TBMI26 – Computer Assignment Report Supervised Learning

Deadline – February 12 2018

Author/-s:
David Tran davtr766
Jakob Bertlin jakbe457

1. **Give an overview of the data from a machine learning perspective. Consider if you need linear or non-linear classifiers etc.**
   In general, when data can be linearly classified they are in divided into very separate clusters. See question 7 for further discussion.

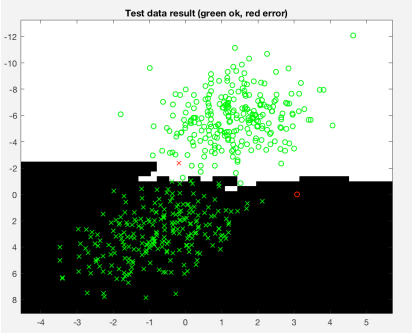
2. **Explain why the down sampling of the OCR data (done as pre-processing) result in a more robust feature representation.**
   When dealing with high resolution images there are a lot of features since every pixel is considered to be an input feature. The down sampling of OCR data is done to reduce the number of feature parameters thus increasing computational efficiency and performance. The down sampling is also done to highlight the most important feature representation.

3. **Give a short summary of how you implemented the kNN algorithm.**
   First calculates the distance from a given point to all training samples and sorts them in ascending order in a new vector, then stores the indexes to the K-closest training points, which are the K first distances in the sorted vector. The most frequently occurring label of the stored indexes is then determined by the built in mode-function.

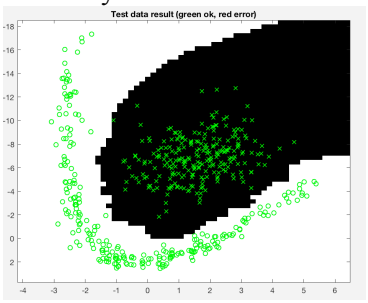
4. **Explain how you handle draws in kNN, e.g. with two classes (k = 2)?**
   If it's a draw the mode function will then return the label it encountered first, which in this case means the label of the point which is closest.

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 determined the most optimal k for each dataset by using 4-fold cross validation on the given dataset. For every fold we compute for every k the accuracy and store the highest accuracy and the associated k value.

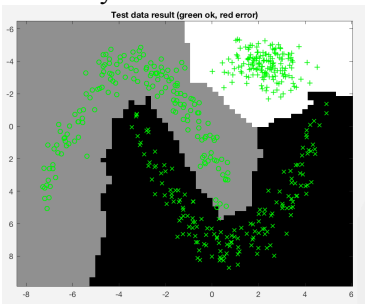
   Accuracy for dataset 1: 0.996



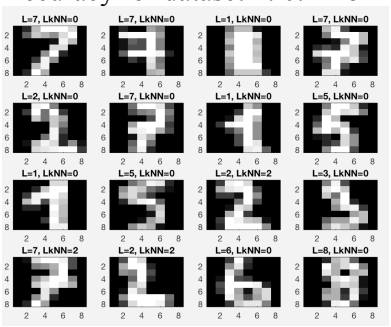
Accuracy for dataset 2: 1.0



Accuracy for dataset 3: 1.0



Accuracy for dataset 4: 0.1225



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

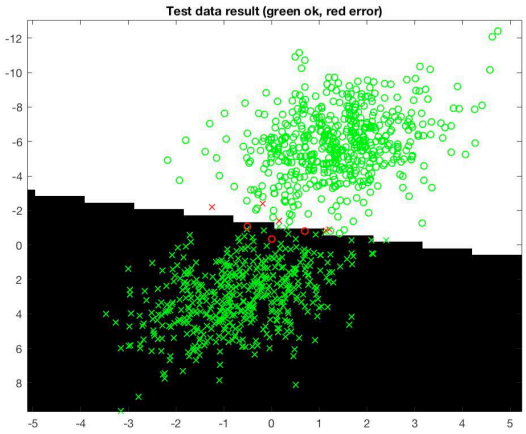
   First a bias is added to the neural network and then starts off by initializing the weight matrix between -1 and 1. Then run the initialized network and use the error derived from that to calculate the gradient for the weights. Then repeat this as many times as specified by amount of iterations.

