TTT4275

Classification of the Iris flower and handwritten numbers

Felicia Rahim

Version: 1.1 Date: 30.04.2018

Summary

The course TTT4275 Estimation, Detection and Classification gives an introduction to the basic techniques for estimation, detection and classification with focus on information and communications technology (ICT) applications and signal processing aspects.[4] As a part of the course the students are given a choice of choosing a project related to either estimation, detection or classification. This rapport focuses only on classification, where two given tasks related to the subject are solved.

In the first task a linear classifier is designed and evaluated. Its purpose is to classify different species off the flower Iris from each other. Further, the importance of four features in each species are analyzed. The linear classifier was not error free as some elements of the data set was misclassified. However, the error rate was small and therefore insignificant. The majority of the data sets was classified correctly.

The second task creates classifiers that classify handwritten number using 10 000 images. A NN-classifier, clustering-based classifier and a KNN-classifier were designed for this task. To compare the classifiers, their performance and processing time were evaluated. As a result when choosing a classifier based on the best performance, the NN-classifier should be used. However, if the processing time is the biggest concern, the clustering-based classifier is considered the best choice.

Contents

1 Introduction **1**

2 Task description and theory **2**
2.1 The project descriptions 2
2.2 Task descriptions 2
2.3 Theory 3

3 Implementation **8**
3.1 Iris 8
3.2 Handwritten numbers from 0-9 8

4 Results **11**
4.1 Iris 11
4.2 Handwritten numbers 14

5 Conclusion **19**

References **20**

Appendix A **21**

Appendix B **27**

Appendix C **32**

Appendix D **33**

1 Introduction

Artificial intelligence and machine learning are increasingly popular topics. Even though the ideas and the algorithms has been around for many years, it is not until now with the high quality technology it is possible to realize such intelligence. Examples of this are self-driven cars and face-recognition. This report the has its focus the algorithm associated with machine learning.

"Machine learning is a method of data analysis that automates analytical model building. It is a branch of artificial intelligence based on the idea that systems can learn from data, identify patterns and make decisions with minimal human intervention."[3] In this project machine learning will be implemented in the form of classification.

The goal of this project is to design classifiers that are capable to find the label of the inputs, i.e classify them. The project is separated into two parts. The first part focusing on unsupervised learning, where the goal is to classify three different variants of the flower Iris. The second task is focused on supervised learning, where a data set with images of handwritten number from 0-9 will be classified.

In Chapter 2 the project descriptions will be presented, in addition the theory needed to understand the task. In Chapter 3 one can read about how the classifiers will be implemented. Furthermore, the results will be presented in Chapter 4. At the end, the conclusion is given in Chapter 5.

2 Task description and theory

This chapter starts by describing the selection of projects and giving the reason for choosing the specific project task. Further on a more detailed description of the project is presented. At the end the theory needed to solve the tasks is presented. At the end of the chapter one can read a short description of the confusion matrix and the error rate.

2.1 The project descriptions

This project is given in the course TTT4275 Estimation, Detection and Classification at NTNU. Each part has their own projects that could be chosen. Estimation had two projects to choose from, investigating a maximum likelihood estimator (MLE) or a best linear unbiased estimator (BLUE). The estimators has different interest areas. The project in Detection is to design a Neyman-Pearson detector to determine the spectrum availability. In Classification there was two projects that were divided into two parts. The first part was equal for both the projects, and consisted in designing a classifier to classify three variants of the Iris flower. The goal of the first project was to design a classifier that could classify a full set of twelve vowel classes. The second project consists of designing three different classifiers to classify handwritten numbers from 0-9.

The classification project last mentioned was the most appealing and was therefore chosen. The reason for this is that it is directly connected to machine learning, which is indeed an interesting topic. To have knowledge about machine learning and more over artificial intelligence is valuable for the future.

2.2 Task descriptions

Iris

The purpose of the first task is to design and train a classifier with a given database. The classifier receives an input that contains information about three different species of Iris, and based on this manages to classify the three species. This problem is close to be linear separable, and the focus is therefore to design a linear classifier to conquer the task. Further the importance of the features in the database with respect to linear separability will be analyzed.

Handwritten numbers

In this part of the project the task is to make an algorithm that can take an input with 10 000 images of handwritten numbers, and give the numbers written as output. The first part of the task is to design a Nearest Neighbor (NN)-classifier using the whole training set as reference. The second part is to design a NN-classifier based on clustering. The last part is to make a KNN-classifier, with K references from the whole training set. For comparison of performance and processing time, the confusion matrix and the error rate for each classifier will be calculated.

2.3 Theory

The theory used to accomplish the project is presented under. First it will be given a short description of the Iris flower. The linear classifier will further be defined. Next will three variants of the NN-classifier be described.

The Iris Flower

The Iris flower has several variants, including three called Setosa, Versicolor and Virginica. The three mentioned species can be discriminated by the different lengths and widths of the petal and sepal. It is these four features which will be used to classify the three species.

The linear classifier

The three variants of Iris are close to linear separable, thus a linear classifier will be designed. In reality more or less all practical problems are non-separable, thus a linear classifier will usually give a worse performance than a nonlinear classifier. However a linear classifier is sometimes preferred due to its simplicity.

When training a classifier, there are some important factors that should be taken into consideration for the data set. The data set should be large enough for the classifier to classify the input data. The data set should also be representative, i.e. all variants of input/class pairs should be present in a large enough scale, and unbiased. Furthermore, the input data should be separated into a training set and test set.

The training set should be applied several times such that the algorithm gives better performance (correctness) on the training set for each iteration. This is called supervised learning. The test set is necessary for the evaluation of the true performance.

The linear discriminant classifier for C $>$ 2 classes and with D columns can be written in a compact matrix form by

$$g_i = Wx_i + w_0, i = 1, ..., C \tag{1}$$

where g and w_0 are vectors of dimension C (number of classes), W is a C×D matrix. The classifier is unbiased, thus w_0 can be ignored and g rewritten to $g = Wx$, where g is the output vector that corresponds to a single input x. The output should match a target vector with binary values $\{0, 1\}$, which can be done with a sigmoid function defined by

$$g_k = \text{sigmoid}(x_k) = \frac{1}{1 + e^{-z_k}}, \tag{2}$$

where $z_k = Wx_k$ for k sample in the training set or test set. The total matrix W is trained as one single entity, i.e. the total training set for all classes is used simultaneously.

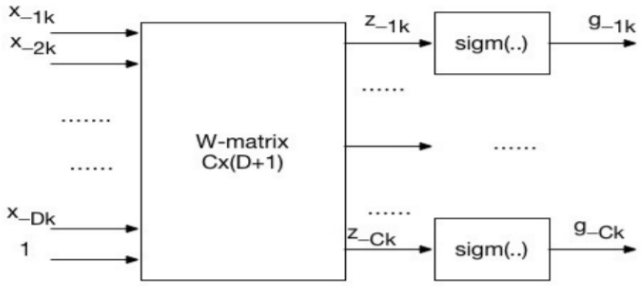

Figure 1: A linear classifier with binary targets [1]

The structure of the linear classifier is shown in Figure 1. The classifier receives (D+1) vectors simultaneously. The vectors are input training samples from each data set when training, or they are the test samples when testing the classifier. At the output of the classifier a new vector is returned.

