TBMI26 – Computer Assignment Report
Supervised Learning

Deadline – February 12 2018

Author/-s:
Albin Bergström albbe673
Lovisa Hassler lovha997

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.**
   Dataset 1 contains a cloud of two classes of dots, which are almost linearly separable but a non-linear classifier would not perform much better than a linear one.

   Dataset 2 is similar to dataset 1 but is not linearly separable and thus needs a non-linear classifier.

   Datset 3 contains three classes of dots which are not linearly separable either and requires a non-linear classifier.

   Dataset 4 contains images of 8x8 pixels containing handwritten digits (0-9), which requires a non-linear classifier.

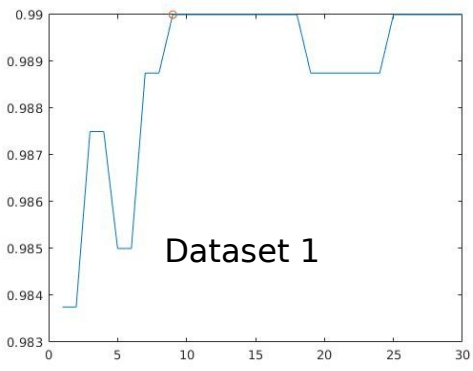
2. **Explain why the down sampling of the OCR data (done as pre-processing) result in a more robust feature representation. See** http://archive.ics.uci.edu/ml/datasets/Optical+Recognition+of+Handwritten+Digits
   It reduces dimensionality and gives invariance to small distortions. Noise is reduced and the algorithm is more generally trained, e.g. not overtrained on very specific samples.

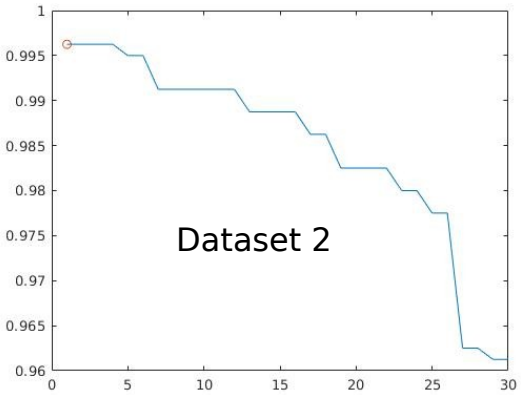
3. **Give a short summery of how you implemented the kNN algorithm.**
   We used the MatLab function knnsearch() to find the closest samples to each samples. It returns, for each sample, a vector containing the indices of the k nearest samples.

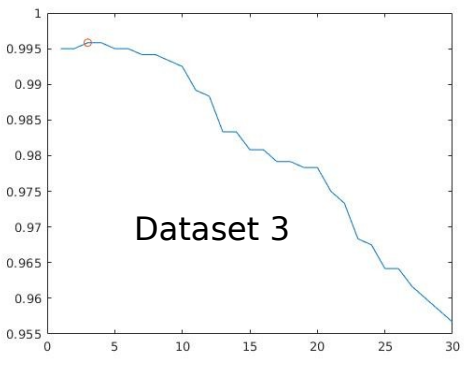
4. **Explain how you handle draws in kNN, e.g. with two classes (k = 2)?**
   After the closest neighbours has been found, the number of samples for each classes is counted. If the two classes with the highest number of close samples have the same number of samples (a tie) the sample that is farthest away is discarded, and then the samples are counted again to see if the tie was solved. This is repeated until there is no tie.

