TBMI26 – Computer Assignment Report  
Supervised Learning

Deadline – March 15 2019

Author/-s:

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.**

The target size of the data is a variable with classes. A linear classifier exists if a hyperplane can be drawn as the class boundary to classify the data. When evaluating a classification problem with many classes, as is the case here, it is very likely that the classes overlap and the Hyperplane no longer separates linear classes from each other. Therefore we do not need linear classifiers here.

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)

Unfortunately the OCR data would lead to very inaccurate results, if it wouldn`t be preprocessed prior to analysis. This is especially needed in this field because of low quality of the images, different writing style, orientation and complex background of the image. That could all lead to a misclassification in the analyzed text.

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

The kNN algorithm is implemented as follows:

1. Defining the training and test dataset
2. Calculating pairwise eucledian distances for the test points to all the training observations by using the function pdist2.
3. Sorting the distances by columns, selecting the k smallest distances and then fetching the indices of the closest data points, to get the labels of them in the end. This is done by fetching the class by taking the values of the L1 vector at the indices of the closest data points.
4. The classes of the k nearest neighbors are saved in the variable labels
5. The matlab- built-in function mode returns the majority class of each column (which means, for every observation it returns the one class it is classified as.)
6. **Explain how you handle draws in kNN, e.g. with two classes (k = 2)?**

With an even number of nearest neighbors and, for example, only two groups to be classified, it can often happen that the majority vote cannot return a unique winner. One solution is to weight the points according to how close they are to the point to be classified. Points that are further away have less influence on the decision than points that are closer.

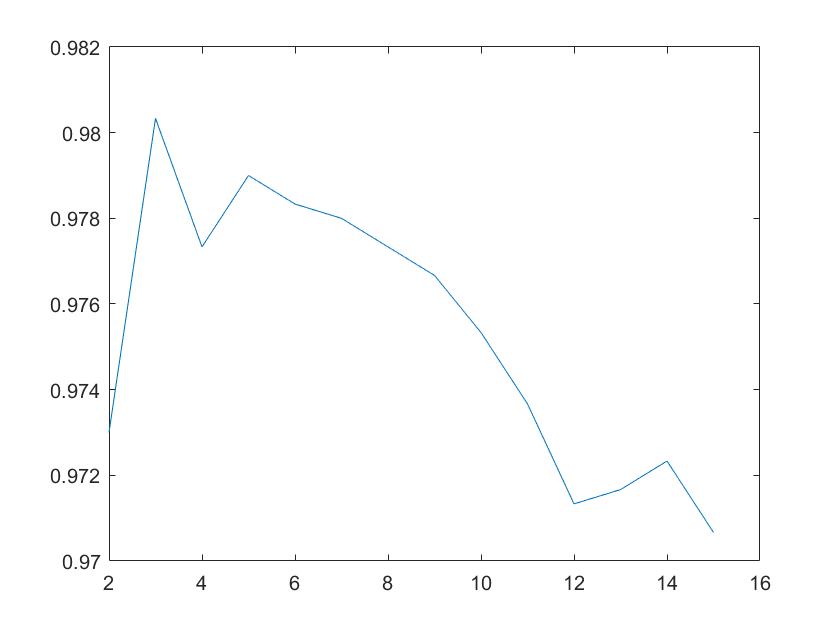
1. **Explain how you selected the best k for each dataset using cross validation. Include the accuracy and images of your results for each dataset.**

For calculating the accuracy, we created a cross validation model. That means, for every value of k we want to evaluate, we evaluated the kNN model n times (where n is the number of folds that is used by cross validation). For n folds, the full data set is divided into n partitions. The following picture explains how the CV works:



For every fold as a test data set, we will get an accuracy value for the classification by the kNN model. This accuracy is averaged over the number of made folds.

In the end, for every value of k, an averaged accuracy value is saved in a vector. Plotting this shows in one picture which value of k is best to use:

By averaging over the accuracies of all the folds for each k, a stable solution can be reached. Also, the higher the number of folds, the higher the stability of the solution. But it should be noted that it lowers again, the lower the number of training points get.

