Home exam machine learning Q and A

## 2.1 What is clustering?

*Explain what clustering is, and how it is different from classification. Give an example of a problem where clustering might be the appropriate solution.*

Clustering is a machine learning method that groups data together without any labels. It is an unsupervised learning method that focuses on underlying patterns and space in vectors to group together samples. Unlike classification it does not need to be trained beforehand, and its only goal is to group together data.

**Example of a clustering problem**: Clustering could be used to group houses and apartments based on features like size, number of bedrooms, location, and price. Without predefined categories (such as "house" or "apartment"), clustering can automatically find groups of properties that are similar, such as high-end homes or small, affordable apartments, revealing underlying patterns in the housing market

## 2.2 K-mean clustering

*Describe the k-means algorithm. Explain how the cluster centroids are initialized and updated. What is the stopping criterion for the algorithm? Typically, k-means is used on vectors, but how can you use it on 2D image data?*

**K-means algorithm:**

The model’s primary goal is to group data into a pre-selected number of clusters, often notated as **“k”**. Each sample is represented as a vector, where each feature is displayed a seperate dimension. The algorithm creates centroids with some given conditions. It then moves the centroid toward the average position of all the closest samples. Here is a breakdown of how the algorithm manipulates the centroids into clusters of data:

**How centroids are initialized:**

There is primarily two ways to create centroids:

1. **Random**

Centroids are placed randomly within the data space. This method is simple but risky, as randomly chosen centroids may end up too close to each other or too far from the actual data clusters. This could lead to poor clustering results, as certain clusters might overlap or remain underrepresented. However, if the random centroids are placed near the true cluster centers, this approach can work well.

1. **Using samples**

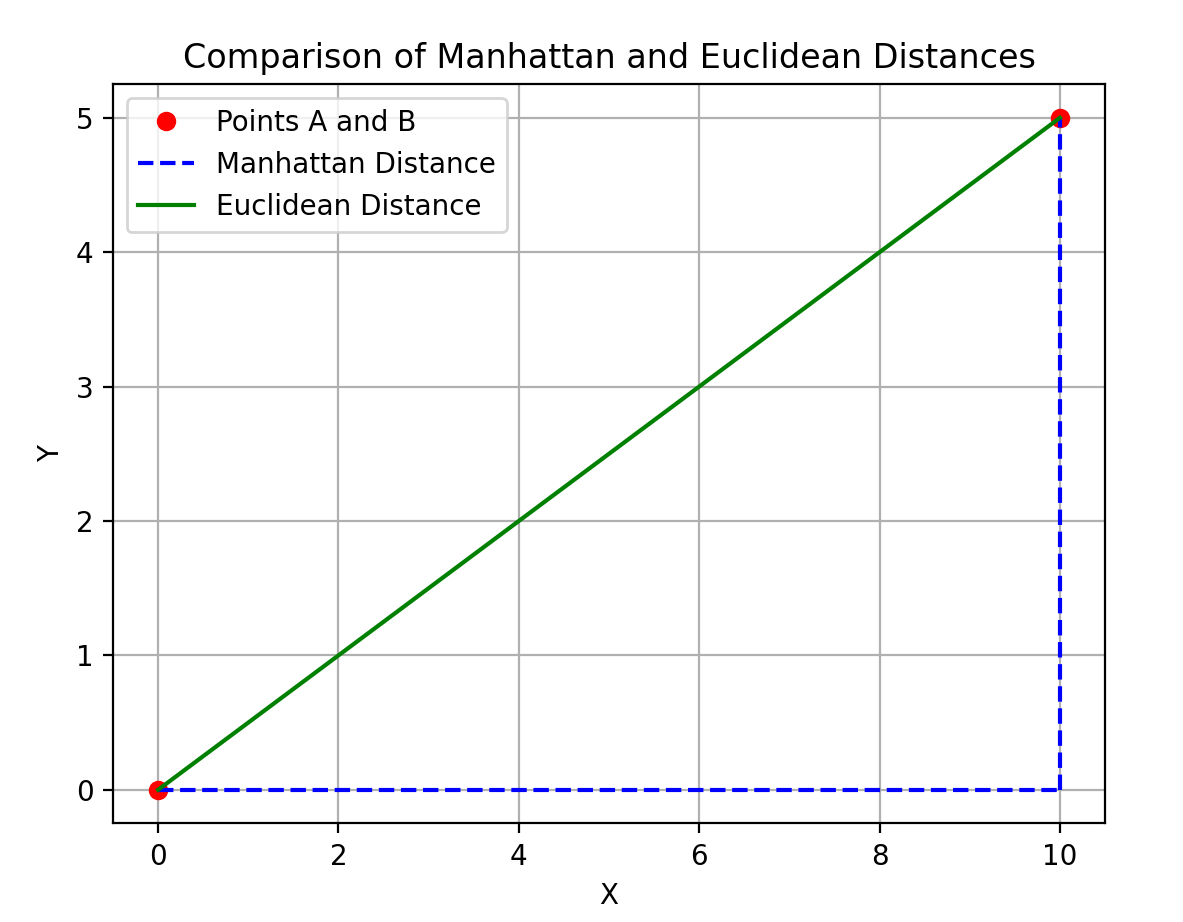
Centroids can also be created by using the vector position of samples in the dataset. This ensures that the centroids are close to the data points we wish to cluster. There is still a chance that we pick samples that is close to each other, therefore it’s important to look at the samples when using samples for centroids. This could be hard to do in high dimensional vector space.

