# 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

### 1. 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.

### 2. 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.

### 3. 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, 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 table below 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 accuracy

## 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 calculated by adding all the images of a class and dividing the result by the number of images in that 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.

But up site 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 (%) | 73.27 | 73.27 | 73.27 | 73.27 |
| Time (ms) | 65.8 | 61.16 | 68.42 | 63.55 |

Table 2: K-Means accuracy

Because the cluster means are initialized from all the data the results of each run are the same.

## K-means with Advanced Clustering

### Implementation

# Bibliography

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