TBMI26 – Computer Assignment Report  
Supervised Learning

Deadline – March 15 2020

Fredrik Johansson

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.**

The first two datasets consist of data 2D data with two classes. The data in the first dataset form two convex clusters which means that we can use a linear classifier. In the second dataset, the data form one convex and one non-convex cluster. Because we have one non-convex cluster, we must use a non-linear classifier.

The third dataset consist of 2D data with three classes and two of them are non-convex clusters. Therefore, we must use a non-linear classifier. The last dataset consists of 10 classes with 64 features each. With that many features a non-linear classifier is preferable.

1. **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**](http://archive.ics.uci.edu/ml/datasets/Optical+Recognition+of+Handwritten+Digits)

When writing numbers on a paper, the edges between the numbers and the paper will consist of a grayscale. By down sampling the grayscale can be removed so the data is reduced to binary, either paper(white) or number(black). Since all the numbers have this grayscale it is not unique for each digit. Therefore, by removing it we achieve a more robust feature representation.

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

I iterate through all the datapoints and for each point I calculate the distance to all other points. Then I store the k nearest datapoint indices, and with the indices I check which class I should predict based on the training data. Lastly, the final prediction is based on the class with the most amount of predictions.

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

I use the MATLAB function mode, if it is a tie it returns the class with the smallest number. It may not be the best approach, but it is okay for the performance needed. A better approach would for too example be to return the class with the smallest mean distance.

1. **Explain how you selected the best k for each dataset using cross validation. Include the accuracy and images of your results for each dataset.**

I split the data into three subsets, and I calculate the accuracy with cross validation in a loop. The loop iterates nine times and k is incremented by one for each iteration. This means that the algorithm is tested for nine different k values.

Dataset 1:

k = 5

Average accuracy = 99.83%

En bild som visar text

Automatiskt genererad beskrivningEn bild som visar text, karta

Automatiskt genererad beskrivning

Dataset 2:

k = 1

Average accuracy = 100%

En bild som visar text, karta

Automatiskt genererad beskrivningEn bild som visar text, karta

Automatiskt genererad beskrivning

Dataset 3:

k = 1

Average accuracy = 99.89%

En bild som visar text, karta

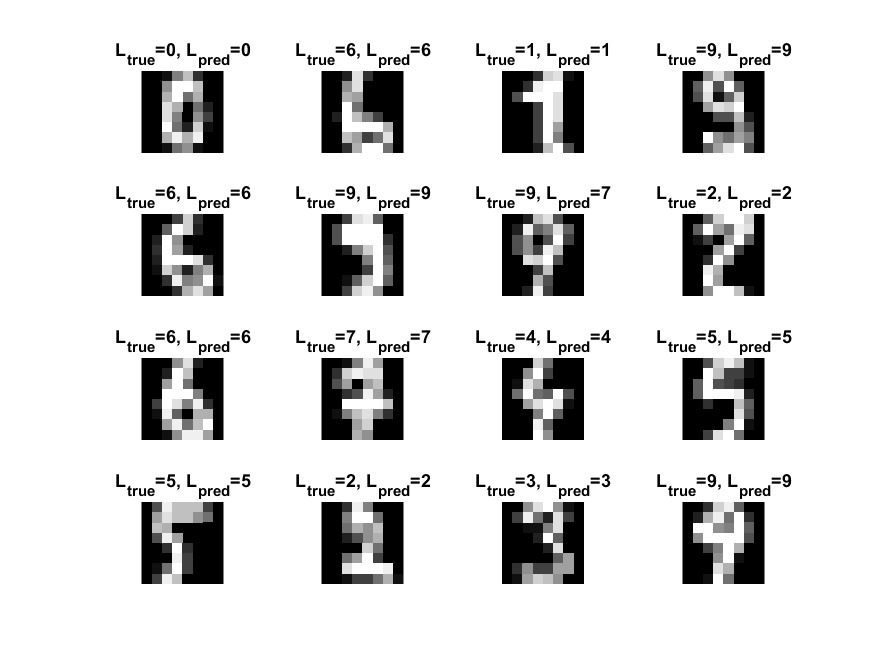
Automatiskt genererad beskrivningEn bild som visar text

Automatiskt genererad beskrivning

Dataset 4:

k = 1

Average accuracy = 98.27%



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

Single: The algorithm begins by initializing the weights in a matrix of the size MxN, where M equals the number of input features (input nodes) and N equals the number of classes (output nodes). The algorithm continues by calculating the initial error. This is done by multiplying the training and test input separately with the weight matrix. After this step an iterating process begins. For every iteration the algorithm calculates the gradient of the cost function and takes a learning step in that direction. The gradient is found by back propagating the network and the purpose of the learning step is to lower the error. Once a new weight has been found the iteration continues until the algorithm reaches a local or global minimum of the cost function. The algorithm will also step if the maximum number of iterations have been reached.

