TBMI26 – Computer Assignment Report Supervised Learning

Deadline – March 14 2021

Authors: Alexander Bois & Daniel Bissessar

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.**
   - For Dataset 1 a linear classifier seems appropriate since the data seems to be linearly separable in the current dimensions. This data contains 2 features and two classes.
   - For Dataset 2 a non-linear classifier seems appropriate since there isn't a clear way to separate the data linearly. This data contains 2 features and two classes.
   - For Dataset 3 a non-linear classifier seems appropriate since there isn't a clear way to separate the data linearly. This data contains 2 features and three classes.
   - For Dataset 4 a non-linear classifier seems appropriate since there isn't a clear way to separate the data linearly. This data contains 64 features and ten 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**
   - It reduces dimensionality and gives invariance to small distortions, which makes it more robust.

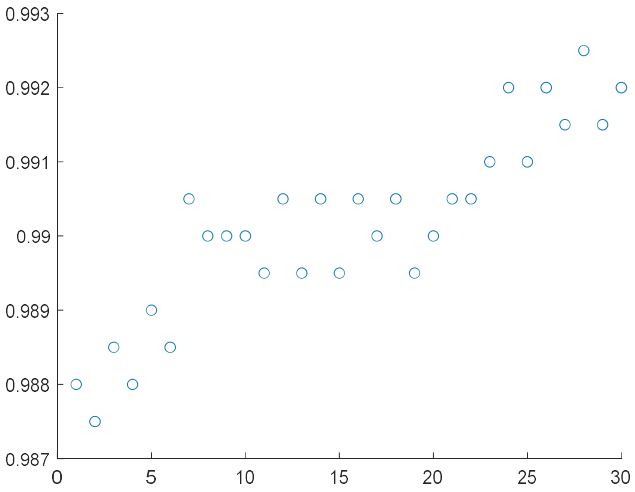
3. **Give a short summary of how you implemented the kNN algorithm.**
   - First we calculate the Euclidean distance from all points in X to all points in XTrain.
   - From the distance we identify the k smallest distances
   - We identify their labels, and a majority vote decides how we classify the point in X.

4. **Explain how you handle draws in kNN, e.g. with two classes (k = 2)?**
   - The "smallest" class label is the "winner" in a draw.

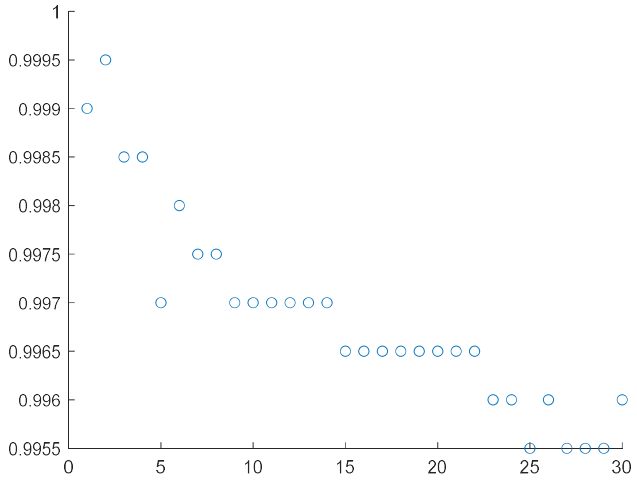
5. **Explain how you selected the best k for each dataset using cross validation. Include the accuracy and images of your results for each dataset.**
   - We choose the K with the largest average accuracy for all datasets. In case of several K with the same accuracy we choose the largest to decrease model complexity. With

a kNN model the complexity of the model decreases with higher k since more points are considered when classifying new points, making it less complex.

