TBMI26 – Computer Assignment Report Supervised Learning

Deadline – February 12 2018

Author/-s:
Villiam Rydfalk
Jonathan Sjölund

In order to pass the assignment you will need to answer the following questions and upload the document to LISAM. 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: Works with a linear classifier.
   Dataset 2: Clearly needs a non-linear classifier.
   Dataset 3: Clearly needs a non-linear classifier.
   Dataset 4: A lot of numbers have points in the same places and thus we will need 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**
   Reduces dimensionality, which improves the calculation time, and gives invariance to small distortions.

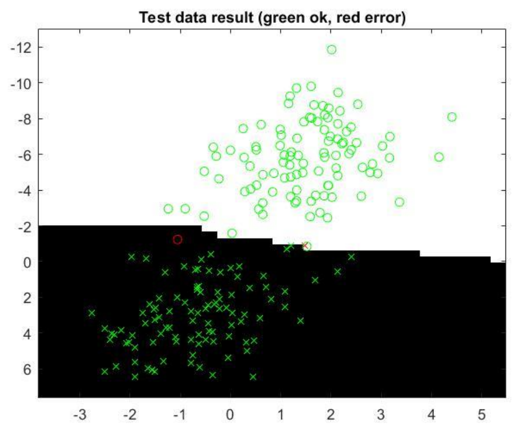
3. **Give a short summery of how you implemented the kNN algorithm.**
   First, a distance matrix between all observations are calculated. This gives us a matrix with the distance between all observations. For each observation in the distance matrix we extract and sort the distances from shortest to longest. We then extract the k-nearest neighbours for that observation. We then count how many neighbours has a specific class and then take the max to get the class with the most amount of nearest neighbours. That class is then the predicted class for that observation.

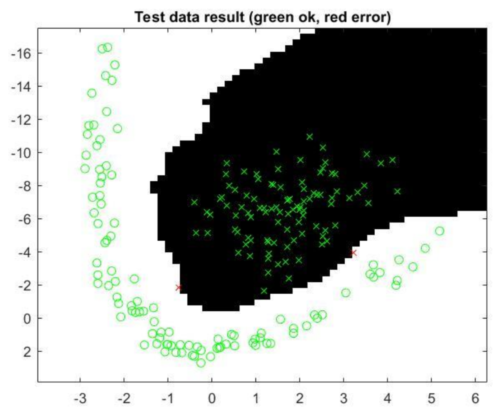
4. **Explain how you handle draws in kNN, e.g. with two classes (k = 2)?**
   Count the number of nearest neighbours for each class and put them in a vector and take max of the vector. If two values have the same max, then the max function will pick the first one.

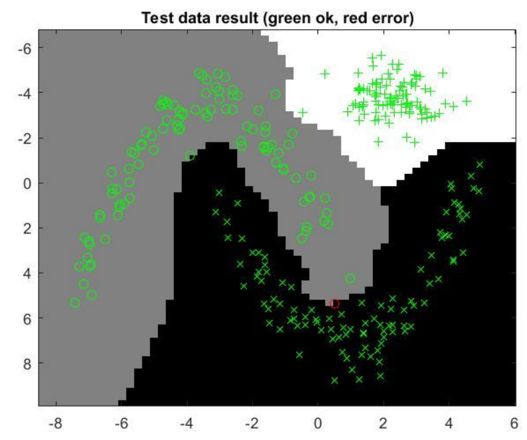
5. **Explain how you selected the best k for each dataset using cross validation. Include the accuracy and images of your results for each dataset.**
   Using 5-fold cross validation the data was divided into 5-folds of equal size. One-fold was chosen as test data and the rest as training data. K-NN was then used for five different *k: s* (1,2,3,4,5) and the accuracy was saved for each test of *k* for a specific combination of folds. After all permutations of test and training data had been used to calculate the K-NN for each *k* an average accuracy was calculated for the *k: s*. The k with the largest average was chosen as the best *k* for that dataset.