It can be seen, that the best k to use is 3, because the averaged cross- validation accuracy is the highest there.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Dataset | acc | k | optimal k | Classification |
| 1 | 98.5 % | 3 | Figure 1 | Figure 1001 |
| The optimal k for dataset 1 can be found at a k – value of 6 to 14. To reach an accuracy of at least 98 %, a k of 3 is enough. That way not that many neighbors have to be taken into account. | | | | |
| 2 | 99.5 % | 4 | Figure 1 | Figure 1001 |
| For dataset 2 the optimal k can be found at a value of 4. There an accuracy of 99.5% can be reached. The results for the classification on the test dataset can be seen above on the right. As the data here can not be classified by a linear separator, the model needed to be a little bit more complicated, aka taking more neighbors into account then for the first dataset. That way a better and more exact picture of the surrounding can be created. | | | | |
| 3 | 99.67% | 4 | Figure 1002 | Figure 1001 |
| 4 | 98.3% | 3 | Figure 1001 | Figure 1003 |

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

Single Layer Backpropagation Network:

* First all variables were initiated
* The initial error is calculated by a multiplication of the initial weight matrix with X(train and test) and put through an activation function (Which is tanh in this case). This returns two initial errors.
* For the rest of the iterations the multiplication of the weight matrix with the training data is used to calculate the gradient:

where Dt is a vector with the real labels, Xt is the training dataset.

* The weights are then updated by multiplying the old weights with the Learning rate and the gradient.
* In the end, the training and test errors are calculated with the new weights and saved in a vector.

Multi-Layer Backpropagation Network:

The Multilayer Network works similar in a sense.

* Here also the initial errors are calculated for the first iteration. By having one hidden layer, the data is first multiplied with the initial weights, put through an activation function and then added to a bias. In the following step, the resulting values are again multiplied with a second set of initial weights, put through an activation function and then added to a bias. With the resulting values the predicted labels can be taken out, compared to the true values and then taken for the initial error calculation.
* For the backpropagation step both weight vectors must be updated. This is done for every iteration. For that, two gradients are calculated, multiplied with the learning rate and then subtracted from the old weights.
* The new weights are then taken to calculate errors for the training and test set again.

1. **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.**

Multi Layer:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Data set | number of Iterations | learning Rate | accuracy | Error | Classification  Train | Classification  Test |
| 1 | 7  400 | 0.05 | 98.8% | Figure 1101 | Figure 1103 | Figure 1104 |
| Here a low learning Rate, low number of iterations and a low number of neurons in the hidden layer is already good for a classification accuracy of 98%. That is, because the data could almost be divided by a straight line and the accuracy would still be high. | | | | | | |
| 2 | 15  4000 | 0.05 | 99.6% | Figure 1101 | Figure 1103 | Figure 1104 |
| Looking at the data, it can be seen that the two classes must be separated by something like a parabola for the classificator to perform well. A parabola is by itself already more complex than a straight line, which could be used as a separator for the first dataset. Therefore the model had to be made a bit more complicated to reach an accuracy of 99.6%. Increasing the number of neurons in the hidden layer makes the model being able to differentiate between more features. Also increasing the number of iterations the algorithm “has more time” to learn how to build the best separator for the two classes. | | | | | | |
| 3 | 25  4000 | 0.05 | 99% | Figure 1101 | Figure 1103 | Figure 1104 |
| The same applies for tuning the parameters for this dataset. As the data here can be classified into three instead of two different groups, the model gets more complicated. These three classes can neither be separated by a linear classifier nor a parabola. For the model to differentiate by multiple features and to have more time to calculate, we set the number of neurons in the hidden layer to 25 and the number of calculating iterations to 4000. In combination with a learning rate of 0.05 an accuracy of 99% could be reached | | | | | | |
|  |  |  |  |  |  |  |

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

Best generalization when the data is split into train validation and test set.

1. **Give a final discussion and conclusion where you explain the differences between the performances of the different classifiers. Pros and cons etc.**

KNN:

KNN classification is a very easy technique with a generally low required computational power. For such an „easy“ method, the results measured on the accuracy of the classification could easily be brought into the range of 98 to 99%.

The downside here is though that all the training points have to be stored. That did not cause a problem in this assignment, but could be a negative aspect when using big datasets.

NN:

Neural Networks are very good in terms of learning any class boundaries. With a multi-layer Neural Network there are a lot of parameters to tune and optimize the performance and adapt to different data conditions.

A downside of this classification method though are that the model tends to overfitting with a very high number of iterations and a low learning rate as well as the long training times.

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

As already said in task 2, especially for image recognition it is helpful to preprocess the data to get a better classification result in the end.