TBMI26 – Computer Assignment Report Supervised Learning

Deadline – March 15 2019

Authors: Anubhav Dikshit <anudi287>
Hector Plata <hecpl268>

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.**
    - Among the four datasets, only the first one is linearly separable, thus even a simple algorithm such as logistic regression, neural network with one single layer is enough to separate the data classes.

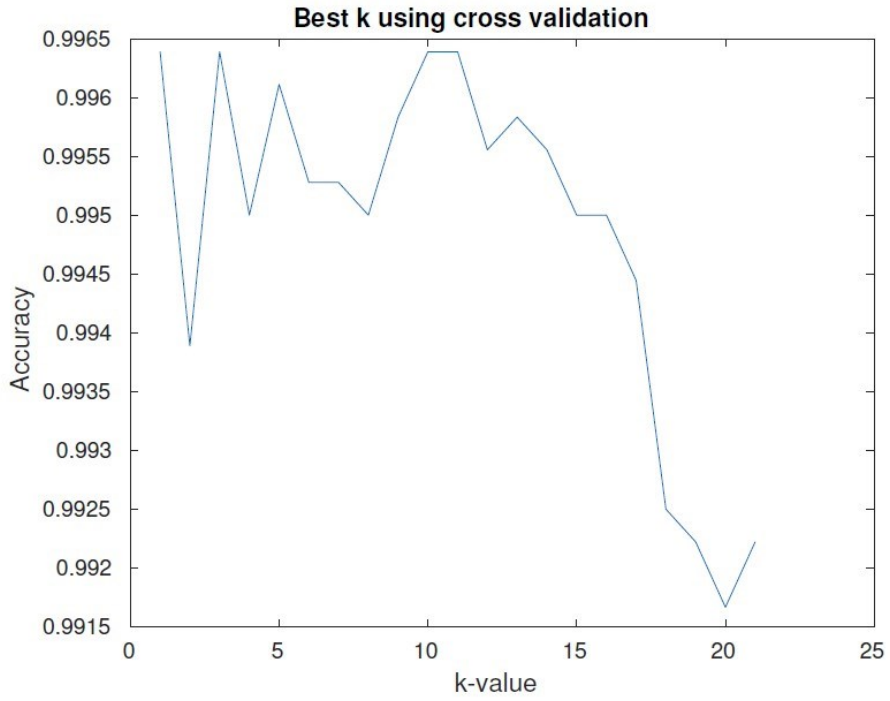
2. **Explain why the down sampling of the OCR data (done as pre-processing) result in a more robust feature representation. See http://archive.ics.uci.edu/ml/datasets/Optical+Recognition+of+Handwritten+Digits**
    - We know that the more the number of feature the higher is the complexity of the algorithm need to resolve this, one easy way of reducing this is to use dimensionality reduction which here is replaced by down sampling.
    - Performing down-sampling has another benefit of reducing noise in the data such that our algorithm doesn't start focusing on handwriting style but focus only on digit recognition (over-trained)

3. **Give a short summery of how you implemented the kNN algorithm.**
    - We first create a distance map for all data points to be classified
    - We sorted them by distance and of the closest neighbors (defined by k) we extracted the index.
    - Using mode function on the neighbors' class, we chose the class to be classified. This is same as majority voting.

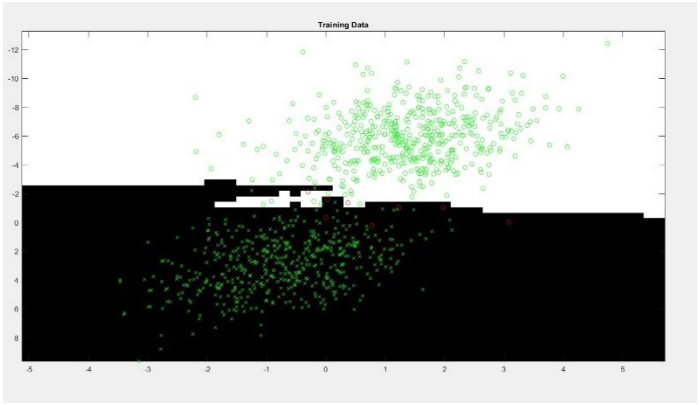
4. **Explain how you handle draws in kNN, e.g. with two classes (k = 2)?**
    - The ideal scenario is to always have even number of k however even this might result in having too many ties.

