TBMI26 – Computer Assignment Report Supervised Learning

Deadline – March 16 2020

Authors: Karolin Stork & Viktor Gustafsson

Revision 1

Q1. Give an overview of the data from a machine learning perspective. Consider if you need linear or non-linear classifiers etc.
When data is linearly separable, a linear neural network will be able to separate the data points. If the data is not linearly separable, as in the x-or problem for instance, a non-linear activation function is needed. As can be seen in the figures in Q7, the data in dataset 1 is linearly separable. Dataset 2, 3 and 4 are not linearly separable.

Q2. 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
Down-sampling the image size to 8x8 results in a dataset and classifiers that are less sensitive to small variety in the data. This simply because smaller details are not represented in the data.

Q3. Give a short summary of how you implemented the kNN algorithm.

We check the closest labels (euclidean distances, cut off based on given k) and save all that are present in uniqueClosestLabels. If the size of this array is 1 (ie if there is only one class present) we set the sample to that class. However, if the size is larger than 1 we compile a "dictionary" (containers.map) were we have class as the key and append the distances as values. We check the number of distances stored for each label and find out what the largest amount of distances stored is. The keys that have that size of their corresponding array with values are saved in equalLabels. If equalLabels only consists of 1 class, we set the sample to that class (majority voting). If there is a draw we sum the distances for each label involved in the draw and the label with the smallest summed distance is set as the label for that sample.

Q4. Explain how you handle draws in kNN, e.g. with two classes (k = 2)?
We sum the euclidean distance for the labels that are equal in representation within the cluster. The label with the smallest summed distance is classified. We utilize the previously mentioned containers.map, where we found labels with the same max amount represented. These label's distances are summed from the dictionary and compared as specified.

Q5. Explain how you selected the best k for each dataset using cross validation. Include the accuracy and images of your results for each dataset.
Cross validation is used to find k by observing the increase in accuracy for both training and testing, and choosing the k that minimizes the test error. Cross validation trains the data by having 10 bins and altering which bin is taken out for testing and training on the 9 others. This is done 10 times and an average of the accuracy is calculated. The result can be seen in the figures below.

