KNN and K-means Clustering

A comparison of the two classification algorithms

Vasilis Kyriafinis, 9797

Aristotle University Of Thessaloniki
Neural Networks
November 2022

Contents

Introduction

Assignment description

The main point of focus of this project is to compare the KNN ans the K-means classification algorithms. For the purposes of this assignment the MNIST data set was used. The code for this assignment can be found in this GitHub repository.

The MNIST dataset

"The MNIST database of handwritten digits, available from this page, has a training set of 60,000 examples, and a test set of 10,000 examples. It is a subset of a larger set available from NIST. The digits have been size-normalized and centered in a fixed-size image.

It is a good database for people who want to try learning techniques and pattern recognition methods on real-world data while spending minimal efforts on preprocessing and formatting." [1]

Each image is a 28x28 pixel gray-scale image. All the images are labeled with the correct digit. The purpose of this assignment is to create two + 1 classification algorithms that are able to identify the images in the test dataset.

State of the Art

Due to the fact that the MNIST dataset is a very well known dataset, there are many successful attempts in the classification. The majority of them use th python language and a relative library.

Objective and Methodology

Objective

This assignment will implement the algorithms found in those libraries in C++ to the best ability of the author. The main focus is to achieve good accuracy in a reasonable amount of run time. For this reason parallel programming techniques will be used.

Equipment

All of the test were run on a 64-bit Manjaro Linux system with an AMD Ryzen 7 5700G processor with 8 cores and 16 threads. The processor has a base clock of 3.8 GHz and a boost clock of 4.6 GHz. The system memory is 16 GB DDR4 3200 MHz.

Methodology

KNN

The KNN algorithm is a supervised learning algorithm. The only parameters of the algorithm that is changed in this implementation is the number of neighbors.

The algorithm requires to calculate the distance between the test image and all the training images. The distance is calculated using the Euclidean distance formula for each pixel of the image.

K-means

The K-means algorithm is an optimization of the KNN algorithm. Instead of calculating the distance between the test image and all The training images, the K-means algorithm calculates the distance between the test image and the centroids of the clusters. The centroids are the mean of the images in the cluster. This K-means implementation uses 10 clusters. The plain K-means algorithm has not parameters to change.

K-means with advanced clustering

The next step in the optimization of the K-means algorithm is to use a more advanced clustering algorithm. The clustering algorithm used in this implementation is the K-means++ algorithm. The K-means++ algorithm classifies the data into more clusters than the original ones. With this optimization more features can be extracted from the data. The K-means++ algorithm has quite a few parameters to change such as the number of clusters, the number of iterations and the portion of the data to use for the initial centroids. The exact implementations for all the algorithms will be explained in the next chapter.

Algorithms and Results

KNN

Implementation

The KNN algorithm is based on the fact that similar objects are close to each other. The distance between two images denotes the similarity between them. If two images have overlapping features, from the Euclidean distance formula (1), those features will cancel each other out and the final distance will be small.

$$d = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{1}$$

The actual implementation of the Euclidean distance in the code does not use the square root. This is because the square root is a relatively expensive function to implement. Also the square root is a monotonic function. This means that the order of the distances is preserved.

An other optimization of the current implementation is that the final distances array is not sorted. The K nearest neighbors are acquired by finding the minimum value of the distances array. The time complexity of this algorithm is O(kn) instead of O(nlogn).

The K factor is the number of neighbors that the algorithm will use to classify the image. The values of K that were used for testing in this implementation are 1 and 3. In general we want to use an odd number for K. This is because if we use

an even number for K we might end up with a tie. Finally, the MNIST dataset has overlapping classes. This means that the classes are not linearly separable. For this reason larger numbers for K lead to very low accuracy.

Results

The KNN accuracy table (1), shows the accuracy and the execution time of the KNN algorithm for different values of K. The accuracy is the percentage of the correctly classified images. The execution time is the time it took to classify all the images in the test dataset. The execution time is measured in seconds. 16 threads were used for the parallelization of the algorithm.

K	1	3	5	7
Accuracy (%)	96.91	97.05	96.88	96.94
Time (s)	39.63	41.92	44.67	46.52

Table 1: KNN performance

K-means

Implementation

The K-means algorithm is an optimization of the KNN algorithm. Instead of calculating the distance between the test image and all the training images, the K-means algorithm calculates the distance between the test image and the centroids of the clusters. The centroids are the mean of the images in the cluster.