The classifier should be trained until the minimum amount of errors when classifying have been found. Minimum Square Error (MSE) finds average errors made by the classifier and is defined by

$$MSE = \frac{1}{2}\sum_{k=1}^{N}(g_k - t_k)^T(g_k - t_k), \tag{3}$$

where t_k are so called targets that represents class labels. The target can be of real numbers or it can have vectorial form $t_k = [0,...,0,1,0,...,0]^T$. The goal is to predict the target with g, such that when subtracting t from g in Eq. (3), g should be close or equal to t to get a small MSE as possible.

Furthermore, the classifier should be trained with minimal errors. It is therefore necessary to take the gradient of the MSE. Thus, rewriting Eq. (3) and taking the gradient gives

$$\nabla_W MSE = \sum_{k=1}^{N}[(g_k - t_k) \circ g_k \circ (1 - g_k)]x_k^T, \tag{4}$$

where $g_k \circ (1 - g_k)$ means elementwise multiplication. In update the classifier whilst it is training, the following formula can be used

$$W(m) = W(m-1) - \alpha\nabla_w MSE, \tag{5}$$

where m gives the iteration number. When choosing the step factor α, its important that its not to small or to big. [1]

The NN-classifier

A NN-classifier is a type of classifier where the learning will be unsupervised. This signifies that the data given as input is not classified beforehand nor labeled, and the machine has

to operate without any other information or earlier training as guidance. This is in unlike the linear classifier which is supervised. In other words this classifier is more designed to sort than to learn. For the NN-classifier to classify data it finds the nearest neighbors of the vectors in the test set to the training set.

$$d(x, ref_{ik}) = (x - \mu_{ik})^T (x - \mu_{ik}) \tag{6}$$

In this project the Euclidian distance, given in Eq.(6), is found between each test vector and each training vector. The test and training vector with the smallest separating distance, will be set to be the most equal vectors. As a result the number presented in the test vector will be classified as the same number in the training vector. Other distance approaches are also possible, such as the Mahalanobis and the Chebychev distance. In addition there exist different variants of this classifier, which two of the classifiers described below are examples of.

Clustering-based classifier

A variant of the NN-classifier is the classifier based on clustering. With this method the training set is first of all sorted into classes and each class are divided into clusters with smaller set of references. These references are used in the same manner as described in section above. The references are sorted by finding the Euclidian distance between the vectors, and sorting the closest vectors in each cluster.

Since the references are fewer in this method, the processing time will be shorter than by using the whole training set as references, i.e it will be fewer operations. But, as a consequence the performance will be worse.

The KNN-classifier

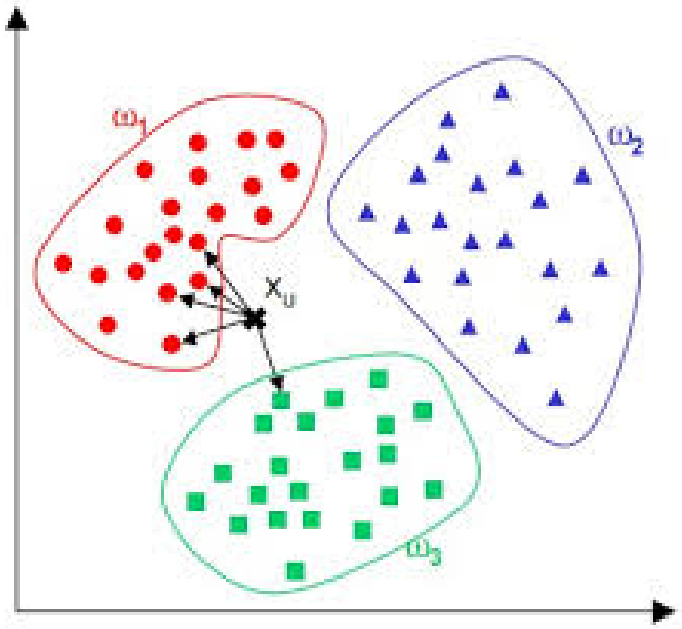

Figure 2: An illustration of the KNN-classifier.

Another variant of the NN-classifier is the KNN-classifier. In this method K numbers of training vectors are found to be the K nearest references to the test vectors. In Figure 2 the classifier is illustrated with three classes. Class 1, ω_1 in red, class 2, ω_2 in blue and class 3, ω_3 in green. The black cross, X_u is the test vector that is going to be classified. Here K = 5, and five distances are being measured. The training vectors in the class with the shortest distance to the test vector, will be set as the vector most equal to the test.

Confusion Matrix and Error Rate

When evaluating the performance of a classifier, one can use the confusion matrix and calculate the error rate of the classifier to visualize the performance.

Table 1: A confusion matrix.

n = 165	**Classified labels**		
True labels	**50**	10	**Class 1**
	5	**100**	**Class 2**
	Class 1	**Class 2**	

In the confusion matrix, the rows are the true labels and the columns are the predicted or classified labels. Shown in Table 1 the confusion matrix in this case has two classes with in total 165 labels that has been classified. 50 labels from class 1 has been classified as a label from class 1, and five labels from the same class has been classified as a label from class 2,

i.e misclassified. As goes for the labels from class 2. The ideal classifier is capable to classify every label correctly, which gives a diagonal confusion matrix.

The error rate is defined as

$$\text{error rate} = \frac{\text{Misclassified labels}}{\text{Total number of labels}} \tag{7}$$

An ideal classifier will always have a error rate equal to 0.

3 Implementation

In this chapter the practical choices in the algorithms are substantiated. Further details of the algorithm can be read in the comments in Appendix A and B. To solve this project the algorithm is written i MATLAB.

3.1 Iris

The algorithm can be found in Appendix A under "Algorithm for IRIS" and "Algorithm for classifying IRIS with reduced features".

Data set

Three data sets are produced, consisting of 50 samples of each of the three variants/classes. The data sets have a dimension of 50×4, where the four columns contains information about respectively sepal length, sepal width, petal length and petal width.

Table 2: Samples from the data sets used in the training set and test set.

Type of set	**Samples**
Training set	1:30
Test set	31:50

Before training the linear classifier, the data sets were divided into a training set and a test set, as illustrated in the Table 2. Thus, the classifier received a training set with 90 samples, 30 sample from each data set, and afterwards a test set with 60 samples, 20 samples from each data set.

Table 3: Different samples from the data sets used in the training set and test set.

Type of set	**Samples**
Training set	21:50
Test set	1:20

To examine whether the confusion matrix and the error rate changes when varying the samples in the training set and test set, the data set was redivided as illustrated in Table 3.

There were also created four histograms whose purpose visualize the features and compare them to the equivalent features in the other classes. By examining the histograms one could tell which features overlapped the most. One and one feature was removed from the whole data set based on how big the overlap was within each feature. This showed how removing the data would affect the performance of the classifier.

3.2 Handwritten numbers from 0-9

The dataset – MNIST

The classifier is used on the dataset, MNIST, designed by NIST, contains 70 000 images of handwritten numbers from 0-9. To classify the numbers, 10 000 images, which are further on

called the test set, are compared with remaining 60 000 images, called the training set. The numbers are written by 250 persons, which makes the data set diverse. It is worth mentioning that the 28×28-images has been centered, scaled and been processed to be presented as a 1×784 vector. Included in the database are two vectors with the numbers written as integers, respectively to the training and test set.

The NN-classifier

The algorithm can be found under "Algorithm for the NN-classifier" in Appendix B.

The first part was to divide the vectors used. The data set contains a large number of vectors. To make the algorithm for the NN-classifier more efficient, both the training vectors and the test vectors are divided into groups of 1000 vectors. The algorithm is based on classifying 1000 test vectors in each for loop, with 60 groups of training vectors.

The second part is to classify the test vectors.