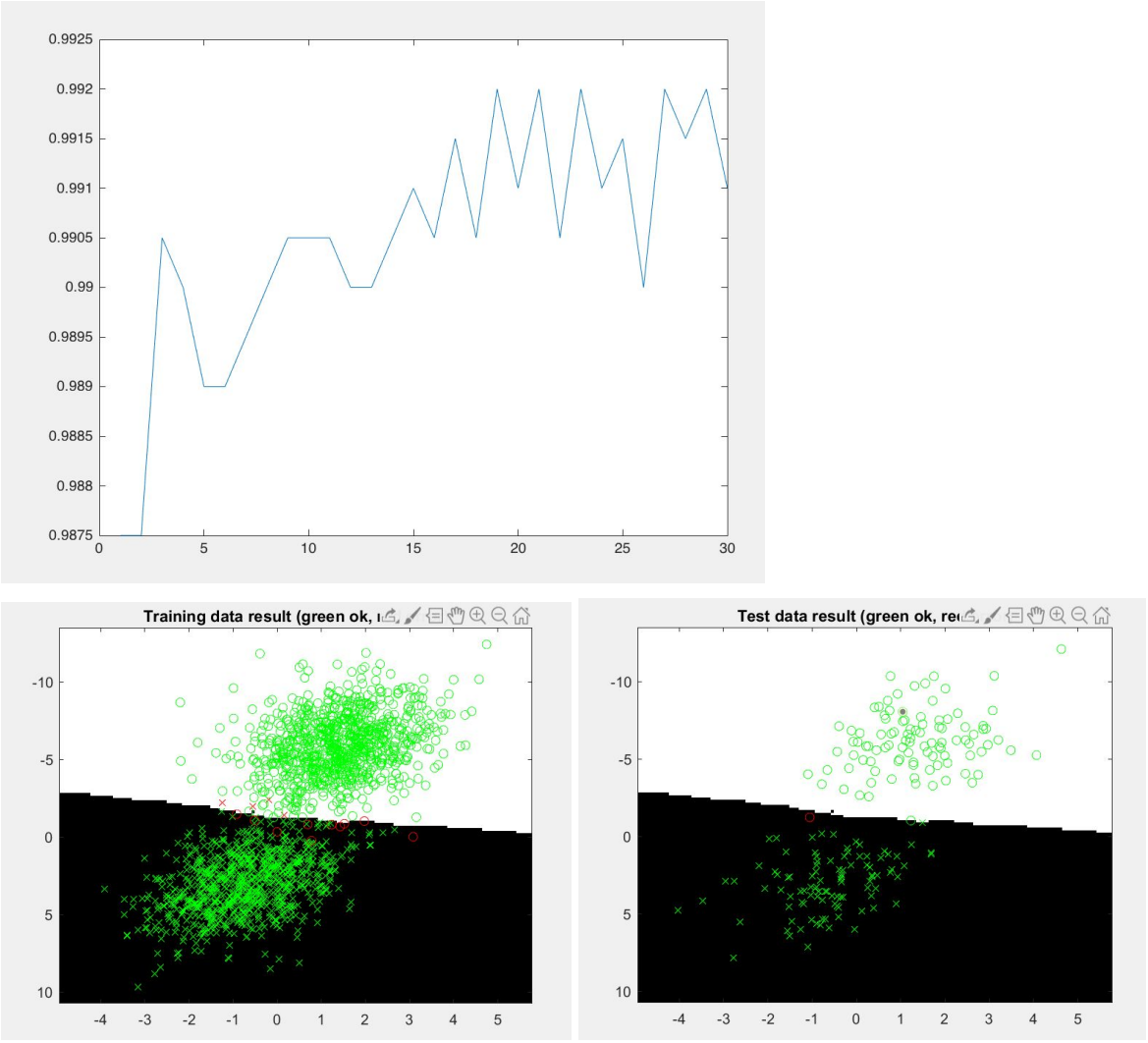
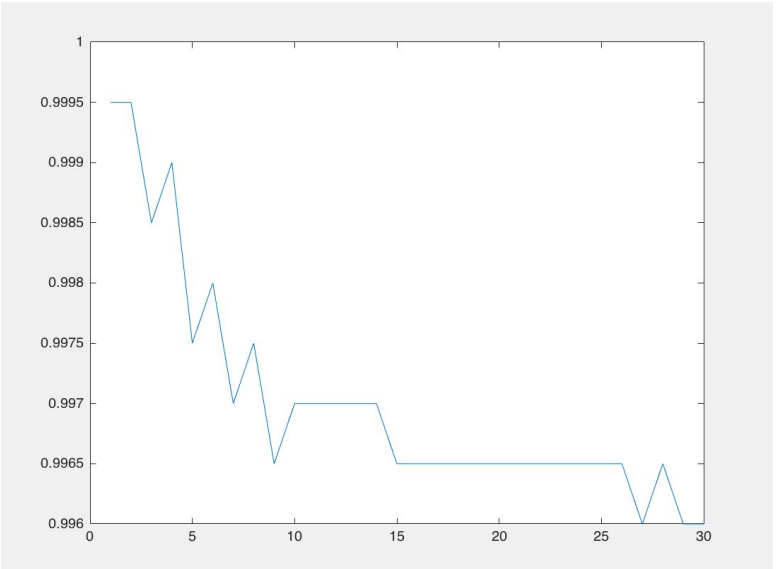


Fig 1. Accuracy plotted against the number of nearest neighbors (k) for dataset 1. k=20 gives the highest accuracy 99.20% with the classification results plotted above.