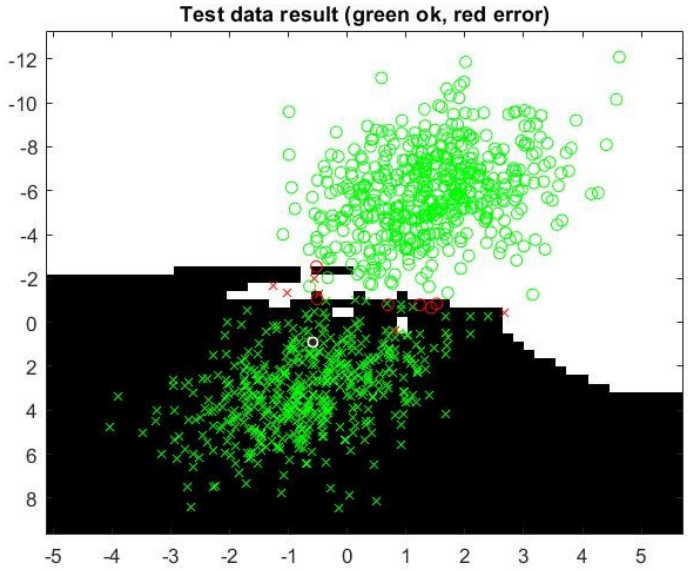
- To resolve this some of the suggestions include adding a small gaussian noise to the data, the tie points and then measure the neighbors again, the practical version of this can be seen in tools like 'Weka' used in data mining course.
- Other methods include measuring distance also in the event of a tie (similar to K-Means) but measuring distance with neighbors.

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 bin size as 2, we run iterations with k varying from 1 to 10, the standard train and test cross validation method was used.

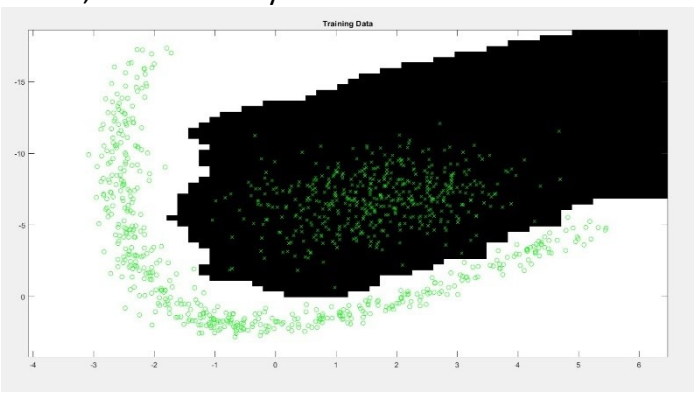


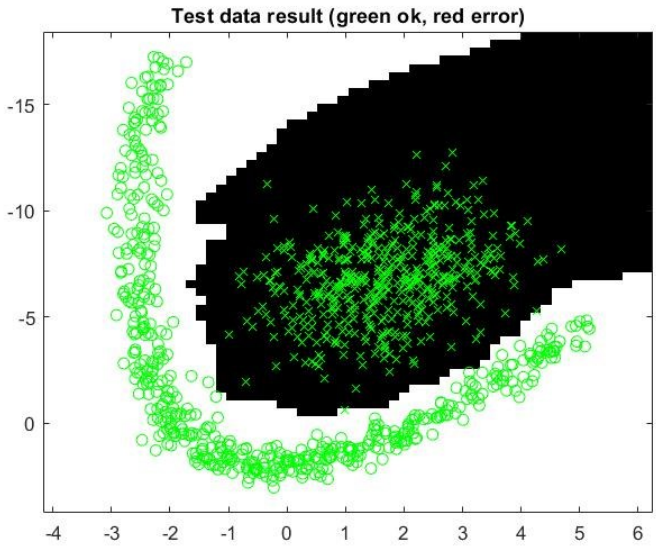
   - The k which yields the lowest validation error/highest accuracy is selected.
   - Dataset1:
     - K=1, test accuracy = 98.80%
     - K=2, test accuracy = 98.40%
     - K=3, test accuracy= 98.70%
     - K=1, train accuracy = 99.00%
     - K=2, train accuracy = 99.20%
     - K=3, train accuracy= 98.70%

