TBMI26 – Computer Assignment Report Supervised Learning

Deadline – March 15 2019

Author/-s:
Martin Smelik (marsm914)
Stefano Toffol (steto820)

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**. 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 data from a machine learning perspective. Consider if you need linear or non-linear classifiers etc.**

    - **Dataset 1:** The first dataset show two clusters linearly separable at values of Y=-2 approximately with an almost flat line. A linear classifier will be sufficient to distinguish the groups.
    - **Dataset 2**: This time the two classes are arranged in a quadratic form, therefore not linearly separable, however the boundary should be easily obtained analytically.
    - **Dataset 3**: Of the three clusters only one is linearly separable (top-right corner). The remaining two follow a quadratic trend with different sign and position of the vertex. A non-linear classifier is needed to perform the task.
    - **Dataset 4:** A series of hand-written digits are present and labeled with the digit they are supposed to represent. Only non-linear classifier can perform the digit recognition task (each pixel is a value between 0 and 16, representing its shade of grey).

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**

Reducing the number of pixel and taking their sum brings to a higher flexibility of the model: ignoring small differences between the position of the lines in the pictures make the algorithm less sensible to any type of bias.

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

   We first created the distance matrix between training and test values, we added the response vector as a column on the matrix and then sorted the matrix according to each column (an observation on the test dataset): in this way the majority of the first k values of the sorted response vector gave the resulting estimate of y.

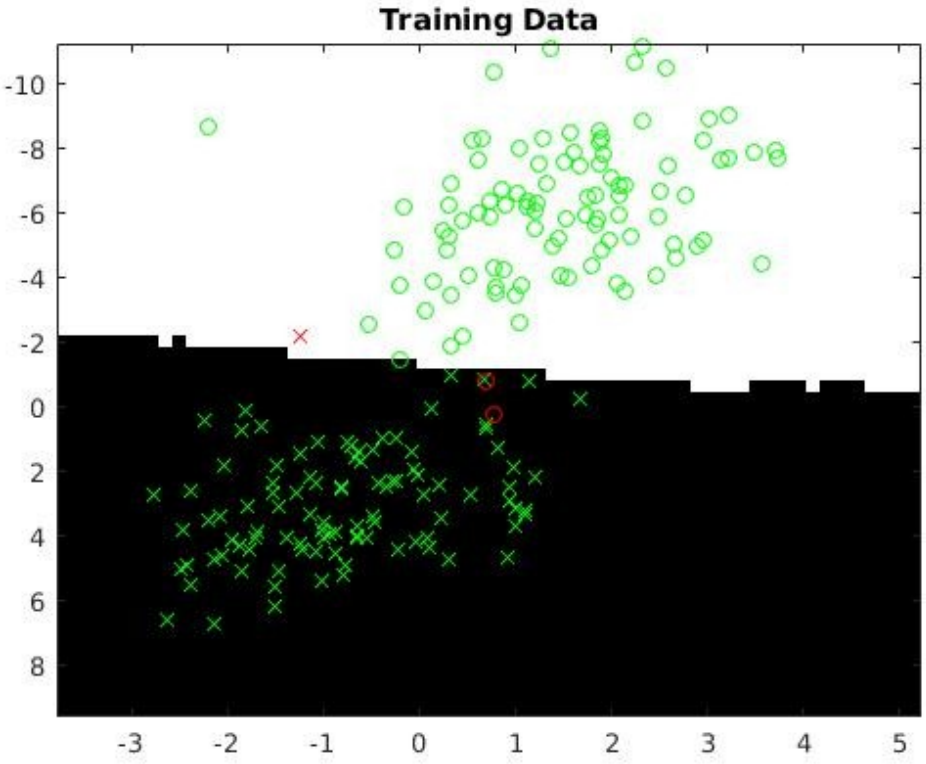
4. **Explain how you handle draws in kNN, e.g. with two classes (k = 2)?**

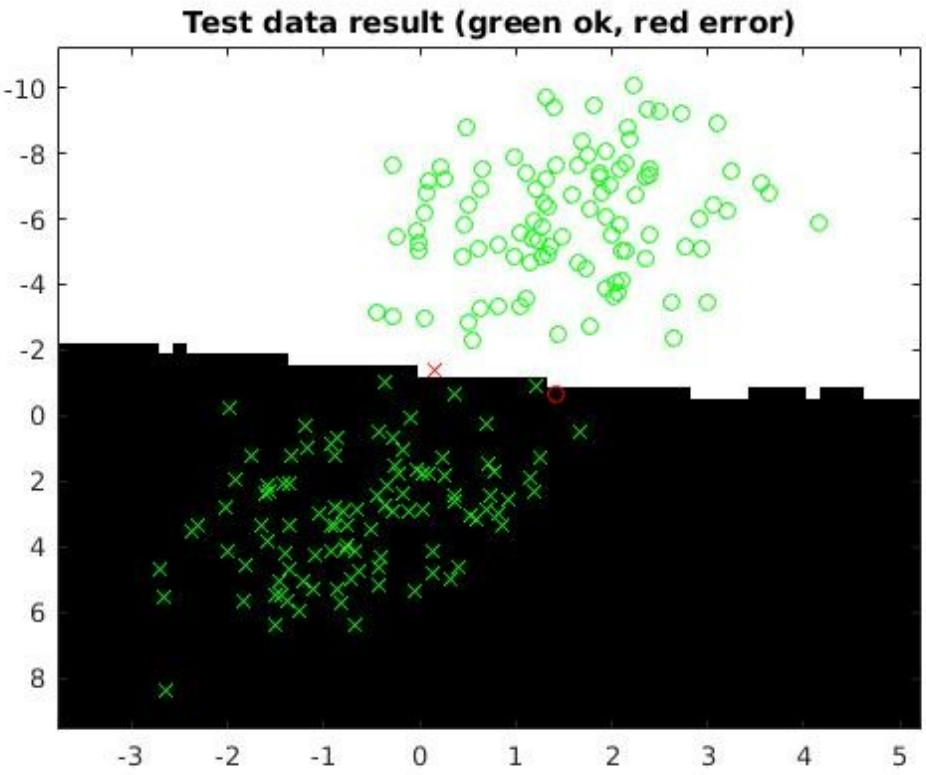
   We decided the class randomly, in order to not have any unbalance/bias in the estimations.

