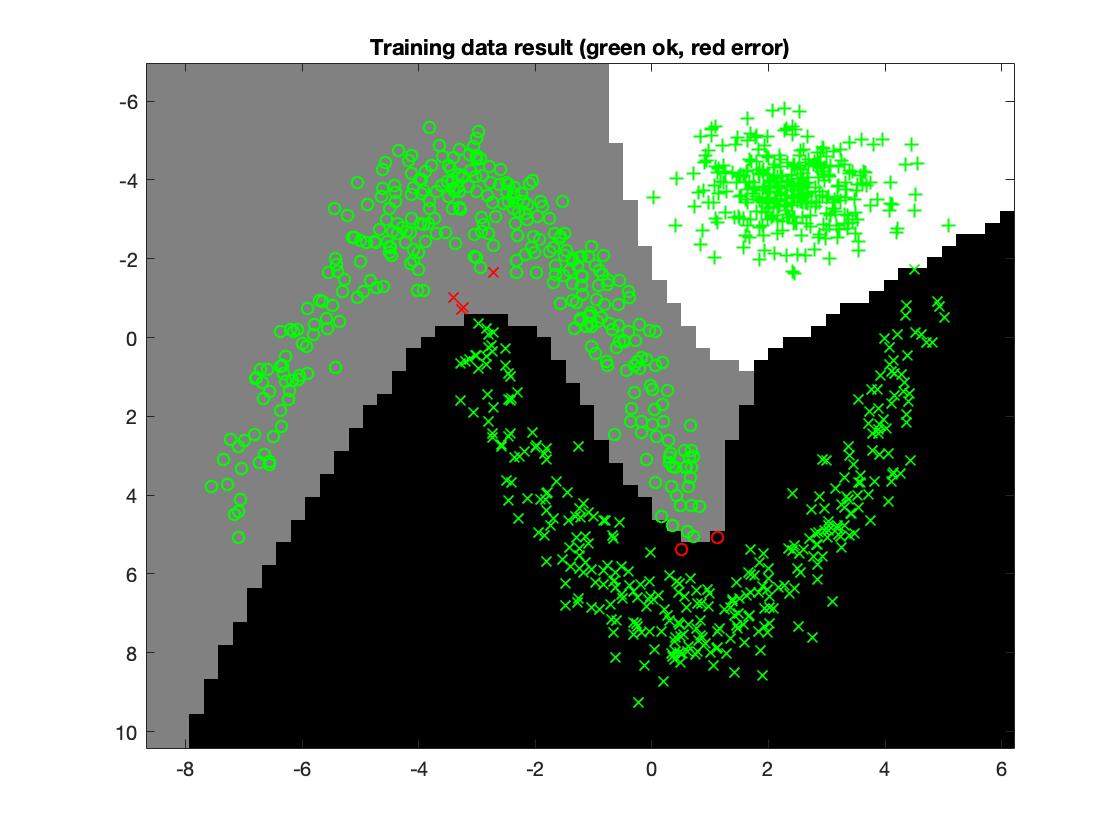
****

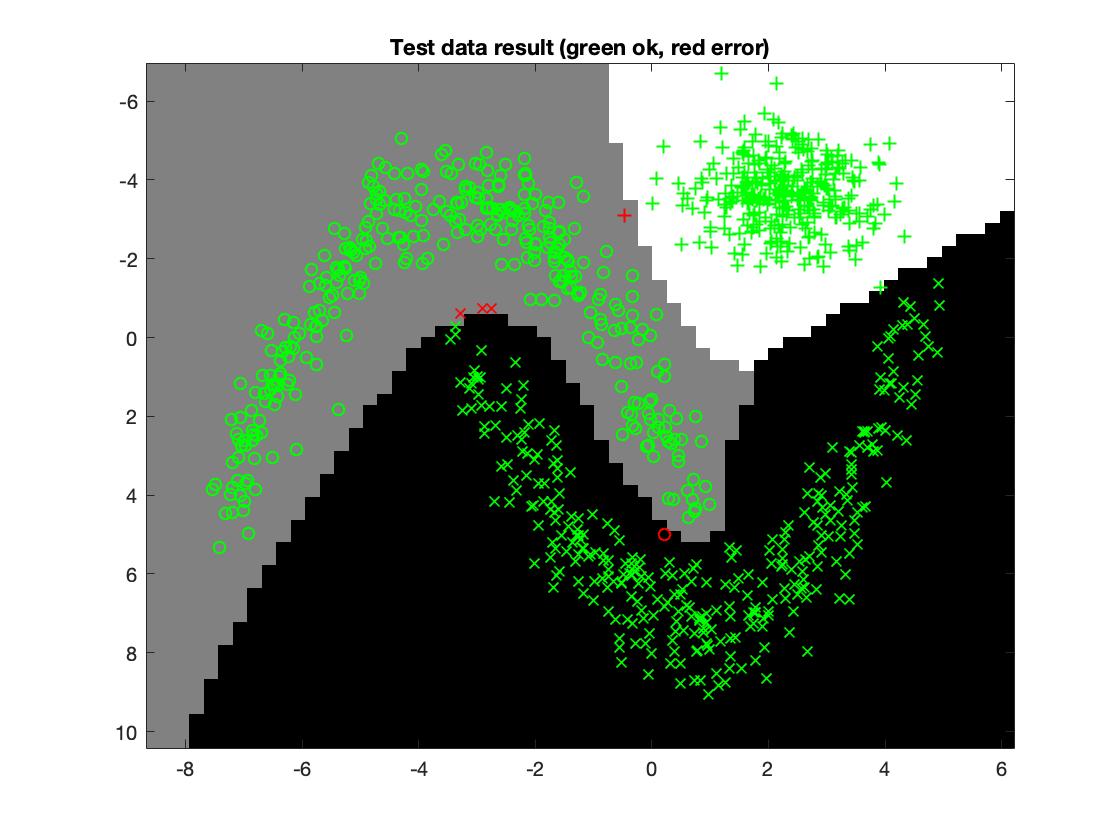
For data set 2 we used the multilayered network. Since this data set needed a pretty complex classifier we used 10 hidden neurons and 7000 iterations was set, though it usually stopped earlier to avoid overfitting. The learning rate was 0.01. With these setting we got an accuracy of 99.5%. 