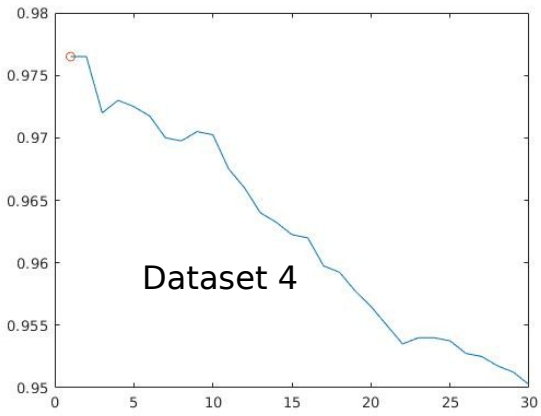
5. **Explain how you selected the best k for each dataset using cross validation. Include the accuracy and images of your results for each dataset.**
   By having a loop that goes from k = 1 to k = 30 and then inside that loop running knn on 4 different bins of data and taking the average accuracy of knn on the 4 bins. This way we get an average accuracy for each k, created from 4 different bins of one dataset, and then select the k with highest accuracy.

   The bins are randomly selected data from the dataset, to ensure an even distribution. In the plots below, x-axis = k and y = accuracy.







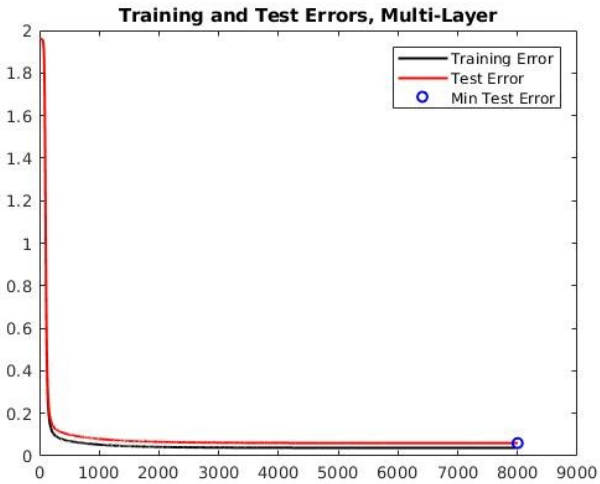


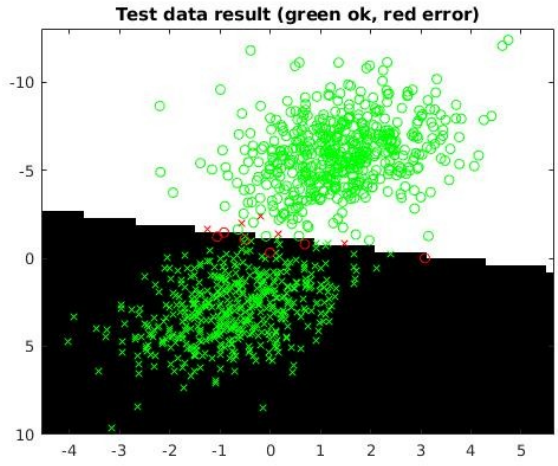
6. **Give a short summery of your backprop network implementations (single + multi). You do not need to derive the update rules.**
   Both the single- and multi-layered network used the hyperbolic tangent (tanh) as activation function.

   The network is created with a bias weight both in the input layer and in the hidden layer. In the multi-layered network this has to be taken into consideration while backpropagating, which we did by discarding the first column in Vout as well as the first row in U.

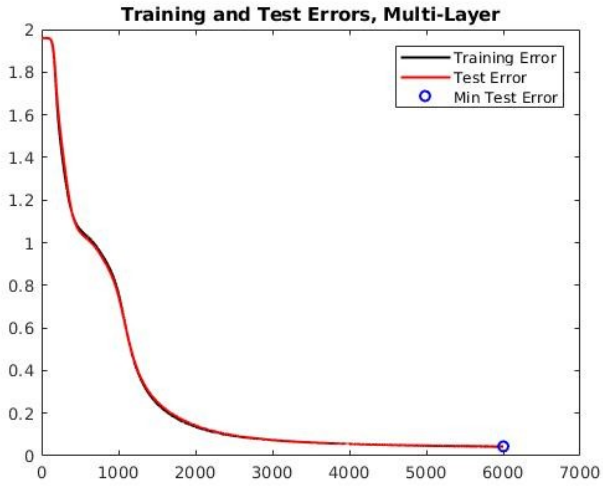
7. **Present the results from the backprop training and how you reached the accuracy criteria for each dataset. Motivate your choice of network for each dataset. Explain how you selected good values for the learning rate, iterations and number of hidden neurons. Include images of your best result for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