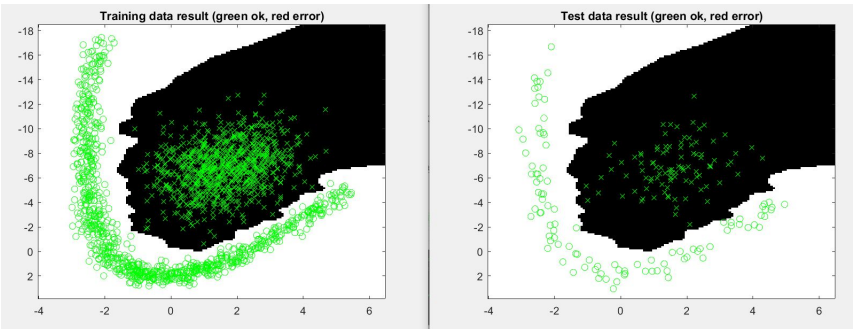
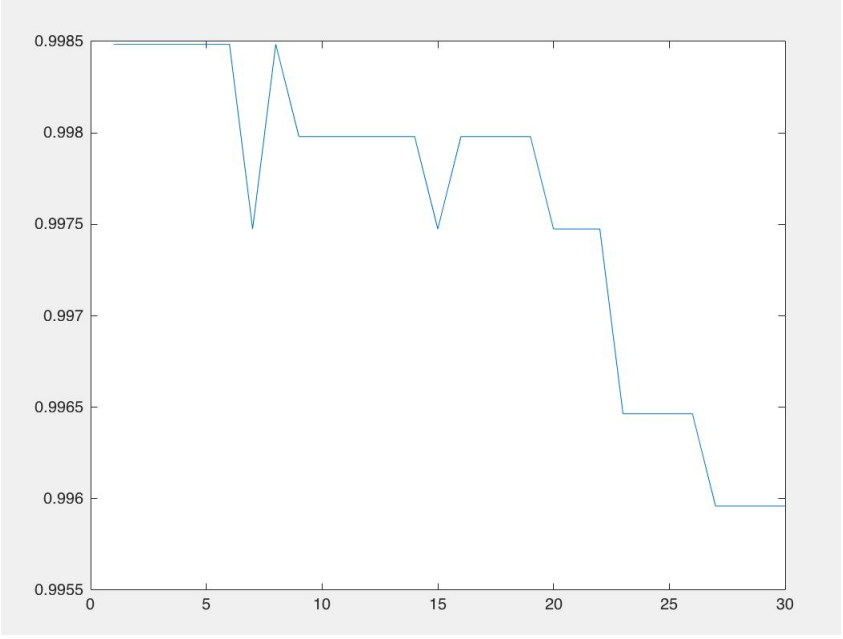


Fig 2. Accuracy plotted against the number of nearest neighbors (k) for dataset 2. k=1 gives the highest accuracy 99.95% with the classification results plotted above.



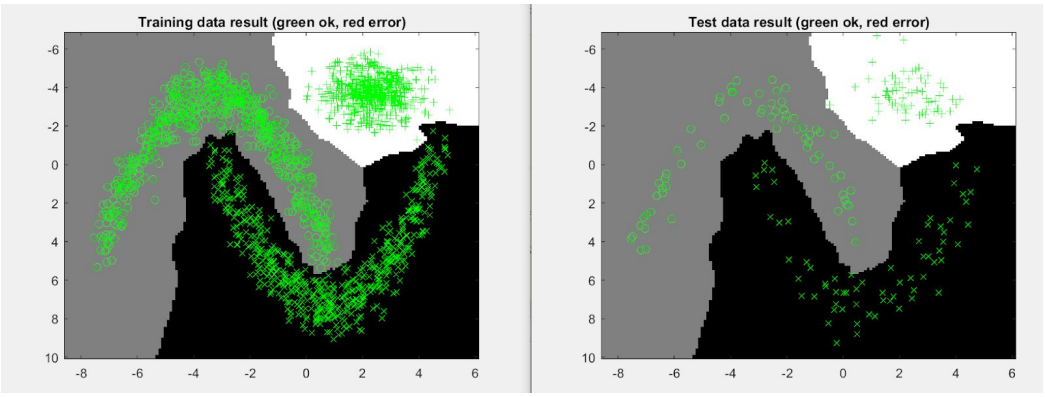
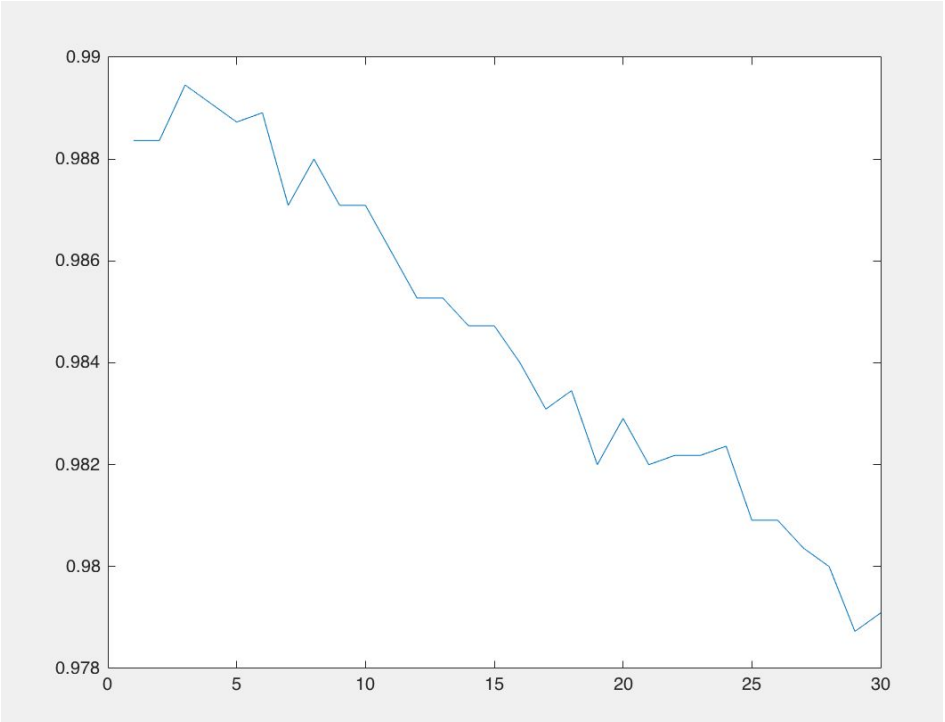