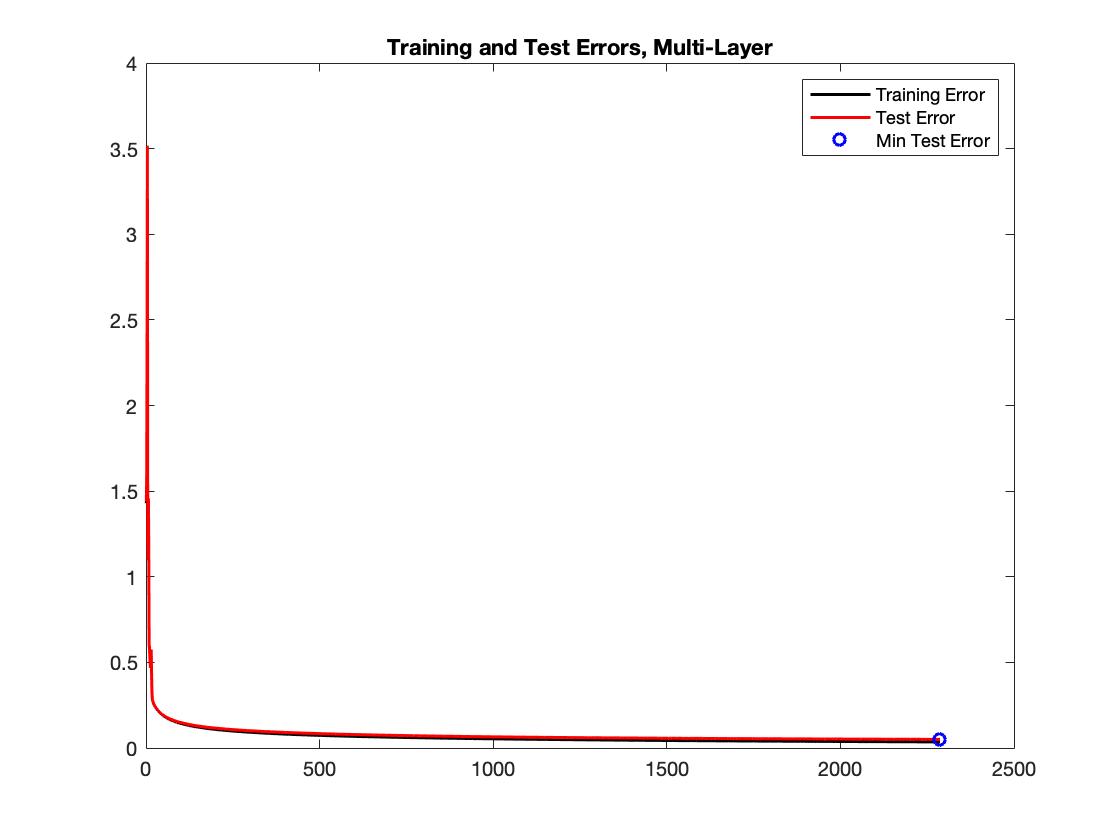


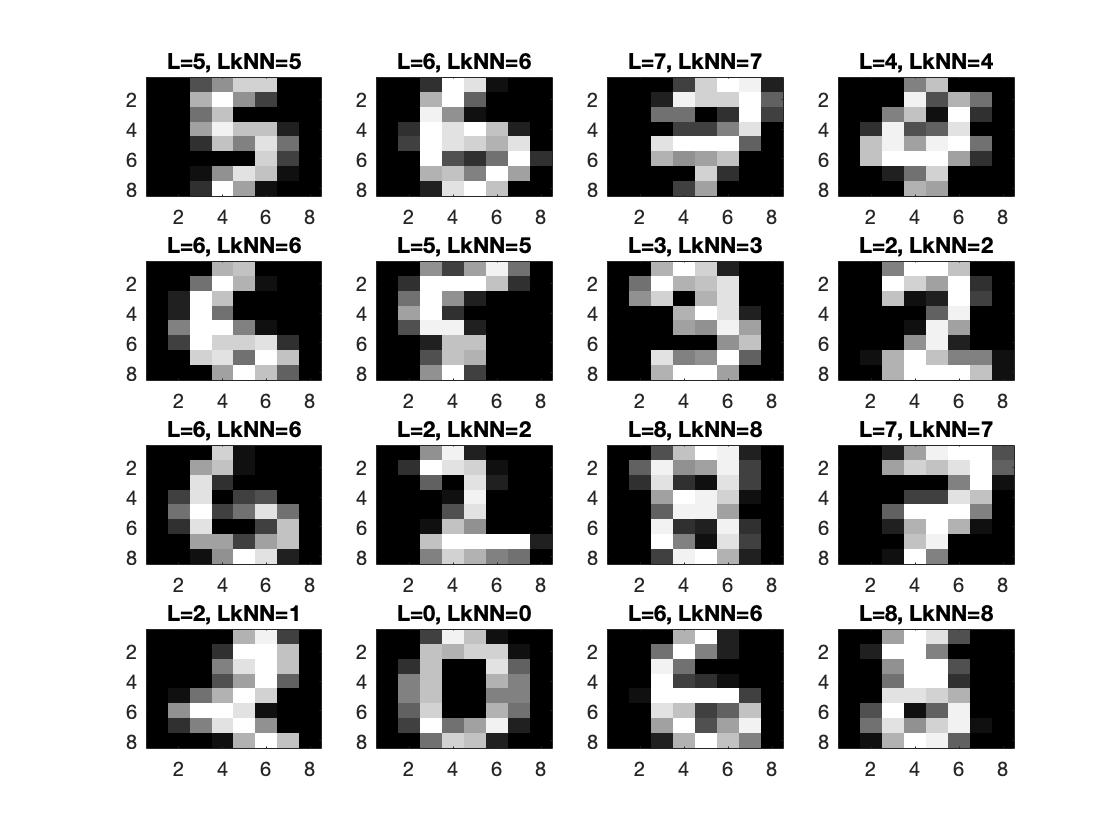


For the third data set, an even more complex classifier was needed so we had to increase the number of neurons in the network. With 30 neurons in the hidden layer, 7000 iterations and a learning rate of 0.01 we got an accuracy of 99.5%. This time the network needed all 7000 iterations. 



