Project report classification MNIST database

Leik Lima-Eriksen and Torbjørn Bratvold

April 2020

Abstract

This project report describes the process of classifying the hand written numbers in the MNIST database[5] using two basic K nearest neighbour algorithms (KNN) and one template based KNN using clusters. The two basic KNN used K = 1 and K = 4. Testing with 1000 hand written numbers The K = 4 KNN gave a lower error rate than the K = 1 KNN of 46% verses 51.6%. Both had big problems classifying allot of none "one" numbers as the number "one". The classification time being ≈ 1.4 seconds per image (on the computer used). The template based KNN performed exept for the initial clustering time (clustering can be done pre usage) much better for both error rate and classification time. The error rate was $\approx 4.8\%$ and the classifying time ≈ 0.14 seconds per image. We confirmed that there is a linear relationship between the number of images to be classified and the processing time for all three classifiers. For the template KNN there seems to be some correlation between badly written numbers and wrongly classified numbers. There also seems to be a factor of randomness as some of the numbers are easily identified by a human.

Contents

1 Introduction

The project described in this report has the goal of classifying the handwritten numbers in the MNIST database[5]. Three ”k-nearest neighbour” algorithms are used to classify the images. Each classifier will be evaluated by performance and processing time.

Moving from the analog to the digital age a lot of literature and information are still stored on handwritten pages. Manually converting these pages to a digital searchable format takes allot of time and human resources. Automating this process would thus be a great leap towards giving people all over the world access to these sources. While we in this project only look at classifying numbers it is still an important step towards achieving this. It also has its standalone uses e.g. for answer sheets and written exams were only the input numbers are of interest.

2 Theory

2.1 K-nearest neigbour

The k nearest neighbour (KNN) algorithm has a simple decision rule were the input data is compared to a set of reference data. Reference data closest to the input data is then used to classify the input. [1] In this implementation the reference and input data are n dimensional vectors were the euclidean distance between the input and a reference point is used to express similarity (smaller distance equals more similar).

The two first KNN algorithms used are quite straight forward. The first basic NN algorithm simply considers the single closest reference point to the input data for classification. While straight forward and easy to implement it is quite sensitive towards outliers. The second algorithm considers a number ”k” nearest reference points and uses a majority vote to classify the input. This gives more robustness towards outliers, but k must be carefully tuned as to large values will lead to local clusters being ignored.

The third and last uses a template version of the basic NN algorithm were $k = 1$, using clustered reference data for classification. Each reference class is arranged into a number ”M” clusters. Each cluster is then represented by the average vector point and used as reference points for the NN algorithm. While harder to implement it gives robustness towards outliers and makes sure local clusters are accounted for. It will also perform better with large reference and input data-sets as the clusters can be calculated pre input and will thus consider a much smaller amount of references per input.

2.2 K-means clustering

While not going quite in dept, basic information about the cluster algorithm will be provided as it is quite essential for the NN template classifier. To create a number of M clusters the clustering is initialized by spawning M points in the coordinate system. The algorithm then follows iterations of two steps.

1. All reference points are assigned to the closest spawned point and becomes a cluster.
2. The spawned points are then re-spawned to the center of its corresponding cluster

These two steps continue until convergence has been reached.

2.3 Processing time

For the two basic KNN algorithms each classification period uses the same amount of time and we should thus see a linear increase in processing time for the number of test images. We also expect the $K > 1$ algorithm to use a bit more time as it also need to do a majority vote.

For the template version the classification time should be significantly lower as it needs to consider fewer references than the two other versions. The clustering should take the most amount of time, but since it only needs to be done once the template version should have lower processing time for big test sets. We should also here see a linear increase in processing time as the number for tests increases and it will have a base processing time equal to the clustering time.

3 Implementation

The classifiers were implemented in python. To run the code you need to download the MNIST data-set from [5], gunzip the files and put them into a directory named "mnist_datasets". The directory must be in the same directory as the "mnist.py" file.