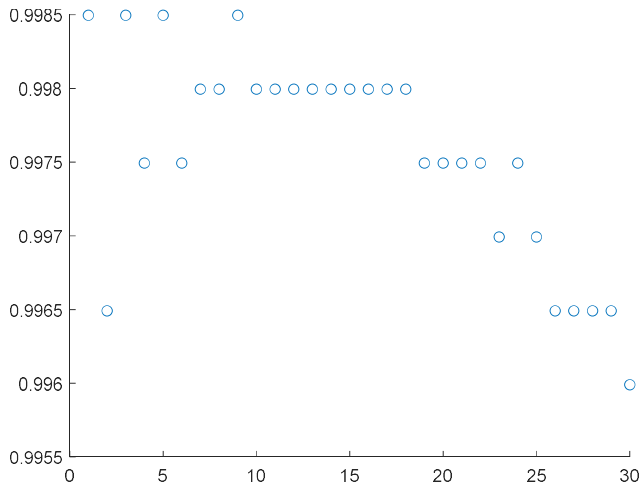
- 1: K=28, accuracy=99.25%

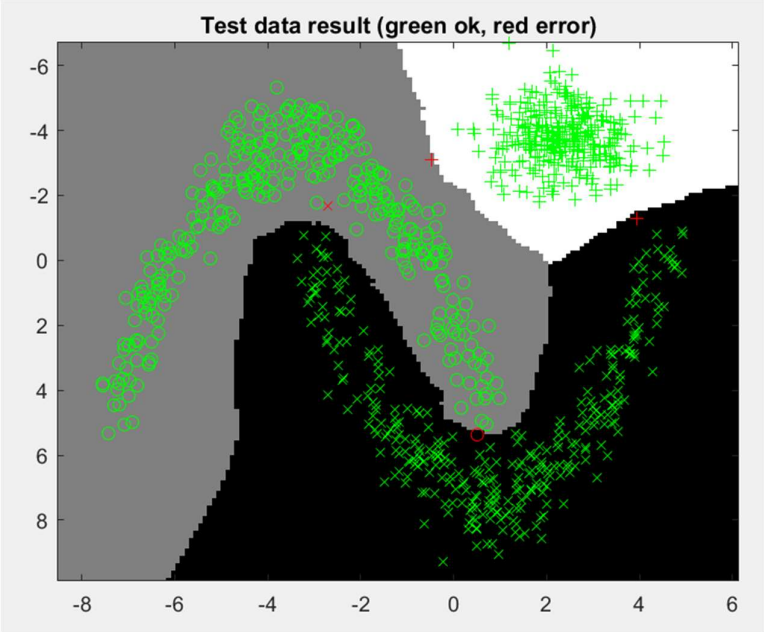


- 2: K=2, accuracy=99.95%

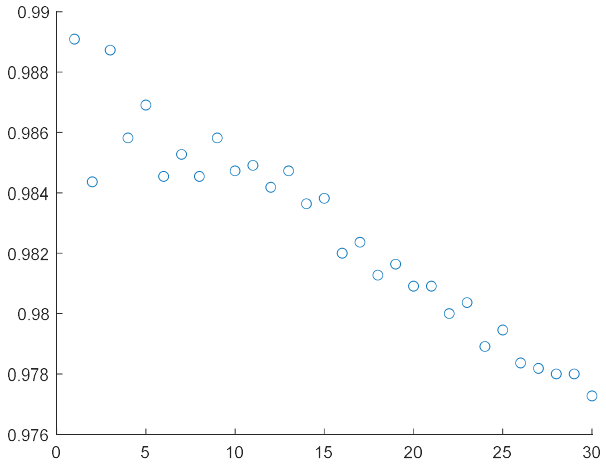


- 3: K=9, accuracy=99.85%.