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.**

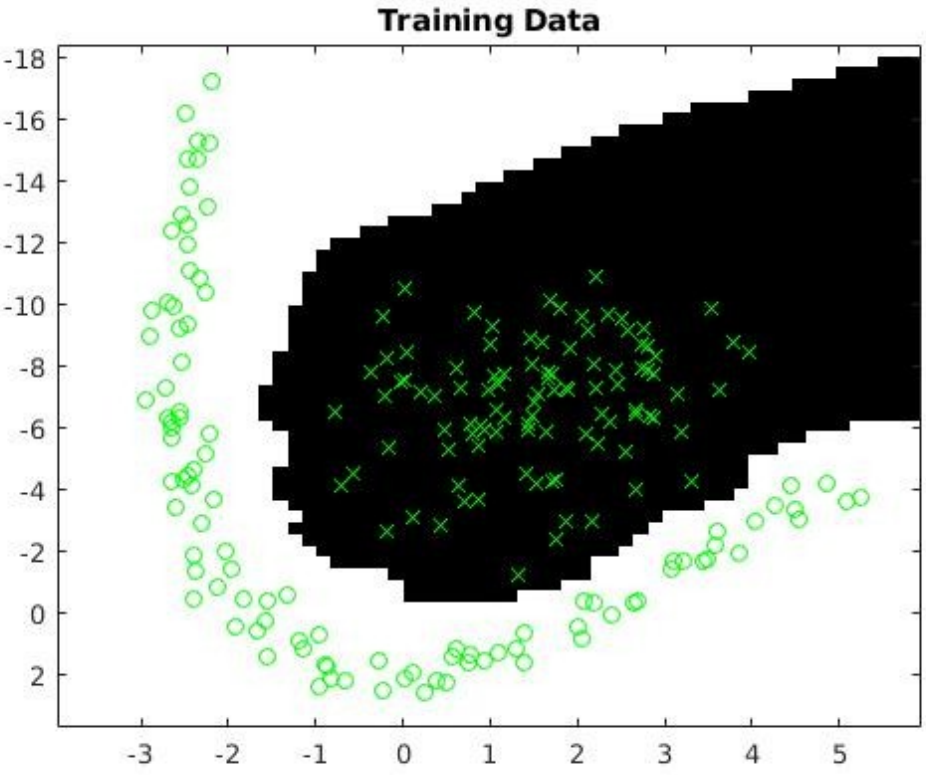
   For each dataset we run a 4-folds CV (for each iteration, ¼ was used as test) and tested on k = 1:40. The optimal K was the one giving the lowest mean error in the test dataset.

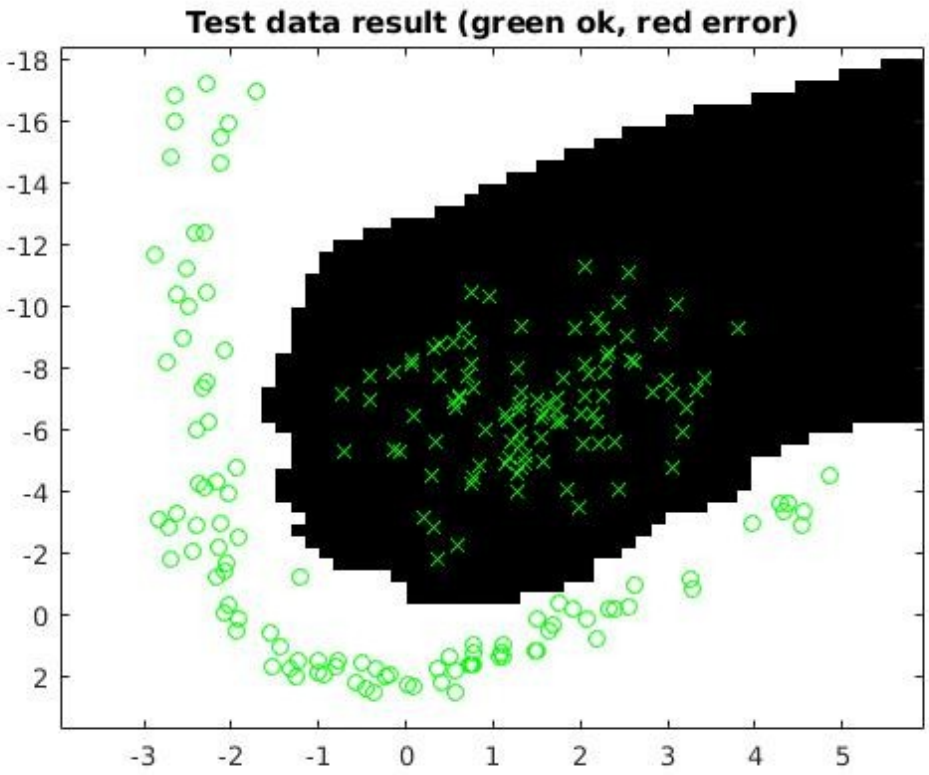
   - **Dataset 1 ;** Accuracy: 0.99 ; Optimal K = 20 ; Seed = 1530.