For finding the Euclidian distance the Matlab command "pdist2()" was used, on line 14. This command is repetitive through this part of the project. This command takes in two matrices, X with the size MX×N and the Y with the size MYxN. The output is two matrices, one MX×MY matrix with all the distances, and one matrix with the same dimensions with the indices from X pointing to the training vector with the smallest distance to the test vectors.

When saving all the indices in "index_all(i,:)", in line 18, the variable index is scaled according to which group of the 60 groups of training vectors is operating at that for loop. This is to have a correspondence with the vector "trainlab", which holds all the numbers as integers. After all the test vectors has been classified, the number presented in the images are fetched from the vector, "trainlab".

The third part is to plot misclassified and classified images.

To observe the performance, ten misclassified and 20 classified images were printed. A test group of 50 people were asked to classify eight of these images to see if the human brain would agree or disagree with the algorithm. The participants were allowed to freely classify the number they thought is was, and was informed that is was a number between 0-9. Four of the images were misclassified images and the other half correctly classified. Both clearly written and poorly written numbers were presented for the test group.

The clustering-based classifier

The algorithm can be found under "Algorithm for the clustering-based classifier" in Appendix B.

When designing a clustering-based classifier the algorithm will first do clustering on 6000 training vectors for each class into M = 64. For doing the clustering, a command called "kmeans(A, M)" is used. This command does the process presented in Chapter 2.3. A is the matrix to be clustered.

The KNN-classifier

The algorithm can be found under "Algorithm for the KNN-classifier" in Appendix B. As explained in Chapter 2.3 a K has to be chosen to find the K nearest neighbor references. The K is set to be 7. For designing the KNN classifier the MATLAB command "knnsearch()" is used. K = 7. The command finds the seven nearest neighbors to all the test vectors.

4 Results

In this chapter the results from the two main tasks will be presented and discussed. First the results from design of the linear classifier for the Iris data will be presented. Following, the results for the classifiers for the handwritten numbers. At the end of this chapter the algorithm will be compared with the human brain.

4.1 Iris

In order to evaluate how well the classifier manages to classify the three species of Iris, a confusion matrix was created and the error rate calculated.

Table 4: Confusion matrix for the training set (samples 1-30).

True label \ Classified label	Class 1	Class 2	Class 3
Class 1	**30**	0	0
Class 2	0	**28**	2
Class 3	0	1	**29**

Table 5: Confusion matrix for the test set (samples 31-50).

Test label \ Classified label	Class 1	Class 2	Class 3
Class 1	**20**	0	0
Class 2	0	**18**	2
Class 3	0	1	**19**

Table 6: Confusion matrix for the training set (samples 21-50).

True label \ Classified label	Class 1	Class 2	Class 3
Class 1	**30**	0	0
Class 2	0	**27**	3
Class 3	0	2	**28**

Table 7: Confusion matrix for the test set (samples 1-20).

True label \ Classified label	Class 1	Class 2	Class 3
Class 1	**20**	0	0
Class 2	0	**20**	0
Class 3	0	0	**20**

Table 4 and 5 shows the confusion matrix for the total training sets and test sets using different samples. In Table 4 one can see that the classifier did not classify all the samples correctly, where two samples were misclassified to class 3 rather than correctly to class 2, and 1 sample was misclassified to class 2 rather than class 3. This is also seen in the confusion matrix for the test set in Table 5 where two samples are misclassified similarly to the training set. Thus, the classifier have not been trained properly most likely because the data set isn't large enough for the classifier to generalize the input data.

When using different samples from the data set in the training set and test set, the confusion matrices in Table 6 indicates two more misclassified samples compared the previous classified training set. However, the confusion matrix for the test set in Table 7 shows that the classifier should work optimal. The reason why the training set has not been classified properly might be because the features in these samples are too similar. Thus, the classifier is having a harder time separating the species from each other.

Table 8: Error rate of the total training set and test set with samples from the data set specified.

Data set	**Error Rate** [%]
Training set [1:30]	3.33
Test set [31:50]	3.33
Training set [21:50]	5.56
Test set [1:20]	0

Table 8 shows the error rate for the total training set and test set. As expected from the confusion matrices, the error rate is higher for the training set using the last thirty samples from the data set. The test set has a lower error rate for the first twenty samples compared to the last twenty samples. This proves that the thirty first samples are easier to classify compared to the thirty last ones.

