MODELING AND SIMULATION OF A K-MEANS CLUSTERING ALGORITHM IN MATLAB AND PYTHON

Ștefana DUȚĂ[1], Laura-Elena DOROBANȚU[2], Dan-Gabriel RADU[3]

[1]*Department of Information and Computer Systems Engineering, 411 IISC, Faculty of Electronics, Telecommunications and Information Technology, Politehnica University of Bucharest, Bucharest, Romania*
[2]*Department of Measurements, Electrical Devices and Static Converters, SIIM1, Faculty of Electrical Engineering, Politehnica University of Bucharest, Bucharest, Romania*
[3]*Department of Electrical Engineering, IEIA2, Faculty of Electrical Engineering, Politehnica University of Bucharest, Bucharest, Romania*

*Corresponding author: Ștefana DUȚĂ, stefana.duta@stud.fim.upb.ro

Scientific coordinator: Felix Constantin ADOCHIEI, PhD, DMEDSC

Abstract. *This paper presents an analysis of the K-means algorithm, a popular unsupervised learning technique used for clustering data based on similar features. For this, the implementation and performance of the K-means algorithm were compared in the Python and MATLAB programming languages. In addition, modeling and simulation of the algorithm were performed to compare the computation time and clustering quality in the two languages. The experimental results demonstrate that the K-means algorithm is a powerful tool for data clustering and can be used effectively in various applications. Although both Python and MATLAB are languages capable of efficiently implementing the K-means algorithm, MATLAB resulted in a shorter computation time.*

Keywords*: K-means, MATLAB, Python, modeling.*

Introduction

Supervised and unsupervised learning are the two main categories of machine learning. Supervised learning algorithms aim to learn a function that maps an input to its corresponding output using available labeled data. Obtaining labeled data is often a challenging task in supervised learning [1]. In situations where dataset labels are unavailable, unsupervised learning is typically used. Clustering algorithms are used in this framework to group data points that have similar characteristics, exploiting the underlying structure of the data distribution [2-3]. Unlike supervised learning, where the algorithm has access to labeled data, unsupervised learning algorithms do not have prior knowledge of the actual labels of the dataset and instead draw conclusions from the data itself [3-4]. The k-means algorithm is a widely used and popular clustering technique due to its ease of implementation and low computational complexity. It is often applied to image compression, where it can reduce the number of colors used to represent an image while maintaining its visual similarity [5-8].

Fig. 1 depicts a typical K-means clustering procedure, where the input dataset is initially represented in a 2D space, as shown in Fig. 1a. Following the K-means clustering process, which involves setting the number of centroids to K=3, the resulting clustered dataset is presented in Fig. 1b. The K-means algorithm requires users to set the number of clusters and initialize the centroids randomly, but this can lead to poor performance due to the dependence on the initial selection of clusters. This issue is more pronounced for large datasets, and determining the optimal number of clusters is a complex task [9]. Another significant limitation of the algorithm is the use of Euclidean distance as a similarity measure, which restricts its ability to detect clusters of non-standard shapes and makes detecting overlapping clusters challenging.

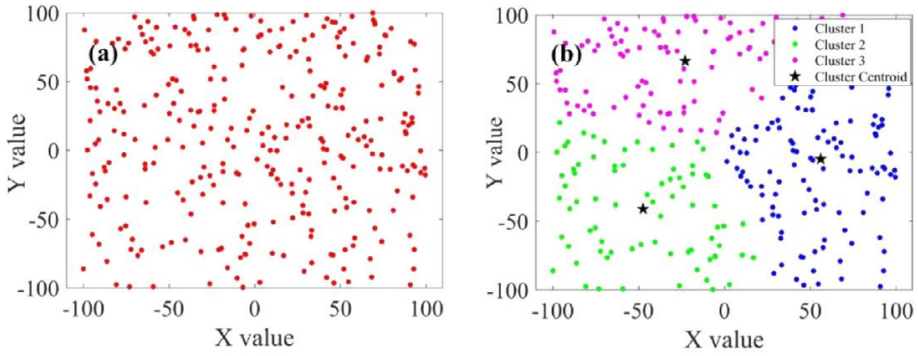

Fig. 1. The K-means clustering algorithm is depicted in (a) with randomly distributed datasets, followed by (b) the identification of the closest centroid for each of the three classes [8]

The K-means clustering algorithm is a heuristic iterative process that can be broken down into three main steps, as described by [7]:

a) Initialization involves setting the central points with a given number of K clusters.

b) The algorithm then assigns all data points to one of the K clusters based on the current centroids.

c) The centroids are then updated based on the newly formed clusters.