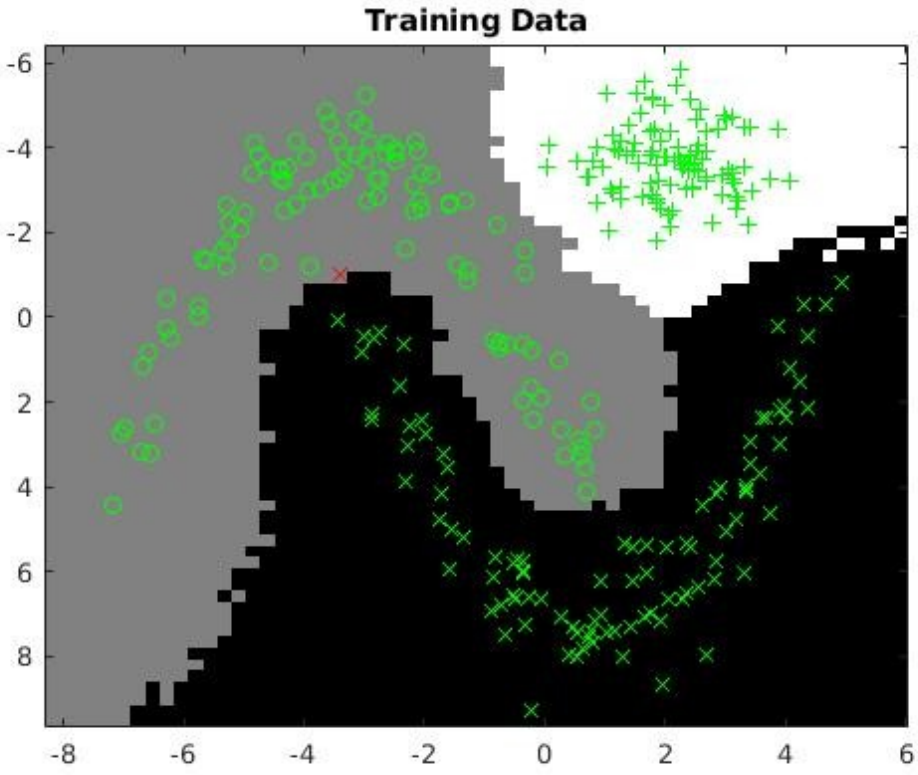


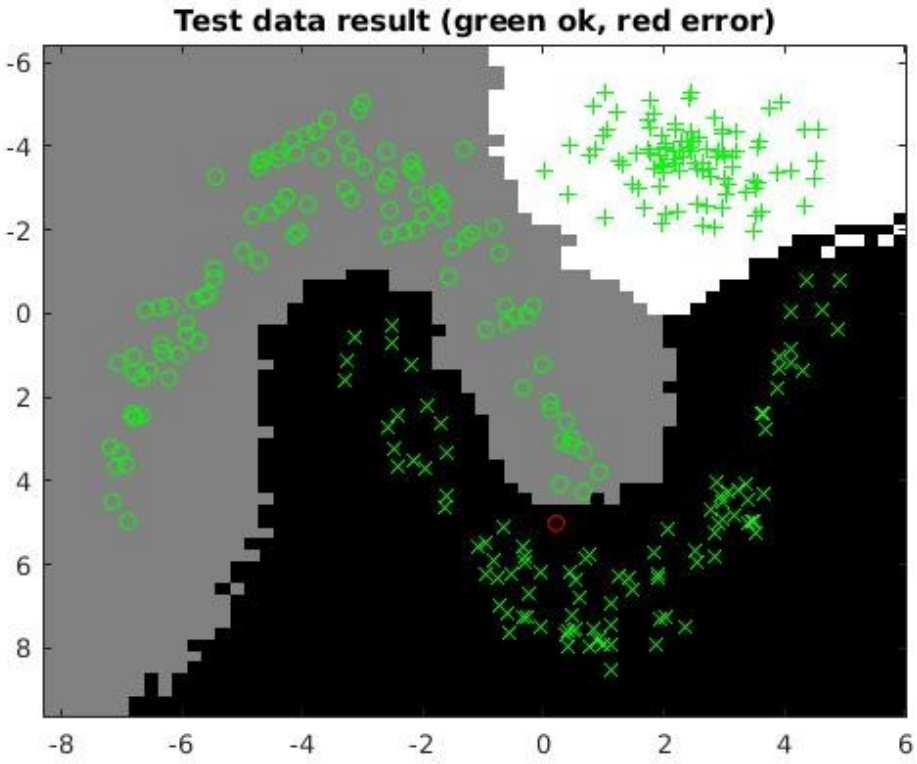
- **Dataset 2 ;** Accuracy: 1.00 ; Optimal K = 1 ; Seed = 1530.



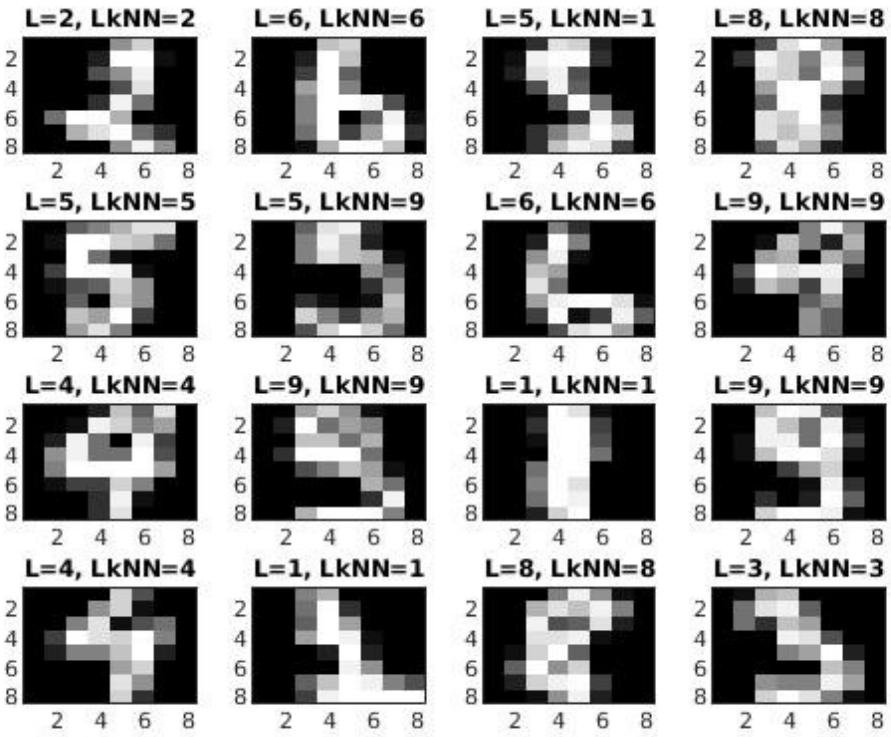


- **Dataset 3 ;** Accuracy: 0.9967 ; Optimal K = 5 ; Seed = 1530.





- **Dataset 4 ;** Accuracy: 0.98 ; Optimal K = 1 ; Seed = 1530.



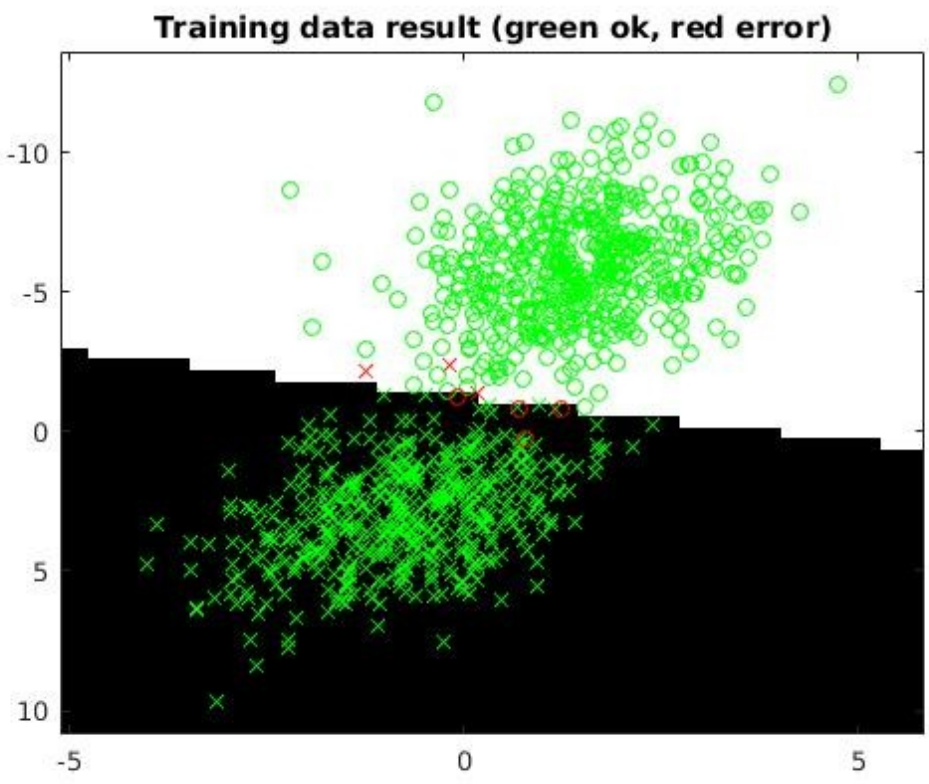
6. **Give a short summery of your backprop network implementations (single + multi). You do not need to derive the update rules.**