Each method has its own use places depending on the goal of the algorithm and the dataset. Therefore, it is important the developer of model considers which option is the best.

**Updating centroids:**

1. Each vector is assigned to the closest centroid
2. Centroid is moved toward the average position of all the closest centroids.
3. These steps are repeated until a stopping criterion is met.

There are multiple ways to determine the distance between vectors. The most common is Euclidean distance and Manhattan distance. Under is a plot I made to visualize the difference:



Stopping criterions:

1. Max iterations: The model has run for a predefined number of iterations.
2. Minimal change: There is no longer significant change in centroid position from last iteration.

**Using k-means on 2D images:**

By representing each pixel as a feature, we can use K-means algorithm to cluster together images. We can use information such as color or intensity of each feature to make groups of the pictures.

Steps:

1. Flatten the images into a vector of pixels, and give each pixel numerical values determined by their color/intensity
2. Apply k-mean clustering to the vectors
3. Reconstruct the images

## 2.3 Display of dataset

*From the images you see, do you have any intuition on what clusters the k-means algorithm will yield?*

A close-up of a person's face

Description automatically generated

Display of 5 random pictures from the dataset

Given that the angle, lighting, and placement of the faces are consistent across the dataset, I predict that the k-means algorithm will primarily group images based on facial expressions. With relatively few pixels representing each image, capturing finer facial details is challenging. Therefore, the algorithm is likely to focus on more prominent differences, such as variations in facial expression.

## 2.4 Implementation of the K-means algorithm

The k-means algorithm was implemented using the principals in **2.2**, in **summary**:

1. ***Initialization***

The centroids are initially created as random samples within the dataset. I chose this approach because of the dataset. Each sample has 28x20 features (560), so with creating a random centroid, there is a chance the vector would land outside of the data’s range. Leading to poor clustering. Using samples is a safer approach when its impossible to visualize the vector space.

1. ***Distance metric***

In this implementation of the k-means algorithm, the user can choose between Manhattan and Euclidean distance. When testing the program, I did not notice any difference between the methods. Given that the Manhattan distance is computed simpler, I opted for it.

1. ***Stopping criterion***

The model runs either until a maximum number of iterations is reached or until the centroids no longer change significantly between iterations.Specifically, the algorithm stops when the change in centroid positions is smaller than a set tolerance. Due to computational precision issues with very small changes, I set the tolerance to 10-8.

1. ***Data handling***

The dataset is already preprocessed into a 1D array of numerical values. After the clustering process, the pictures are reconstrued using the pyplot import from matplotlib. Chatgpt helped me with the reconstruction.

The full snippet of the code is included in **Appendix B (link her din skibidi).**

## 2.5 The models result

In this implementation, the user has the option to choose the number of clusters (k), Here is the result of different k values:

A collage of different facial expressions

Description automatically generated

K= 3

A collage of different facial expressions

Description automatically generated

K=6

A collage of different facial expressions

Description automatically generated

K=10

*Based on the visualization, what do the individual clusters represent? Does what you observe align with your initial intuition?*

Based on the visualizations, the model effectively clustered pictures of the same person, despite accounting for image flips, facial expressions and different angles. This outcome surprised me, as I initially believed that facial expressions would be the dominating factor. The result suggests that face shape, positing of eyes and facial features are the dominating factors. This also proves that k-means clustering is not only a simple model, but also an effective one.

*What do you think would happen if the faces in the dataset were not perfectly centred in the image but randomly (slightly) shifted to the left or the right? Would you get the same results? why?*

If the pictures were not perfectly centered, I think it would negatively affect the model. Simple algorithms like K-means are sensitive to positioning because each pixel is treated as a feature. When the pixels shift, they would end up being compared with different features. This means prominent features like face shape, eye positioning, and facial details wouldn’t be matched correctly anymore. As a result, the algorithm's ability to detect underlying patterns would significantly weaken, and we would not get the same results.

## 3.2 Appendix B

A screenshot of a computer program

Description automatically generated

A screen shot of a computer program

Description automatically generated

Reconstruction of the images:

A computer screen shot of a program

Description automatically generated