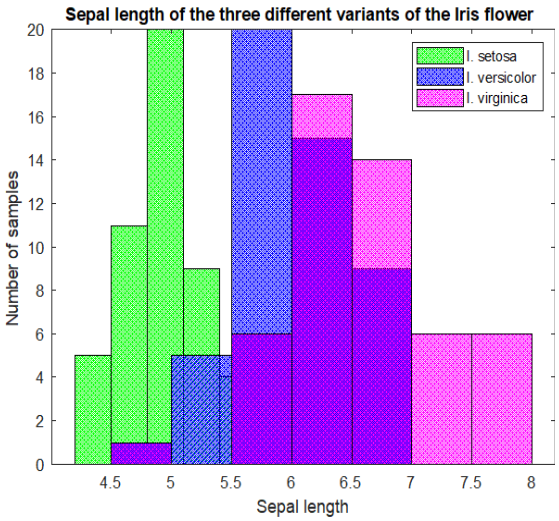

Figure 3: Sepal length

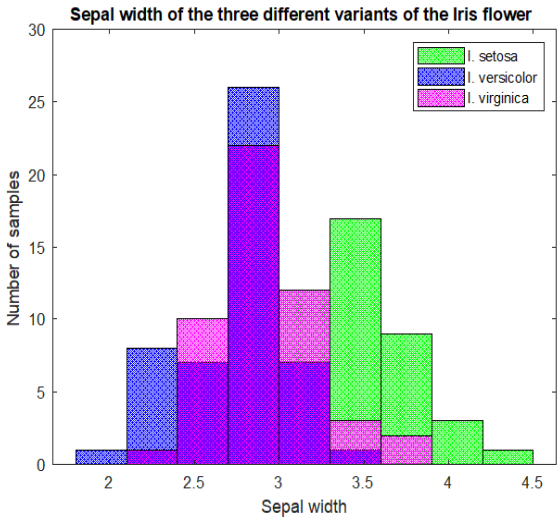

Figure 4: Sepal width

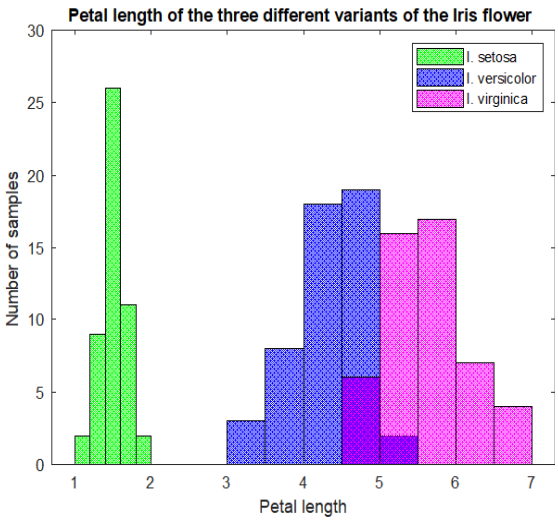

Figure 5: Petal length

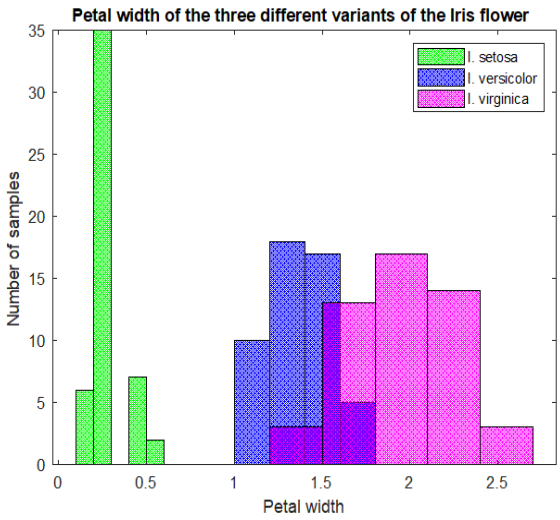

Figure 6: Petal width

Figure 3, 4, 5 and 6 shows histograms for each feature in the three classes. It appears that for Figure 4 the bins overlap the most, meaning that these features are harder to distinguish and are less linearly separable. The bins for the petal lengths overlaps the least and are therefore easier to distinguish from each other, as shown in Figure 5. Thus, it is easier to distinguish the three species from each other for these features. Since the features overlap the most in the sepal length, it is desired to examine whether removing particular features causes the linear classifier to classify better.

Table 9: Error rate of the total training set and test set with removed features.

Data set	**Error Rate** [%]
Training set (three features)	3.33
Test set (three features)	5.00
Training set (two features)	13.33
Test set (two features)	10.00
Training set (one feature)	20.00
Test set (one feature)	13.33

Table 9 shows the error rate of the training sets and test sets with fewer features in each set. The confusion matrices for these sets can be found in Appendix D. When analyzing the error rates, it is noticeable that it increases when there are fewer features in the sets. Also the confusion matrices have more misclassified samples now compared to before. This is reasonable because there are fewer data to train the classifier with, such that the classifier can't differentiate the species apart from each other as optimal as before. With less information to train the system with, the less it will learn and will therefore make more mistakes. This makes sense as the golden rule for learning is "...the more data the better...", despite the data being approximately linearly separable. [1]

4.2 Handwritten numbers

To be able to compare the three different variants of the NN-classifier, was, as mentioned, the confusion matrix plotted and the error rate was calculated. These parameters are presented below.

Table 10: Confusion Matrix for the NN-classifier.

	Classified labels										
True labels	**973**	1	1	0	0	1	3	1	0	0	**0**
	0	**1129**	3	0	1	1	1	0	0	0	**1**
	7	6	**992**	5	1	0	2	16	3	0	**2**
	0	1	2	**970**	1	19	0	7	7	3	**3**
	0	7	0	0	**944**	0	3	5	1	22	**4**
	1	1	0	12	2	**860**	5	1	6	4	**5**
	4	2	0	0	3	5	**944**	0	0	0	**6**
	0	14	6	2	4	0	0	**992**	0	10	**7**
	6	1	3	14	5	13	3	4	**920**	5	**8**
	2	5	1	6	10	5	1	11	1	**967**	**9**
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	

Table 10 presents the confusion matrix for the NN-classifier. The error rate was calculated to be 3.09% which corresponds to 310 misclassified images. As seen in the table the largest numbers lies on the diagonal, which is a desirable result. As for the processing time, the whole algorithm took 295.21 seconds. This is a relative number, and will later be compared with the other classifiers.

Table 11: Confusion Matrix for the classifier based on clustering.

	Classified labels										
True labels	**961**	1	4	1	0	3	6	1	2	1	**0**
	0	**1132**	2	0	0	0	0	0	0	1	**1**
	11	7	**980**	8	0	0	4	9	12	1	**2**
	0	0	4	**935**	0	22	0	11	25	13	**3**
	2	5	3	0	**920**	0	6	2	3	41	**4**
	5	1	1	16	2	**843**	6	3	7	8	**5**
	8	3	1	0	1	3	**939**	1	2	0	**6**
	1	18	9	0	7	1	0	**965**	1	26	**7**
	6	0	7	17	3	20	1	5	**909**	6	**8**
	6	5	3	6	32	5	0	17	6	**929**	**9**
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	

Table 11 presents the confusion matrix for the clustering-based classifier. Also in this case the matrix has the largest numbers on its diagonal. The error rate is 4.44%, which is an error of 444 images. The classification process took 51.89 seconds.

Table 12: Confusion Matrix for the KNN classifier.

True labels \ Classified labels	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0	**971**	1	1	0	0	0	5	1	1	0
1	0	**1127**	1	3	0	0	4	0	0	0
2	16	11	**965**	8	2	0	6	18	5	1
3	1	1	5	**949**	1	22	3	10	8	10
4	0	11	0	0	**916**	0	7	5	2	41
5	7	3	0	15	2	**838**	11	2	7	7
6	7	3	0	0	3	3	**942**	0	0	0
7	0	28	7	0	3	0	0	**972**	1	17
8	9	2	5	26	5	25	7	9	**870**	16
9	8	5	5	5	24	3	0	13	6	**940**

The confusion matrix for the last classifier, the KNN-classifier, is presented in Table 12. As one can see also this classifier had the right behavior. It has a error rate on 5.10 % and a processing time at 309.94 seconds.

Table 13: The error rate for the three classifiers.

Classifier	**Error Rate**[%]
NN	3.09
Clustering	4.44
KNN	5.10

Table 14: The processing time for the three classifiers.

Classifier	**Processing Time [s]**
NN	295.21
Clustering	51.89
KNN	309.94

From the results in Table 13 and 14 one can observe that the NN-classifier has the lowest error rate, and the cluster-based classifier has the shortest processing time, but the worst error rate. The behavior of the last mentioned classifier was as expected. It was fastest due to not using the whole training set at reference.

In the case of the KNN-classifier the processing time was longer from than predicted. Before implementation it was assumed that the KNN-classifier had the shortest processing time, due to fewer references. To test if the processing time could be reduced, the test set was divided into ten parts, operating with 1000 test vectors at the time. This did not give any significant difference. Despite the failed optimization attempt, there might by other approaches for improvements.

Human against algorithm

As the report introduced, AI is becoming increasingly popular. Further reading will show in how big range the algorithm and the human brain agree. By using the algorithm, "Code for plotting the misclassified and classified numbers" in Appendix B, images was printed. The images was classified by a group of 50 anonymous people.

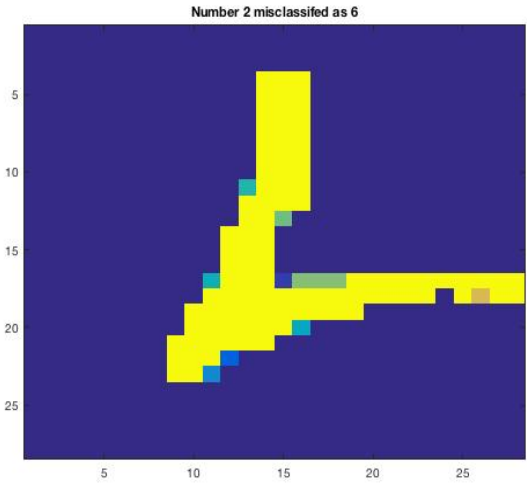

Figure 7: The number 2 misclassified as 6.