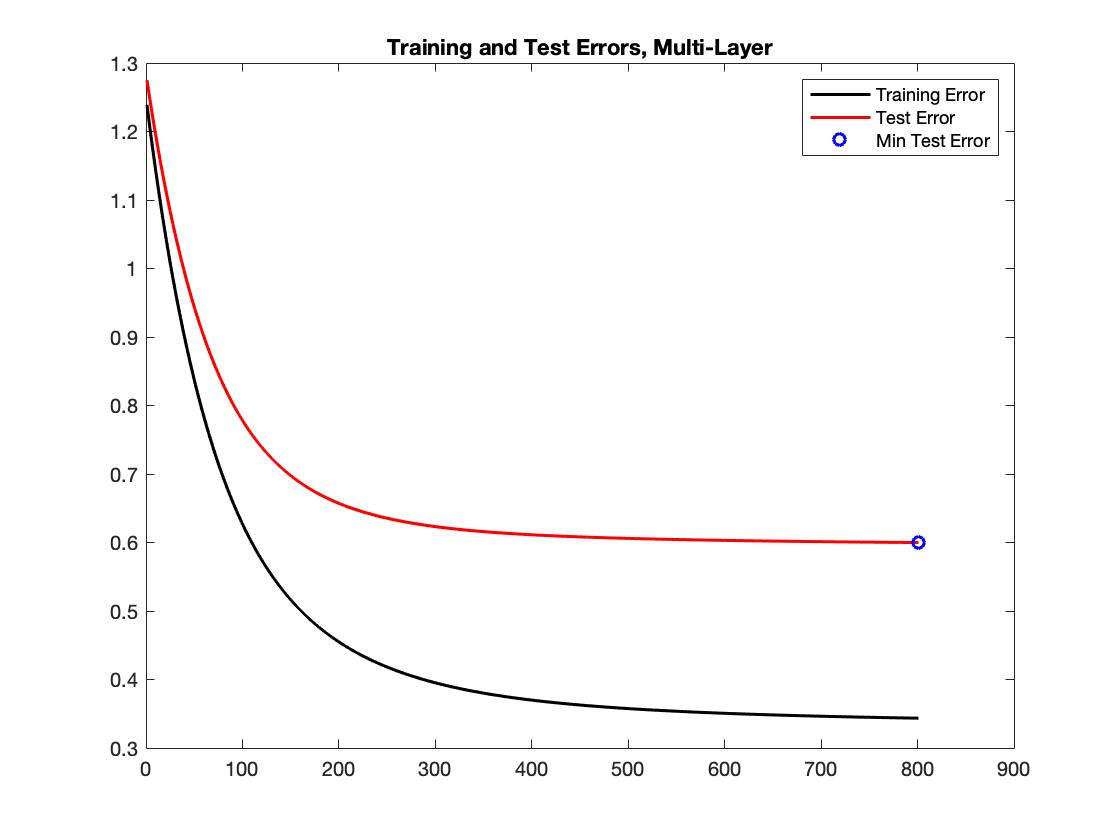


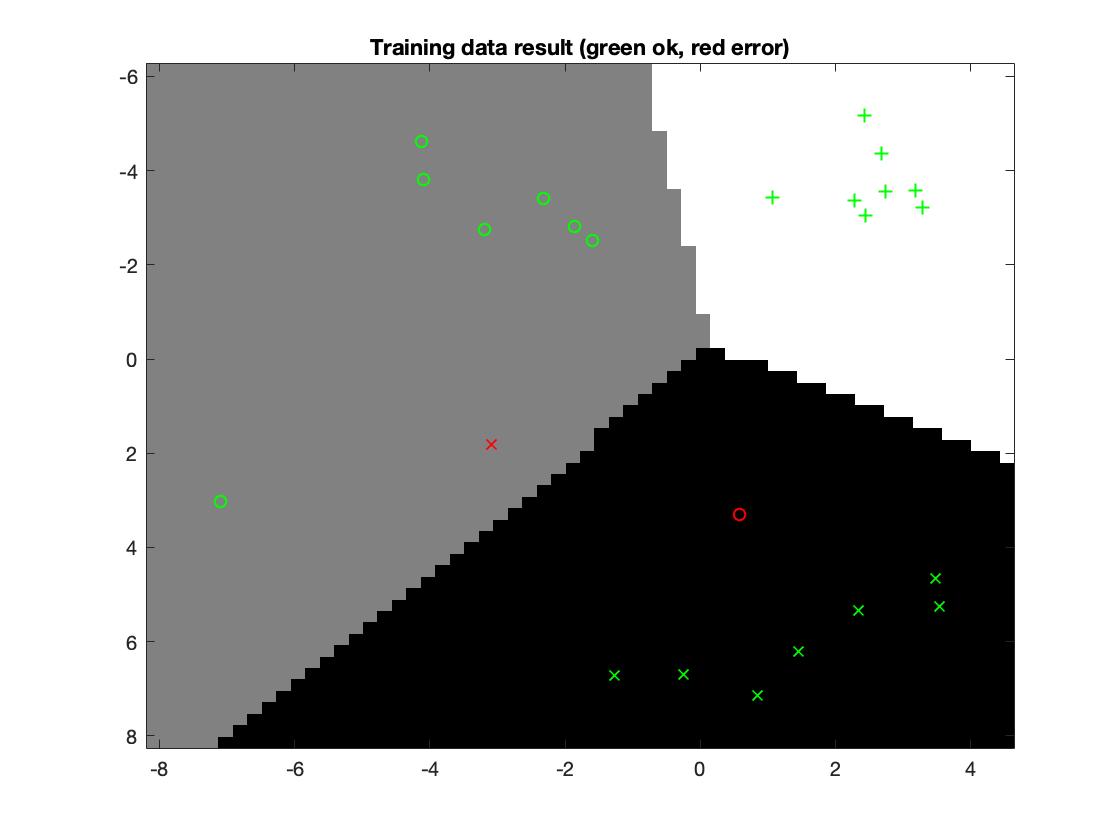
In the last data set, 60 hidden neurons were needed to always get above 96% accuracy. With the same number of iterations and learning rate, 96% accuracy was achieved. It always stopped before reaching 7000 iterations.