By repeating steps b) and c) over several iterations, the K-means algorithm eventually converges and can effectively detect ball-shaped or spherical clusters. However, this is limited using the Euclidean metric as a distance measure [8, 10].

Fig. 1.2 presents the steps of a K-means algorithm.

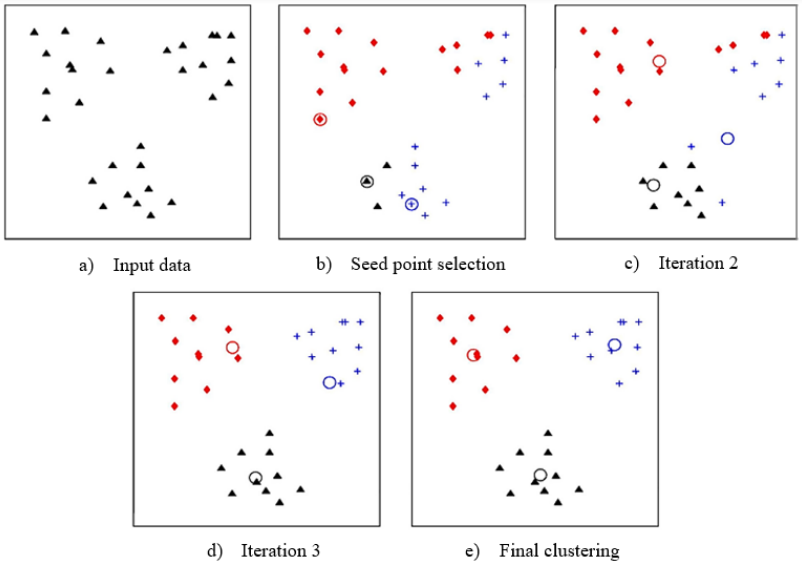

Fig. 2. The K-means algorithm is illustrated in the following steps: (a) A two-dimensional input dataset with three clusters. (b) Selection of three seed points as cluster centers, followed by the initial assignment of the data points to these clusters. (c) & (d) Intermediate iterations involving the updating of cluster labels and their centers. (e) The final clustering obtained by the K-means algorithm at convergence [11].

In the K-means algorithm, the goal is to group n multidimensional data points in a given dataset *X* into *K* categories. To achieve this, the Euclidean distance is utilized as the similarity index, and the algorithm seeks to minimize the sum of squares of different types, referred to as the "minimization" step. [10, 12]. This step is carried out according to Eq (1) [12]:

$$d = \sum_{k=1}^{k} \sum_{i=1}^{n} \left|\left| x_i - u_k \right|\right|^2 \quad (1)$$

The variables used in the K-means algorithm are defined as follows: *k* represents the *K* cluster centers, u_k represents the *k*-th center, and xi represents the *i*-th point in the given dataset.

The solution of center u_k is shown in Eq (2):

$$\frac{\partial}{\partial u_k} = \frac{\partial}{\partial u_k} \sum_{k=1}^{k} \sum_{i=1}^{n} (\mathrm{x_i} - u_k)^2$$
$$= \sum_{i=1}^{n} 2(x_i - u_k) \quad (2)$$

Modeling and Simulation

This paper compares the implementation and performance of the K-means algorithm in Python and MATLAB d.

A. Data generating:

Random data are generated from three normal distributions with known parameters. First, each class used several data points to study the algorithm's computation time evolution. Next, the data is divided into three classes and randomly mixed.

B. The starting assumption

The three starting points for the algorithm are set.

C. Setting the algorithm parameters.

An expected precision of 0.001 and a substantial error starting value of 10000 are chosen.

D. Expectation

The expectation is the first function, which associates the closest point to each center with its cluster and handles updating the data labels according to the Euclidean distance.

E. Maximization

This function updates the center of the clusters according to the update of the labels in the previous step by averaging the coordinates of the points in each class on each axis.

F. Error Update

At each iteration, the desired and the calculated error are compared. If the calculated one is higher, it is updated with a new value calculated according to equation (1) presented in the previous chapter.

This study uses an algorithm based on the K-means implementation developed by Reza Ahmadzadeh, available in the GitHub repository at https://github.com/rezaahmadzadeh/K-means [13. The presented algorithm was trained using a Lenovo device with an ADM Ryzen 5 3500U processor and 8 GB RAM, using the Windows 11 64-bit operating system. The training process did not involve the GPU.