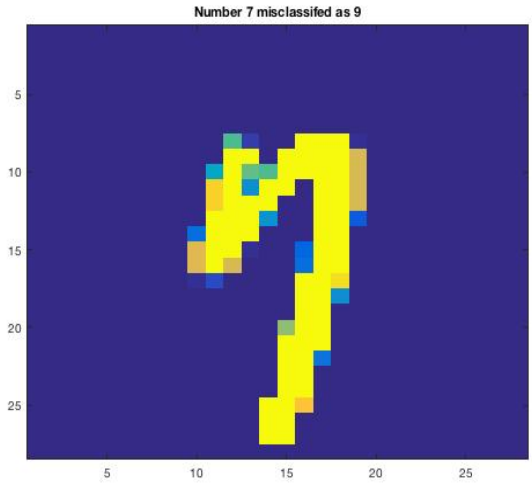

Figure 8: The number 7 misclassified as 9.

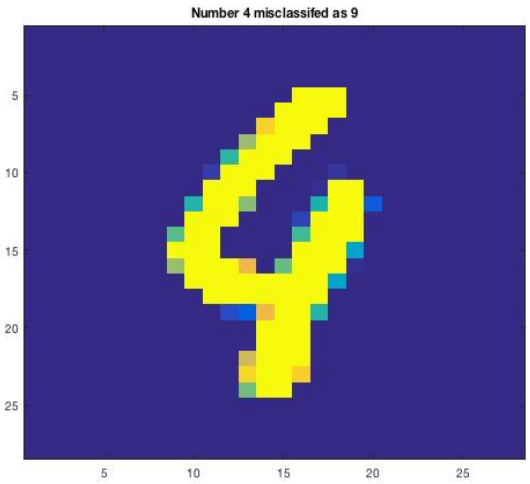

Figure 9: The number 4 misclassified as 9.

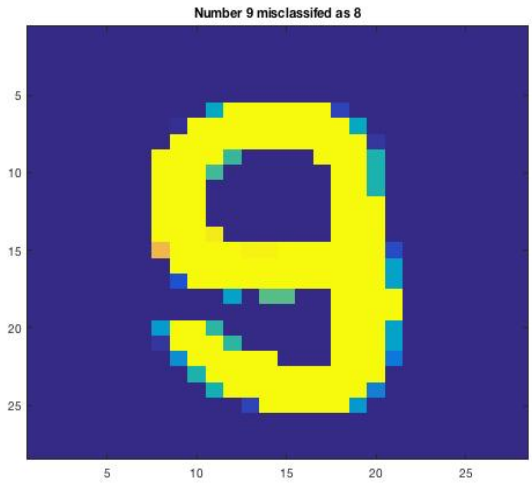

Figure 10: The number 9 misclassified as 8.

The images above is a selection of the images that got misclassified of the NN-classifier. Figure 7 and 8 show the numbers 2 and 7, respectively, and got classified as 6 and 9. Figure 9 and 10 show the numbers 4 and 9. They got classified as 9 and 8.

When asking the test group they disagree with the algorithm on every image. Except from

the image in Figure 8 where 44.9% of the participants answered that the number was 1, the big majority classified the numbers correctly.

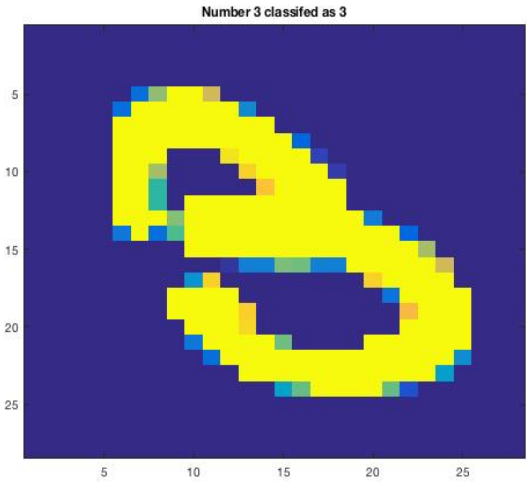

Figure 11: The number 3 classified as 3.

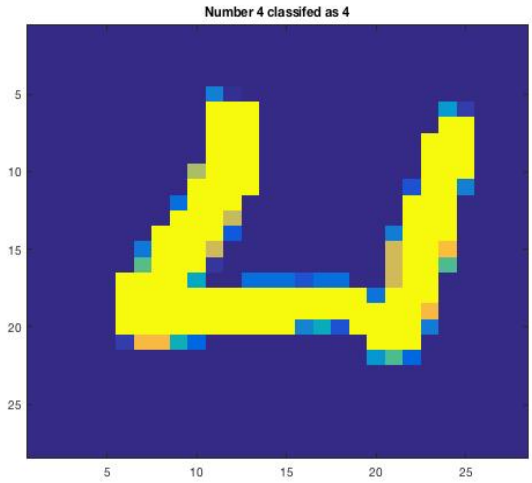

Figure 12: The number 4 classified as 4.

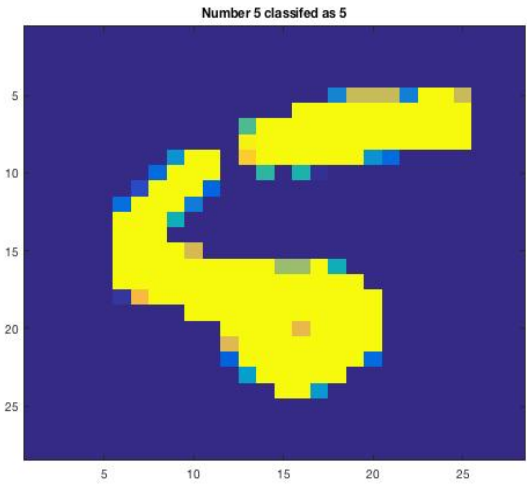

Figure 13: The number 5 classified as 5.

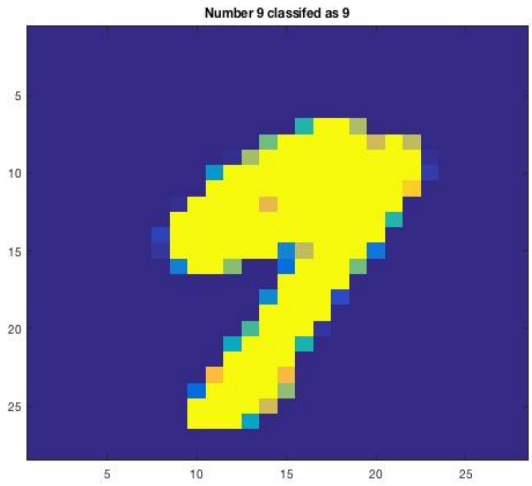

Figure 14: The number 9 classified as 9.

The handwritten numbers in the Figure 11, 12, 13 and 14 are numbers classified correctly. These, as one can see, are more difficult to classify.

When asking people to classify the numbers, 90% of the participants classified the numbers in Figure 11 and 14 correctly, where 76 % managed to correctly classify the number in Figure 12. On the finale image in Figure 13 52% classified it as a 6, and the algorithm conquered the humans. The complete result of the survey is to be found in Appendix C. The results are presented in a confusion matrix inspired matrix.

In total, the general human brain managed to correctly classify six out of eight images, in comparison to the algorithm that managed four out of eight, in this context. In three cases the human and the algorithm agreed.

The human brain can, as shown, easily classify the clear written images. The reason is the earlier experience with handwritten numbers and the cognitive ability of perception. In comparison, the algorithm has only approximately 6000 images for each handwritten number to compare with. The poorly written numbers were more difficult for the test group to classify. Therefore the misclassification of the machine is understandable. When that being said, the classifier did manage to classify some difficult cases too, such as the image in Figure 13.

