Unsupervised Learning algorithms for digits recognition

https://github.com/nossa-y/Unsupervised_learning

Nossa Iyamu, Yanis Halloum
Telecom SudParis
Institut Polytechnique de Paris

Abstract

Unsupervised machine learning plays a pivotal role in various applications, and its application to digit recognition is of particular interest due to its potential impact on fields like computer vision. This project focuses on implementing unsupervised machine learning algorithms such as K-Means, K-Medoids, DBSCAN, and Gaussian Mixture for digit recognition using the MNIST dataset.

The primary objective of this research is to explore and apply unsupervised learning techniques to efficiently recognize and cluster digits within the dataset. The algorithms aim to uncover inherent patterns and structures in the data without explicit guidance, showcasing the potential of unsupervised approaches in the domain of digit recognition. The project uses the power of algorithms to autonomously discern features and group similar digit images, demonstrating the feasibility and efficacy of unsupervised learning in handling large-scale image datasets.

Keywords: Unsupervised Learning, Machine Learning, Digit Recognition, MNIST Dataset, Clustering, Feature Extraction.

Introduction

Digit recognition is a fundamental problem in the realm of image processing and machine learning, with applications ranging from optical character recognition to automation and robotics. The MNIST dataset, consisting of 10,000 handwritten digits, has been a benchmark for evaluating the performance of various machine learning models. In this study, we focus on the application of unsupervised machine learning techniques to the MNIST dataset, specifically addressing the recognition and clustering of digits.

The use of unsupervised learning in digit recognition offers a unique perspective, as it enables the algorithms to autonomously identify patterns and structures within the dataset without relying on explicit labels. The pattern recognition can be useful to know in a population those who have a handwriting issue. This project explores the implementation of various unsupervised learning algorithms, such as clustering and feature extraction, to discover and exploit the inherent relationships among the digit images.

The ultimate goal is to showcase the efficacy of unsupervised learning in discerning similarities and differences among digit images, leading to the formation of meaningful clusters without prior knowledge of the digit labels. Through this exploration, we aim to contribute to the broader understanding of unsupervised learning techniques and their potential applications in the domain of digit recognition.

1 MNIST Dataset

The dataset comprises 10,000 images represented as pixel vectors, where each pixel is identified by its position, ranging from pixel1 to pixel784. The pixel values range from 0 to 254, reflecting the intensity of the grayscale, with 0 representing white and 254 representing black. Additionally, each image is associated with its label, indicating the corresponding digit from 0 to 9 with different handwriting.

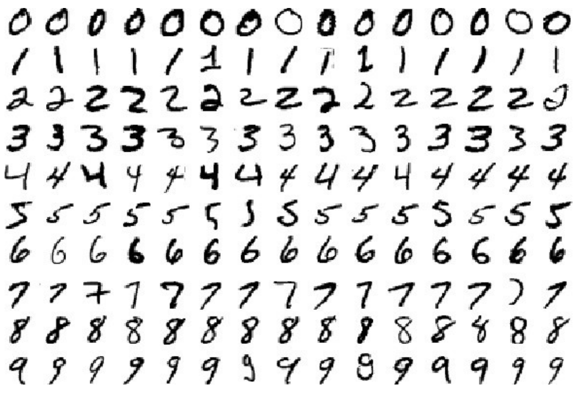


Figure 1: Illustration of the MNIST database structure.

2 K-Means

The K-means algorithm is a hard clustering method aiming at partitioning a dataset into a non-overlapping number of clusters. Each data point is made part of the nearest cluster in terms of mean which serves as the centroid of the cluster. We want to use this method to cluster our digits. Thus, two of the most important parts of this algorithm is the choice of the clusters' centre and their number. In our pursuit of implementing unsupervised machine learning algorithms for digit recognition, we are confronted with the inherent variability in how individuals may write the same digit. While our initial inclination is to consider 10 clusters, each corresponding to a specific digit, we recognize the potential richness in exploring alternative cluster configurations. This stems from the acknowledgment that a single digit can manifest in various writing styles, leading to the discovery of nuanced patterns within the data.