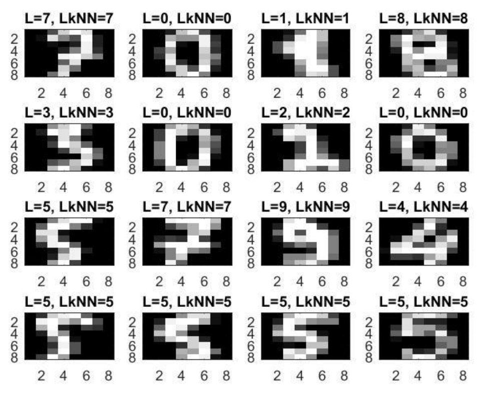
Table 1: *Accuracy for each dataset when using k-nn*

Datasets (1,2,3,4)	Best *k*	Accuracy
1	1	0.99
2	1	0.99
3	1	0.9967
4	1	0.98









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

   Training and test data was manipulated as following:

$$\text{From } X = [x_1,\ldots,x_n]^T \text{ to } X = [1,x_1,\ldots,x_n]$$

   where X contains *n* features plus the bias = 1.

   The weights were initialized by having the number of unique classes (outputs) as rows and number of features (inputs) as columns.

   For the forward propagation the outputs where calculated as

$$Y = \sigma(W*X)$$

   where σ is the activation function. For single layer neural network

$$\sigma(W*X) = W*X$$

   while for multilayer neural network

$$\sigma(W*X) = \tanh(W*X)$$

   was the activation function.

   The gradient was calculated as the following:

$$\varepsilon(W) = \sum_{i=1}^{N}(\sigma(W^T X_i) - y_i)^2 \rightarrow \frac{d\varepsilon}{dw} = 2\sum_{i=1}^{N}(\sigma(W^T X_i) - y_i)\sigma'(W^T X_i)X_i).$$

   The weights were updated as

$$W_{t+1} = W_t - \eta\frac{d\varepsilon}{dW}.$$

   For single layer it was

$$\text{grad\_W} = (Y_{\text{training}} - D_{\text{training}})*\ \sigma(W_{\text{out}} * X_{\text{training}})^T$$

   and for multilayer it was

$$\text{grad\_V} = (Y_{\text{training}} - D_{\text{training}})*\ \sigma(W_{\text{out}} * X_{\text{training}})^T$$

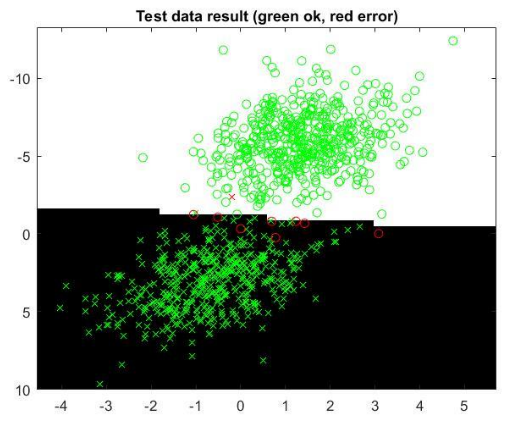
$$\text{grad\_W} = V_{\text{out}}^{T*}(Y_{\text{training}} - D_{\text{training}}).*(1-\tanh(W_{\text{out}}*X_{\text{training}}).\hat{}2)*\ X_{\text{training}}^T.$$

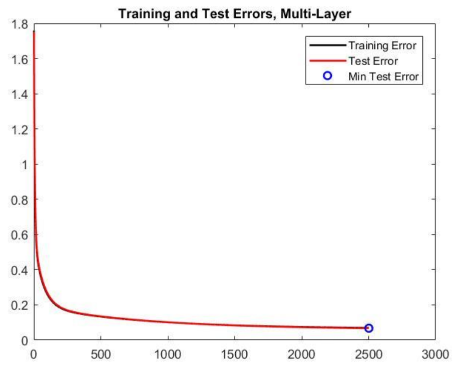
7. **Present the results from the backprop training and how you reached the accuracy criteria for each dataset. Motivate your choice of network for each dataset. Explain how you selected good values for the learning rate, iterations and number of hidden neurons. Include images of your best result for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