- 

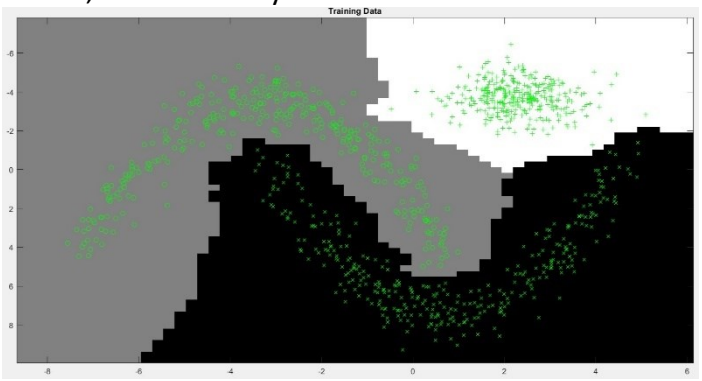


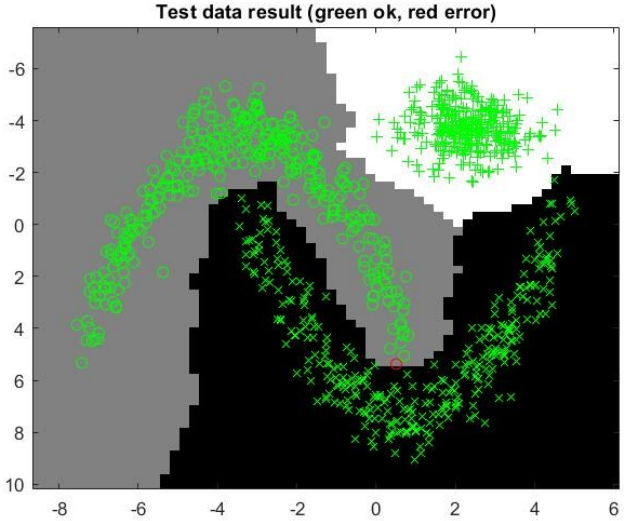
- Dataset2:
  - K=1, test accuracy = 100%
  - K=2, test accuracy = 100%
  - K=3, test accuracy = 99.80%
  - K=1, train accuracy = 99.99%
  - K=2, train accuracy = 100.00%
  - K=3, train accuracy= 99.70%



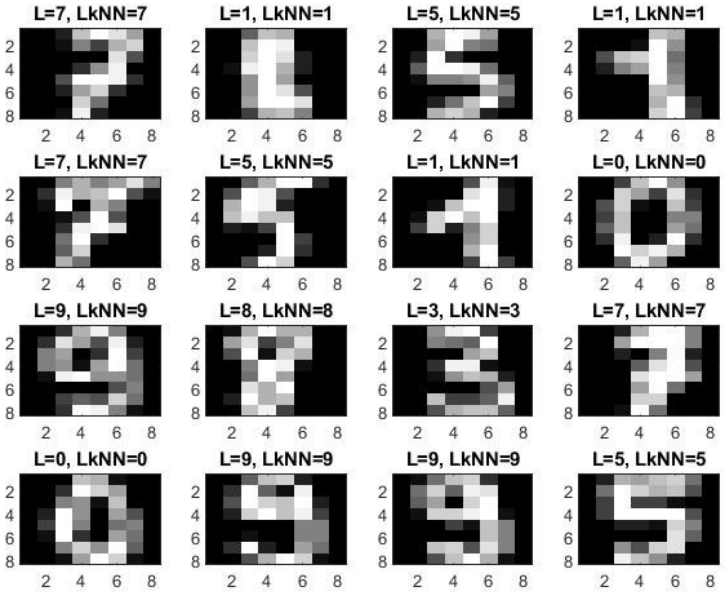


- Dataset3:
  - K=1, accuracy = 99.90%
  - K=2, accuracy = 99.60%
  - K=3, accuracy = 99.80%
  - K=1, train accuracy = 99.99%
  - K=2, train accuracy = 99.60%
  - K=3, train accuracy= 99.70%