To elaborate further on this idea, we propose the exploration of a range of cluster options, extending beyond the conventional 10 clusters. By considering, for instance, 5 classes, our objective shifts towards identifying digits that are likely to be written in similar or identical ways. This approach allows us to delve into the nuances of writing styles within a reduced cluster space, unveiling patterns of similarity in how certain digits are represented.

Conversely, expanding our perspective to encompass 15 classes provides us with a unique opportunity to uncover the spectrum of diversity in writing styles for each digit. This exploration into a more extensive set of clusters enables us to discern which digits exhibit the most varied representations. In doing so, we aim to capture the breadth of potential writing styles within the dataset, offering valuable insights into the intricacies of digit representation.

In essence, this flexible approach to cluster configuration allows us to adapt our model to the inherent complexity of digit writing variations. By considering different numbers of clusters, we enhance our capacity to unravel the diverse patterns embedded in the dataset, ultimately contributing to a more comprehensive understanding of the nuances in handwritten digit recognition.

2.1 Inertia

Using the KMeans algorithm, we will use the Inertia cost function:

$$J = \sum_{i=1}^{K} \sum_{x \in \text{Cluster}_i} \|x - \mu_i\|^2$$

With

$\mu_i = \frac{1}{n_i} \sum_{x \in \text{Cluster}_i} x$, the centroid of the cluster i,

K, the number of clusters

n_i, the number of data assigned to the i-th Cluster

The cost function plays a pivotal role in unsupervised machine learning algorithms, particularly in the context of clustering. In the case of our digit recognition project, the cost function is instrumental in evaluating the performance of our model by quantifying the distance between each data point belonging to a cluster and its corresponding centroid.

The centroid, a fundamental concept in clustering, serves as a representative point for a cluster and is not an actual data point present in the dataset. Rather, it is computed as the mean of similar data points within the cluster, determined based on certain criteria. This criteria could be defined by the algorithm itself, such as minimizing the intra-cluster variance or optimizing a specific objective function.

By computing the distance between each data point and its cluster's centroid, the cost function provides a measure of how well the data within a cluster aligns with the cluster's representative point. Common distance metrics used in this context include Euclidean distance or Mahalanobis distance, depending on the nature of the data and the characteristics of the clustering algorithm.

The role of the centroid as a mean of similar data points reflects the central tendency within a cluster, capturing the essence of the group. This abstraction allows the model to generalize the characteristics of the cluster and aids in making predictions for new, unseen data points.

Ultimately, the cost function serves as a guide for the unsupervised learning algorithm to iteratively adjust the positions of the centroids, seeking to minimize the overall distance and enhance the cohesion of data within each cluster. This iterative process of updating centroids and recalculating the cost function contributes to the model's ability to discern meaningful patterns in the data, facilitating the effective clustering of digits in our digit recognition project.

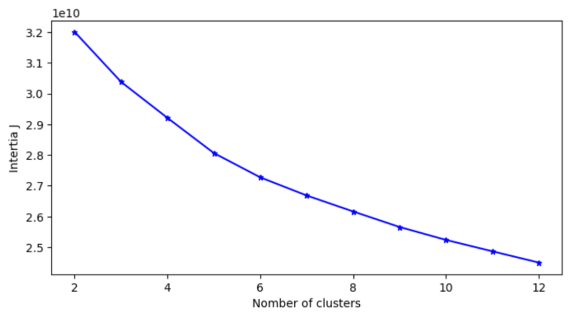


Figure 2: Plot of the Inertia as a function of the number of clusters

By simulating this cost function according to the number of clusters, we consider the "elbow" method to discover the optimal number of clusters to minimize the cost function. The "elbow" method consists in identifying the moment the curve varies sharply and get the number of clusters K that makes this effect. However, the curve is almost linear, so we cannot identify the "elbow". Thus, we cannot deduce the optimal number of clusters with this method.

2.2 Silhouette

Let's study another method: the silhouette method:

The silhouette method is a technique for evaluating the quality of a clustering solution. It measures how similar each data point in a dataset is to its own cluster (cohesion) compared to other clusters (separation). This measure provides an indication of the quality of the separation between clusters.