In figure 1 and 2 we see the flowcharts of the basic KNN and the template based KNN

For clustering we used the function "KMeans" found in the sklearn library[2]. It works by using the algorithm in 2.2. For calculating the euclidean distance between the reference and test points the function "scipy.spatial.distance.euclidean()" from the scipy library was used[4]. It calculates the distance using equation 1. Here d is the distance between the two vectors u_i and v_i.

$$d = \sqrt{\Sigma|(u_i - v_i)|^2} \tag{1}$$

We also used the numpy library for its arrays and fast operations [3]

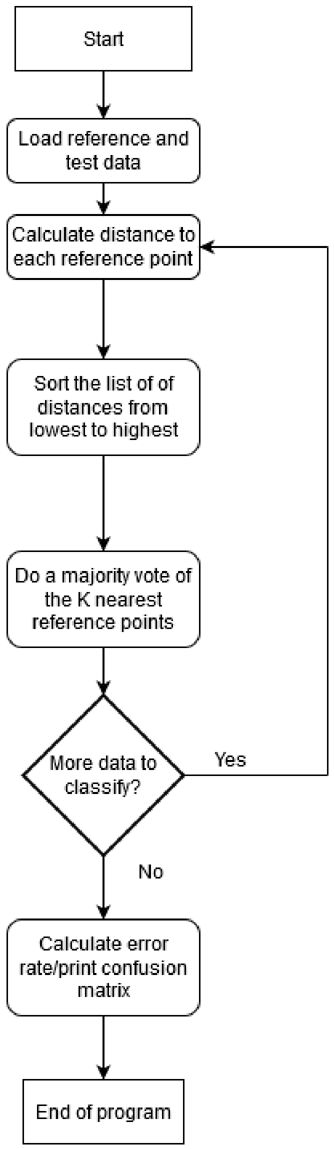

Start
Load reference and test data
Calculate distance to each reference point
Sort the list of of distances from lowest to highest
Do a majority vote of the K nearest reference points
More data to classify?
Yes
No
Calculate error rate/print confusion matrix
End of program

Figure 1: Flowchart for the two basic KNN classifiers

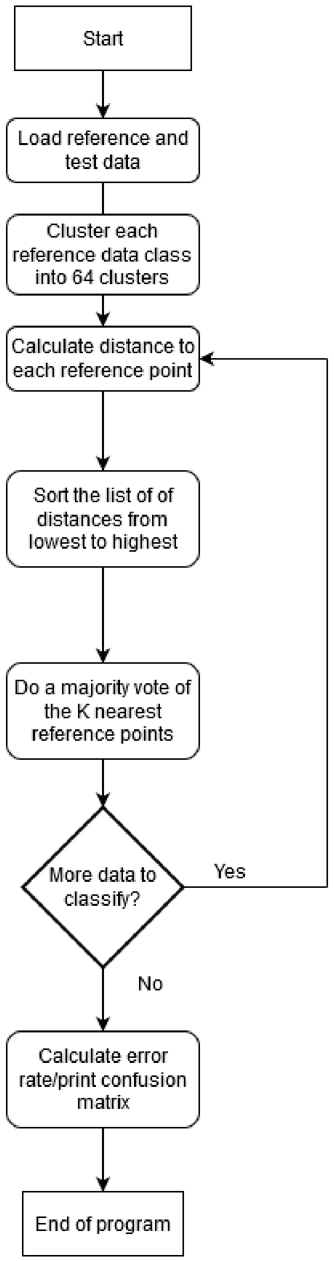

Start
Load reference and test data
Cluster each reference data class into 64 clusters
Calculate distance to each reference point
Sort the list of of distances from lowest to highest
Do a majority vote of the K nearest reference points
More data to classify?
Yes
No
Calculate error rate/print confusion matrix
End of program

Figure 2: Flowchart for the templated based KNN classifier

Images classified	KNN			
	K = 1		K = 4	
	Time used	Error rate	Time used	Error rate
100	132.2s	36.0%	137.8s	33.0%
1000	1354.8s	51.6%	1474.3s	46.0%

