TBMI26
vicli815
jonto406

Supervised learning: kNN and backpropagation

Q1. Give an overview of the data from a machine learning perspective. Consider if you need linear or non-linear classifiers etc.

*In dataset 1 we have two clusters of data. In our opinion a linear classifier would do an okay job, but the two classes are a bit mixed up which will lead to a lower accuracy. Also a non linear classifier will not get 100 % accuracy unless it overfits the data.
*In the second dataset we have one round cluster of data and the other class distribution is shaped like an arc. This will require a non linear classifier.
*In the third data set we have three clusters with some mixed up data points. The form of these clusters are two arcs and one circular cluster. A non linear classifier will be required to separate the data.
*In the last data set it contains “numbers” in a 8x8 matrix in black and white. It looks hard if not impossible to separate using a linear classifier.

Q2. 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

In the OCR data it has been pre processed into an 8x8 matrix. This is called downsampling and is used to reduce the complexity of the model (by reducing dimensionality and noise) and therefore eases the data classification. Less features also leads to shorter training time.

Q3. Give a short summary of how you implemented the kNN algorithm.

In the kNN algorithm you calculate the distance to the k closest neighbors and then let them have a majority voting regarding class.

First we calculated the euclidean distance (used a row sum) from all training samples to the sample that were to be classified. These distances were then added to a matrix together with the training sample feature. So first column distances and the second classes. We then sorted this matrix from the lowest to highest distance. We then let the k first have a majority vote using the matlab function mode.

Q4. Explain how you handle draws in kNN, e.g. with two classes (k = 2)?

As mentioned before we used mode to settle the voting. In case of a draw this built in matlab function picks the lowest. For example (3, 4) → LPred = 3.

Q5. 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 selected the best k for each dataset by looping through all k values from 1 to 50 and then plot a graph with accuracy on the y-axis and different k values on the x-axis. See attached plots.

TBMI26
vicli815
jonto406

Dataset 1

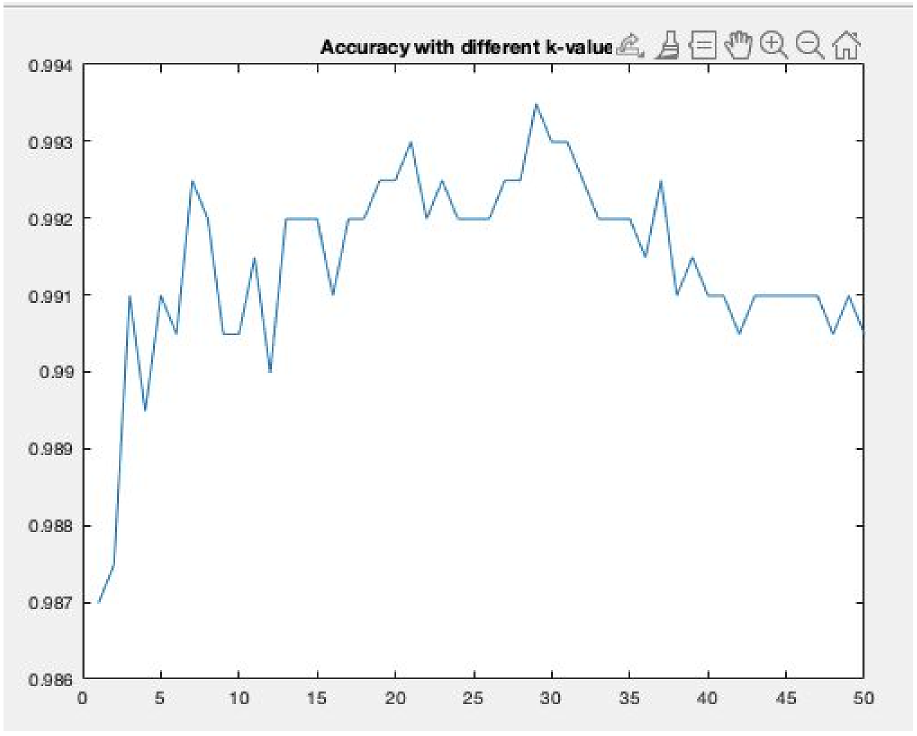


Figure 1: Accuracy with different k-values for dataset 1.