Here's how the silhouette method works:

1. **Distance Calculation:** For each data point in the dataset, the average distance (or a dissimilarity measure) is calculated to all other points in the same cluster (cohesion) and the average distance to points in the nearest neighboring clusters (separation).

2. **Silhouette Score Calculation:** For each point, the silhouette score is calculated using the formula:

$$\text{Silhouette Score} = \frac{b - a}{\max(a, b)}$$

3. **Overall Score Calculation:** The overall silhouette score for the entire clustering is then obtained by taking the average of the silhouette scores for all points.

4. **Interpretation:** The silhouette score ranges from -1 to 1. A high score indicates good separation between clusters, with high cohesion within clusters and clear separation between them. A score close to zero suggests overlapping clusters, and a negative score generally indicates that points might be better assigned to a different cluster.

In summary, the silhouette method provides a quantitative measure of the quality of a clustering solution by assessing both intra-cluster cohesion and inter-cluster separation.

Let's draw the silhouette function according to the number of clusters.

We can observe a maximum of the function for $K = 9$ if we don't consider $K = 2$ because this value is not valid in our study. With the value $k = 9$, the model is going to provide the best separation among the different classes according to the similarity of the images. The choice of K=9 holds significance in our exploration, despite the conventional expectation of aligning the number of classes with the distinct digits (10 in this case). As elucidated earlier, this departure from the intuitive alignment introduces a compelling element to our analysis.

Upon setting K=9, an intriguing observation emerges. The initial pattern discernible from the

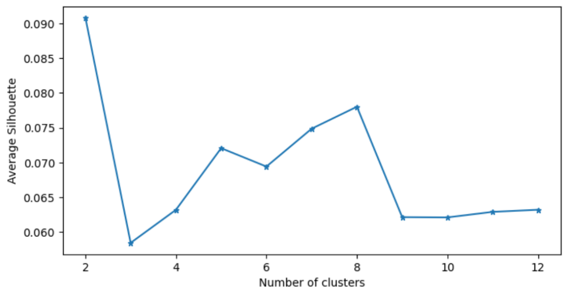


Figure 3: Plot of the Silhouette as a function of the number of clusters

MNIST database, encompassing over 10,000 images, is the remarkable similarity in the way some numbers are written. This revelation suggests a notable challenge in training a model to distinguish between certain digits, as they exhibit resemblances that transcend individual variations.

However, this challenge becomes an opportunity for improvement. Identifying which specific numbers present difficulties in differentiation empowers the model to refine its learning process. Below are the 9 centroids computed by the model, serving as representative points for each cluster and providing valuable insights into the shared characteristics of digits that pose challenges in discrimination.

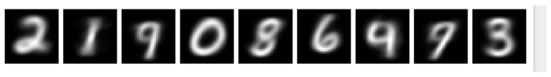


Figure 4: The 9 clusters defined by the K-Means method.

We can see that there is not even 9 digits represented, digits are repeated. An analyse of these centroïds shows us that the 9 is likely to be confused with the 7 and 4. An human eye would see the 9 represented 3 times in different forms. Thus it would be important for the model to take into account every pattern of this number in order to identify dissimilarities between 9, 4 and 7 and be able to tell what number correspond to a data point.

Let's see how the model behaves for 10 clusters, the optimal value found with the silhouette metric:

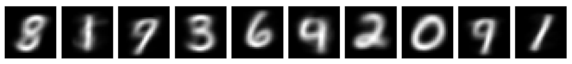


Figure 5: The 10 clusters defined by the K-Means method.

Once again, the cluster distribution is not injective. Hence, either the number of clusters is not optimal or the features treated by the model for finding similarities between data is not valid. This highlights the main default of this method: the dependence on the centroids initialization.

2.3 Calinski-Harabasz Index

To measure how efficient the clustering was, we use the Calinski-Harabasz and Davies-Bouldin indexes:

The Calinski-Harabasz Index measures the ratio of between-cluster variance to within-cluster variance. A higher score indicates better separation between clusters. It measures the inter-cluster dispersion.

