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 2D data with two classes. The data in the first dataset form two convex clusters. Convex clusters are clusters which distances between two arbitrary points are within the cluster. With convex clusters we can use a linear classifier. In the second dataset, the data form one convex and one non-convex cluster. Since 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 images the grayscale can be reduced, resulting in data with distinct edges between paper(white) and number(black). We can do this since all the numbers have the grayscale in common. Therefore, by reducing 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 for each index I check which class it is predicted to belong to, 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 label number. It may not be the best approach, but it is okay for the specifications given in this case. A better approach would for 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 change which subset is used for testing and which are used for training. For every combination I calculate the accuracy of the model and the final accuracy is the mean of these values. This process is done nine times where k is initialized as 1 and is incremented by one for each iteration. This means that the algorithm will test the accuracy of the model for k = 1,…,9.

**Dataset 1:**

K = 5

Time spent classifying 1 sample: 4.4161e-05 sec

Average accuracy = 99.83%

En bild som visar text, karta

Automatiskt genererad beskrivningEn bild som visar text

Automatiskt genererad beskrivning

**Dataset 2:**

K = 1

Time spent classifying 1 sample: 2.4189e-05 sec

Average accuracy = 100%

En bild som visar text, karta

Automatiskt genererad beskrivningEn bild som visar text, karta

Automatiskt genererad beskrivning

**Dataset 3:**

K = 1

Time spent classifying 1 sample: 3.676e-05 sec

Average accuracy = 99.89%

En bild som visar text

Automatiskt genererad beskrivningEn bild som visar text, karta

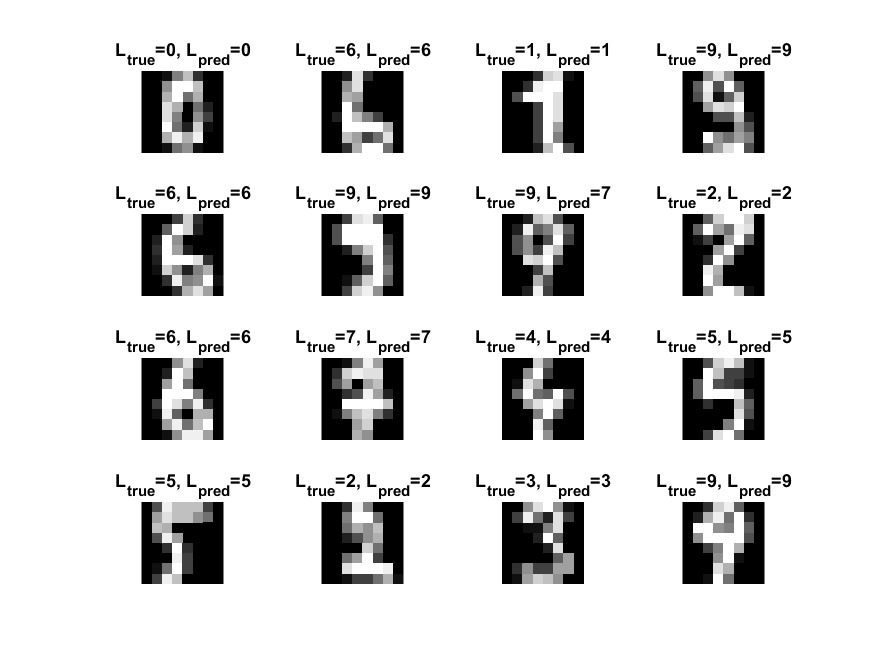
Automatiskt genererad beskrivning

**Dataset 4:**

K = 1

Time spent classifying 1 sample: 0.00018114 sec

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 beforehand determined maximum number of iterations.

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 neurons in the hidden layer. The second weight matrix lies between the hidden layer and the output is of the size NxP, where P equals the number of classes. After this the iterating process begins. This time two gradients are calculated, one for each weight matrix and both are updated accordingly. Otherwise the algorithm behaves the same as before.

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.**

All the parameters for all the datasets except the first one where gathered using trial and error to achieve the accuracy criteria. For the first dataset the elbow method where used to find suitable parameters. A single layer network was used to classify dataset 1 since the data were linearly separable and there is therefore no need to add a hidden layer. Below are the used parameters and the result:

Iterations: 1000

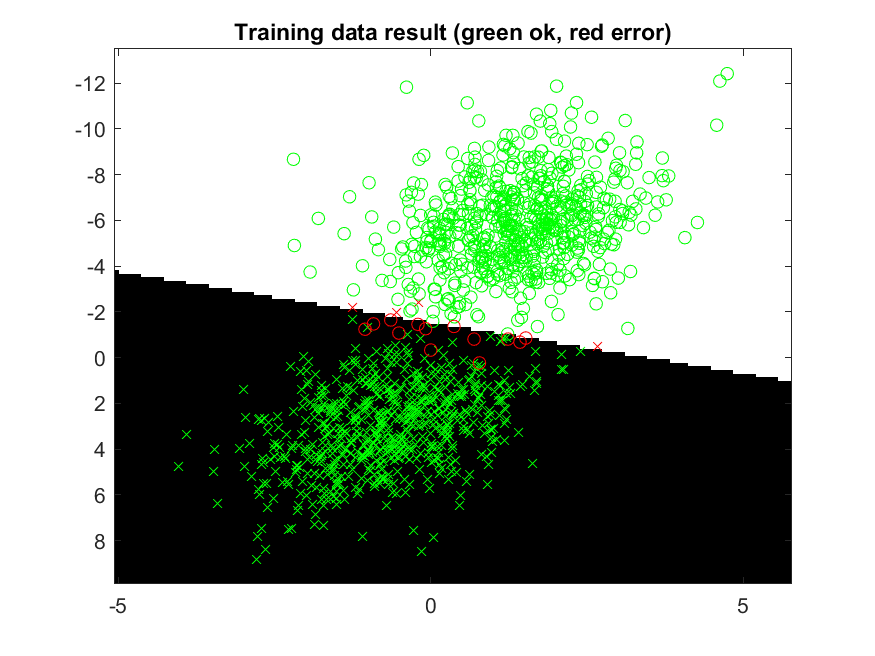
Learning rate: 0.001

Time spent training: 0.029271 sec

Time spent classifying 1 sample: 1.3238e-07 sec

**Accuracy: 99.6%**

En bild som visar karta, text

Automatiskt genererad beskrivning

For dataset 2 I used a multi-layer network since the data were not linearly separable. Below are the used parameters and the result:

Hidden neurons: 20

Iterations: 10000

Learning rate: 0.01

Time spent training: 12.4524 sec

Time spent classifying 1 sample: 8.3193e-07 sec

**Accuracy: 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. Below are the used parameters and the result:

Hidden neurons: 20

Iterations: 10000

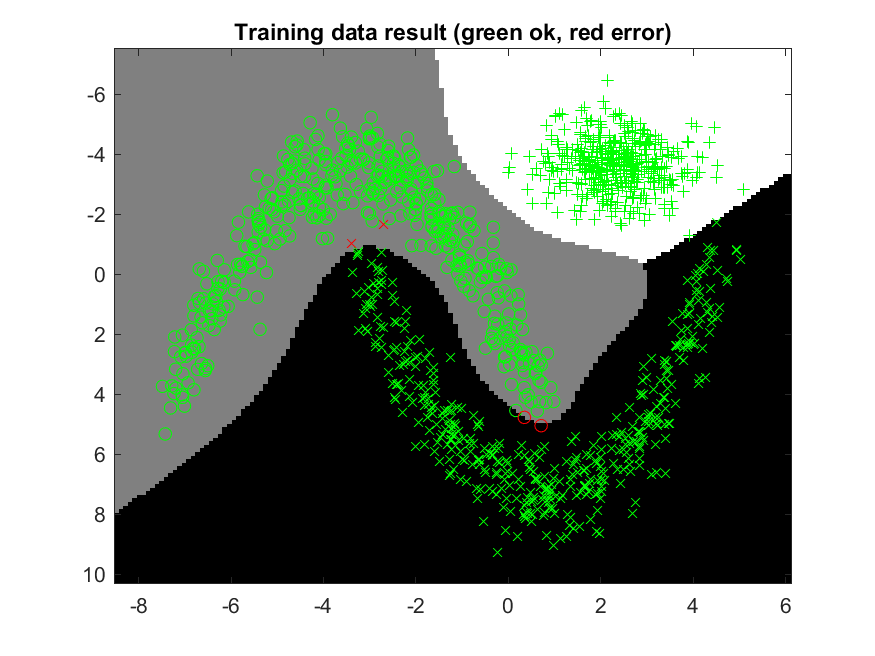
Learning rate: 0.01

Time spent training: 12.2736 sec

Time spent classifying 1 sample: 8.1226e-07 sec

**Accuracy: 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. Below are the used parameters and the result:

Hidden neurons: 30

Iterations: 10000

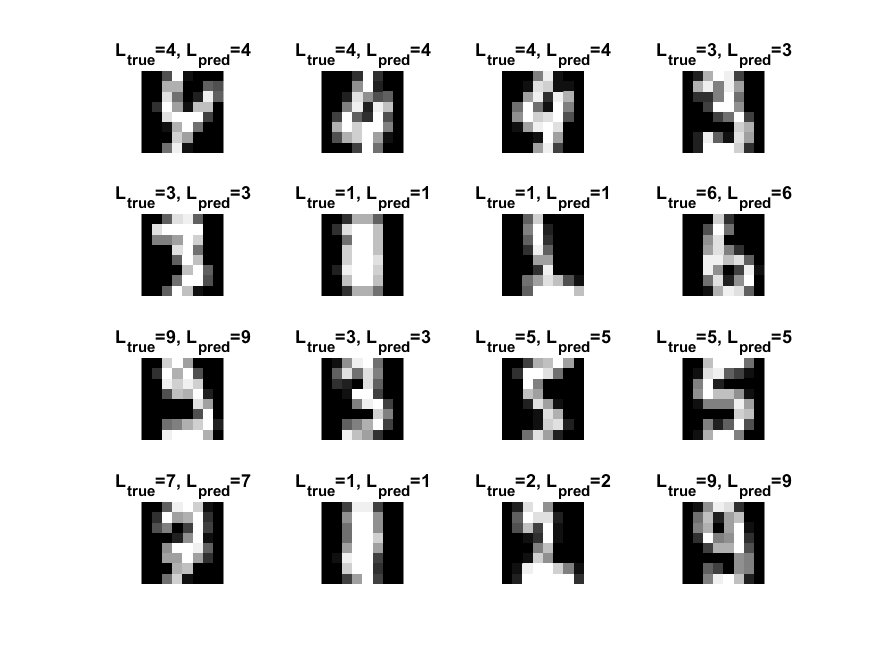
Learning rate: 0.001

Time spent training: 72 sec

Time spent classifying 1 sample: 1.026e-06 sec

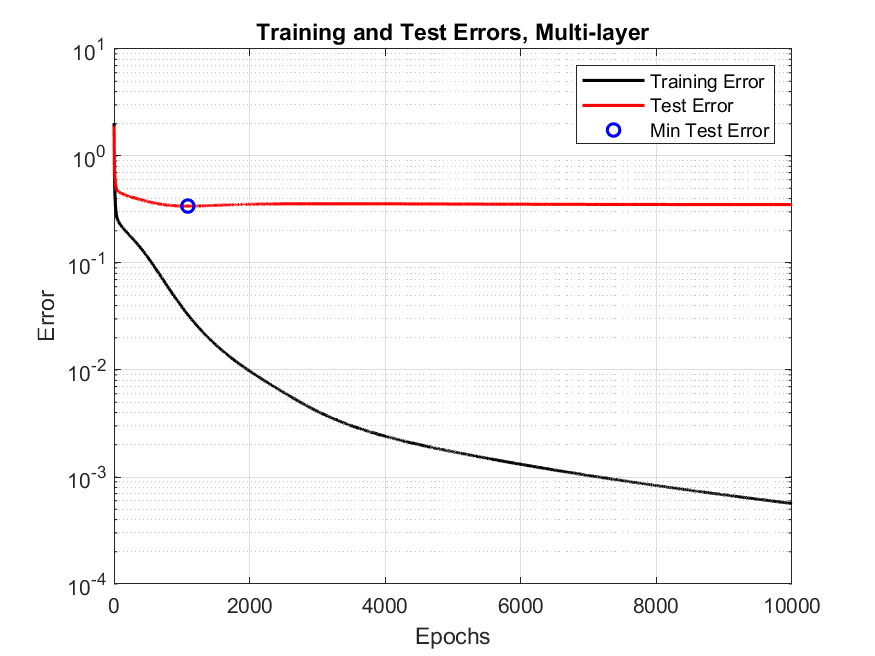
**Accuracy: 96,7%**

En bild som visar skärmbild

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 and the remaining 99 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 dataset 3 in question 7, we can see that the data have some dense clusters, especially in the top corners. From this we can draw the conclusion that it is more likely to draw training data points that lies within these clusters. Furthermore, it is less likely to draw data points from the less dense areas, such as the circle class’s rightmost edge in the lower right corner. By using more training data, the probability to get a data point within the less dense areas increases and therefore makes the model more general. I used the same parameters as in the result of 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.**

The accuracy of the classifiers is very similar. Regarding the performance of the classification, the neural network is faster. However, the neural network requires a training phase which is substantially more time consuming, whereas the kNN algorithms does not. The kNN algorithm were easier to implement than the neural network and it only have one parameter to tune compared to the three of the neural network implementations. Once the neural network has been trained the training data can be removed, in contrast to kNN which must have the training data available to classify.

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

We could add a momentum term to the weight increment. This is done by adding the previous gradient value multiplied by a momentum factor to the weight increment. This tends to keep the gradient in the same direction as previous iteration, resulting in less oscillations.