Fig 3. Accuracy plotted against the number of nearest neighbors (k) for dataset 3. k=1 gives the highest accuracy 99.85% with the classification results plotted above.



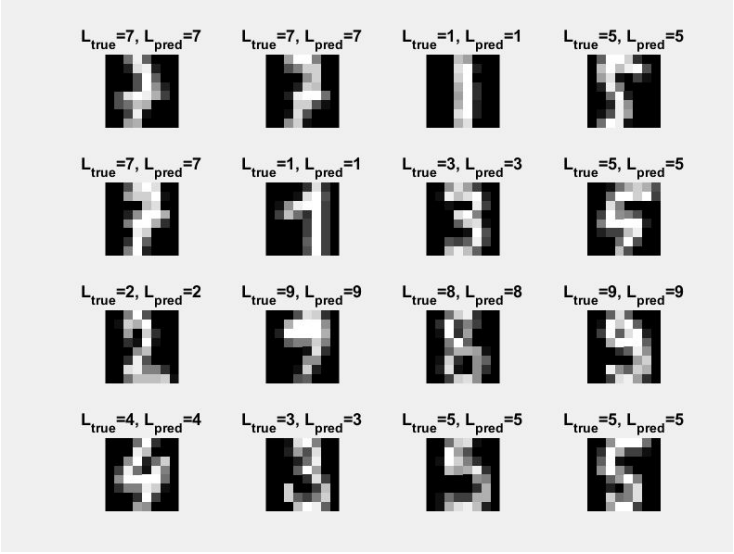


Fig 4. Accuracy plotted against the number of nearest neighbors (k) for dataset 4. k=3 gives the highest accuracy 98.95% with the classification results plotted above.

Q6. Give a short summary of your backprop network implementations (single + multi). You do not need to derive the update rules.

The single layers back prop algorithm updates each weight wij by 2*(aj-yj)*xi.

```
grad_w = 2 * (YTrain - DTrain)' * XTrain / NTrain
```

The multilayer backward propagation updates each weight Vjk in the final layer in the same way as above, but with the output of the hidden layer instead of xi. The update for each Vjk becomes 2*(ak-yk)*aj.

```
grad_v = 2 * H' * (YTrain - DTrain) / NTrain
```

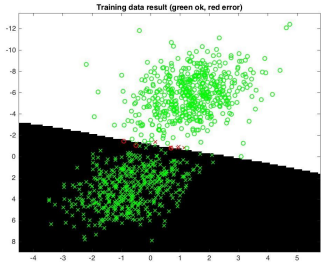
The hidden layer is a bit more complicated, as all nodes in the final layer are affected by each hidden layer weight. This is because each hidden layer node is connected to each node in the final layer. Thus the update rule for the weight wij becomes sum_k(2*(ak-yk)*wj) *aj *tanhprim(zj) * xi.