The formula for the Calinski-Harabasz Index is as follows:

$$\text{C-H Index} = \frac{\text{Between-group variance}}{\text{Within-group variance}} \times \frac{N-K}{K-1}$$

Davies-Bouldin Index

The Davies-Bouldin Index measures the "compactness" of clusters and the "separation" between clusters. A lower score indicates better clustering quality. It measures the intra-class dispersion.

The formula for the Davies-Bouldin Index is as follows:

$$\text{Davies-Bouldin Index} = \frac{1}{K}\sum_{i=1}^{K}\max_{j\neq i}\left(\frac{\mathrm{S}_i + \mathrm{S}_j}{\mathrm{D}_{ij}}\right)$$

In summary, the Calinski-Harabasz Index favors clusters that are compact and well-separated, while the Davies-Bouldin Index favors clusters that are both internally compact and well-separated from each other. These indices are used to assess the quality of clusters generated by clustering algorithms, such as K-means. A higher score for the Calinski-Harabasz Index and a lower score for the Davies-Bouldin Index typically indicate better clustering quality.

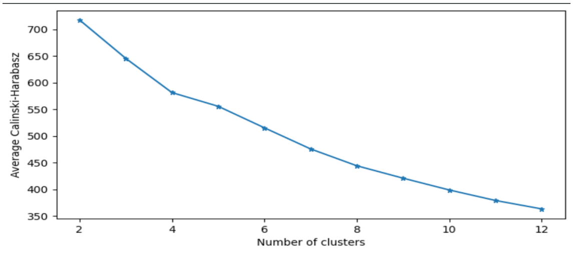


Figure 6: Average Calinski-Harabasz as a function of the number of clusters.

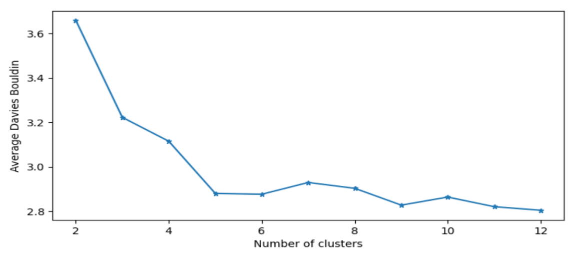


Figure 7: Average Davies-Bouldin as a function of the number of clusters.

We can see that the Calinski-Harabasz Index increases when K decreases and Davies-Bouldin Index decreases when K increases. There is not an explicit method to find the best compromise i.e find the best weights for the formula:

$$score = w1 \times CH\_score \quad + w2 \times \frac{1}{DB\_score}$$

According to these metrics, $K = 8$ seems to be a better compromise than $K = 9$ because the Calinski-Harabasz Index for $K = 9$ is higher than for $K = 8$ and Davies-Bouldin Index is lower for $K = 9$ than for $K = 8$.

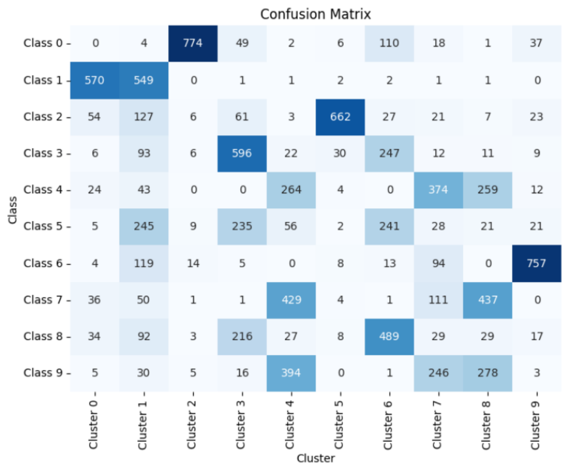
2.3.1 Summary

Finally, we found that the silhouette method suggests $K = 9$ for the best clustering and K decreases, and Davies-Bouldin Index suggests $K = 8$, because several factors need to be considered:

1. **Sensitivity to Cluster Shapes:** The Calinski-Harabasz and Davies-Bouldin indices may be sensitive to certain shapes of clusters, favoring compact, well-separated clusters. If the clusters in your data have complex shapes or if they overlap, these indices might not perform optimally.
2. **Metric Assumptions:** Different indices make different assumptions about the nature of clusters. For example, the silhouette method focuses on the compactness and separation of clusters, while Calinski-Harabasz and Davies-Bouldin indices consider variance and distance measures.
3. **Trade-offs:** The silhouette method often seeks a balance between compactness and separation, aiming for clusters that are well-defined but not too dispersed or too close to each other. Other indices might emphasize one aspect over the other.
4. **Noise Sensitivity:** Noise or outliers in the data can influence clustering results. Some indices may be more or less sensitive to noise, leading to differences in the recommended number of clusters.
5. **Dataset Characteristics:** The nature of the data, including its dimensionality, distribution, and inherent structure, can impact the performance of clustering indices. Different datasets may require different clustering evaluation approaches.

Thus we need to deeply dive into the context of the problem and the objectives to know which metric of error it is more accurate to use.

Ultimately, it would be useful to use a confusion matrix to verify which number of clusters provides the most accurate clustering, but in unsupervised learning, it would mean checking manually if each number corresponds to its cluster which can be time consuming when we don't have any label. In this case, we have the labels associated with each data, we can compute a confusion matrix comparing labeled data with the label predicted by our model.



	Cluster 0	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5	Cluster 6	Cluster 7	Cluster 8	Cluster 9
Class 0	0	4	774	49	2	6	110	18	1	37
Class 1	570	549	0	1	1	2	2	1	1	0
Class 2	54	127	6	61	3	662	27	21	7	23
Class 3	6	93	6	596	22	30	247	12	11	9
Class 4	24	43	0	0	264	4	0	374	259	12
Class 5	5	245	9	235	56	2	241	28	21	21
Class 6	4	119	14	5	0	8	13	94	0	757
Class 7	36	50	1	1	429	4	1	111	437	0
Class 8	34	92	3	216	27	8	489	29	29	17
Class 9	5	30	5	16	394	0	1	246	278	3

Figure 8: Confusion matrix for 9 clusters.

In our analysis, it becomes evident that clusters 2, 5, and 9 exhibit a distinctive characteristic: they predominantly gather numbers unique to their respective clusters, belonging to only one class. Consequently, these clusters demonstrate the lowest variance and boast high accuracy in their predictions. This suggests that the model excels in accurately capturing the patterns associated with these particular clusters.

Contrastingly, clusters 6, 7, and 8 present a different scenario. Their composition includes numbers that span multiple classes, leading to higher variance and reduced accuracy. These clusters seem to pose challenges for the model in achieving precise predictions, indicating potential areas for improvement.

Examining individual clusters, cluster 2 emerges as a strong candidate for representing class 0. This inference suggests that the model effectively identifies and utilizes relevant features to predict the occurrence of 0 in a given data point.

On the other hand, considering individual classes, class 5 appears distributed across various clusters, implying that the model encounters difficulties in accurately predicting this class. This observation underscores a potential limitation in the model's performance, particularly in discerning the distinctive features associated with class 5.

In summary, our analysis sheds light on both the strengths and weaknesses of the model, emphasizing the need for further refinement to enhance its predictive capabilities.

For this model, the average entropy is 6.68.

3 K-Medoids

The K-Medoids method shares similarities with the K-Means algorithm, but with a key distinction in how cluster centroids are defined. In K-Medoids, unlike K-Means, the centroid of each cluster is an actual data point from within the cluster, and this point is referred to as the medoid.

The medoid is chosen to minimize the sum of distances between itself and every other data point within the same cluster. In other words, it represents the most centrally located point in the cluster with respect to the dissimilarity (or distance) metric used. This choice of a medoid as the centroid enhances the robustness of the algorithm, especially in scenarios where mean-based centroids might be sensitive to outliers.