Dataset 2

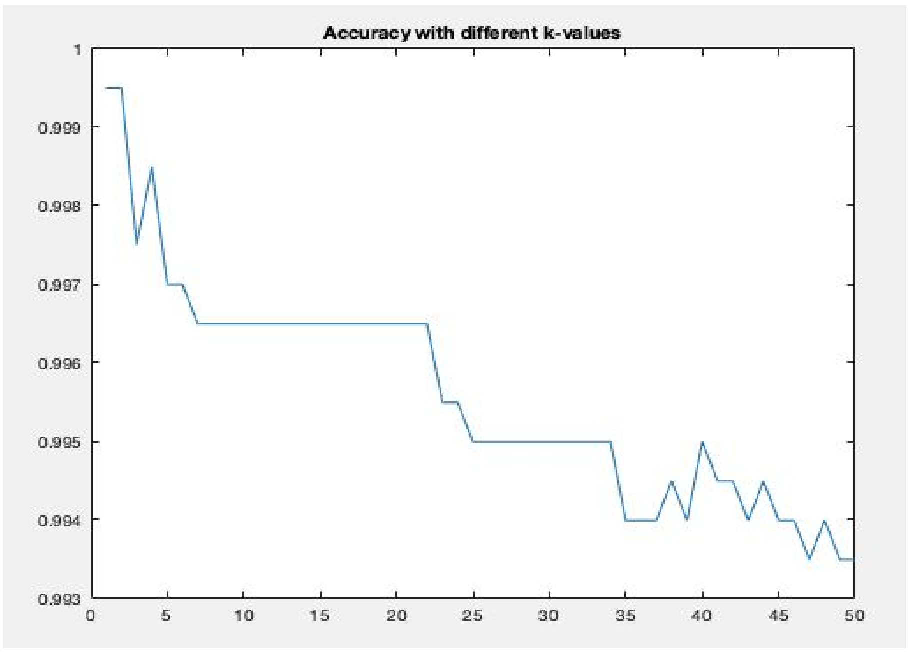


Figure 2: Accuracy with different k-values for dataset 2.

TBMI26
vicli815
jonto406
Data set 3

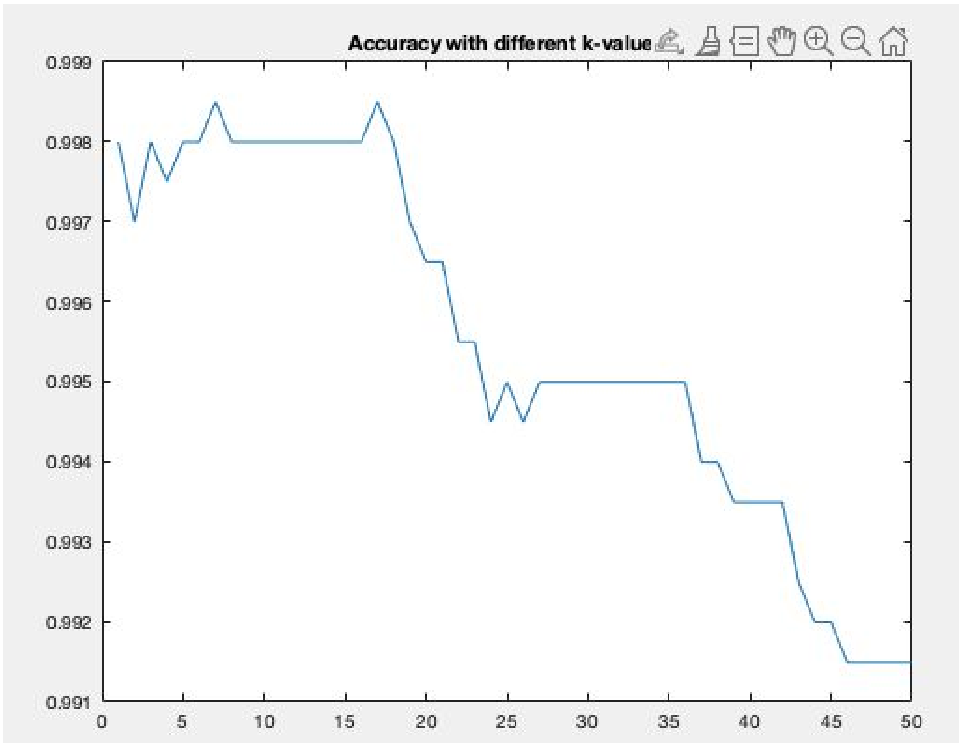
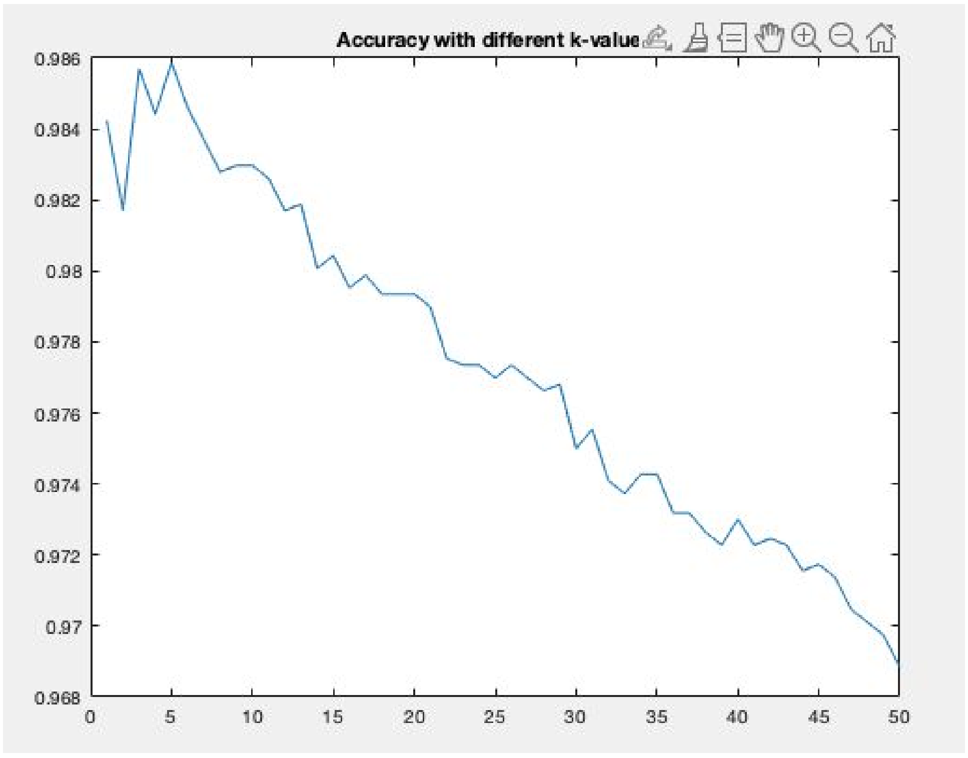