5 Conclusion

A linear classifier was designed and trained with three data sets. The data sets contained 50 samples each, with information about four features for three species of the Iris flower. The classifier was trained with a training set and was afterwards tested with a test set. In order to have a minimal amount of misclassified species, the training went on until the minimum MSE was found. Thus, the three species were only misclassified with a error rate on 3.33% and 5.56% with the training set and 3.33% and 0% with the test set. This was also illustrated with confusion matrices for both the training set and test set. The linear classifier was not error free because the data set wasn't large enough and features weren't all linearly separable. When classifier was trained using only one feature, the error rate increased to 20.00% for the training set and 13.33% for the test set. Hence, the classifier is having a hard time generalizing the input data when the data isn't large enough. The species will in that case be poorly classified.

After designing and testing three variants of NN-classifiers, the result is that what makes the best classifier best is relative. The first classifier, the NN-classifier has the lowest error rate on 3.10%, but has a longer processing time. On the contrary the cluster-based classifier six times faster than the NN-classifier, but has a lower error rate. The KNN-classifier designed in this project has the worst error rate and the longest processing time of all of the three. It was attended to approve the classifier, but with no success. To examine if humans would agree in the classification done by the algorithm a survey was arranged. 50 people participated to classify eight images by only observing the images. The results showed that in three out of eight cases the human and the algorithm agreed.

References

[1] Johnsen, M. H. (2017) *Classification.* Trondheim, Norway. NTNU.

[2] February 2, 2018.*Iris flower data set.* Wikipedia. Accessed: April 28, 2018. Available at: `https://en.wikipedia.org/wiki/Iris_flower_data_set`

[3] *Machine Learning – What It Is and Why It Matters.* SAS Institute Inc. Accessed: April 29, 2018. Available at: `https://www.sas.com/en_us/insights/analytics/machine-learning.html.`

[4] *TTT4275 - Estimation, Detection and Classification* NTNU. Accesses: April 29, 2018. Available at: `https://www.ntnu.edu/studies/courses/TTT4275#tab=omEmnet`

[5] Castle, N. *Supervised vs. Unsupervised Machine Learning* Accesses: April 30, 2018. Available at: `https://www.datascience.com/blog/supervised-and-unsupervised-machine-learning-algorithms`

Appendix A

Algorithm for IRIS

```
1
2  x1 = load('class_1'); %opens data set for class 1
3  x2 = load('class_2');
4  x3 = load('class_3');
5
6  x1tr = x1(1:30, :); %training set of samples 1 to 30 (total of 30
       samples) for class 1, dimension of 30x4
7  x1te = x1(31:50, :); %test set of samples 31 to 50 (total of 20
       samples) for class 1, dimension of 20x4
8  x2tr = x2(1:30, :);
9  x2te = x2(31:50, :);
10 x3tr = x3(1:30, :);
11 x3te = x3(31:50, :);
12
13 x_all = [x1tr; x2tr; x3tr]; %Put all the training sets into one
       matrix, dimension 90x4
14
15 xclass = [x_all, ones(90,1)]; %Input to the linear classifier with
       a dimension 90x5,
16 %where the fifth column is ones to make the input linear.
17
18 W = abs(randn(3,5)/10); %Starts the classifier as a Cx(D+1) (3x5)
       matrix. Dividing the matrix with a down scaling it to make it
       less complicated.
19
20 threshold = 100; %Set a threshold such that the classifier will
       continuously classify until the error is
21 % small.
22
23 while threshold > 1/1000
24 z = xclass * W.' ; %formula for z.
25 g = 1./(1+exp(-z)); %formula for the sigmoid
26 MSE = 0;
27
28 for i = 1:90 %for loop that runs through all of the samples in the
       total training set
29     if i<31
30         t = [1,0,0]; %three classes: target for class 1
31     elseif i<61
32         t=[0,1,0]; %three classes: target for class 2
33     else
34         t=[0,0,1]; %three classes: target for class 3
35     end
```

```
36      MSE = MSE + [(g(i,:)-t).*g(i,:).*(1-g(i,:))]'*(xclass(i,:)); %
            formula for the gradient of the MSE
37  end
38
39  W = W - 0.01*MSE; %W is updated by the gradiant of MSE, and keep it
        in continuously training in a while loop.
40
41  threshold = abs( sum(sum(MSE)))/15; %Stops the while loop when the
        error(MSE) is small.
42  end
43
44
45  [mx, ind] = max(g'); %Decision rule. The maximum of g is found in
        order to find the numbers that are closest to 1.
46  class1 = ind(1:30); class2 = ind(31:60); class3 = ind(61:90); %
        Finding labels from the decision rule.
47
48  %Making a confusion matrix, dimension 3x3, for the training set
49  confmat = [sum(class1(:)==1), sum(class1(:)==2), sum(class1(:)==3)
        ;...
50        sum(class2(:)==1), sum(class2(:)==2), sum(class2(:)==3);...
51        sum(class3(:)==1), sum(class3(:)==2), sum(class3(:)==3)];
52  disp(confmat);
53
54
55  %Error rate of the training set. How many of the input samples were
        classified
56  %correctly or not.
57  amountOfWrong = sum(confmat(1, 2:3)) + confmat(2,1) + confmat(2,3)+
        sum(confmat(3, 1:2)); %Finding where in the confusion matrix
        there are errors
58  ERR_training = 100*(amountOfWrong/90);
59  fprintf('This is the error rate for the training set: %.3f‘\n',
        ERR_training);
60
61  %This part runs the test set through the classifier. This is not
        run through the while loop because we do not wish to train the
        classifier, we only want to run the test set through the
        classifier to see if it gives reasonable outputs
62  xtest = [x1te; x2te; x3te]; %Making a test vector that goes as the
        input at the linear classifier.
63  xclasstest = [xtest, ones(60,1)]; %60x5 dimension (60 samples in
        the test set, from three classes)
64  ztest = xclasstest*transpose(W);
65  gtest = 1./(1+exp(-ztest));
66
67  gtestmax = max(gtest'); %Decision rule for the test set
```

```
68 [mxtest, indtest] = max(gtest');
69 class1test = indtest(1:20); class2test = indtest(21:40); class3test
       = indtest(41:60);
70
71 %Making a confusion matrix, dimension 3x3, for the test set:
72 confmattest = [sum(class1test(:)==1), sum(class1test(:)==2), sum(
       class1test(:)==3);...
73     sum(class2test(:)==1), sum(class2test(:)==2), sum(class2test(:)
           ==3);...
74     sum(class3test(:)==1), sum(class3test(:)==2), sum(class3test(:)
           ==3)];
75
76 %Error rate of the test set.
77 antallfeiltest = sum(confmattest(1, 2:3)) + confmattest(2,1) +
       confmattest(2,3)+ sum(confmattest(3, 1:2));
78 ERR_test = 100*(antallfeiltest/60);
79 fprintf('This is the error rate for the test set: %.3f\n', ERR_test
       );
```