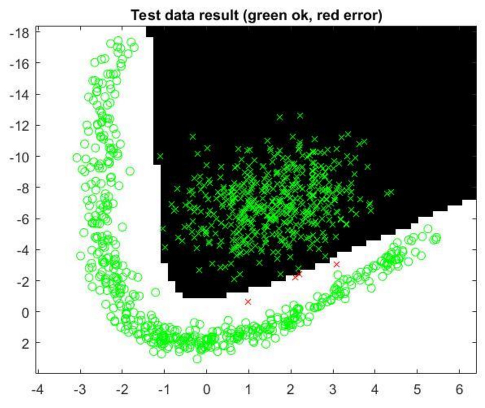
   The selected values in Table 2 where chosen primarily by testing. If the error does not diverge we decrease the learning rate, and if we see that the error still has not converged we increase the number of iterations. If the error is too large after converges we increase the number of hidden neurons.

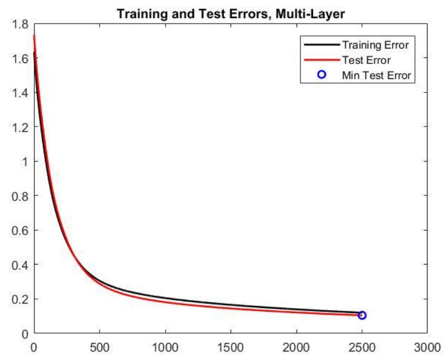
Table 2: *Parameter choice for multilayer neural network*

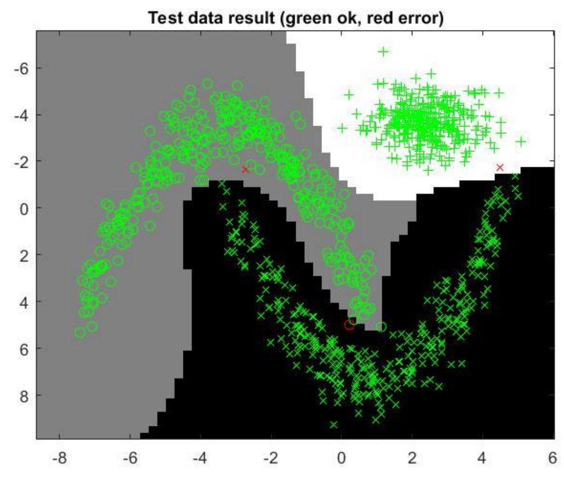
Datasets (1,2,3,4)	**Learning rate**	**Number of iterations (epochs)**	**Number of hidden neurons**	**Time spent training (sec)**	**Accuracy (%)**
1	0.000005	2500	30	10.7908	0.994
2	0.000005	2500	30	11.6327	0.993
3	0.00005	2500	30	11.061	0.99399
4	0.000005	2500	100	117.6394	0.97545