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.**

Every parameter chosen is determined by the complexity of the dataset and through trial and error.

Dataset 1:

- Number of hidden units: 1
- Number of iteration: 2000
- Learning rate: 0.03

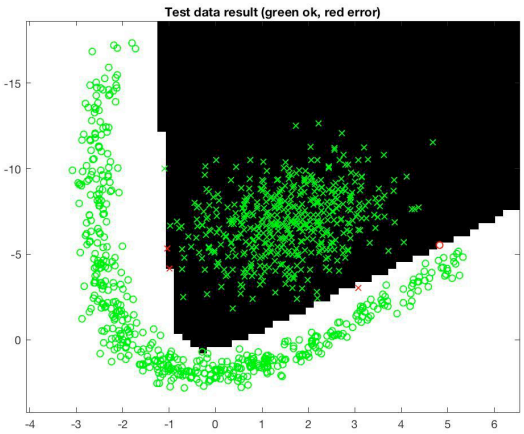


Time spent training: 0.77973 sec
Time spent classifying 1 feature vector: 2.9267e-07 sec
Accuracy: 0.992

Motivation: From the plot, we can see that dataset 1 can be easily divided with a linear classifier. Therefore, our neural network does not have to be so complex. One hidden unit is enough and the number of iteration is quite low for the neural network.

Dataset 2:

- Number of hidden units: 3
- Number of iteration: 10 000
- Learning rate: 0.01

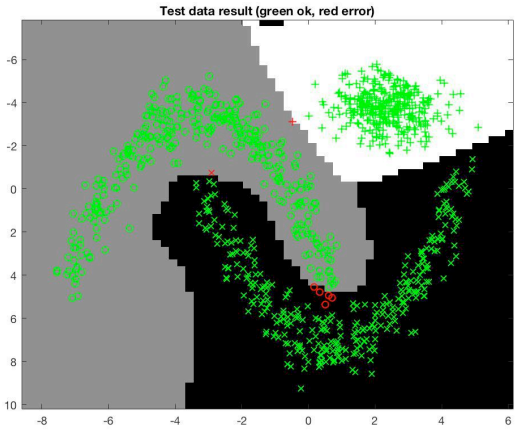


Time spent training: 5.0933 sec
Time spent classifying 1 feature vector: 1.9562e-07 sec
Accuracy: 0.996

Motivation: From dataset 2, we can see that the dataset can't be linearly classified and therefore the neural network has to be more complex because in this case, the dataset has to be classified non-linearly. Since the dataset still contains two classes it is enough with three hidden units. However, a higher number of iteration is more plausible compared to dataset 1.

Dataset 3

- Number of hidden units: 11
- Number of iteration: 15 000
- Learning rate: 0.013

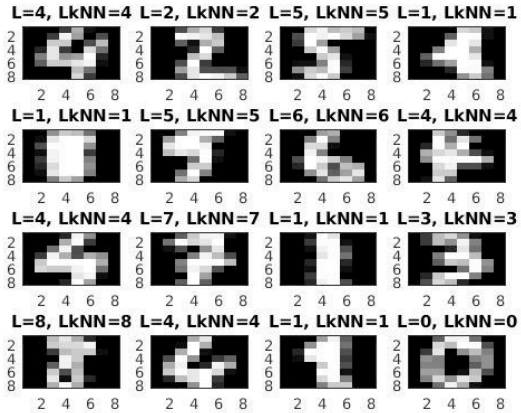


Time spent training: 22.6203 sec
Time spent classifying 1 feature vector: 4.3874e-07 sec
Accuracy: 0.99299

Motivation: Dataset 3 has three classes which has to be classified. The neural network has to be significantly more complex and therefore we have chosen number of hidden units to 11 and 15 000 iterations.