Figure 3: Accuracy with different k-values for dataset 3.

Data set 4



TBMI26
vicli815
jonto406
Figure 4: Accuracy with different k-values for dataset 4.

Data set 1 - classified with kNN

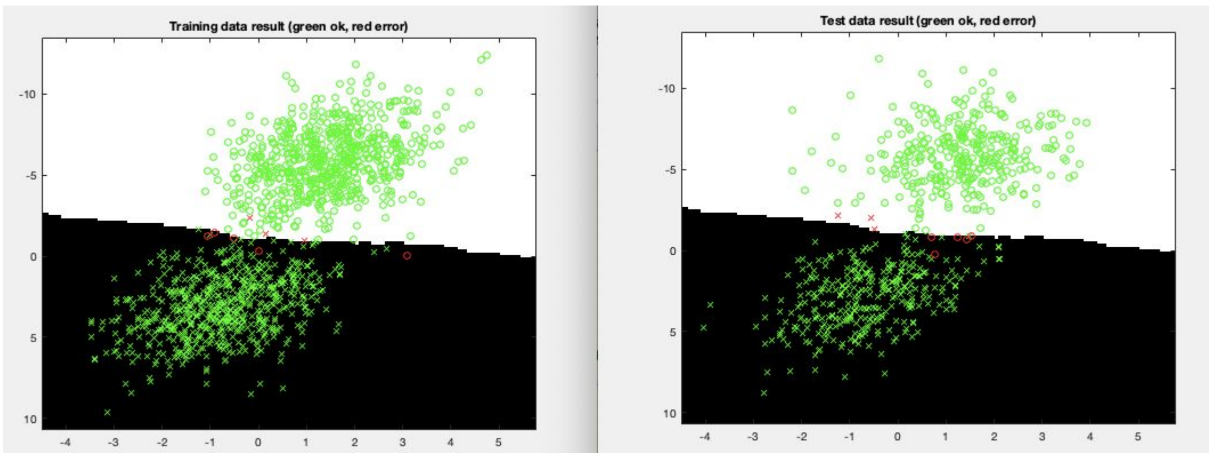


Figure 5: Classification of dataset 1 with 99.3 % accuracy using kNN.