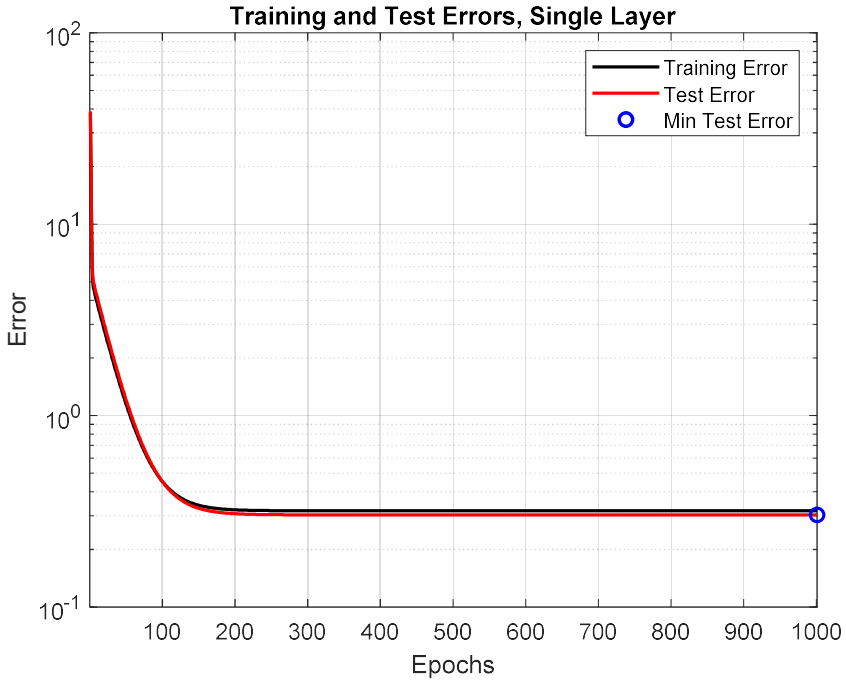
- 4: K=1, accuracy=98.91%



6. **Give a short summary of your backprop implementations (single + multi). You do not need to derive the update rules.**
   For the Single layer the backpropagation was simple. We take the difference between the output of our network and the correct output and multiply that with the training features. We also had to multiply the gradient with 2/NTrain to get good results. We updated the new weights using the gradient and the learning rate.

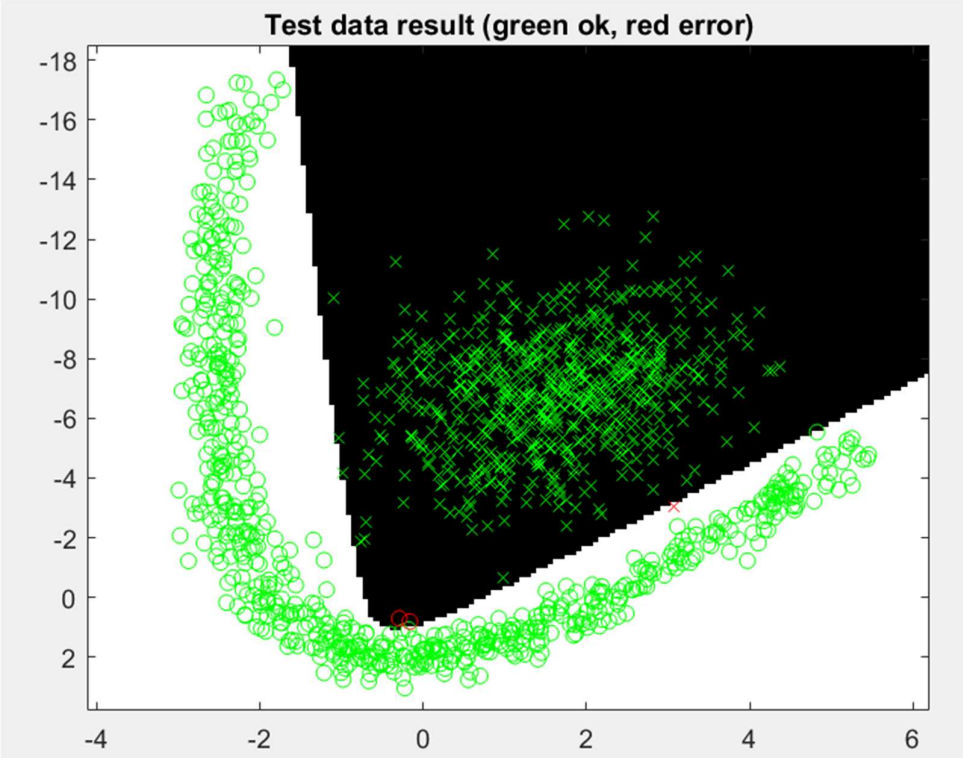
   For the multi-layer we have two sets of weights. One set for the output layer(V), and one for the hidden layer(W). The V set is calculated in the same manner as the single layer, but with the small difference that instead of multiplying with the training features, we multiply with the activations from the hidden neurons. For the set W we multiply the former V weights with the difference between our output and the desired output. This was then element wise multiplied the product of the derivative of the hidden layer output times the input training features.
   The W and V weights were then updated using the gradients.