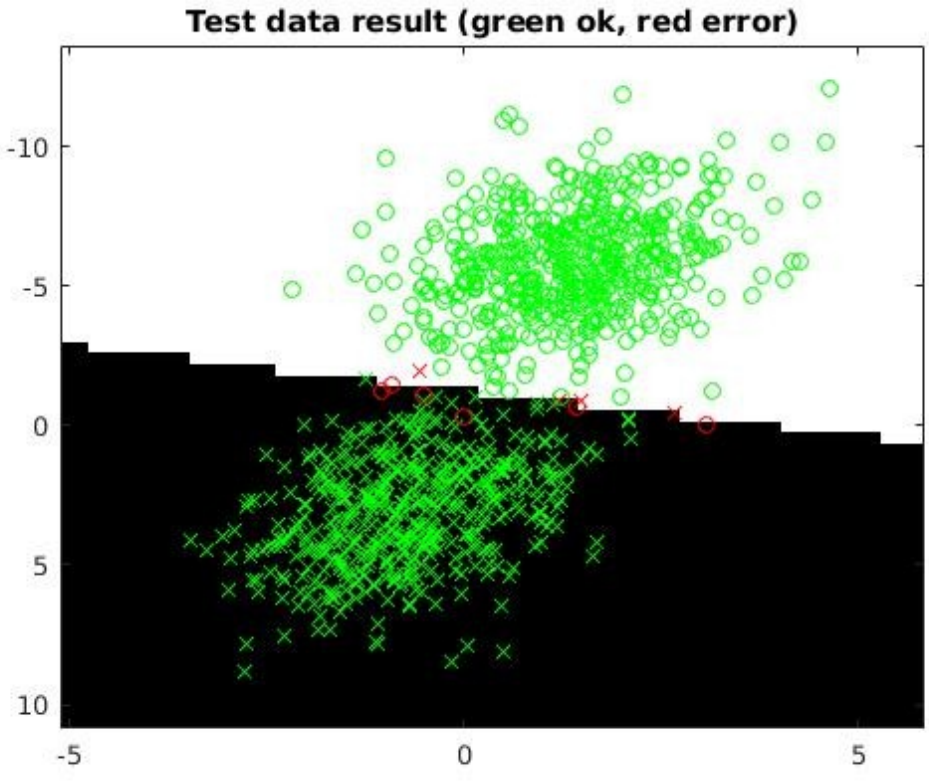
   The main goal of the backprop was to evaluate the gradients of the weights correctly. For this we used the chain rule to derive the formulas. We updated the weights in the direction of the gradients to achieve the minimizing the error function, testing each dataset with different values of the learning rate (which model the learning step of the algorithm), number of iterations and number of neurons in the hidden layer (for multilayer only). The weights were initialized randomly between -0.1 and 0.1.

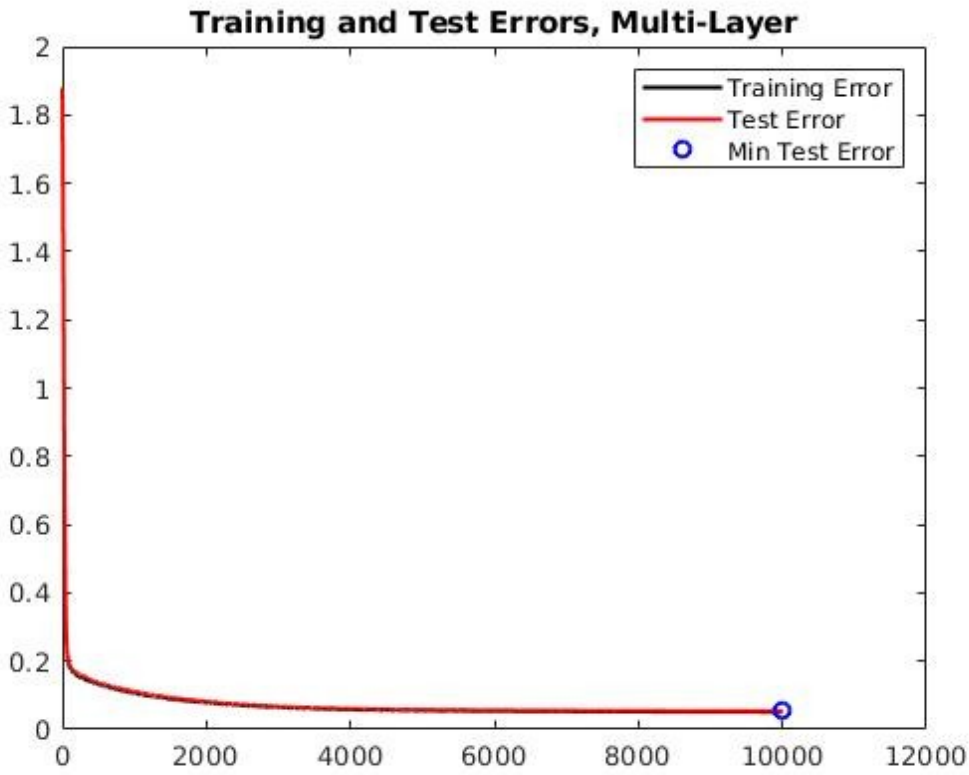
7. **Present the results from the backprop training and how you reached the accuracy criteria for each dataset. Motivate your choice of network for each dataset. Explain how you selected good values for the learning rate, iterations and number of hidden neurons. Include images of your best result for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

   In order to select the best values for the number of hidden neurons, learning rate and iterations we tried combinations of values for each dataset. After a few tests, we realized that the number of iterations were not affecting substantially our estimates, and therefore decided to fix them at 10000 for the first three dataset and 50000 for the OCR one.
   We checked instead a grid of different values, given by all the possible combinations of some values for the learning rate and the number of hidden layers. For dataset 1, 2, and 3 those values were [4, 7, 10, 13] and [0.1, 0.05, 0.01, 0.005, 0.001] for the number of hidden neurons and learning rate respectively.
   For dataset 4, since the data was more complex, different values were necessary: [17,21,25,30] were chosen as values for the number of hidden neurons, while [0.01, 0.005, 0.001, 0.0005, 0.0001] for the learning rate.
   In the case of dataset 1, 2 and 3 the selected combination of the hyperparameters were not the only ones leading to the best performances, but we opted for the least complex model among them.
   The weights were all initialized according to the same seed, 1530. The weights have been drawn uniformly between -0.1 and +0.1.