3.1 Inertia

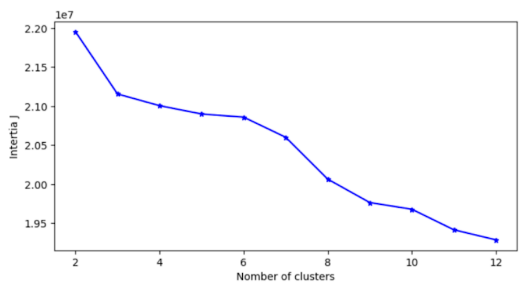


Figure 9: Plot of the Inertia as a function of the number of clusters for the K-Medoids method

The elbow method is not really valid here, the same way it was for K-Means: the elbow cannot be identified. However, we can use the Silhouette metrics.

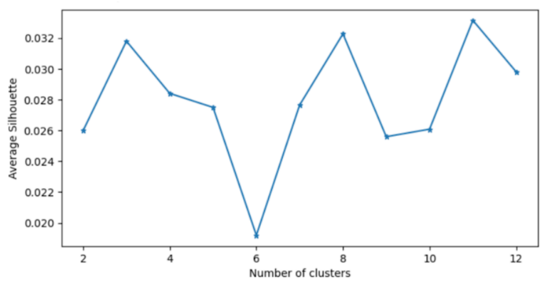


Figure 10: Plot of the Silhouette as a function of the number of clusters for the K-Medoids method

The optimal number of cluster for this metric is 8.

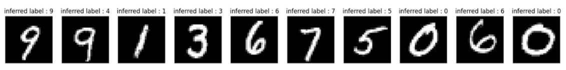


Figure 11: Clusters obtained with K-Medoids method

When plotting the centroids (which are medoids), an human eye can see that only 8 digits are represented. After computing the entropies, the lowest is attached to the ninth cluster (digit 0) and the highest is for the second one (digit 1). Error metrics like Calinski-Harabasz Index Davies-Bouldin Index can be used and we'd find an optimal number of clusters of 8. We can conclude that an optimal number of clusters to group the data is 8, but regarding the entropy, K-means method seems to be more appropriate because the average entropy is lower than kmedoids method, even if the extremes values are more apart.

4 Gaussian Mixture Model

The data is assumed to be generated by a mixture of several gaussian distributions, each associated with a particular cluster or component (to be developed) Thus a data point can belongs to several clusters which can be more logic in some cases (for example a sport shoe belongs to the category sport clothes and shoes

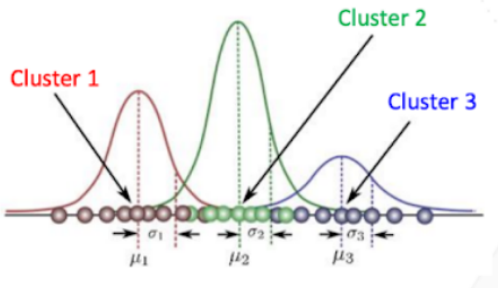


Figure 12: Illustration of the superposition of clusters in the Gaussian Mixture Model.

Each cluster is optimized my maximizing the log-likelyhood.

5 Clustering for image compression

We consider this original photo of a dog:

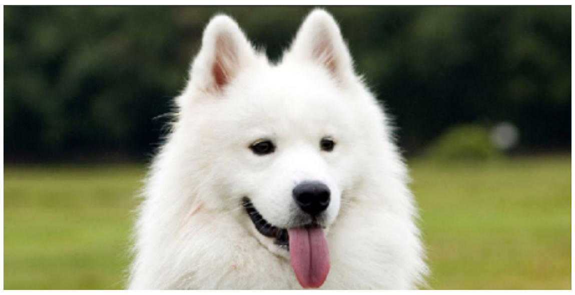


Figure 13: Reference image.

We will apply the K-Means method for compressing this image.

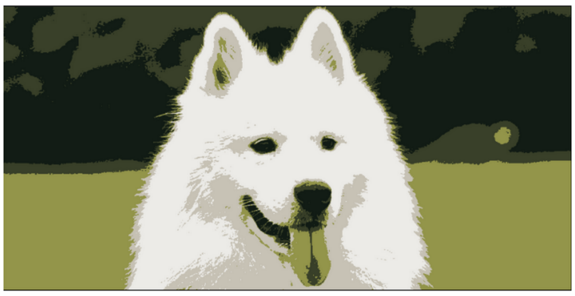


Figure 14: Image compressed for K = 5.