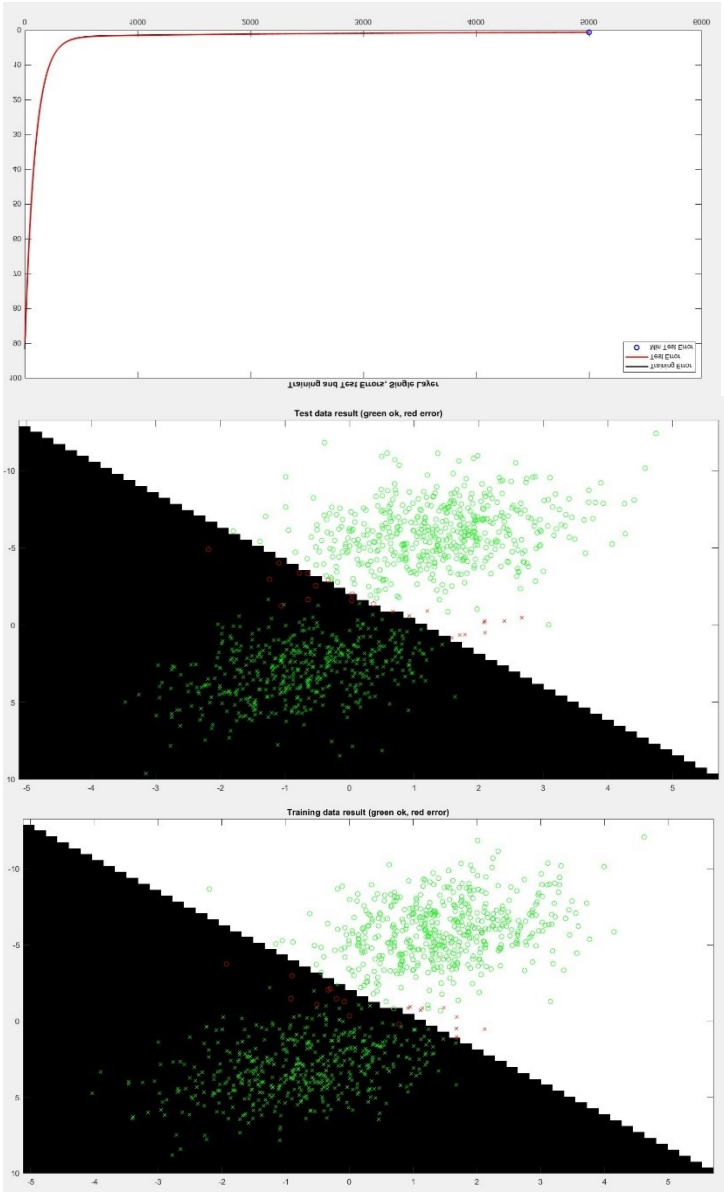
- Dataset4:
  - K=1, accuracy = 98.52%
  - K=2, accuracy = 97.94%
  - K=3, accuracy = 98.70%
  - K=4, accuracy = 98.41%
  - K=1, train accuracy = 98.38%
  - K=2, train accuracy = 97.83%
  - K=3, train accuracy= 98.38%
  - K=4, train accuracy= 98.05%



6. **Give a short summery of your backprop network implementations (single + multi). You do not need to derive the update rules.**
   - For both cases a forward pass was made, saving the neuron activations. Updating the output layer and the hidden layer was inspired by the following source:
   - https://theclevermachine.wordpress.com/2014/09/06/derivation-error-backpropagation-gradient-descent-for-neural-networks/