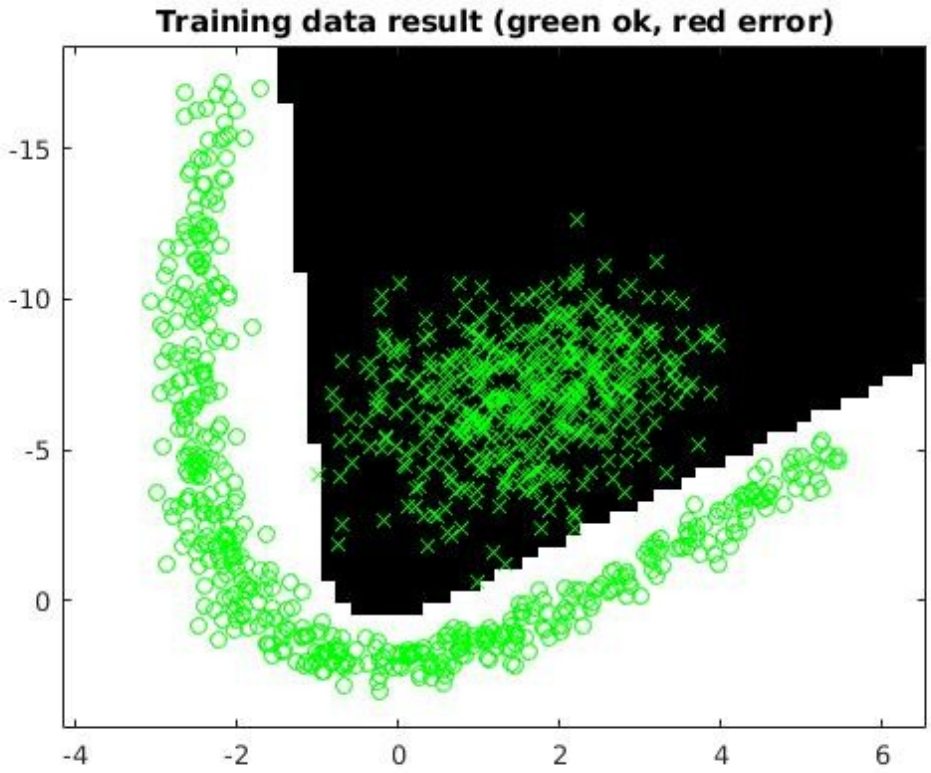
   - **Dataset 1 ;** Accuracy: 0.99 ; #Hidden: 10 ; #Iterations: 10000 ; Learning rate: 0.005.