Dataset 2 - classified with kNN

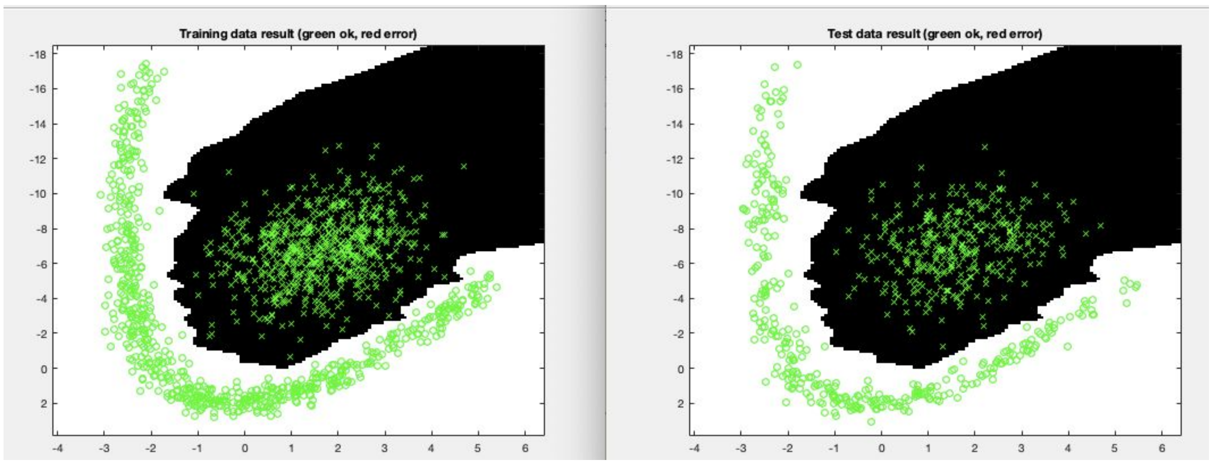


Figure 6: Classification of dataset 2 with 99.9 % accuracy using kNN.

Dataset 3 - classified with kNN

TBMI26
vicli815
jonto406

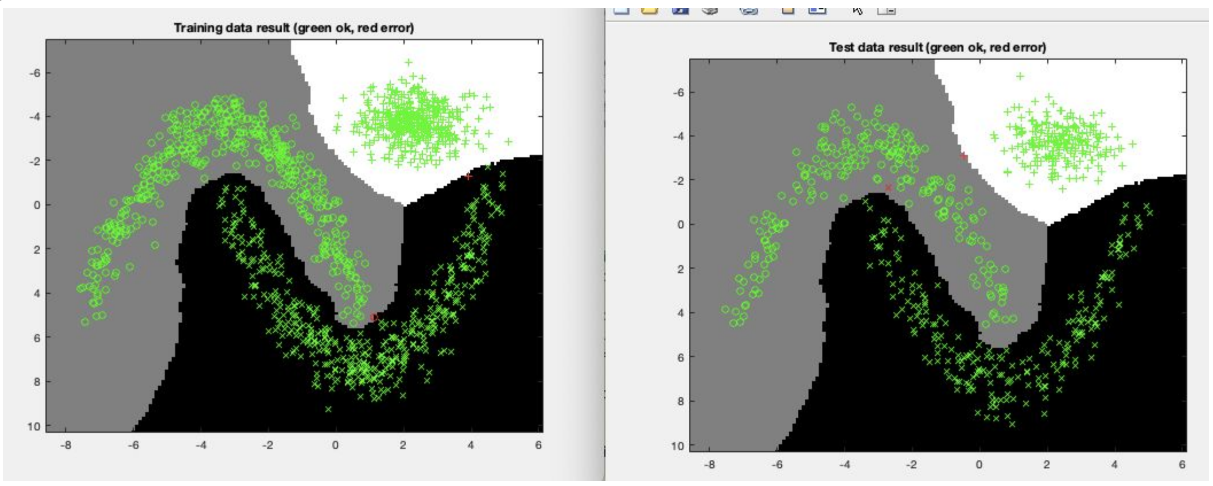


Figure 7: Classification of dataset 3 with 99.85 % accuracy using kNN.