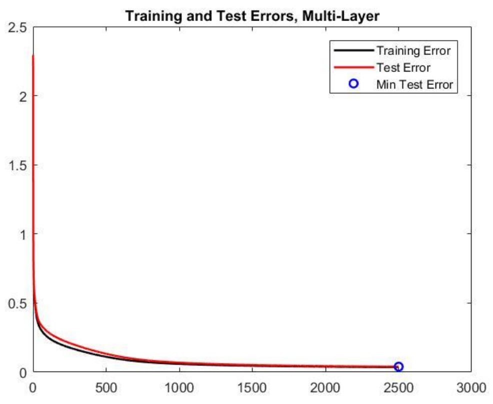




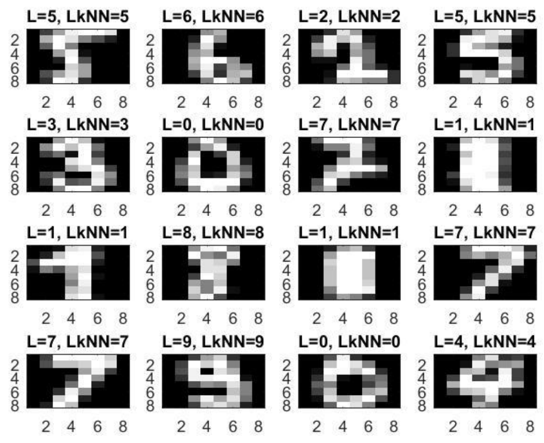




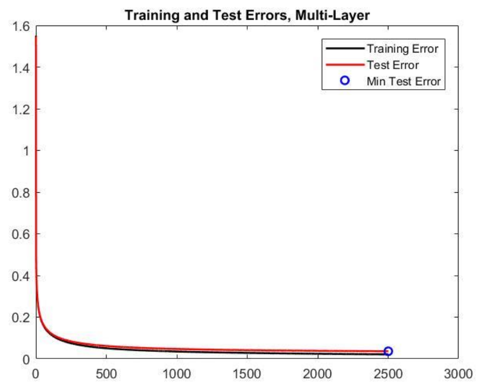
Test data result (green ok, red error)



Training and Test Errors, Multi-Layer
Training Error
Test Error
Min Test Error



L=5, LkNN=5
L=6, LkNN=6
L=2, LkNN=2
L=5, LkNN=5
L=3, LkNN=3
L=0, LkNN=0
L=7, LkNN=7
L=1, LkNN=1
L=1, LkNN=1
L=8, LkNN=8
L=1, LkNN=1
L=7, LkNN=7
L=7, LkNN=7
L=9, LkNN=9
L=0, LkNN=0
L=4, LkNN=4

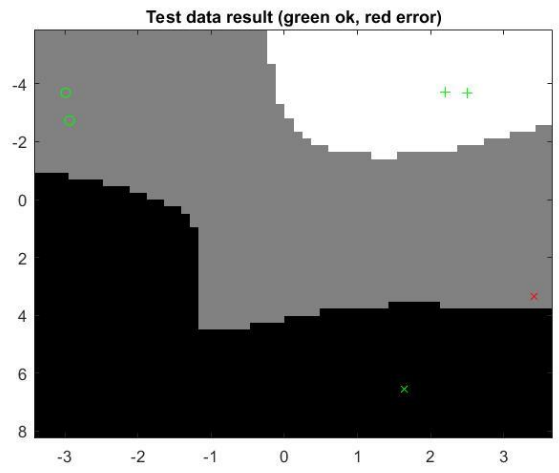


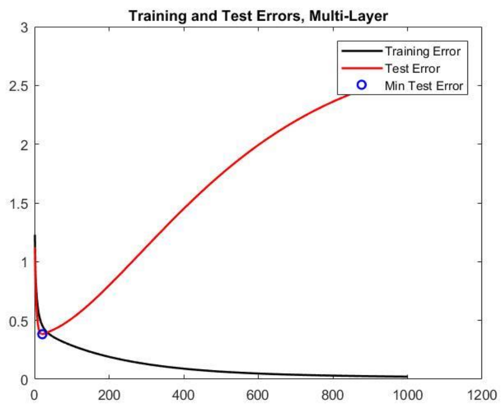
Training and Test Errors, Multi-Layer
Training Error
Test Error
Min Test Error

8. **Present the results, including images, of your example of a non-generalizable backprop solution. Explain why this example is non-generalizable.**
   As we can see in the test data result image if we apply this to other data it will give bad results (bad accuracy). In the training and test error image we can see that the test error (generalization error) is bad and increase after a few epochs.

Table 3: *Parameter choice for non-generalizable backpropagation solution*

Dataset	Learning rate	Number of iterations (epochs)	Number of hidden neurons	Time spent training (sec)	Accuracy (%)
3	0.0005	1000	200	0.278	0.66667





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 gives a good result but requires a lot of memory for big datasets and if we use cross-validation the performance can decrease by a lot for big and complex datasets where we need to test a lot more different k: s.

   Single layer neural network is quick and works well for classifying binary data, for data with more than two classes is problematic with only a single layer and will likely give a bad accuracy (not generalizable to datasets with many classes).

   Multilayer neural network requires good parameter tuning which can be hard to optimize. Performance can also take a big hit if the dataset is very big which this classifier requires to get a good result. However, with good enough parameters it works well for classifying linear and non-linear data.

10. **Do you think there is something that can improve the results? Pre-processing, algorithm-wise etc.**
    For dataset 4 we believe using more hidden layers would improve the result slightly since some of the patterns are more complicated and would be captured better by that. We already have very good result (high accuracy) so improving is difficult.