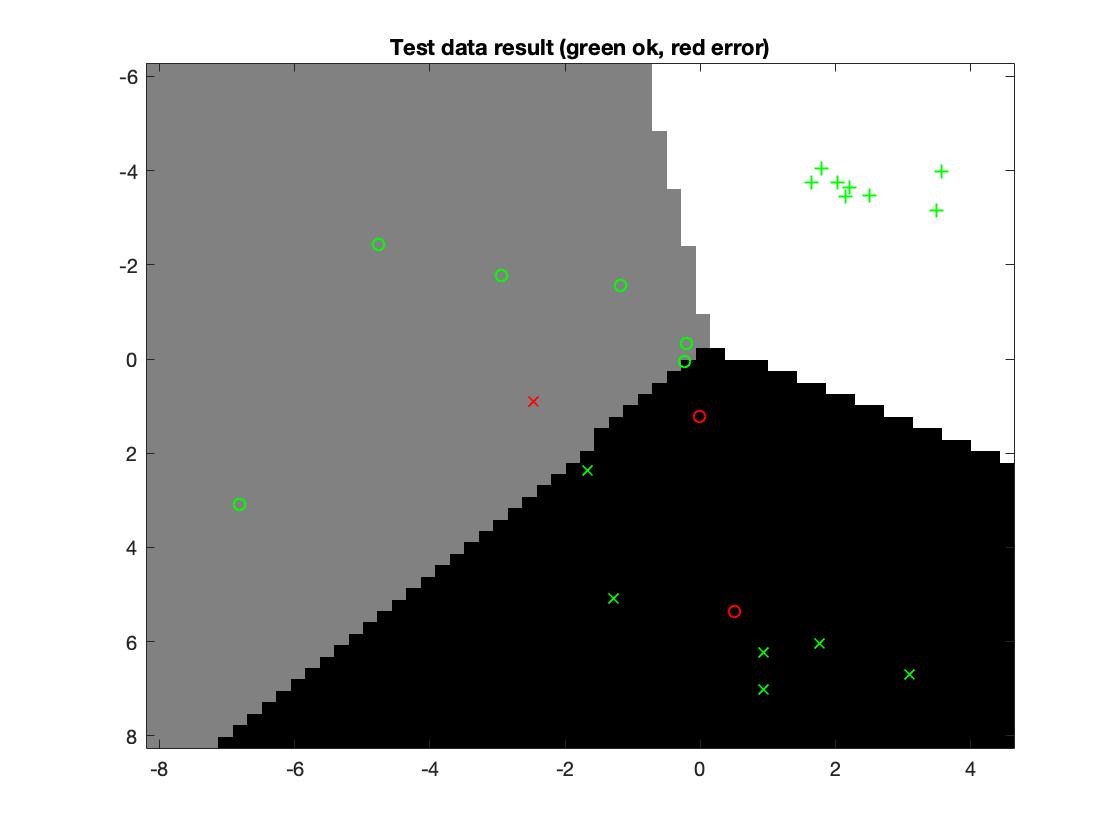


1. **Present the results, including images, of your example of a non-generalizable backprop solution. Explain why this example is non-generalizable.**

When a solution is not generalizable, it means that the network will perform very well for the training set because it memorized the training examples, but it cannot learn to adapt to other situations and data sets. This means that when presented to new data, the network will classify the new data points far worse than it classified the training examples. This can clearly be seen in the figures below. The interesting is the errors, we see that the test error is almost double the train error.





****

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

The kNN classifier is more easily implemented and also could be used on all 4 og the data set without changing the k value and still get pretty good accuracy. Whereas with the neural nets we needed to change the number of neurons to be able to classify different data sets. We could set the number of neurons to a high number for all data sets, but the it takes a lot of time computing the result, even for simple problems. Here kNN has an edge. The problem with kNN is that it is not learning, you need to store all data points to be able to get a good classification, with neural nets you train on one data set and then you can use data points not seen before to be classified without needing to store them. When the network fails to classify new data points, then you store them and retrain your network. We save memory this way.

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

We could use a momentum term in our back-propagation algorithm for example.