Dataset 4 - classified with kNN

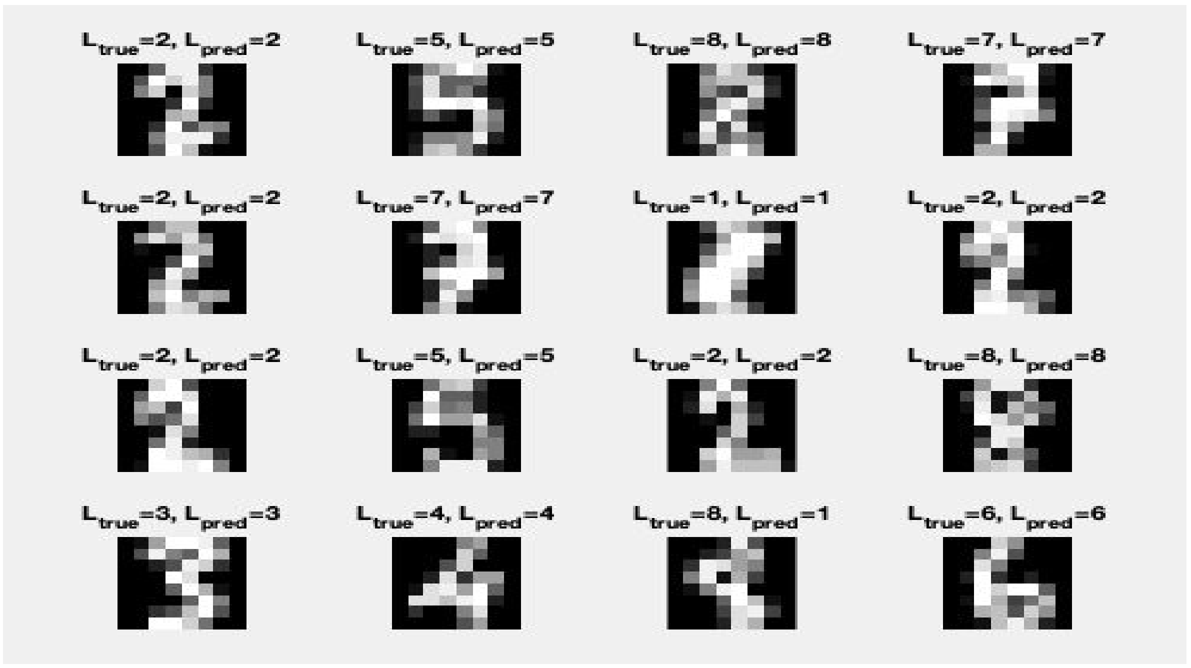


Figure 8: Classification of dataset 4 with 98.6 % accuracy using kNN.

Q6. Give a short summary of your backprop network implementations (single + multi). You do not need to derive the update rules.

Single layer:

First, we added bias to the input by adding a column of ones as the last feature. Then, we initiate the weight W0 to be multiplied to the input. For the single layer, we start by using a random weight in the interval [0, 1]. Other intervals such as [-1, 1] would also have worked. Then we multiply the weight to the input. This is repeated when we train the layer by updating the weight based on the difference between the modified input to output and the correct output, minimizing this difference (minimizing the loss function). The new weights are then sent back to be re-used and this keeps on optimizing the weights to give a desired output. When training is done, the weights have been improved through the backpropagation and at last we run the test data to evaluate the result.

TBMI26
vicli815
jonto406
Multi layer:

The core function of the multi layer is similar to how we handle the single layer. However, instead we have several hidden layers with weights that are being multiplied to the input based on an activation function tanh(X*W). This makes it possible to give non-linear classifiers. Before giving the output, we multiply with the weights of the output layer. For these weights (hidden layers and output layer), we used random weights in the interval [-0.01, 0.01] because larger weights would make the minimize loss function very oscillating. Just as with the single layer, we check for error compared to desired output and optimize the weights in the training function before evaluating by using test data.

Q7. 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.

Results multilayer networks

When trying to increase our accuracy we did much trial and error. We held two parameters constant and moved one. We changed the number of hidden, the learning rate and number of iterations. We also divided our initial random weights by 10 and this helped the accuracy and oscillations. In the first three datasets we did not have any problem with the accuracy and more optimal solutions do exist, but we are happy with our classifier. In the 4th dataset we had to increase the number of hidden layers and reduce the training time to avoid overfitting.

Dataset 1
Number of hidden = 20
Learning rate = 0.0075
Number of iterations = 2000
Acc = 98.8 %

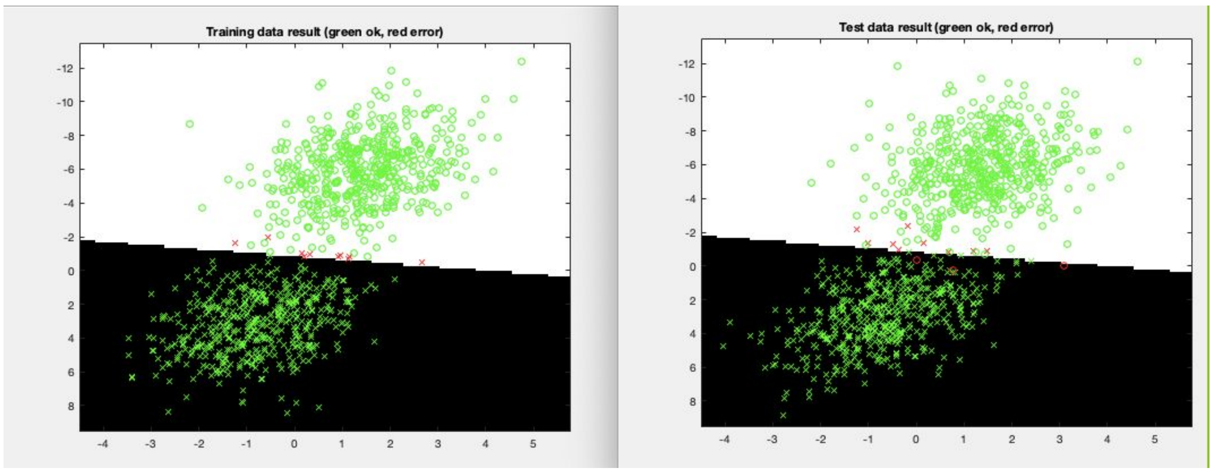


Figure 9: Classification of dataset 1 with 98.8 % accuracy using backpropagation.