In order to start the classification process, the algorithm needs to initialize the centroids. The centroids are initialized by calculating the mean image of every class. The mean image is then used as the centroid of the cluster. The initialization cost of the algorithm is substantially higher than the KNN algorithm. The results of the initialization are shown in the figure (1).

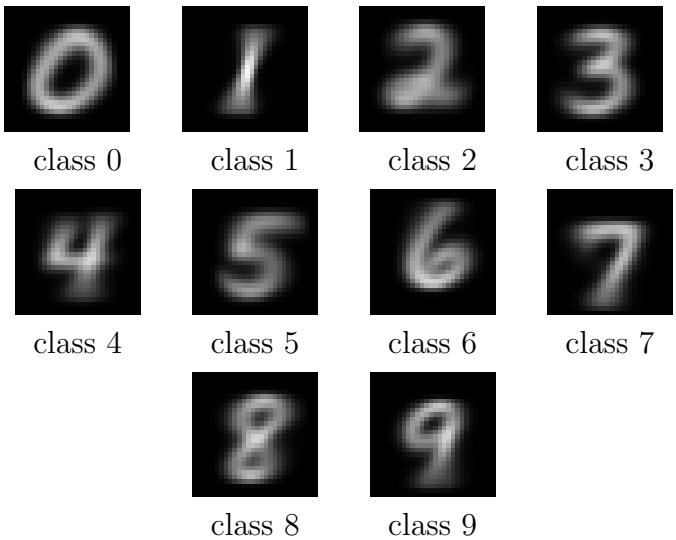


Figure 1: K-means centroids

The centroids above appear to to be fuzzy. This is because of the variation of images in the same class. It is clearly visible that the centroids of the class 1 is comprised of ones that lean to the left and ones that lean to the right. This is bad for the classification as the distance between the centroid and the test images will never be low enough to confidently classify the image.

The up site of the K-means algorithm is that once the centroids are initialized, the algorithm can classify the images in a very efficient way. The centroids can be saved in a file and reused for future classifications. The classification process is the same as the KNN algorithm.

Results

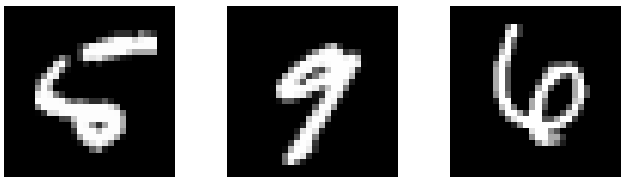
The K-means algorithm does not perform very well. The reason for this is that digits of the same class can appear in drasticaly different positions. This means that the centroids could easily be placed in the middle of another class, thus leading to a lot of misclassifications.

The table below shows the accuracy and the execution time of the K-means algorithm. The accuracy is the percentage of the correctly classified images. The execution time is the time it took to classify all the images in the test dataset. The execution time is measured in seconds. 16 threads were used for the parallelization of the algorithm.

Run	1	2	3	4
Accuracy (%)	82.06	82.06	82.06	82.06
Time (ms)	65.8	61.16	68.42	63.55

Table 2: K-Means performance

Because the cluster means are initialized from all the data the results of each run are the same. Some misclassifications are shown in the figure (2).



classified as 2 classified as 7 classified as 4

Figure 2: K-means misclassifications

The first and second image could easily be misclassified but the third image is a clear misclassification. The image is clearly a 6 but the algorithm classified it as a 4. The distance between the centroid of the class 4 and the image is 757713 while the distance between the centroid of the class 6 and the image is 769147. It is a close call but the algorithm should have classified the image.

K-means with Advanced Clustering

Implementation

The initialization of the k-means algorithm was performed in a supervised way. The centroids were initialized by iterating over all the training images and calculating the mean image of each class. This method is not very efficient in two ways. First, the performance of the algorithm is dependent on the size of the training set. Second the number of classes is fixed. This is a problem because in the MNIST dataset images from the same class can appear in different positions.

The interpretation of this phenomenon is that images from the same class can have different characteristics, such as size, shape etc. For example the number 1 can lean to the left or to the right. To combat this problem, the number of clusters are drastically increased.

The initialization algorithm used is the K-means++ algorithm. The K-means++ chooses the first centroid randomly and after that it tries to choose the next centroids in a way that the distance between the centroids is maximized. The algorithm is described in the paper [2].

After the centroids are initialized, training images are assigned to the closest centroid. The class of each centroid is determined by the most common class of the images assigned to it. The centroids are also updated by calculating the mean image of the images assigned to it.