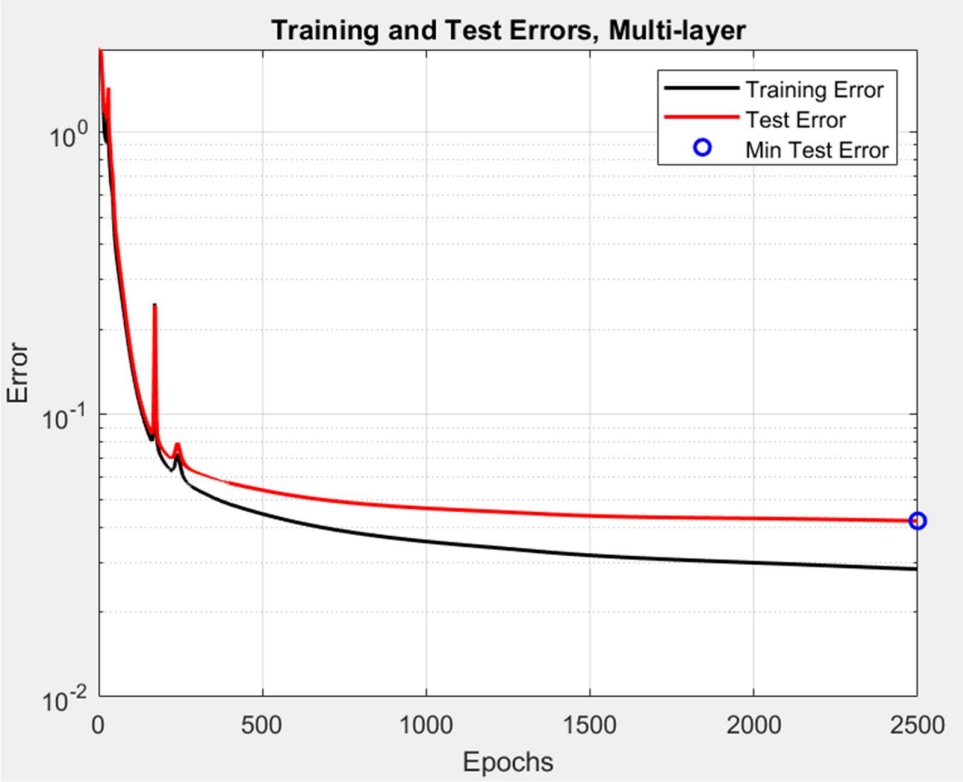
7. **Present the results from the neural network training and how you reached the accuracy criteria for each dataset. Motivate your choice of network for each dataset. Explain how you selected good values for the learning rate, iterations, and number of hidden neurons. Include images of your best result for each dataset, including parameters etc.**
    - **1:** Iterations: 1000, Learning rate: 0.01. We choose the Single layered network to train the model since the data is linearly separable, and therefore a single layer will suffice and take less time to train. We obtained the test accuracy > 99%. We choose the parameters by trying different values.



- **2:** Hidden nodes: 20, Iterations: 2500, Learning rate: 0.1. We choose the Multilayered network since the data isn't linearly separable. We obtained the test accuracy > 99% error rate for several tries. We choose the parameters by trying