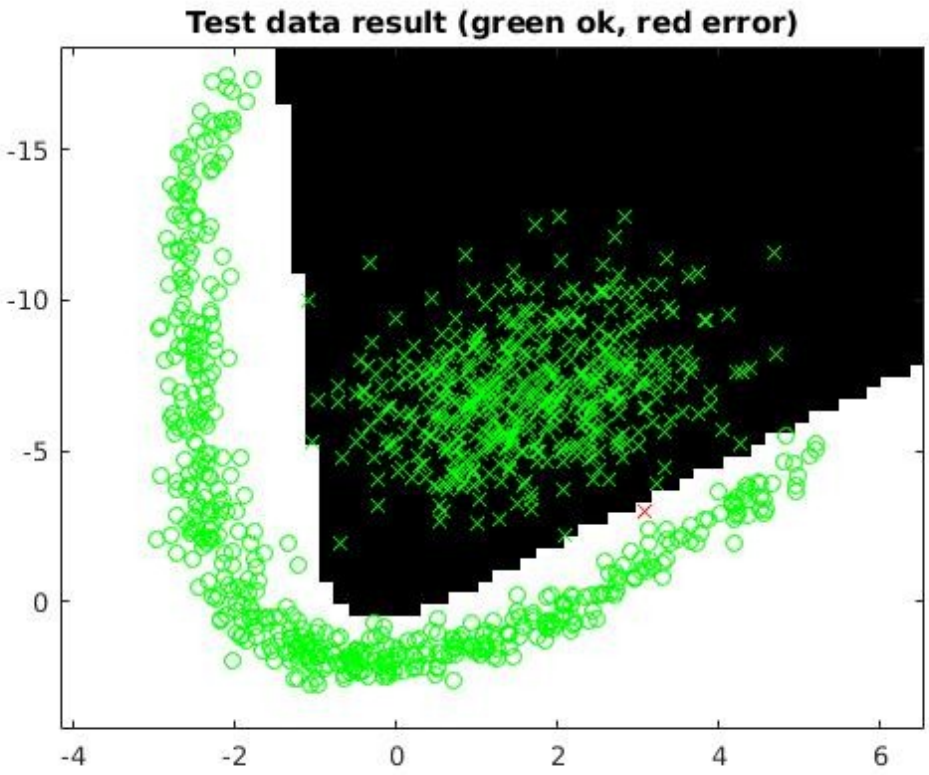

Training data result (green ok, red error)
-10
-5
0
5
10
-5
0
5

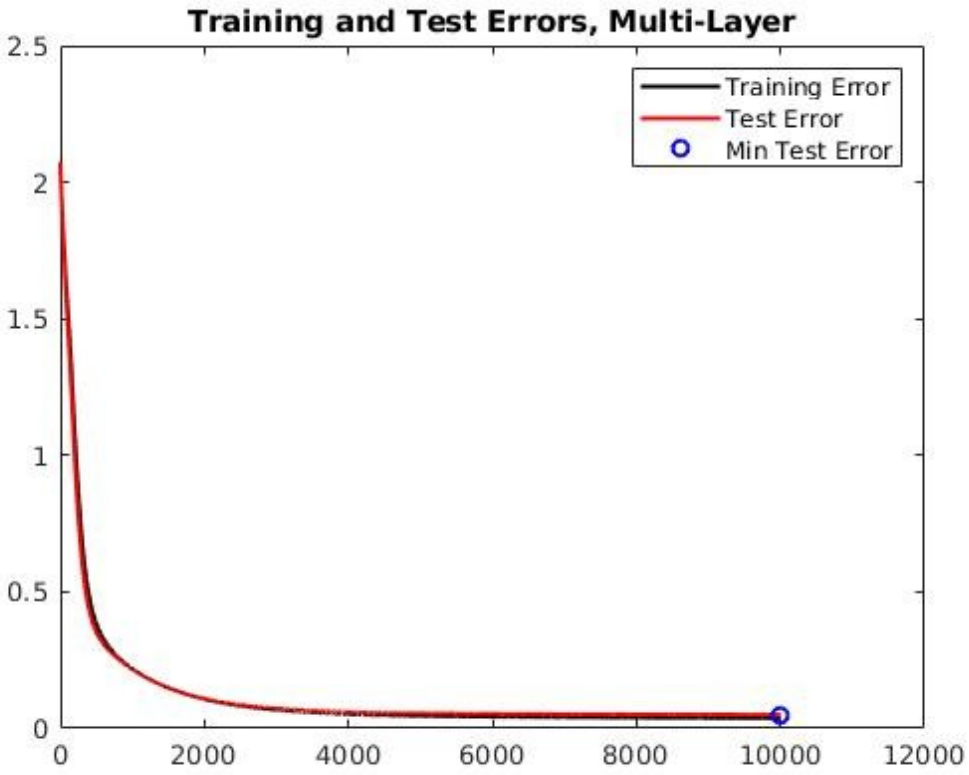

Test data result (green ok, red error)
-10
-5
0
5
10
-5
0
5



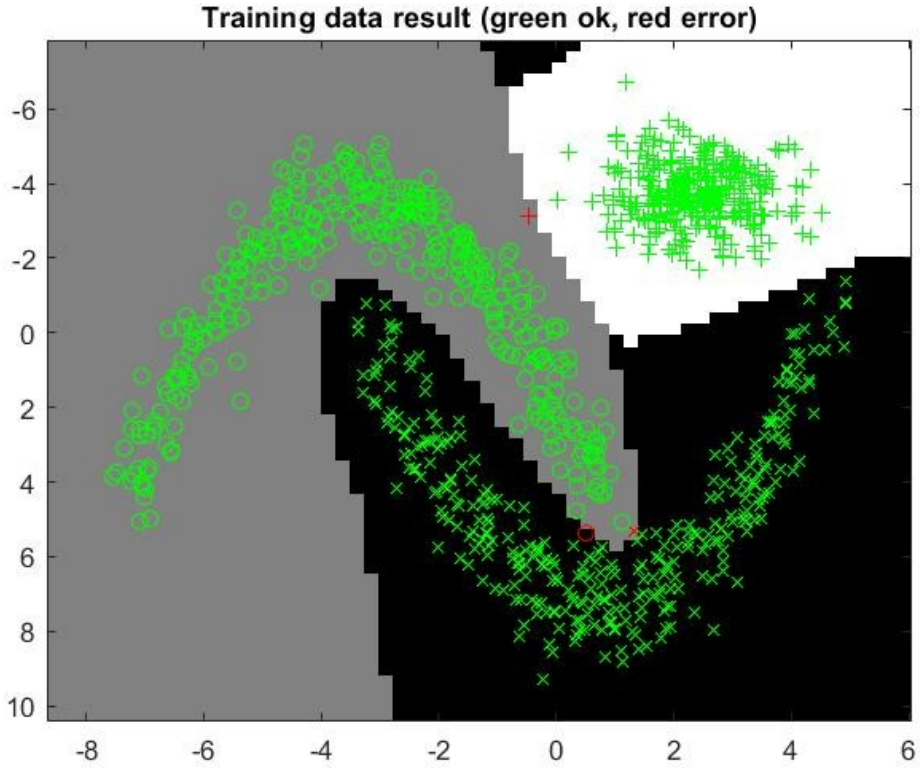
- **Dataset 2 ;** Accuracy: 0.999 ; #Hidden: 7 ; #Iterations: 10000 ; Learning rate: 0.005.