During the selection and later fitting of the centroids, a random subset of the training images is used. This is done to speed up the algorithm but it also prevents the algorithm from overfitting.

The number of the centroids determines the number of distinct features that the algorithm will extract from the images. Below are some selections of centroids for different number of clusters.

Figure 3: 10 Clusters centroids selection

With 10 clusters (3) the centroids average out close to all 0. So the classification is almost random at 20% accuracy.

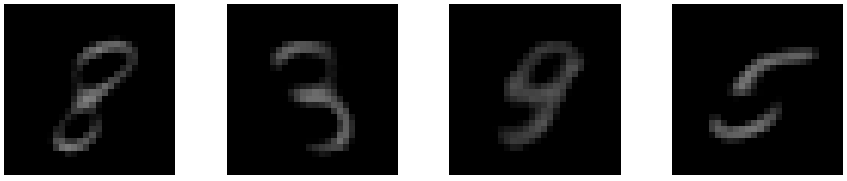


Figure 4: 32 Clusters centroids selection

With 32 clusters (4) the centroids are more distinct. The first image is an 8, the second is probably a 3, the third is a 9 and the fourth is a 5. The accuracy is close to 60%.

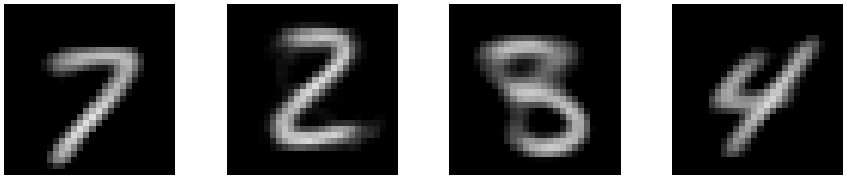


Figure 5: 128 Clusters centroids selection

With 128 clusters (5) the centroids are even more distinct. There are some problems with specific numbers such as 3 and 8. For example the 3rd image could be a 3 or a 8. The accuracy is close to 80%.

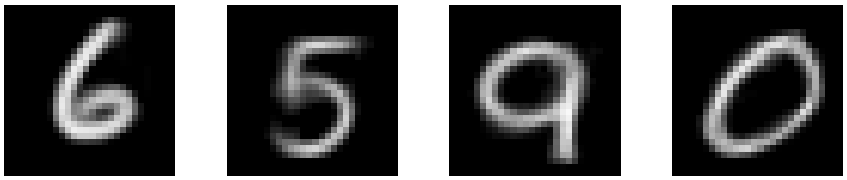


Figure 6: 350 Clusters centroids selection

Finally with 350 clusters (6) the centroids are very distinct. The accuracy is close to 90%. This number of clusters produces the best results.

For all the tests the number of iteration during fitting was set to 30, as this was the number that produced the best results.

Results

Number of clusters	10	32	128	350
Accuracy (%)	15.99	53.53	86.58	91.38
Cluster Creation Time (s)	3.1	7.38	26.35	77.18
Image Classification Time (s)	0.061	0.175	0.677	2.034

Table 3: K-Means++ clustering performance

The increasing cluster creation time is not a problem as it is done only once. The produces centroids can be saved and used for future classification.

Conclusions

The KNN algorithm has the greatest accuracy of the three algorithms. The disadvantage is that the classification time for each image is very high. In the MNIST dataset this is not a big problem because the training set is relatively small. Using bigger dataset would significantly increase the classification time. The time complexity of the KNN algorithm is O$m*n$ where n is the number of training images and m is the number of test images.

The K-means algorithm has a better time complexity than the KNN algorithm because after the initial setup the classification time is O10m where m is the number of test images. The disadvantage of the algorithm is the lack of accuracy.

Finally the K-means++ algorithm has better accuracy than the K-means algorithm but still worse than the KNN. Again after the initial setup the classification time is small. There are a few parameters that can be changed to improve the accuracy of the algorithm. The number of clusters, the number of iterations and the portion of the data to use for the initial centroids, and for the later fitting of them.

In this projects the parameters are probably sub optimal but still produce an above average result. Further optimization of the parameters could improve the accuracy of the algorithm to levels directly comparable to the KNN algorithm.

List of Figures

List of Tables

Bibliography

[1] *MNIST*. http://yann.lecun.com/exdb/mnist/.

[2] David Arthur Sergei Vassilvitskii†. *k-means++: The Advantages of Careful Seeding*. URL: https://theory.stanford.edu/~sergei/papers/kMeansPP-soda.pdf.