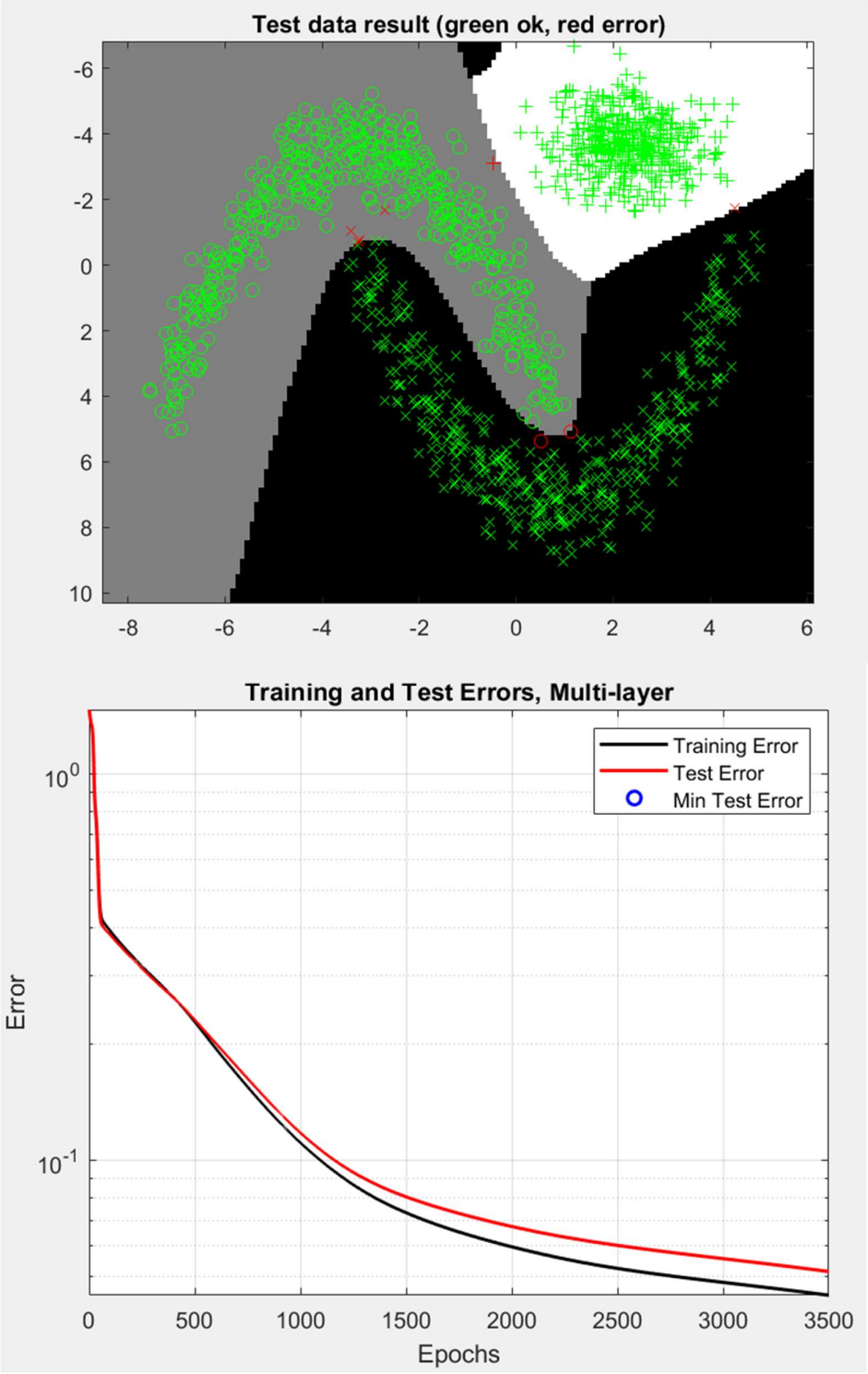
different values.