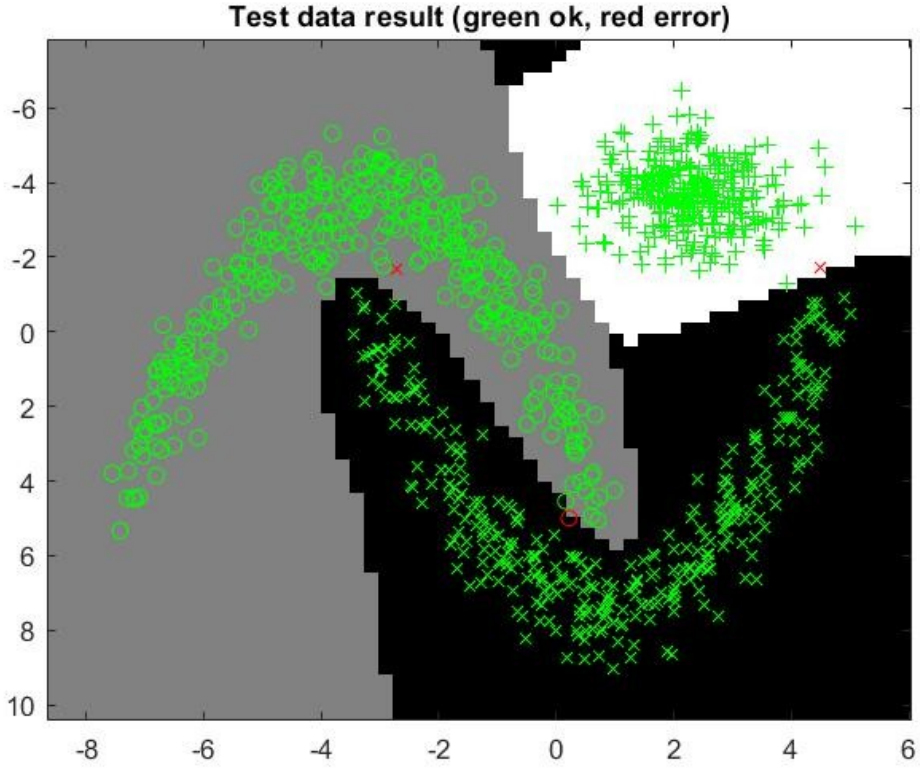


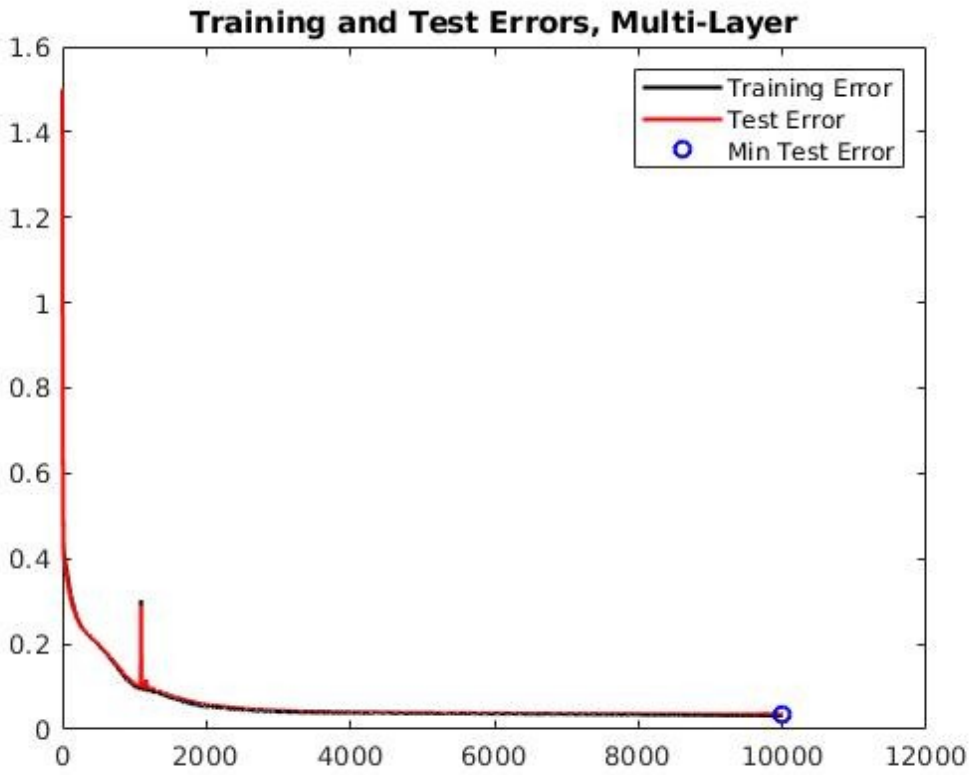




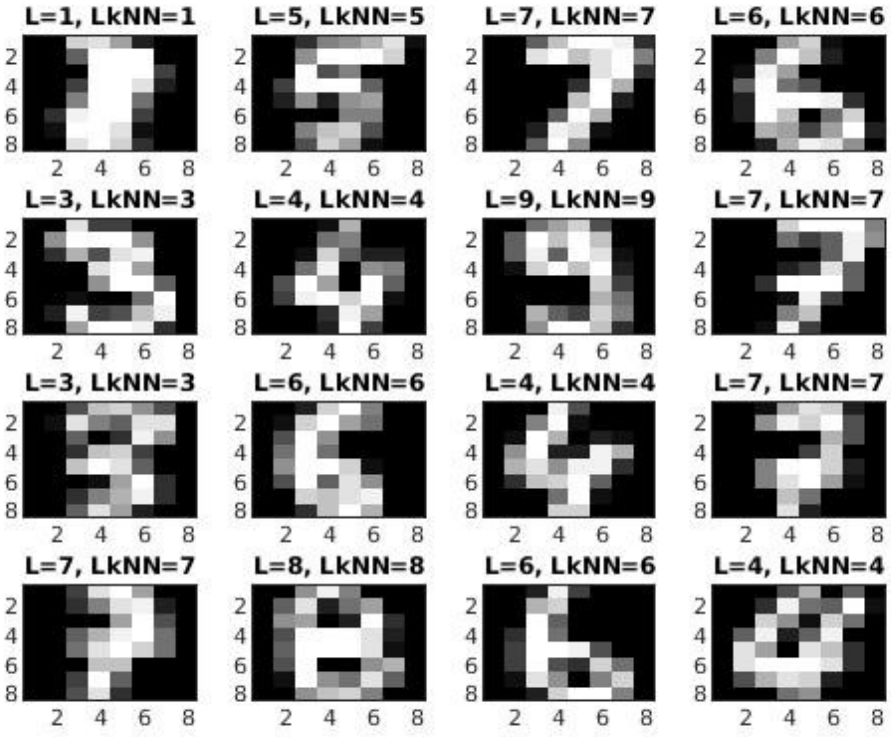
- **Dataset 3 ;** Accuracy: 0.997 ; #Hidden: 10 ; #Iterations: 10000 ; Learning rate: 0.05.

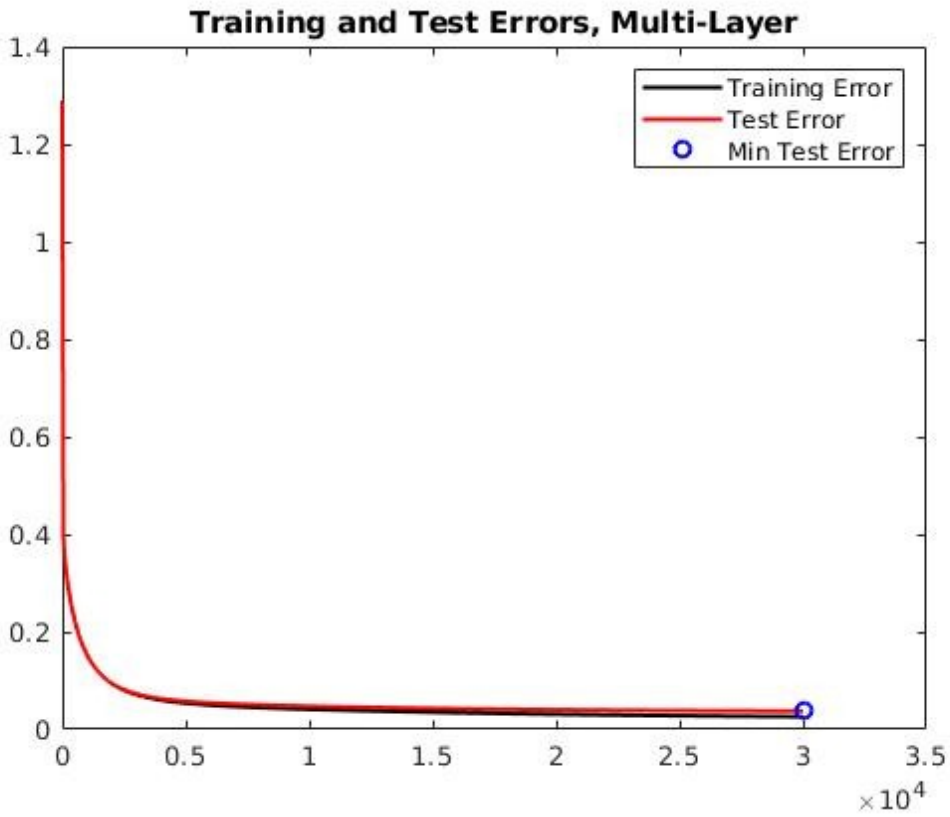

Training data result (green ok, red error)
-6
-4
-2
0
2
4
6
8
10
-8
-6
-4
-2
0
2
4
6


Test data result (green ok, red error)
-6
-4
-2
0
2
4
6
8
10
-8
-6
-4
-2
0
2
4
6



- **Dataset 4 ;** Accuracy: 0.9704 ; #Hidden: 30 ; #Iterations: 30000 ; Learning rate: 0.001.