Dataset 4:

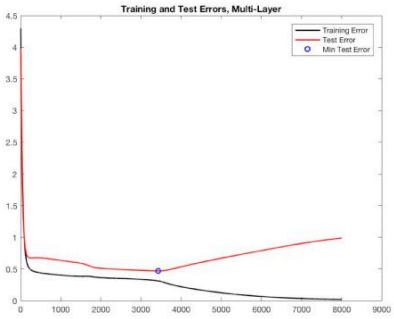
- Number of hidden units: 62
- Number of iteration: 35000
- Learning rate: 0.0012

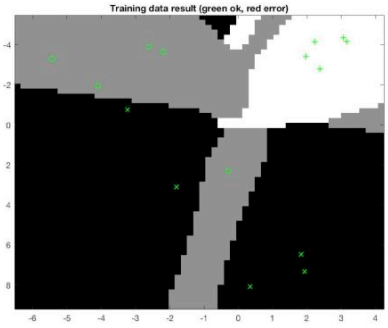


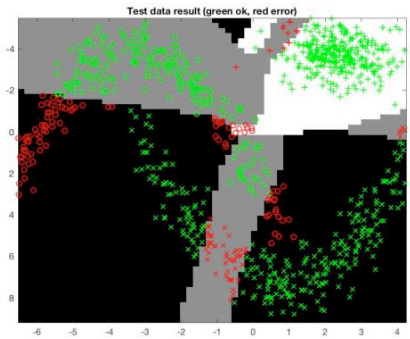
Time spent training: 1026.2169 sec
Time spent classifying 1 feature vector: 1.9303e-06 sec
Accuracy: 0.96282

Motivation: Handwritten numbers are very abstract concept compared to the other datasets. Which is why the number of hidden units is increased substantially. But through testing it was also discovered to need a lot of training iterations with low training rate to classify numbers correctly, otherwise it would always get stuck with around 80-90% accuracy.

8. **Present the results, including images, of your example of a non-generalizable backprop solution. Explain why this example is non-generalizable.**
To overfit the neural network we increase the number of hidden units and number of iterations for dataset 3 and in the same time reducing the training samples. The result is the following:



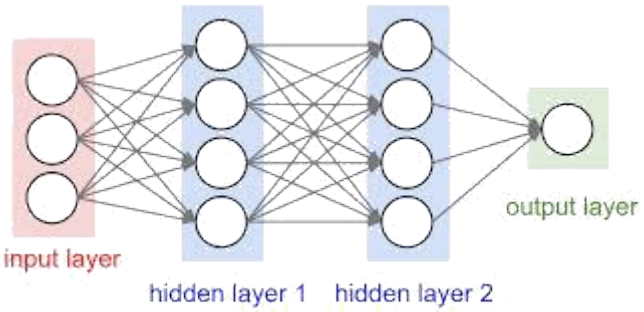




From the plot we can see that the neural network performs well on the training data but as soon as the neural network test on new data it performs worse. From the graph we can also see that the test error increases as the training error decreases which is a strong indication that the neural network is overfitted. The example is non-generalizable because the neural network learns only non-generalizable characteristics in the training data and therefore the trained neural network won't work for other data samples.

9. **Give a final discussion and conclusion where you explain the differences between the performances of the different classifiers. Pros and cons etc.**
kNN is a good classifier when the training data doesn't have to be too large since you always have to carry the training data with you, however you also don't need to train the classifier when using kNN. This is in contrast to neural networks which instead only uses the training data once during the training and then carries a very lightweight classifier only consisting of weight data, neural network equations and no training data at all.

10. **Do you think there is something that can improve the results? Pre-processing, algorithm-wise etc.**
In more complex dataset such as the OCR data, an increase in hidden layer for the neural network should improve the results. For example, consider the neural network below:



The extra hidden layer can improve the results. The first hidden layer extracts the general robust feature representations such as edges, curves etc. that corresponds to a handwritten digit. The second hidden layer will pick up abstract concepts of given handwritten digit and then determines what digit it is in the output layer.

Also some kind of dynamic learning rate would most likely greatly improve results on abstract classifiers such as handwritten digits since there can be an abundance of local minima which you sometimes need to avoid getting stuck in. In short, having a learning rate that can dynamically increase and decrease to find different minima.

Some kind of feature selection before any training can also be implemented to reduce overfitting on features that might not actually be a factor in the result.