Algorithm for classifying IRIS with reduced features

```
1  x1 = load('class_1');
2  x2 = load('class_2');
3  x3 = load('class_3');
4
5  %Producing histograms for each feature and class:
6   for i = 1:4
7   figure(i)
8   histogram(x1(:, i), 'FaceColor', 'g'); %Make histogram for the
       features in class 1
9   hold on;
10  histogram(x2(:, i),'FaceColor', 'b'); %Histogram for the features
       in class 2
11  histogram(x3(:, i),'FaceColor', 'm'); %Histogram for the features
       in class 3
12  hold off;
13
14  if i ==1 %Puts all the samples with same features in one figure
15     title('Sepal length of the three different variants of the Iris
           flower');
16  elseif i==2
17      title('Sepal width of the three different variants of the Iris
            flower');
18  elseif i==3
19      title('Petal length of the three different variants of the
           Iris flower');
20  else
```

```
21      title('Petal width of the three different variants of the Iris
            flower');
22  end
23  end
24
25  %Looking at the histograms, it appears which features overlap the
        most.
26
27  x1tr1 = [x1(1:30, 1), x1(1:30, 3), x1(1:30, 4)]; %New training set
        with one feature removed
28  x1te1 = [x1(31:50, 1), x1(31:50, 3), x1(31:50, 4)]; %New test set
        with one feature removed
29  x2tr1 = [x2(1:30, 1), x2(1:30, 3), x2(1:30, 4)];
30  x2te1 = [x2(31:50, 1), x2(31:50, 3), x2(31:50, 4)];
31  x3tr1 = [x3(1:30, 1), x3(1:30, 3), x3(1:30, 4)];
32  x3te1 = [x3(31:50, 1), x3(31:50, 3), x3(31:50, 4)];
33
34  x_all1 = [x1tr1; x2tr1; x3tr1]; %90x3 dimension, new training
        vector with one feature removed. There are now only three
        features in this vector.
35  xclass1 = [x_all1, ones(90,1)]; %dimension 90x4, where the fourth
        column is the ones at the input of the classifier.
36
37  x1tr2 = [x1(1:30, 1), x1(1:30, 3)]; %New training set with only two
         features
38  x1te2 = [x1(31:50, 1), x1(31:50, 3)]; %New test set with only two
        features
39  x2tr2 = [x2(1:30, 1), x2(1:30, 3)];
40  x2te2 = [x2(31:50, 1), x2(31:50, 3)];
41  x3tr2 = [x3(1:30, 1), x3(1:30, 3)];
42  x3te2 = [x3(31:50, 1), x3(31:50, 3)];
43
44  x_all2 = [x1tr2; x2tr2; x3tr2]; %90x2 dimension, new training
        vector consisting of only two features.
45  xclass2 = [x_all2, ones(90,1)]; %dimension 90x3, where the third
        column is ones at the input of the classifier.
46
47  x1tr3 = [x1(1:30, 1)]; %New training set with only one feature
48  x1te3 = [x1(31:50, 1)]; %New test set with only one feature
49  x2tr3 = [x2(1:30, 1)];
50  x2te3 = [x2(31:50, 1)];
51  x3tr3 = [x3(1:30, 1)];
52  x3te3 = [x3(31:50, 1)];
53
54  x_all3 = [x1tr3; x2tr3; x3tr3]; %90x1 dimension, new training
        vector consisting of only one feature.
```

```
55 xclass3 = [x_all3, ones(90,1)]; %dimension 90x2, where the second
       column is ones at the input of the classifier.
56
57 W_new = abs(randn(3,2)/10); %Finding the parameters W, MSE and g
       again for the training and test sets with fewer features.
58 threshold_new = 100;
59
60 while threshold_new > 1/1000
61 z_new = xclass3 * W_new.' ;
62 g_new = 1./(1+exp(-z_new));
63 MSE_new = 0;
64
65 for i = 1:90
66     if i<31
67         t = [1,0,0];
68     elseif i<61
69         t=[0,1,0];
70     else
71         t=[0,0,1];
72     end
73     MSE_new = MSE_new + [(g_new(i,:)-t).*g_new(i,:).*(1-g_new(i,:))
           ]'*(xclass3(i,:));
74 end
75 W_new = W_new - 0.01*MSE_new;
76 threshold_new =abs( sum(sum(MSE_new)))/15; %stops when the error is
       small
77 end
78
79
80 [mx_new, ind_new] = max(g_new');
81 class1_new = ind_new(1:30); class2_new = ind_new(31:60); class3_new
       = ind_new(61:90);
82
83 confmat_new = [sum(class1_new(:)==1), sum(class1_new(:)==2), sum(
       class1_new(:)==3);...
84      sum(class2_new(:)==1), sum(class2_new(:)==2), sum(class2_new(:)
           ==3);...
85      sum(class3_new(:)==1), sum(class3_new(:)==2), sum(class3_new(:)
           ==3)]; %Finding the confusion matrix of the new training
           sets where some of the features have been removed.
86
87 antallfeiltraining_new = sum(confmat_new(1, 2:3)) + confmat_new
       (2,1) + confmat_new(2,3)+ sum(confmat_new(3, 1:2));
88 ERR_training_new = 100*(antallfeiltraining_new/90); %Error rate for
       the training set where some of the features have been removed.
89 fprintf('Error rate for the training set: %.3f`\n',
       ERR_training_new);
```

```
90
91  %Testing
92  xtest_new = [x1te3; x2te3; x3te3]; %New vector for test set with
        some removed features.
93  xclasstest_new = [xtest_new, ones(60,1)]; %Input vector at the
        linear classifier with some removes features.
94  ztest_new = xclasstest_new*transpose(W_new);
95  gtest_new = 1./(1+exp(-ztest_new));
96  [mxtest_new, indtest_new] = max(gtest_new');
97  class1test_new = indtest_new(1:20); class2test_new = indtest_new
        (21:40); class3test_new = indtest_new(41:60);
98
99  confmattest_new = [sum(class1test_new(:)==1), sum(class1test_new(:)
        ==2), sum(class1test_new(:)==3);...
100      sum(class2test_new(:)==1), sum(class2test_new(:)==2), sum(
             class2test_new(:)==3);...
101      sum(class3test_new(:)==1), sum(class3test_new(:)==2), sum(
             class3test_new(:)==3)]; %Finding confusion matrix for the
             test set with removed features.
102
103 antallfeiltest_new = sum(confmattest_new(1, 2:3)) + confmattest_new
        (2,1) + confmattest_new(2,3)+ sum(confmattest_new(3, 1:2));
104 ERR_test_new = 100*(antallfeiltest_new/60); %Finding the error rate
        for the new test set with some removed features.
105 fprintf('Error rate for the test set: %.3f\n', ERR_test_new);
```

Appendix B

Algorithm for the NN-classifier

```
load('data_all.mat');

z_all=ones(60, 1000);
index_all = zeros(60,1000);
estInd=zeros(1000,1);
estLab=zeros(10000,1);


for j = 1:10
    v=testv((1+(j-1)*1000):(j*1000), :); %Take the 1000*j test
       vectors.

    for i = 1:60
        w=trainv((1+(i-1)*1000):(i*1000), :); %Take the 1000*i
           training vectors.
        z = pdist2(w, v, 'euclidean');%Finds the euclidiaen
           distance between the two matrices operating.
        % The v is 1000x784 and w is 60x784.
      [distanse_mindre,index] = min(z); %Finds the minimum distance
          of the 1000 distances for each 1000 test
      z_all (i, :) = transpose(distanse_mindre); %Works as a memory
         , saving the 60 mimimum distances from each group i.
      index_all (i,:) = transpose(index+(i-1)*1000);%Same as z_all,
          but saves the indices(1-1000) where the minimum distances
          is located in z_all.
    end

[minAll, index_ny] = min(z_all); %Finds the minimum distance of the
    60 distances,and the corresponding indices (1-60).
for k = 1:1000
estInd(k) = index_all(index_ny(k), k); %Saves the index
   corresponding to each test vector where the shortest distance
   for all the training vectors is located.
end
estLab(1+(j-1)*1000:(j*1000))= trainlab(estInd); %Fetches all
   numbers in trainlab on the estInd rows, and saves it is estLab.


end
estLab = transpose(estLab); %Transposes estLab to have the same
   dimension as testlab.

%Confusion matrix
```

```
32 Conf_Mat = confusionmat(testlab,estLab); %Makes the confusion
       matrix.
33 disp(Conf_Mat)
34
35 %Error rate
36 nn_antallfeil = num_test; %Sets numbers of errors equal the total
       number of test vectors before the for loop.
37 %Iterates through the whole matrix. If the row number equals the
       column
38 %number, i.e the diagonal, the number of errors reduces with the
       number on
39 %the diagonal, which is correctly classified.
40 for i = 1:10
41     for j = 1:10
42         if i == j
43             nn_antallfeil = nn_antallfeil - Conf_Mat(i,j);
44         end
45     end
46 end
47
48 nn_errorrate = nn_antallfeil/num_test; %Error rate is calculated by
       dividing numbers of errors with the total number of test
       vectors.
```

