TBMI26 – Computer Assignment Report  
Supervised Learning

Deadline – March 14 2021

Author/-s:

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 3 datasets represent a points that have two attributes and are assigned to two or three classes (dataset3). The fourth dataset is handwritten digits with 64 features and 10 classes.

The only dataset that can be separated linearly is the first dataset. Some of the points do overlap with the other class, but the performance of a linear classifier should be good enough. The other dataset cannot be separated by a linear classifier.

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)

The data is reduced in dimensions to 8x8 bitmaps that can then in turn be represented as 64bits. This dimensional reduction helps standardize all the data, reduce the number of features to be studied and saves computational resources. This pre-processing reduces the burdens of differences in resolution and size that can occur between different elements of the dataset.

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

In our implementation of the kNN algorithm, we calculate the Euclidean distance between a point and all others and sort them by distance and keeping track of the index. We then tabulate the k-first elements and find the label that is the most represented. The label that is the most represented gives the label of our point.

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

For draws we choose the label of the nearest neighbour. This implementation works well enough for the datasets that we have and we did not foresee any issues nor encountered any. A different implementation for solving the draw could be to increase or decrease k by 1 to solve for a particular point.

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

For both the single layer and multilayer implementation we start by initializing the data and separating it into training and test data. We add a bias to both. We then initialize the weights as random numbers between 1 and 0.

In the case of the single layer network. Our objective with the training data is to try to find the set of weights that minimizes the misclassification error. We implement the calculation to find the gradient of the weights in order to update them.

The learning rate then determines how much, in the direction of the gradient, our next guess for the updated weights will be.

For the case of the multilayer, we also initialize the number of neurons in our hidden layers (there are 2 hidden layers). In a similar fashion to the single layer, we implement the calculation to find the updated weights for the first and the second layer and the learning rate determines how far we move in the direction of the calculated gradient for our next guess.

For both cases, we iterate for the number of epochs that we determine heuristically.

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.**
2. **Present the results, including images, of your example of a non-generalizable backprop solution. Explain why this example is non-generalizable.**
3. **Give a final discussion and conclusion where you explain the differences between the performances of the different classifiers. Pros and cons etc.**

The advantage of using the kNN algorithm over neural networks is that it is a fast, understandable implementation of an algorithm that performs well compared to the neural network in this context and with these datasets. However, the drawback of the kNN algorithm is that all the data is stored in order to make a prediction, which in the case of very large dataset is very computationally intensive. This algorithm also faces the issue that finding an optimal k large enough for the calculations to be quick and generalizable, but small enough that it performs well is computationally intensive using cross-validation because we have to store all the data in order to compare. We should also keep in mind of the bias introduced by our decision of how to solve draws in the implementation. Small k will lead to models that are too complex even though it can perform well on non-linear problems. The kNN algorithm seems to be an easy go-to algorithm for exploratory models that can quickly yield insight into the data to be analyzed/classified.

Neural networks require to be trained and can thus be more cumbersome to use compared to the kNN algorithm. The structure of a neural net can however easily be ported to another problem set and has thus more flexibility compared to the kNN algorithm. If we compare the neural networks between each other, we find that multilayer neural nets perform better than single-layer ones. Even if the single layer can theoretically solve any problem with enough neurons, it is much more computationally expensive to do so than to add more depth to the network. We should also add here that the weights of neural networks and the attributes are difficult to interpret which makes them difficult to share and explain.

1. **Do you think there is something that can improve the results? Pre-processing, algorithm-wise etc.**Algorithm wise, we have explained already that for the kNN algorithm, we could improve the resolution of draws by dynamically implementing a new k for the point that has a draw that would either be one more or one less in order to solve the draw and use an odd k.

For the neural networks, the multilayer network could be improved by randomly switching off certain neurons in order for the network as a whole to not rely too much on a single attribute. This has the added advantage to yield a more generalizable solution. The increase in the number of hidden layers could also be implemented, but could add uneccasary complexity.

In terms of pre-processing and for complex data like the hand-written digits, one could imagine a way of handing missing data and outliers statistically by implementing an probabilistic approach to the labeling. In the example of handwritten digits, if we did not know the true labels of the digits, we could use one-hot encoding and probabilities to assign labels to unlabeled data.

1. **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**](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.**