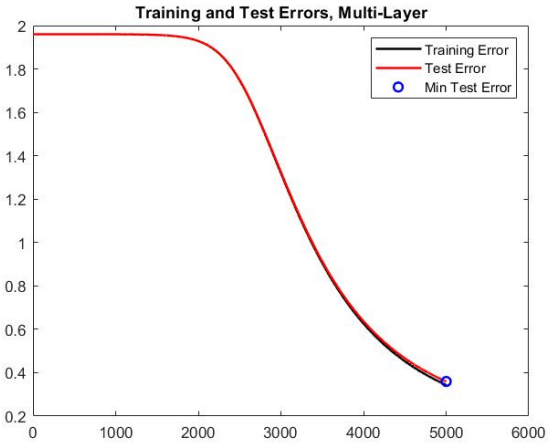
7. **Present the results from the backprop training and how you reached the accuracy criteria for each dataset. Motivate your choice of network for each dataset. Explain how you selected good values for the learning rate, iterations and number of hidden neurons. Include images of your best result for each dataset, including parameters etc.**
   - Dataset 2, 3 and 4 are non-linearly separable, thus we need more than one hidden layer, hence only the multi-layer network results are shown are those other datasets.
   - The multi-layer neural network using tanh as activation function.
   - The learning rate was changed using trial and error method and number of iterations was 5000 and 10000 for complex cases.

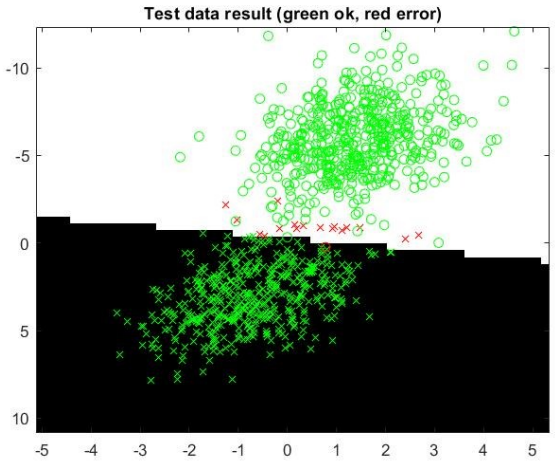
- Care was taken to ensure that the number of neurons remain less since we know that more neurons would always yield to fit any complex function.
- Dataset1:
  - Learning Rate: 0.00001
  - Iterations: 5000
  - Accuracy: 97.70%



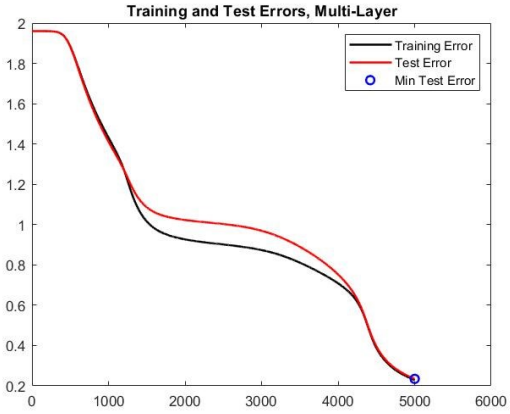
Multi-layer:

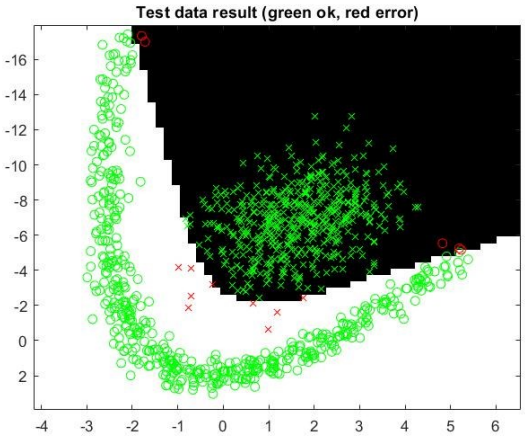
- Dataset1:
  - Learning Rate: 0.0001
  - Hidden Neurons: 2
  - Iterations: 5000
  - Accuracy: 98.20%