Images classified	Cluster NN		
	Clustering	Classifying	
	Time used	Time used	Error rate
100	327.5s	1.4s	5.0%
1000	364.5s	14.8s	4.8%
10000	360.7s	127.3s	4.5%

Table 1: Processing time and error rates of the KNN classifiers

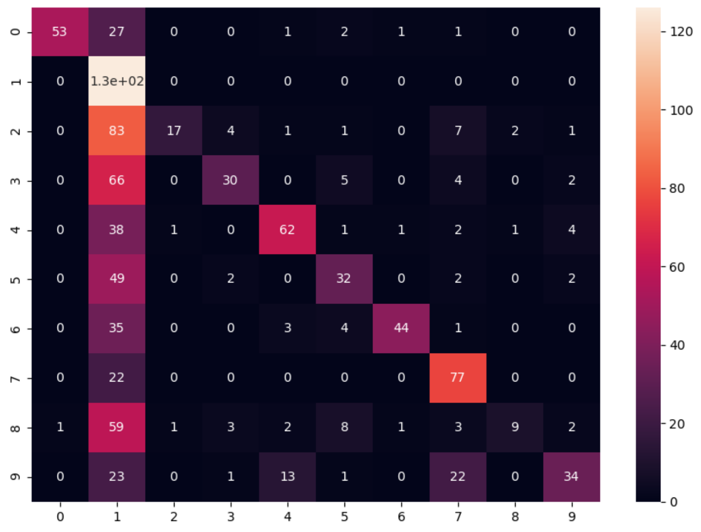
4 Results

In table 1 we can see the processing time and error rate for the three classifiers. The processing time was calculated on a laptop running windows, therefor the time may vary when calculated on other computers, the ratios should however stay roughly the same. The two non template KNN classifiers were not tested with 10000 test images as it would take far to long to process. For these two we can see the the ratio between time used and number of images are almost the same for both 100 and 1000 tested images. The error rate increases as more images were tested. For 100 tests KNN with K = 4 has an error rate that is 3% lower than for K = 1. For 1000 tests the difference increases and becomes 4.6% lower for K = 4

The clustering is clearly the most effective algorithm. The processing time for classification is about 100 times lower than for the non template classifiers. While the clustering uses around 350 seconds to complete, it is relatively constant for both amounts of test data. For an amount of test data over ca. 260 samples the template version will use a lower total time with clustering included.

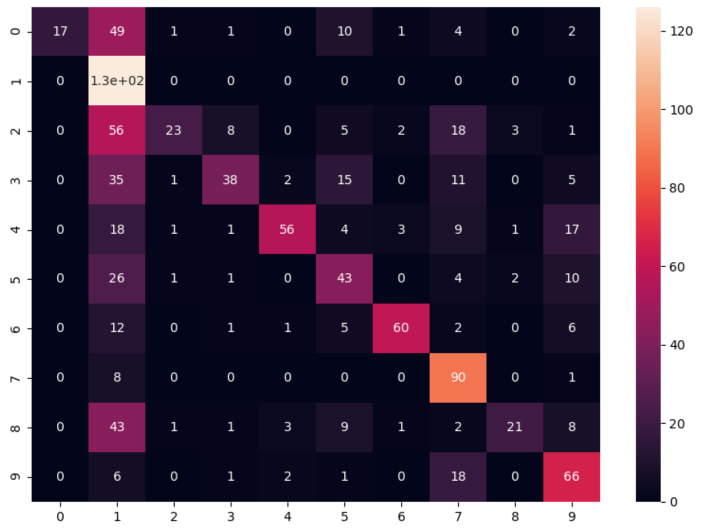
The error rate of the template classifier for 1000 tests is only 4.8% which is nearly 10 times lower than for the K = 4 NN. From the confusion matrixes in figures 3-5 we can see that the template NN is in general not biased towards any specific number while the two KNN have a large amount of numbers mislabeled as "one".