   Dataset 1
   Hidden neurons: 10
   Epochs: 8000
   Learning rate: 0.005
   Weight initialization: random[-0.005, 0.005]
   Accuracy: 0.993

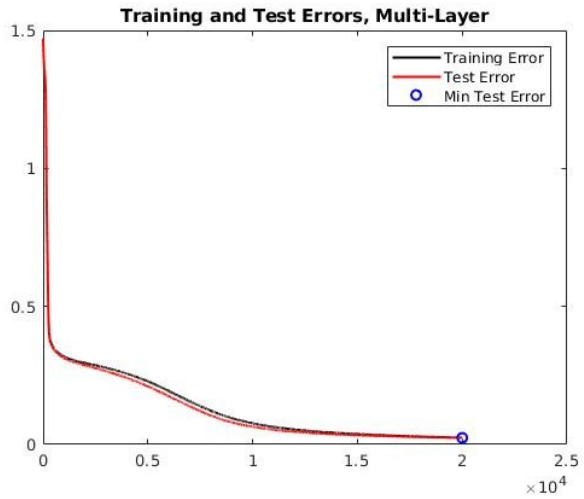


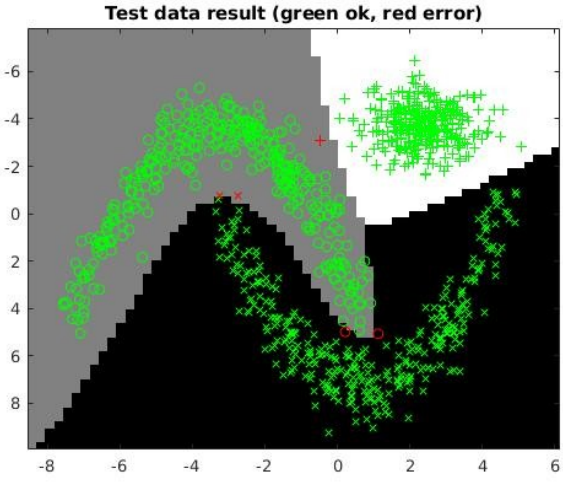


   Dataset 2
   Hidden neurons: 20
   Epochs: 6000
   Learning rate: 0.005
   Weight initialization: random[-0.005, 0.005]
   Accuracy: 0.991



Dataset 3
Hidden neurons: 40
Epochs: 20000
Learning rate: 0.005
Weight initialization: random[-0.005, 0.005]
Accuracy: 0.99099