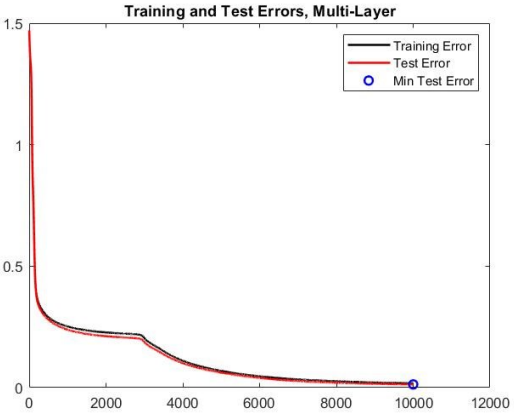


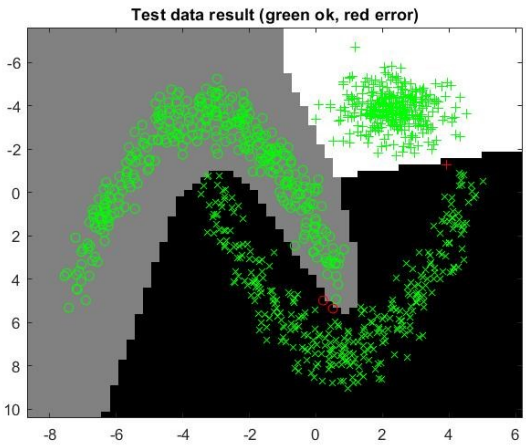
- Dataset2:
  - Learning Rate: 0.0009
  - Hidden Neurons: 2
  - Iterations: 5000
  - Accuracy: 99.20%



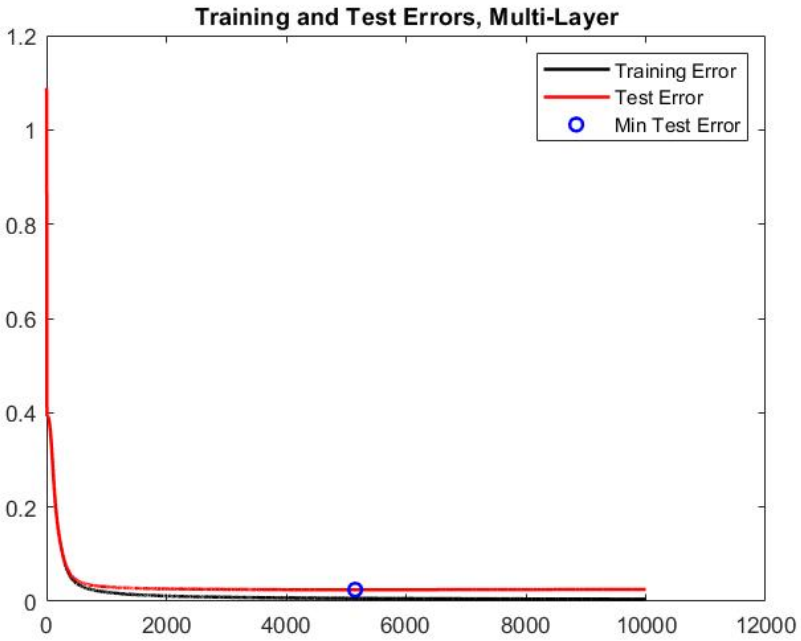


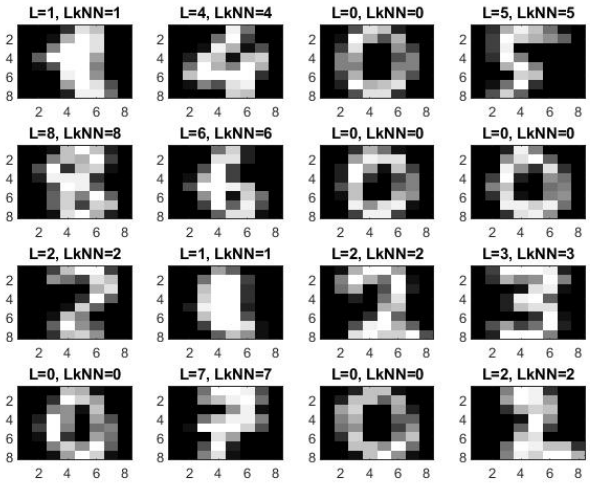
- Dataset3:
  - Learning Rate: 0.006
  - Hidden Neurons: 7
  - Iterations: 10000
  - Accuracy: 99.70%