Dataset 2
Number of hidden = 20
Learning rate = 0.0075
Number of iterations = 2000
Acc = 99.7 %

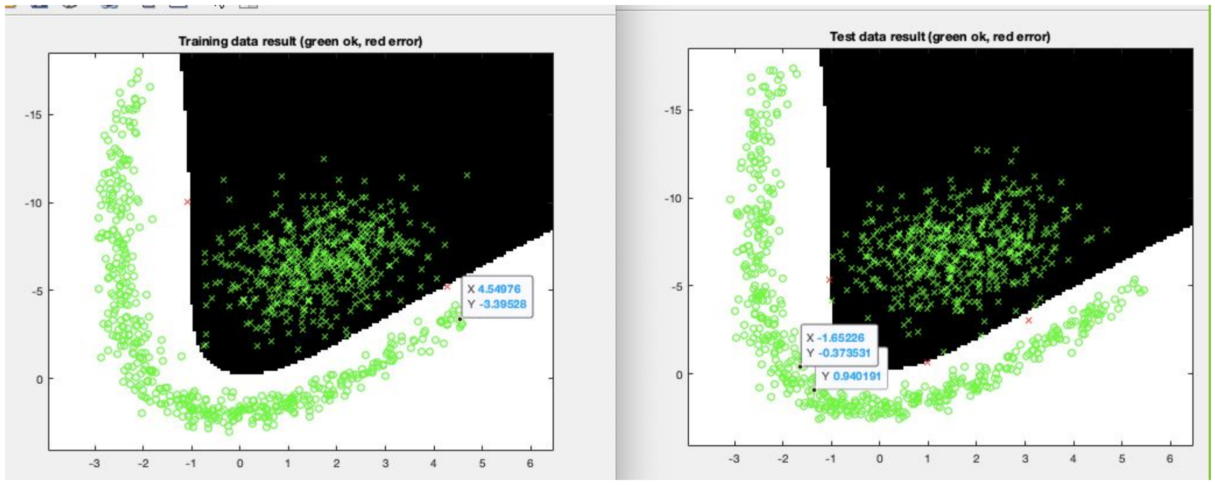


Figure 10: Classification of dataset 4 with 99.7 % accuracy using backpropagation

Dataset 3
Number of hidden = 30
Learning rate = 0.0075
Number of iterations = 6000
Acc = 99.6 %

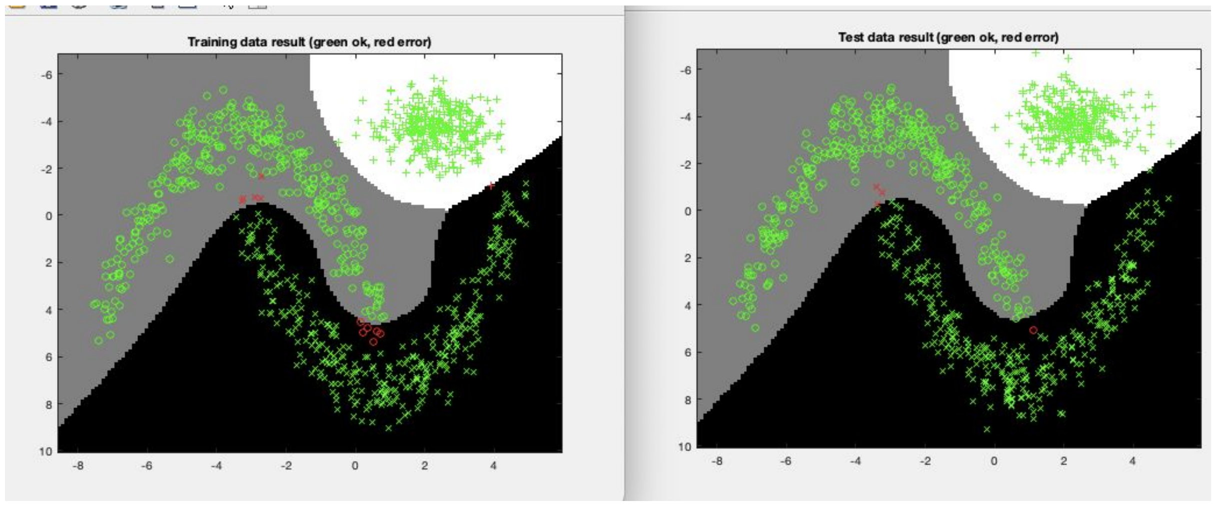


Figure 11: Classification of dataset 4 with 99.6 % accuracy using backpropagation.