Multi: This variation of the algorithms uses the same process but with some changes. There are now two weight matrices. The weight matrix between the input and the hidden layer is of the size MxN, where M still equals the number of input features. However, N now equals the number of hidden neurons in the hidden layer. The second weight matrix between the hidden layer and the output is of the size NxP, where P equals the number of classes.

1. **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.**

For dataset 1 I used a single layer network because the data in linearly separable. I used 10000 iterations with the learning step of 0.001 and the accuracy was 99.7%.

En bild som visar skärmbild

Automatiskt genererad beskrivningEn bild som visar text

Automatiskt genererad beskrivningEn bild som visar text

Automatiskt genererad beskrivning

For dataset 2 I used a multi-layer network since the data were not linearly separable. I used 20 hidden neurons with 10000 iterations and the learning step of 0.01. The resulting accuracy was 100%.

En bild som visar skärmbild

Automatiskt genererad beskrivningEn bild som visar text, karta

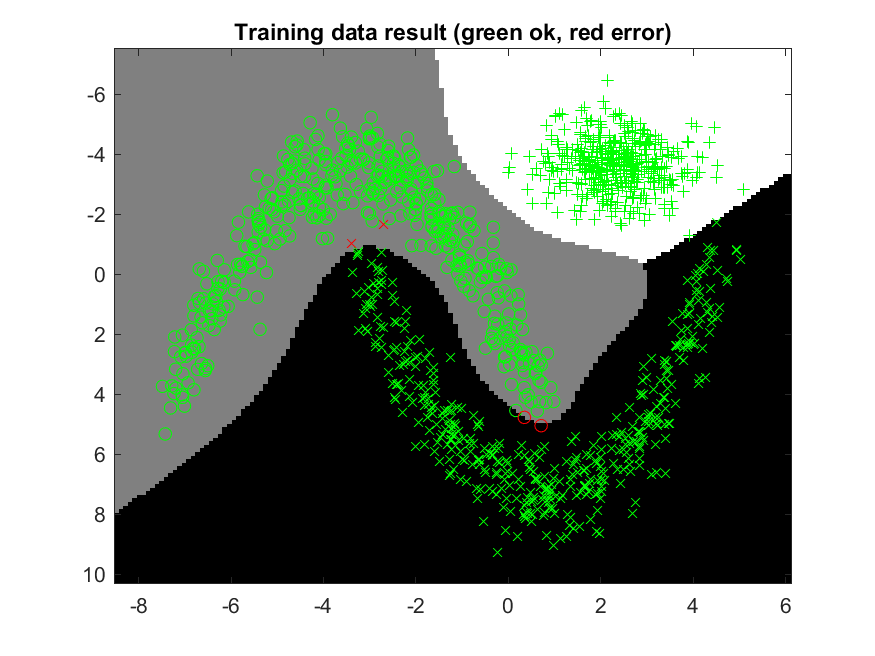
Automatiskt genererad beskrivningEn bild som visar text, karta

Automatiskt genererad beskrivning

For dataset 3 I used a multi-layer network since the data were not linearly separable. I used 40 hidden neurons with 10000 iterations and the learning step of 0.01. The resulting accuracy was 99%.