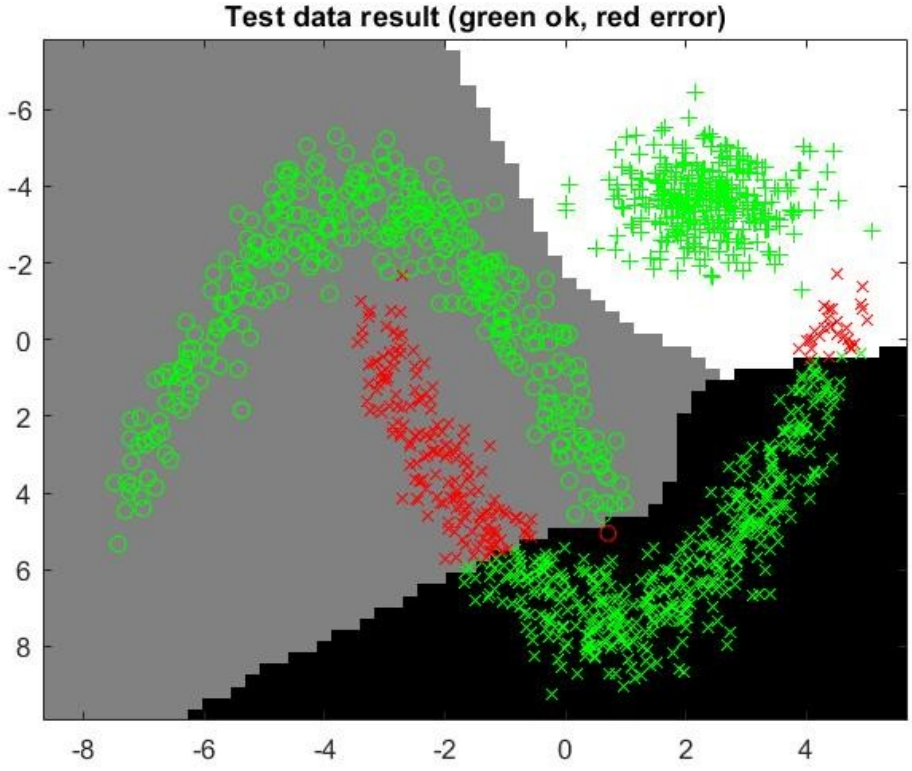


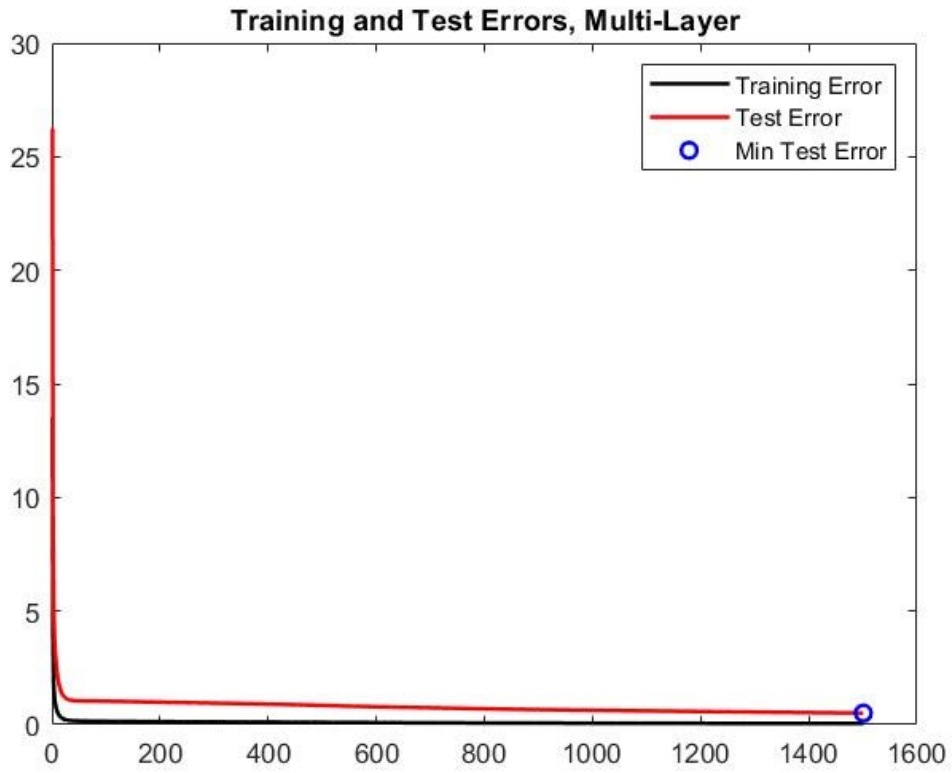


8. **Present the results, including images, of your example of a non-generalizable backprop solution. Explain why this example is non-generalizable.**

   Our approach was that we decided not to shuffle the data before dividing them into train and test data and we chose 333 data points from groups “+” and “o“ and we chose just 34 data points from group “x” for the train data. With this training data we obtained quite good training error (98,9%) however the obtained test error was much worse (84,5%). For the training we used 1500 iterations, 10 hidden neurons and the learning rate 0.05. The test error is so bad because the training data was unequally distributed and the network was not able to cover the left part of “x” data group. In extreme case with using less training data from “x” group or even not using this data for training, we could actually get just 2 classes in the end.


Training data result (green ok, red error)
-6
-4
-2
0
2
4
6
8
-8
-6
-4
-2
0
2
4


Test data result (green ok, red error)
-6
-4
-2
0
2
4
6
8
-8
-6
-4
-2
0
2
4



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

   Starting with KNN algorithm, we must say that it is has a great balance between the simplicity and the accuracy. This algorithm is easy to understand and to explain and we do not need a big computational power. Single layer NN is quite useful if we have linearly separable border. As it is just a single layer, it is not so computationally difficult, however for borders different to linearly separable it is unusable. We are able to obtain the best accuracy for all data sets with multilayer NN. However multilayer is difficult to explain to people with not relevant background and it is computationally difficult to train and even more difficult to find correct parameters.

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

    First of all we ourselves realized the importance of the initialization of the weights: for instance even with what came out to be the best combination of hidden layers, number of iterations and learning rate, setting weights all equal to 0 would bring to the same value for all the predictions, resulting in an accuracy of 1/(# classes). Other pre-determined values, such as 1, will bring to some local minimum different with poor performances. In the end we decided to initialize them randomly between -0.1 and 0.1.
    Secondly a standardization of the dataset may improve the performances of the model: in dataset 1 for instance it raises accuracy of 0.2%. Nonetheless this

transformation not always grants a better result and should be tested with every dataset. For instance, in the OCR dataset it makes the image itself more confusing, even difficult to recognize for a human. In general, it works well when the features were measured on very different scales.
Lastly different values of the learning rate, number of hidden neurons and iterations will, as we have just seen, affect the predictions of the model. However we have tested for computational and time reasons just a very tiny amount of values for these hyperparameters. Other values may lead to more accurate results. We have anyways to consider that extremely big values in the number of hidden neurons and of iterations may overtrain the model and adapt it too well both to train and test dataset.