Code for plotting the misclassified and classified numbers

```
1  %Plotting some of the misclassified pictures
2
3  x = zeros(28,28); %Making the matrix that will hold the image.
4  n =10; %Sets number of images equal 10.
5
6      for i = 1 : num_test %Want to search through the total test
           data set.
7          if testlab(i) ~= estLab(i) %If the number differ from the
               estimated number,
8              x(:)  = testv(i,:);  %x is set to the corresponding
                   test image.
9              n = n-1;
10             text = sprintf('Number %i misclassifed as %i', testlab
                   (i), estLab(i));
11         end
12         if n < 1 %Break when all n images has been printed.
13             break
14         end
15         figure(n)
16
17         image(transpose(x)); title(text); hold on %Print.
```

```
18      end
19      %% Here the correctly classified images are printed. The code
            is same, except the condition in the if loop.
20  n = 10;
21  m = 20;
22       for i = 1 : num_test
23          if testlab (i) == estLab (i)
24              x(:)  = testv(i,:);
25              m = m-1;
26              text = sprintf ('Number %i classifed as %i', testlab(i)
                    , estLab(i));
27          end
28          if m < 1
29              break
30          end
31          figure (n+m) %Since already n images has been printed, (n+m
                ) makes the figure function print another m images.
32
33          image(transpose(x)); title (text); hold on
34      end
```

Algorithm for the clustering-based classifier

```
1   C = 10;
2   M = 64;
3   z_clust_all = zeros(10, 10000); index_clust_all = zeros (10,10000);
4   estLab_clust=zeros(1,10000);
5
6       for k = 1:10
7           trainTemp = trainv(trainlab==(k-1),:); %Finds all indices
                in trainlab there the number is (k-1), and fill
                trainTemp with the vectors from trainv on the rows
                indicated by the indices first found. I.e, sorting by
                class.
8           [idxi, Ci] = kmeans(trainTemp, 64);%Clusters trainTemp with
                 64 clusters.
9           %Ci is a 64x784 matrix.
10
11          z_clust = pdist2(Ci, testv, 'euclidean'); %Finds the
                distance between the current class and all the 10 000
                test vectors.
12          [distanse_mindre_clust,index_clust] = min(z_clust); %Finds
                the shortest distance of the 64 distances.
13          %index_clust returns the row which contains the shortest
                distance between the 64 distances.
14          z_clust_all (k,:) = distanse_mindre_clust; %Saves the
                shortest distances from each class k. (10x10000)
```

```
15
16
17      end
18
19  [minAll_clust, index_ny_clust] = min(z_clust_all); %Finds the
        minimum distance. index_all_clust (1x10000) holds the row number
        in z_clust_all, which correspond to each class where the
        shortest distances of all is located.
20  estLab_clust = index_ny_clust - 1 ; %estLab_clust holds the class
        number. Since class 0 was locates at row 1, etc.
21
22  %Confusion Matrix
23  Conf_Mat_clust = confusionmat(testlab,estLab_clust);
24  disp(Conf_Mat_clust)
25
26  %Error rate
27  nn_antallfeil_clust = num_test;
28  for i = 1:10
29      for j = 1:10
30          if i == j
31              nn_antallfeil_clust = nn_antallfeil_clust -
                    Conf_Mat_clust (i,j);
32          end
33      end
34  end
35
36  nn_errorrate_clust = nn_antallfeil_clust/num_test; %0.0477
```

Algorithm for the KNN-classifier

```
1   %knn-classifier
2   K =7; %Wants 7 references.
3   %knnsearch(....) is a build-in function that finds the K nearest
        neighbors
4   %between training set and the test set with the euclidean distance.
5   [cIdx,cD] = knnsearch(trainv,testv,'K',K,'Distance','euclidean');
6   %cIdx is a 10000x7 matrix which holds the indices where the k
        nearest neighbors is located in trainv.
7   estLab_knn = trainlab (mode(transpose(cIdx))); %The number that
        occurs most frequently is found with mode() and the matrix cIdx
        is transposed to fit the dimension of trainlab.
8   %The nearest neighbor is fetched and the index is used to get the
        number in
9   %trainlab before placed in estLab_knn.
10
11  %Confusion matrix
12  Conf_Mat_knn = confusionmat(testlab,transpose(estLab_knn));
```

```
13 disp(Conf_Mat_knn)
14
15 %Error rate = 0.0510
16 knn_antallfeil = num_test;
17 for i = 1:10
18     for j = 1:10
19         if i == j
20             knn_antallfeil = knn_antallfeil - Conf_Mat_knn (i,j);
21         end
22     end
23 end
24
25 knn_errorrate = knn_antallfeil/num_test;
```

Appendix C

The results from the classification by humans

Table 15: The comparison human classification of images misclassified by the algorithm.

Figure	**True label**	**Classified by human**										**Classified by algorithm**
7	2	1	36	1	1	3	0	5	0	0	2	6
11	3	0	0	45	0	0	0	0	1	1	1	3
9	4	0	0	0	50	0	0	0	0	0	0	9
12	4	1	1	3	38	1	1	0	1	0	4	4
13	5	0	0	0	1	20	26	0	2	0	1	5
8	7	23	0	0	7	1	0	12	0	7	0	9
10	9	0	0	0	0	0	0	0	1	49	0	8
14	9	2	0	0	1	0	0	2	0	45	0	9
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**Other**	

Appendix D

Confusion matrices for training sets and test sets when features have been removed.

Table 16: Confusion matrix for the training set with three features.

True label \ Classified label	Class 1	Class 2	Class 3
Class 1	**30**	0	0
Class 2	0	**27**	3
Class 3	0	2	**28**

Table 17: Confusion matrix for the test set with three features.

Test label \ Classified label	Class 1	Class 2	Class 3
Class 1	**20**	0	0
Class 2	0	**20**	0
Class 3	0	1	**19**

Table 18: Confusion matrix for the training set with two features.

True label \ Classified label	Class 1	Class 2	Class 3
Class 1	**30**	0	0
Class 2	0	**26**	4
Class 3	0	2	**28**

Table 19: Confusion matrix for the test set with two features.

True label \ Classified label	Class 1	Class 2	Class 3
Class 1	**20**	0	0
Class 2	0	**19**	1
Class 3	0	1	**19**

Table 20: Confusion matrix for the training set with one feature.

True label \ Classified label	Class 1	Class 2	Class 3
Class 1	**30**	0	0
Class 2	0	**25**	5
Class 3	0	2	**28**

Table 21: Confusion matrix for the test set with one feature.

True label \ Classified label	Class 1	Class 2	Class 3
Class 1	**20**	0	0
Class 2	0	**19**	1
Class 3	0	1	**19**