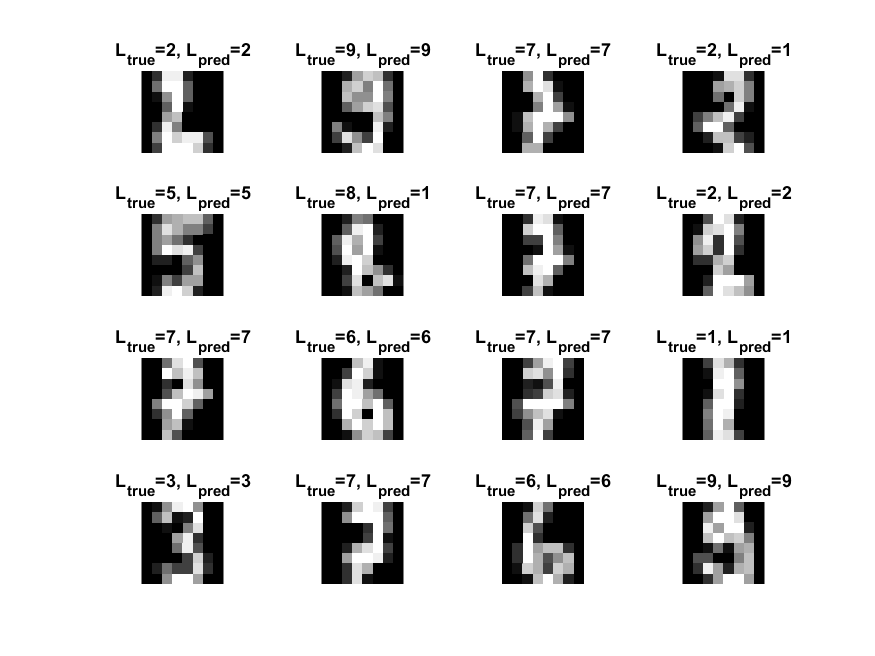
En bild som visar karta, text

Automatiskt genererad beskrivningEn bild som visar text

Automatiskt genererad beskrivning

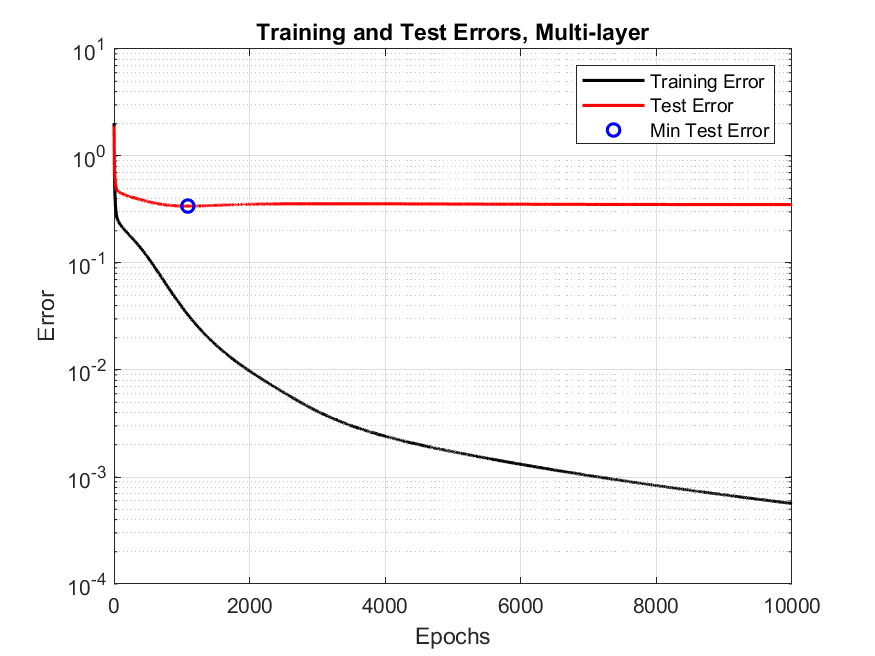
For dataset 4 I used a multi-layer network since the data were not linearly separable. I used 40 hidden neurons with 10000 iterations and a learning step of 0.001. The resulting accuracy was 96%.

En bild som visar skärmbild, karta

Automatiskt genererad beskrivning

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

For this example, I divided the data into 100 samples. The model is trained using one of these samples. The remaining 99 samples were used to test the model. This led to every sample containing 17 data points. Therefore, the model heavily depends on these 17 points in the training phase. If we look at the result from question 8, we can see that the data have two dense clusters in the top corners. From this we can draw the conclusion that it will be more likely to choose training data points which lies within these clusters. It is less likely to choose data points from the less dense areas, such as the circle class rightmost edge. By using more training data this probability increases and makes the model more general. I used the same parameters as the result in question 8 for dataset 3 (20 hidden neurons, 10000 iterations and a step size of 0.01). The result is shown below:

En bild som visar text

Automatiskt genererad beskrivningEn bild som visar text

Automatiskt genererad beskrivning

1. **Give a final discussion and conclusion where you explain the differences between the performances of the different classifiers. Pros and cons etc.**
2. **Do you think there is something that can improve the results? Pre-processing, algorithm-wise etc.**

Momentum to the error func.