In figure 6 and 7 we see some of the correctly and miss-correctly labeled numbers of the template KNN classifier. Here T is the correct number and P is the number predicted by the classifier. The numbers in 7 are quite ordinary and should pose no problem for a human to identify. In figure 6 some of the numbers can be harder to read. Especially the 5 in the second row from the top and the number 3 in the bottom row. While other again seem completely normal like the 5 in the second bottom row and the 0 in the second top row. This indicate



	0	1	2	3	4	5	6	7	8	9
0	53	27	0	0	1	2	1	1	0	0
1	0	1.3e+02	0	0	0	0	0	0	0	0
2	0	83	17	4	1	1	0	7	2	1
3	0	66	0	30	0	5	0	4	0	2
4	0	38	1	0	62	1	1	2	1	4
5	0	49	0	2	0	32	0	2	0	2
6	0	35	0	0	3	4	44	1	0	0
7	0	22	0	0	0	0	0	77	0	0
8	1	59	1	3	2	8	1	3	9	2
9	0	23	0	1	13	1	0	22	0	34

Figure 3: K = 1 NN, 1000 test samples



	0	1	2	3	4	5	6	7	8	9
0	17	49	1	1	0	10	1	4	0	2
1	0	1.3e+02	0	0	0	0	0	0	0	0
2	0	56	23	8	0	5	2	18	3	1
3	0	35	1	38	2	15	0	11	0	5
4	0	18	1	1	56	4	3	9	1	17
5	0	26	1	1	0	43	0	4	2	10
6	0	12	0	1	1	5	60	2	0	6
7	0	8	0	0	0	0	0	90	0	1
8	0	43	1	1	3	9	1	2	21	8
9	0	6	0	1	2	1	0	18	0	66

Figure 4: K = 4 NN, 1000 test samples

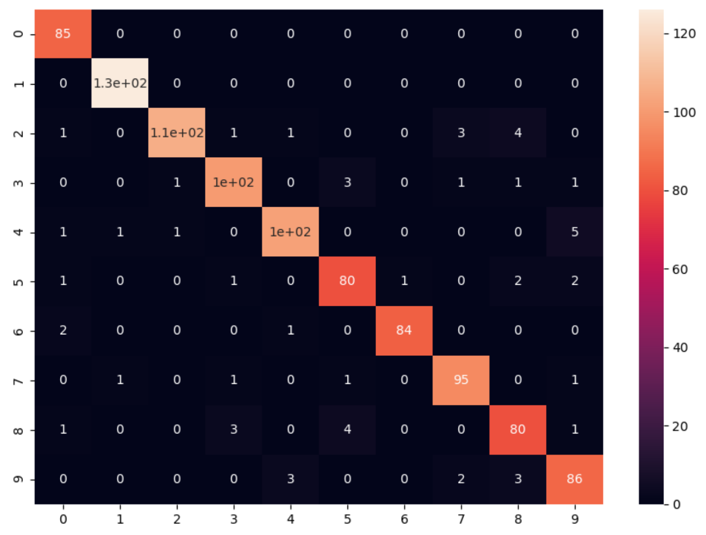


Figure 5: Template NN using clustering, 1000 test samples

that while there is some correlation between badly written numbers and the classifiers inability to predict them correctly, there also seem to be an aspect of randomness involved.