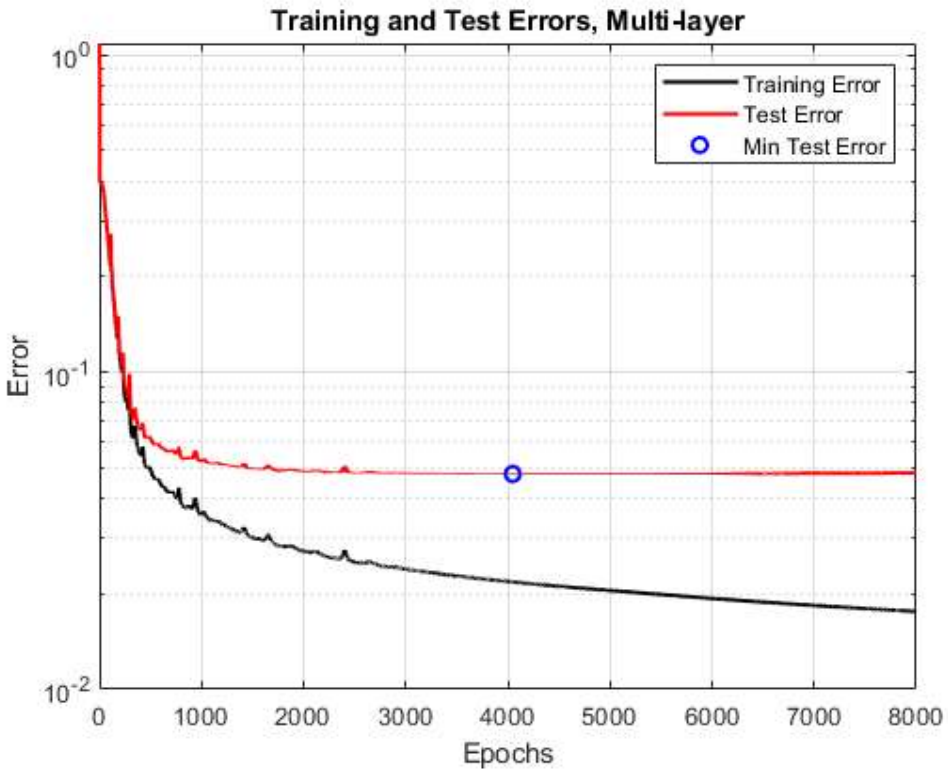


- **3:** Hidden nodes: 10, Iterations: 3500, Learning rate: 0.03. We choose the Multilayered network since the data isn't linearly separable. We obtained the test Accuracy > 99% error rate for several attempts. We choose the parameters by trying

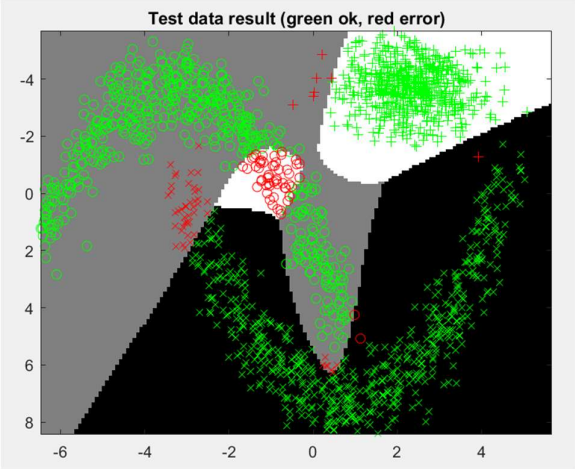
and stopping when obtaining a good enough result.



- **4:** Hidden nodes: 64, Iterations: 8000, Learning rate: 0.01. We choose the Multilayered network since it isn't linearly separable. We obtained the error: 96.2%. We selected the values above through by trying different values and achieving good result with the chosen ones.



8. **Present the results, including images, of your example of a non-generalizable backprop solution. Explain why this example is non-generalizable.**
    - This is one example of a solution of a non-generalizable solution. Where the error for test data is close to perfect, but the model cant generalize to the test data in a good way.



9. **Give a final discussion and conclusion where you explain the differences between the performances of the different classifiers. Pros and cons etc.**
   The differences are that kNN compared to the neural networks isn't trained in the same manner but is bases its decision on a stored data set. This makes kNN a slower method to use when classifying, since we iterate over all the data every time with kNN, while with the networks all the work is done in training, but it is fast to use when trained. kNN is a simple method that can capture complexities in data that other models are having trouble with, especially models that depend on linear separability. The difference between a single layer and a multilayer-network is that multilayer-networks can be used to classify data that is not linearly separable, but is often slower to train due to the increased complexity.

10. **Do you think there is something that can improve the results? Pre-processing, algorithm-wise etc.**
    - The data can be normalized to between -1 and 1 to have better results within the tanh function.
    - One could change the activation function to a more simple one to ease backpropagation

11. **Optional task (but very recommended). Simple gradient decent like what you have implemented can work well, but in most cases we can improve the weight update by using more sophisticated algorithms. Some of the most common optimization algorithms are summarized nicely here: https://towardsdatascience.com/optimizers-for-training-neural-network-59450d71caf6 Implement one or a few different optimizers and compare the speed at which the training converges. A good starting point is to implement momentum gradient decent.**