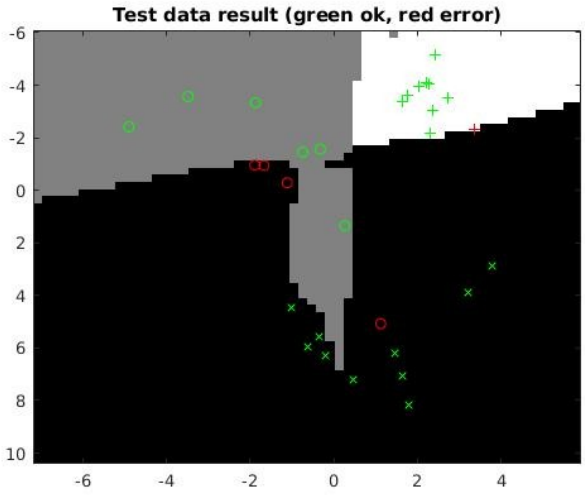


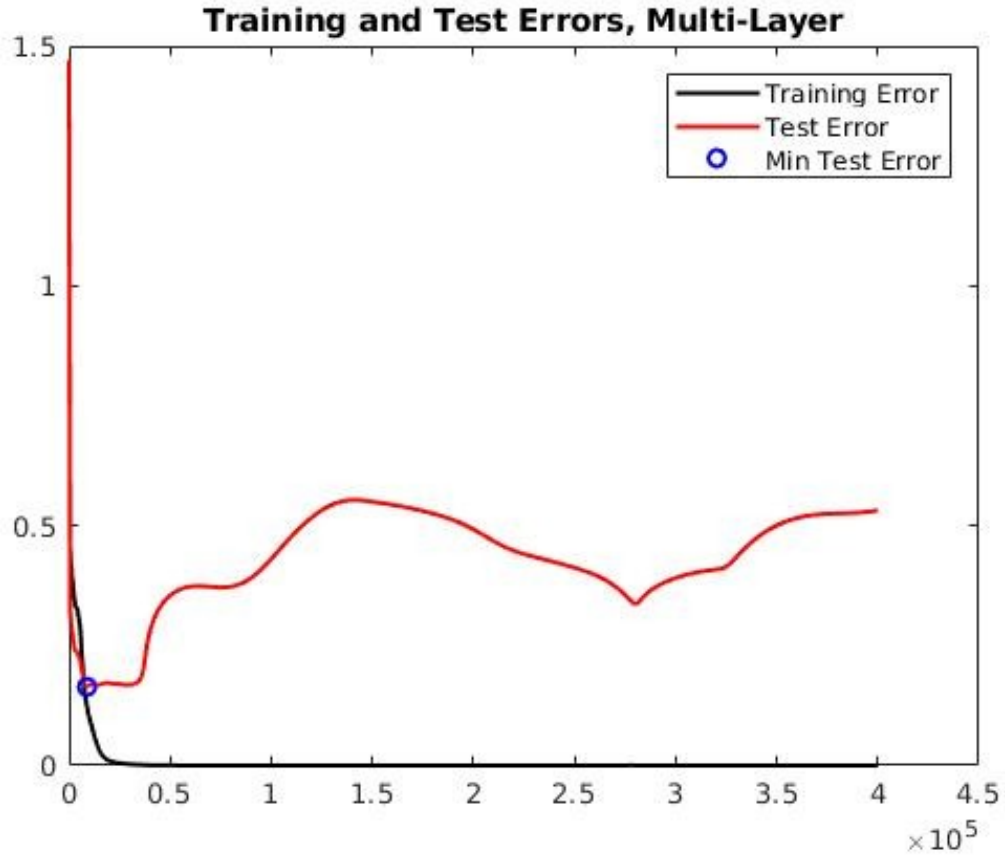


Dataset 4
Hidden neurons: 80
Epochs: 40000
Learning rate: 0.001
Weight initialization: random[-0.0005, 0.0005]
Accuracy: 0.95993 = 0.96

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

The system has been overtrained, as can be seen in the graph where the training error converges to 0 while the test error increases with the increasing number of epochs trained after a certain point.





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

The knn requires no previous training but is run directly on the dataset. It is also pretty fast in these circumstances, but since it is

dependent on the number of data samples as it needs to compare each sample to each other sample it will get slower.

The neural network and the backpropagation algorithm requires some pre-processing where the network is trained on the data. This also means that the network can get overfitted to a certain dataset whereas the knn algorithm never has this problem. It is however independent of the size of the dataset that is going to get classified and can therefore outperform the knn algorithm once it is trained and in cases where the dataset is large enough.

One advantage with the knn over the neural network is the simplicity of it and the time required to implement it.

10. **Do you think there is something that can improve the results? Pre-processing, algorithm-wise etc.**
For the neural network an activation function can be added to the ouptut layer to decrease the time it takes for the network to converge.

The input data could be normalized between -1 and 1 to better fit the tanh() activation function.