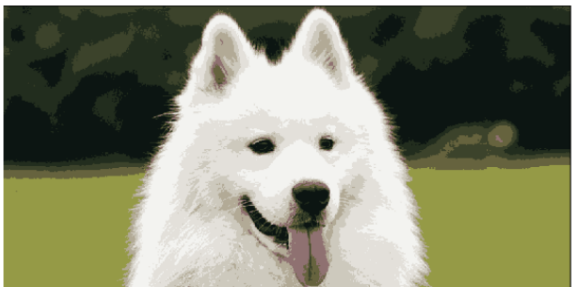


Figure 15: Image compressed for K = 10.

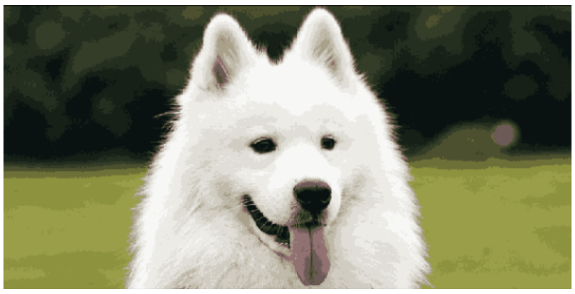


Figure 16: Image compressed for K = 16.

We can see that as we rise the value of K, the compressed image is more and more looking like the original one. The number of clusters represent the number of color shades displayed. With K = 5, only 5 colors are displayed, that is why the image looks so blurry.

6 Conclusion

Strenght and weaknesses of the K-Means

The K-Means algorithm is easy to implement as it only takes the number of clusters as input and has an efficient mean complexity of O(tKN) with N samples, K clusters, t iterations. However, the algorithm is not not usable when the attributes are not intervals because the mean needs to be computed. In addition, the number of K cluster needs to be optimal and the algorithm depends too much on the initialization.

We can also highlights the limits of the K-Means method with a simple data repartition example:

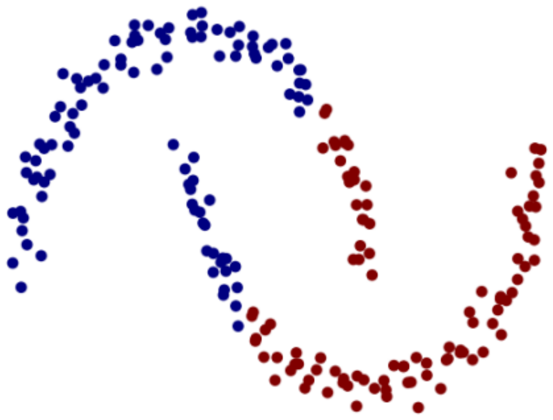


Figure 17: K-Means behaviour for 2 clusters.

Now, let's apply DBScan to that exact example:

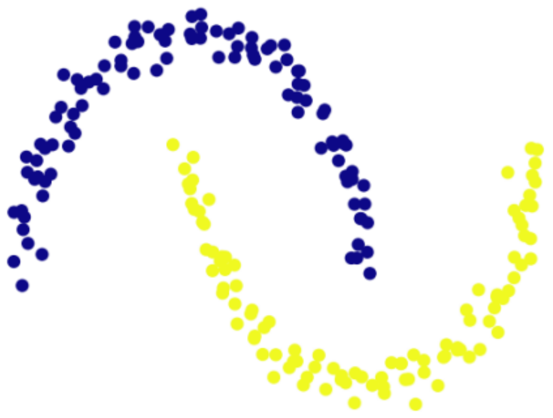


Figure 18: DBScan behaviour for 2 clusters.

K-Means can only form spherical clusters because it is based on the distance to the center. It is not applicable when the data is not of that shape. On another hand, as the DBSCAN method is based on the data points density, it can cluster arbitrary shapes.