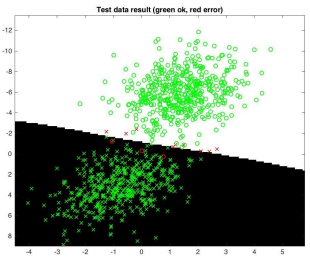
```
grad_w = (2 * (Vout(1:end-1,:) * (YTrain - DTrain)'.*(tanhprim(A))')*XTrain / NTrain)'
```

Q7. 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 foreach dataset, including parameters etc.
We used the multilayer network for each dataset as this network gave the best result for each dataset except dataset 1, which is linearly separable. We chose to use multilayer for dataset 1 as well for simplicity's sake.

We tried different settings to achieve the required accuracy. Epochs were increased when we saw that the test/train error graph hadn't plateaued yet. We increased the learning rate to achieve faster training, and increased the number of hidden neurons to increase the complexity of the model.

	Learning rate	Hidden	Epochs	Test Accuracy
Dataset 1	0.001	7	10000	98.8%
Dataset 2	0.005	7	7000	99.8%
Dataset 3	0.005	7	12000	99.1%
Dataset 4	0.005	50	5000	97.0%





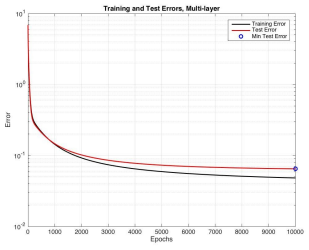


Fig 1. Results from dataset 1

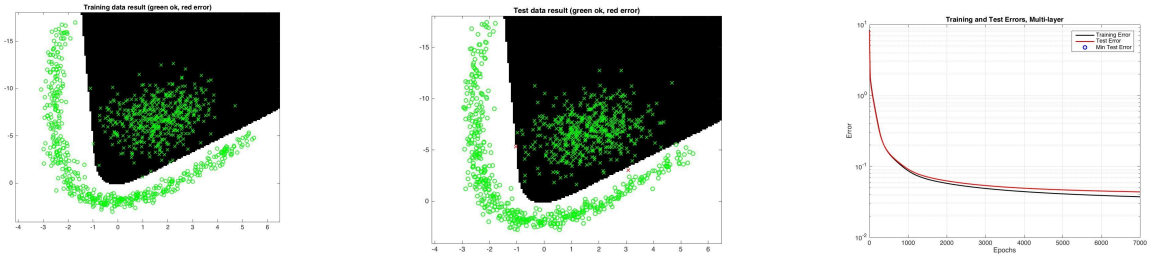


Fig 2. Results from dataset 2

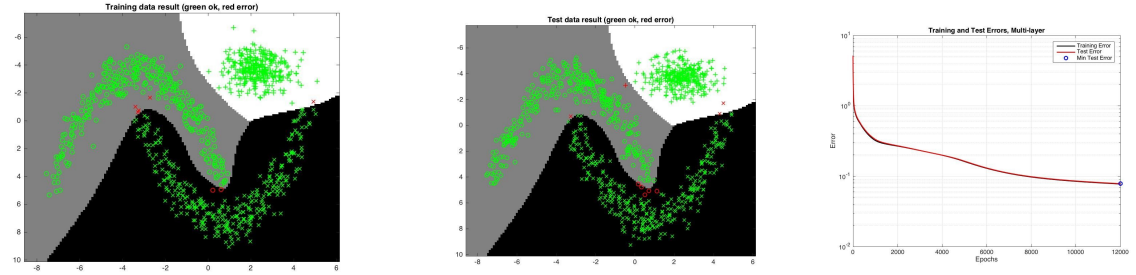


Fig 3. Results from dataset 3

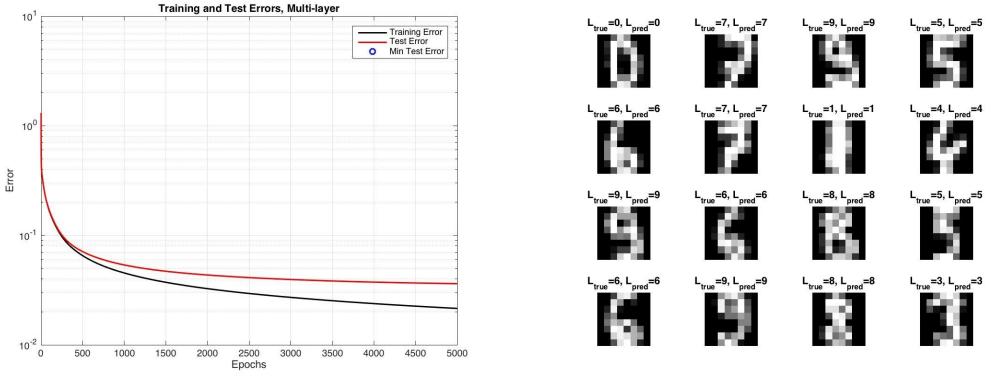
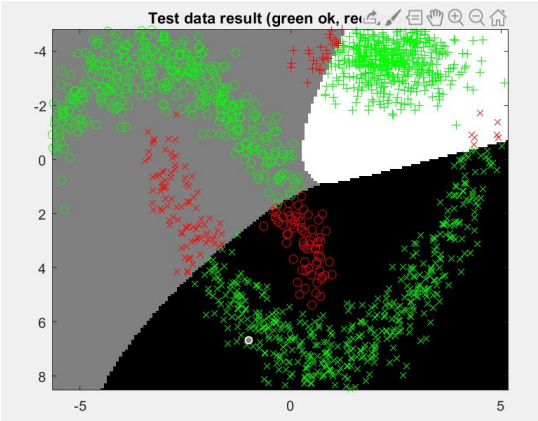
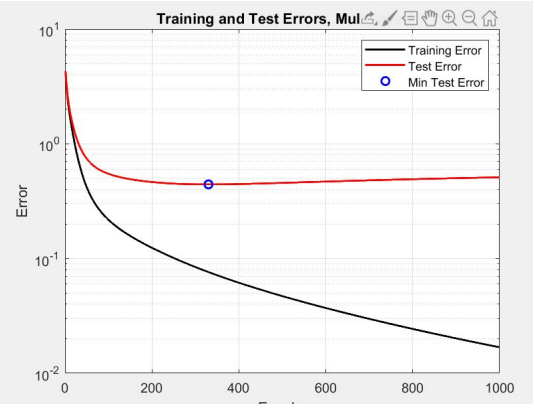


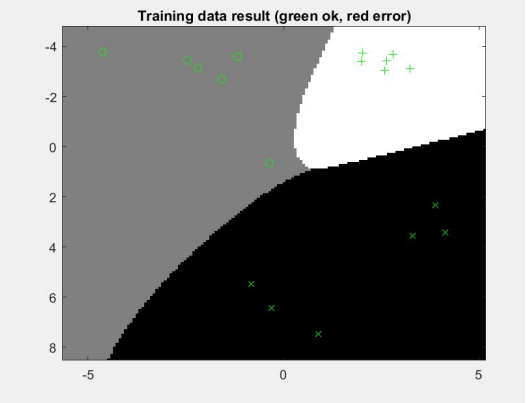
Fig 4. Results from dataset 4

Q8. Present the results, including images, of your example of a non-generalizable backprop solution. Explain why this example is non-generalizable.

Dataset	3
Number of hidden	7
Number of epochs	20
Learning rate	0.005
Training test ratio	1:99
Accuracy	57%







The training data is not representative for the complete data distribution, ie the clusters. Therefore the backpropagation can't fit the model to the correct clusters and performs poorly on the test data.

Q9. Give a final discussion and conclusion where you explain the differences between the performances of the different classifiers. Pros and cons etc.

The two classes in dataset 1 are close in proximity with a slight overlap and the best k for that dataset was therefore a lot higher than for dataset 2 where there was a bigger margin between the two classes. KNN performed the best for all datasets. The drawback in KNN is that you need more training data to be able to predict accurately and if the dataset includes outliers or is asymmetric in its distribution the KNN algorithm will probably perform worse than the networks. Compared to single layer networks, in KNN the classes don't have to be

linearly separable. There is also the issue with storing all training data in KNN in comparison to the networks as we don't have any parameters that "learn" in KNN and the whole process depends on having training data to compare to. The algorithm for KNN is also easier to understand than backpropagation.

Q10. Do you think there is something that can improve the results? Pre-processing, algorithm-wise etc.

One common pre-processing step in machine learning is data-normalization because we want the input data to be of a common scale (zero mean and unit variance). By doing this the model will be able to learn faster, i.e faster convergence. Otherwise the gradient can oscillate and take longer to reach optimum.