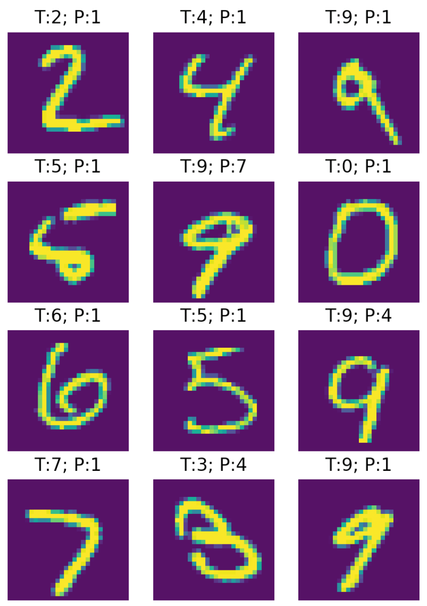


Figure 6: Pictures of correctly classified numbers using template KNN

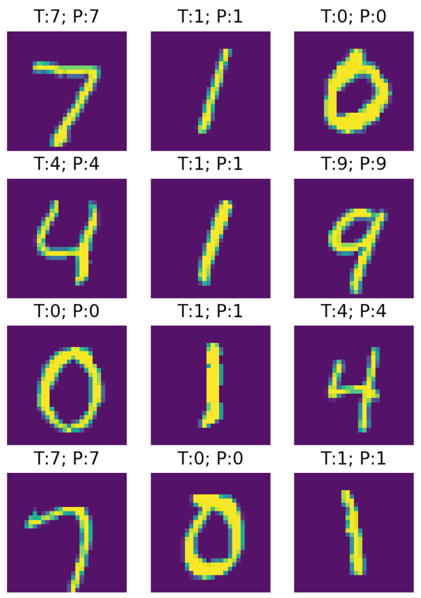


Figure 7: Pictures of wrongly classified numbers using template KNN

5 Discussion

As predicted in the theory part 2.3 the classifiers show a linear trend in processing time as the number of test data increases. While the classification of a single image is fast enough for real-time usage for all three classifiers the template version is clearly superior in both time and error rate. It should be used for all purposes were a low error rate is important or when clustering can be utilized before test data is inputted. The error rate of the two non template KNN classifiers are so high that they are probably not usable without being heavily monitored. We also see a large discrepancy in error rate between a 100 and a 1000 test images. This is probably from a 100 test samples not being large enough to correctly estimate the true error rate of the classifier thus an error rate of around 50% will be more in line for standard use. The main error factor as shown in the confusion matrixes in figures 3-5 comes from other numbers being miss-classified as "one". If the number "one" is not used as reference or for testing the two non template KNN might give a lower error rate than they currently do. This should be looked into before a possible use.

6 Conclusion

As discussed in the theory part the classifiers had a linear increase in processing time in relation to the amount of test data. The K = 1 and K = 4 KNN classifiers had an error rate of about 51.6% and 46% respectively. The lower error rate for K = 4 The time to classify a single number was ≈ 1.35 second with marginally higher processing time for K = 4 . While the template based NN classifier is initially slower because of the clustering time of ≈ 350 seconds, the classifying itself has a speed of ≈ 0.14 seconds per number which is ≈ 100 times faster than the two basic KNN classifiers. The clustering can be performed pre classifying and should thus not add to any real-time processing. The template error rate was also superior with an error rate of ≈ 4.8% for all number of test data. Here the wrongly predicted numbers seem to have some correlation with how badly they are written, but some of the miss-predicted numbers are easily identified by a human eye. Thus it seems there is also an aspect of randomness involved for the miss-predicted numbers. The two non template KNN classifiers had a big tendency to miss-classify other numbers as "one". For these to be used in any viable way one should evaluate their performance when the number "one" is not part of the reference or test data.

References

[1] Magne H. Johnsen., December 18, 2017, *TTT4275- Estimation, detection and classifying : Compendium in classification for TTT4275*

[2] scikit-learn developers., April 2020, *https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html*

[3] NumPy developers., April 2020, *https://numpy.org/index.html*

[4] The SciPy community,. April 2020

https://docs.scipy.org/doc/scipy/reference/generated/scipy.spatial.distance.euclidean.html

[5] Yann LeCun, Professor The Courant Institute of Mathematical Sciences, Corinna Cortes, Research Scientist Google Labs, New York, New York University,. April 2020 *http://yann.lecun.com/exdb/mnist/*