Dataset 4
Number of hidden =40
Learning rate = 0.0075
Number of iterations = 3000
Acc = 96.3 %

TBMI26
vicli815
jonto406

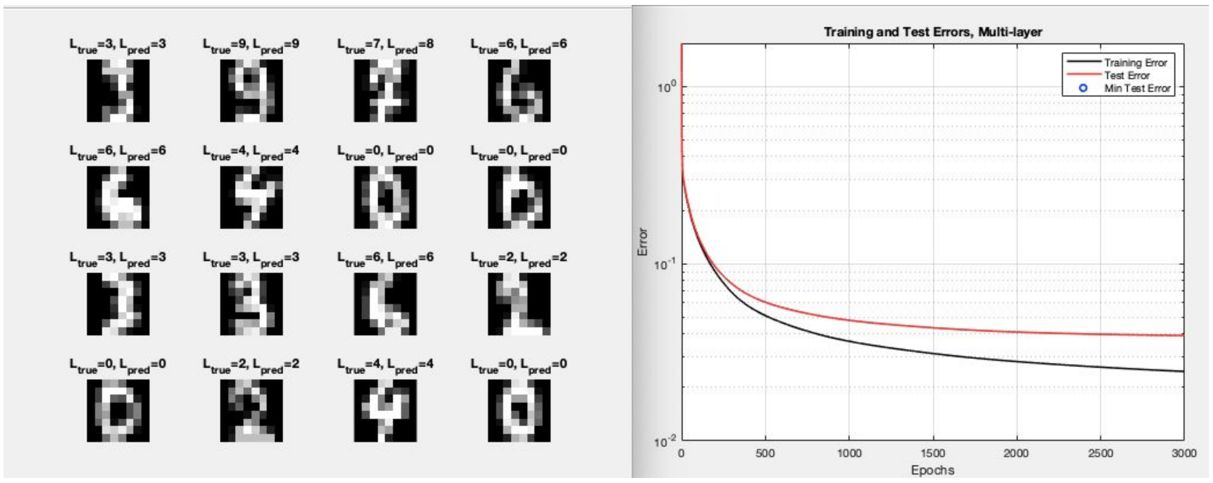
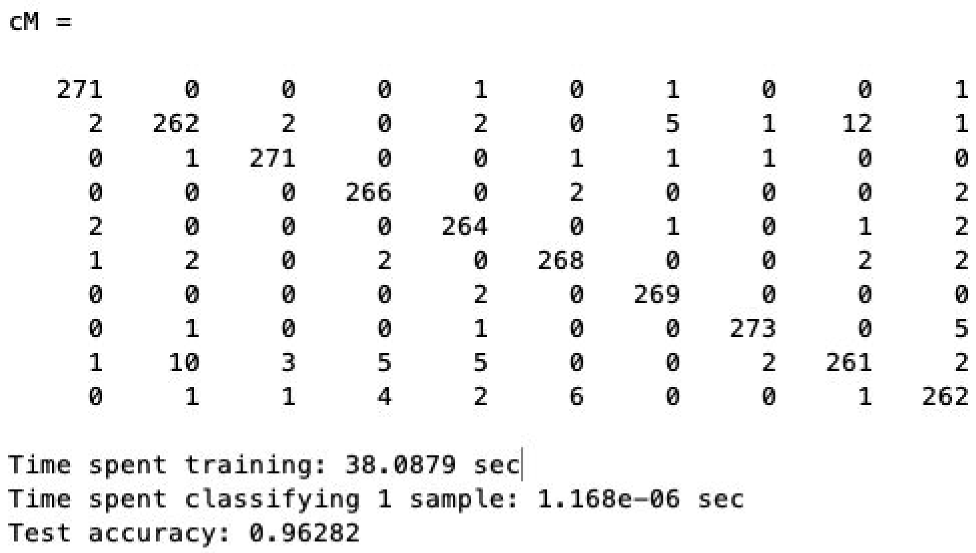


Figure 12: Classification of dataset 4 with 96.3 % accuracy using backpropagation.



```
cM =

   271     0     0     0     1     0     1     0     0     1
     2   262     2     0     2     0     5     1    12     1
     0     1   271     0     0     1     1     1     0     0
     0     0     0   266     0     2     0     0     0     2
     2     0     0     0   264     0     1     0     1     2
     1     2     0     2     0   268     0     0     2     2
     0     0     0     0     2     0   269     0     0     0
     0     1     0     0     1     0     0   273     0     5
     1    10     3     5     5     0     0     2   261     2
     0     1     1     4     2     6     0     0     1   262

Time spent training: 38.0879 sec
Time spent classifying 1 sample: 1.168e-06 sec
Test accuracy: 0.96282
```

Figure 13: confusion matrix of dataset 4 using backpropagation.