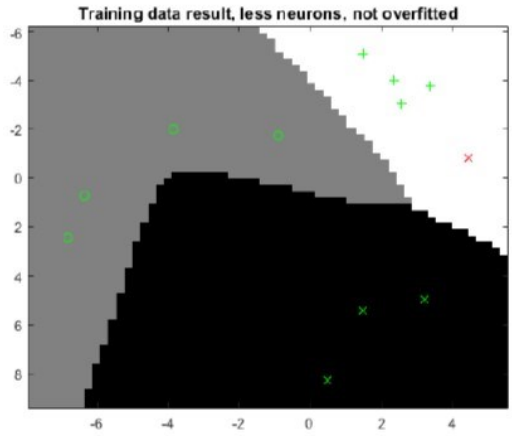


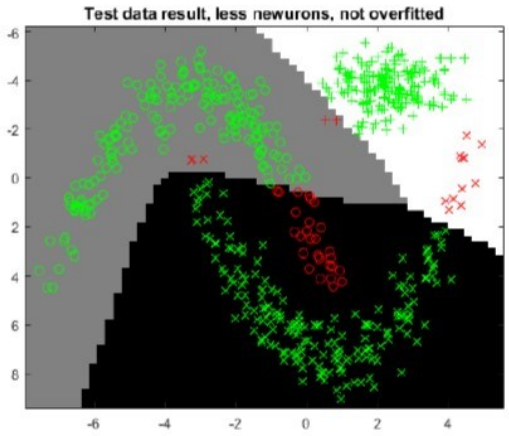
- Dataset4:
  - Learning Rate: 0.005
  - Hidden Neurons: 50
  - Iterations: 10000
  - Accuracy: 96.66%

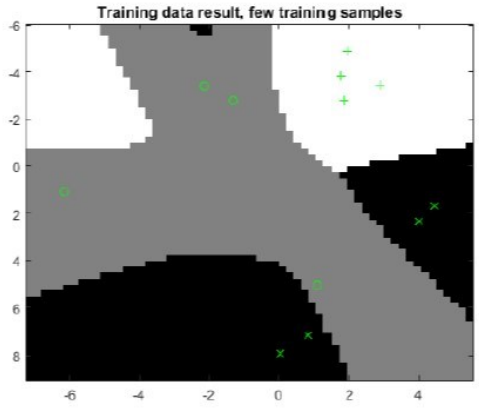


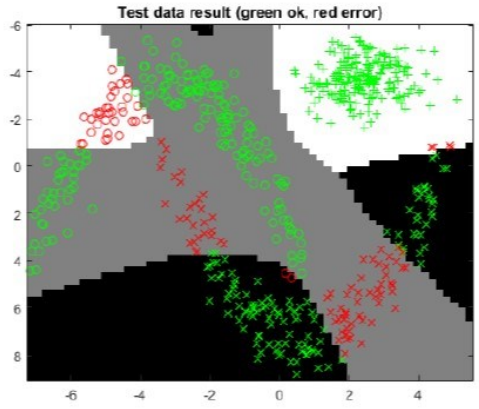


8. Present the results, including images, of your example of a non-generalizable backprop solution. Explain why this example is non-generalizable.









- A non-generalizable solution can be obtained by using very few training samples, as in the example above. Another type of non-generalizable solution is a solution which is over-fitted or has more classification boarders than required.
- In the second case a lot more neurons were used which in combination with a small amount of datapoints result in an over flexible network.

9. **Give a final discussion and conclusion where you explain the differences between the performances of the different classifiers. Pros and cons etc.**
    - KNN: KNN is easy to implement and requires hardly any training time, making it a good choice for a baseline to compare other models. It does however require access to all the training data for every new classification when it compares the new data to *all* old data, making classifications computationally expensive. KNN also requires one to select the k value which can be however easily gotten using cross validation.

    - Neural Network: Training an neural network is computationally expensive since one needs to rely on iterative methods and finding a global minimum does not realistically happen. Never the less a good enough solution can usually be found. Compared to KNN the data does not need to be stored after the network is trained and classification is done using matrix multiplications instead of exhaustive search, however the computational time can grow exponentially.

10. **Do you think there is something that can improve the results? Pre-processing, algorithm-wise etc.**
    - Regarding datasets 1, 2 and 3 there is not much that can be done to improve performance. The data looks like it is generated from the same distribution making it easy to achieve high performance and hard to over fit. To achieve better performance on dataset 4, a more complex model such as deeper network could be used, which would of course highly increase the training time. When pre-processing one thing to consider is that the dimensionality is already quite low, so simply reducing the resolution might lead to too big a loss of information. Perhaps some latent features can be found using PCA that would allow for additional dimensionality reduction.