TBMI26 – Computer Assignment Report Supervised Learning

Deadline – March 15 2020

Author/-s:
Thomas Indrias

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.**
   It is seen in the datasets that the first set can separated by a line (linearly separable) therefore a linear classifier is sufficient. However, the other datasets need non-linear classifiers because they're not linear 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**
   Down sampling reduces noise and the dimensions of the data which results with better features.

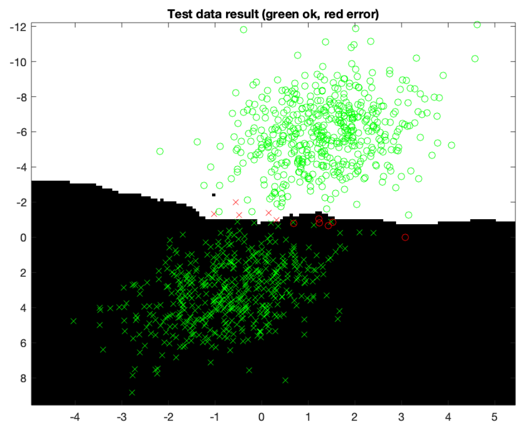
3. **Give a short summary of how you implemented the kNN algorithm.**
   Firstly, we calculate the distance for a point X1 to all other points XTrain.
   After finding the distance from all X to all XTrain, we get a matrix where each row indicates the distance from point Xi to all other XTrain.

   Secondly, we sort each row which gives us the smallest distance for all X and then extract k closest points to a new matrix. The classification is then done by the most frequent values in each row of the new matrix.

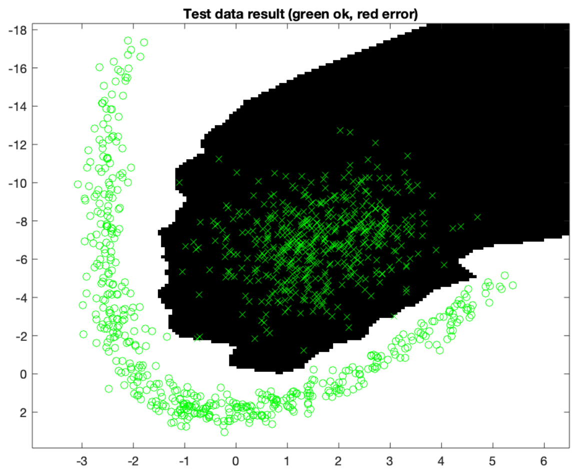
4. **Explain how you handle draws in kNN, e.g. with two classes (k = 2)?**
   It is handled by taking the smallest value in the k sized array for a point. [1 2 1 2] => 1
   Another way to handle draws is to decrease k until there's no tie.

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.**
   To select the best k, I made a script that divides the dataset into bins and uses one for testing and the rest for training. The accuracy is then calculated by dividing the validation error by the number of bins. This is then iterated for k = 1, 2, …, 30. Function returns a vector

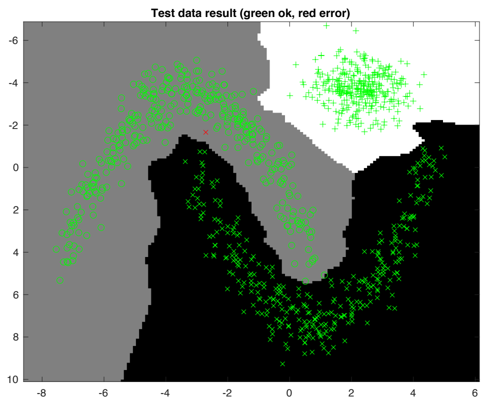
containing a score for each k. The best k is selected by that vector.
numSamplesPerLabelPerBin is set as infinite.



1st dataset:
Acc: 99.00%
k: 6

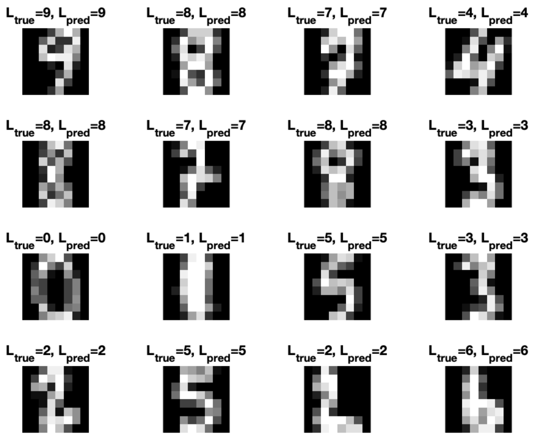


2nd dataset:
Acc: 100%
k: 1



3rd dataset:

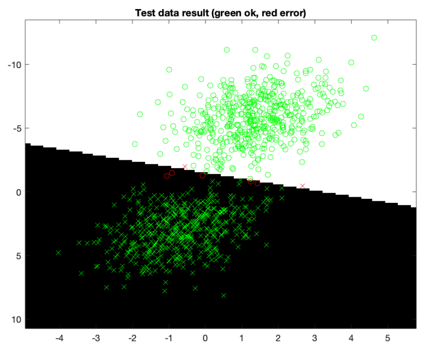
Acc: 99.90%
k: 1

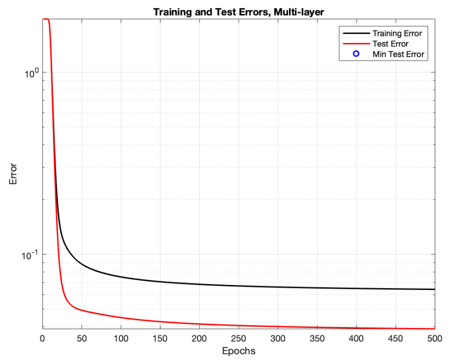


4th dataset:
Acc: 98.41%
k: 5

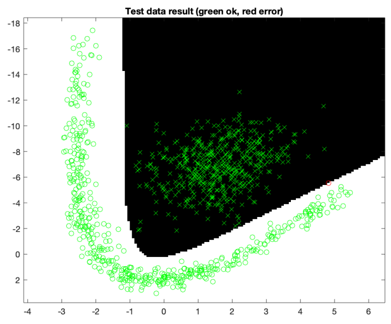
6. **Give a short summary of your backprop implementations (single + multi). You do not need to derive the update rules.**
In both implementations a vector of ones had to be added to the training and test set so that each bias weight could be calculated. The weight matrix, W0 was initialized with random scalars derived from a standard normal distribution with the size NxM. The size of our weight doesn't change for number of samples. The size indicates the amount of input and output nodes. For the hidden layer in multi, a hyperbolic tangent was used as the activation function.

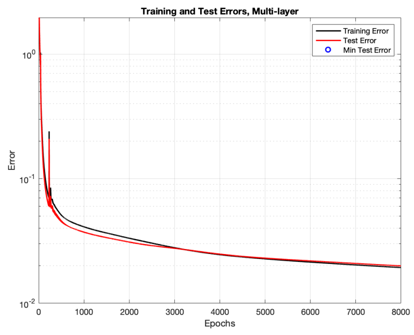
7. **Present the results from the neural network training and how you reached the accuracy criteria for each dataset. Motivate your choice of network for each dataset. Explain how you selected good values for the learning rate, iterations and number of hidden neurons. Include images of your best result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





1st dataset:
Acc: 99.3 %
Hidden neurons: 1
# Iterations: 500
Learning Rate: 0.1





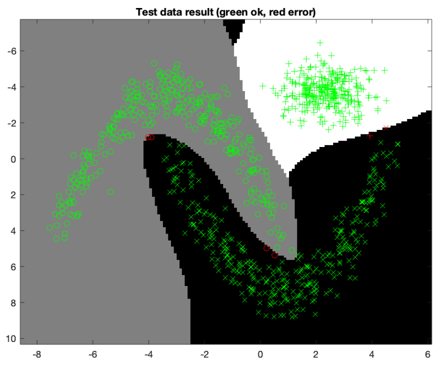
2nd dataset:

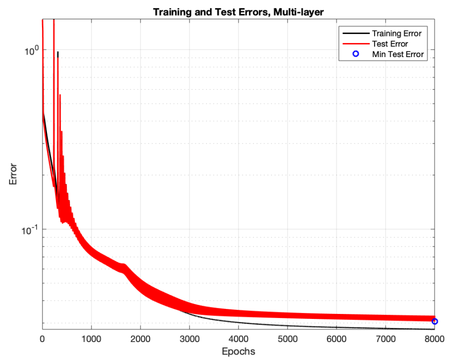
Acc: 99.9 %

Hidden neurons: 10

# Iterations: 8000

Learning Rate: 0.1





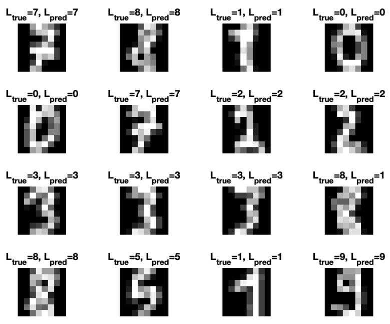
3rd dataset:

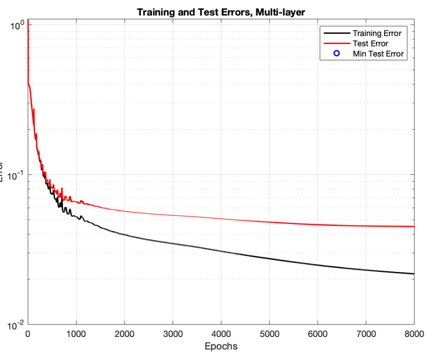
Acc: 99.4 %

Hidden neurons: 20

# Iterations: 8000

Learning Rate: 0.1





4th dataset:

Acc: 96.787 %

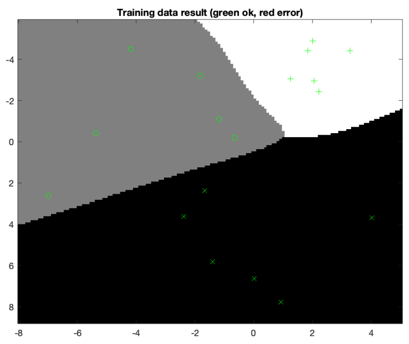
Hidden neurons: 50

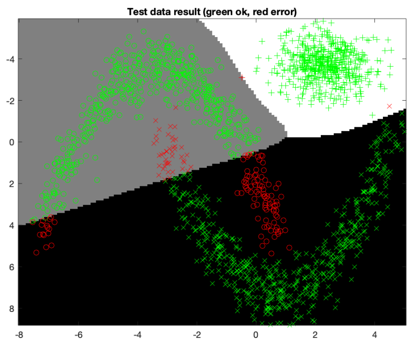
# Iterations: 8000

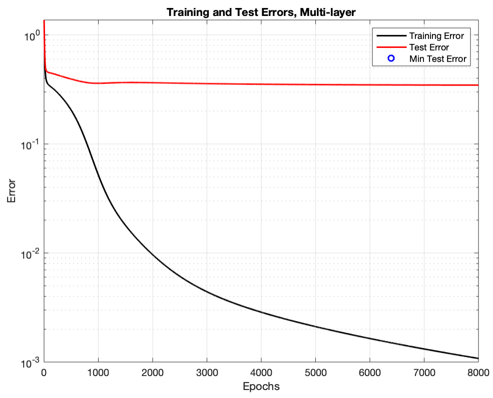
Learning Rate: 0.01

The first dataset could easily be classified through one neuron since it's linearly separable. However, for the other sets, the neurons had to be increased to correctly classify them. Especially the last set, which has 10 classes to be classified and in turn, result to a more complex problem. Therefore, the number of neurons had to be increased to around 80 neurons to satisfy the accuracy limit. The amount of iterations was decided by looking at the error and choose the epoch where the error starts to converge or is at a global minimum. The smaller learning rate used, the more iterations we had to go through to find a suitable minimum error.

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







From the example above, we're trying to generalize with training set that's one tenth of the data set (1% training set, 99% test set). This is done by creating 100 bins and combining them except for the first bin which is used as training data. For the example, the parameters were set to 50 neurons, 8000 iterations and 0.01 steps to fit the model. As expected, the results were poor as the model was no longer generalizable and representative to the test set. This can be seen in the figure "Training and Test Errors, Multi-layer".

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

k-NN is easy to implement and to understand. It requires no training nor modeling. However, it's not computationally efficient for large sets of data because it has to compare the test data to all training points. It is also very sensitive to outliers if k is low as it chooses the neighbors based on some distance formula.

Artificial Neural Network (ANN) is very good to use if you want to generalize non-linear data. Once trained the predictions are fast. On the downside, it's relies on iterative methods and steps which is computationally expensive and time consuming. ANN also requires a large amount of training data to correctly generalize a model, otherwise it might lead to overfitting.

10. **Do you think there is something that can improve the results? Pre-processing, algorithm-wise etc.**
    There's not much to pre-process in the provided datasets except to normalize the values in the datasets. Additionally, in the fourth dataset, PCA (Principal Component Analysis) could be used to extract features and reduce dimensionality.