Q8. Present the results, including images, of your example of a non-generalizable backprop solution. Explain why this example is non-generalizable.

A non generalizable solution performs badly on the test data and well on training data. This could be done with limiting the amount of training data. In the figure below the single layer network (dataset 1) is used but the training data is limited to 400 (instead of 1000). One can see that the testing error increases. This is an example of a non generalizable backpropagation solution.

TBMI26
vicli815
jonto406

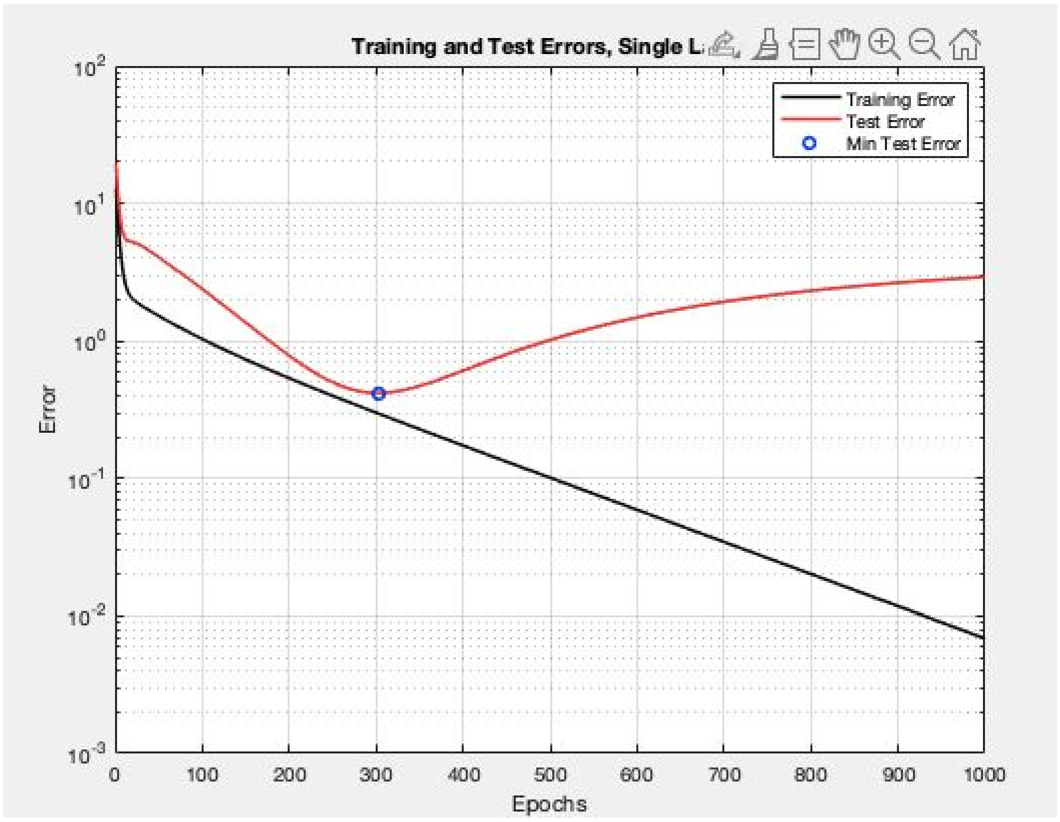


Figure 14: Training and test error of non-generalizable backpropagation solution.

Q9. Give a final discussion and conclusion where you explain the differences between the performances of the different classifiers. Pros and cons etc.

The kNN is easy to understand and to implement. The downside to this method is that it is computational heavy and hard if not impossible to do on larger datasets.

The neural network is more complex and harder to understand by intuition. The pros are that you do not have to save all data as is done in kNN. You just need to save the parameters. Furthermore, neural networks are good with higher dimensions and non-linear data. Another con is that these kinds of networks may need more training data for the gradient search.

Q10. Do you think there is something that can improve the results? Pre-processing, algorithm-wise

Our kNN network is pretty slow due to the sorting of all distances. This could have been done by and if statement instead and just store the k shortest distances. We believe this would reduce the running time of our code.

The results might also be improved by using another activation function and by adding an adaptive step length for each weight (or at least some hidden layers) from gradients. If the same sign on several succeeding gradients → increase step length. However, it would need a lot of tuning to be optimized and would be another parameter to experiment with for different datasets just as we did with the number of hidden layers and the number of iterations.

We could improve the accuracy in the multilayer network by creating a function for optimizing the parameters (number of layers, learning rate and training iterations) instead of just doing trial and error like we did.