Results

CASE I – 50 values for each class

Actual values of the centroids of the three classes:

- (4, 0); (0, 4); (-3, 3).

Initial guess values:

- (0, 0); (2, 2); (-1, -1).

Tab. 1 shows the running times for the two programming languages.

Table 1

Experimental Results with 50 values/class – Running Time

Running Time (s)	**MATLAB (4 iterations)**	**Python (4 iterations)**	**Python (2 iterations)**
	0.0025	0.0128	0.0149

With the help of Python, a result can be reached with only two iterations, whereas in MATLAB, a minimum of 4 iterations are required to achieve a minimum error value (0). The result obtained after the same number of iterations for both programming languages, named after four iterations, is also observed.

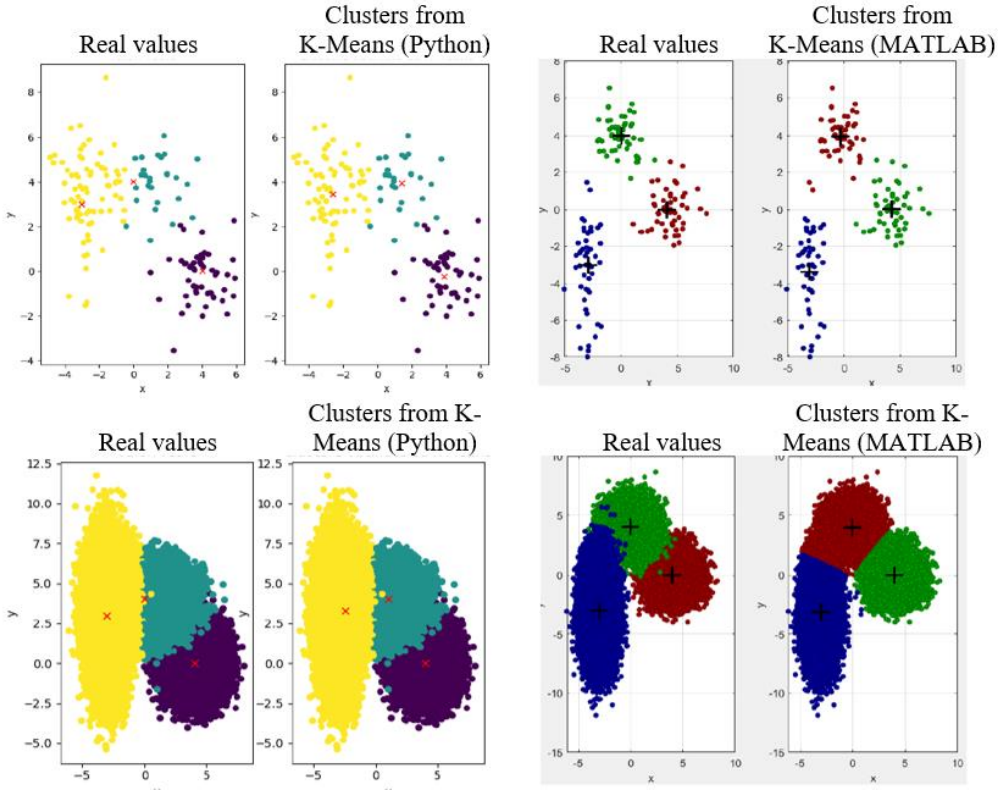

Fig. 3. Algorithm results: CASE I – up and CASE II- down. Python – left and MATLAB - right. Dividing the data into the three classes and determining the centroids

CASE II – 50000 values for each class

Actual values of the centroids of the three classes:

- (4, 0); (0, 4); (-3, 3).

Initial guess values:

- (0, 0); (2, 2); (-1, -1).

Tab. 2 shows the running times for case II. With Python, a result can be reached with six iterations, whereas in MATLAB, at least nine iterations are needed to get a minimum error value.

Table 2

Experimental Results with 50000 values/class – Running Time

Running Time (s)	**MATLAB (9 iterations)**	**Python (6 iterations)**
	0.1940	33.7897

Experimental results show that MATLAB is faster than Python when implementing the K-means algorithm. For example, the computation time in MATLAB was five times faster than in Python when using a set of 50 values. Furthermore, it is found that if the number of data is increased to 50000, MATLAB is 174 times faster than Python.

It was also found that the quality of the clustering results was similar in both languages, as can be seen in Fig. 3.

Conclusions

In conclusion, this paper compared the implementation and performance of the K-means algorithm in Python and MATLAB programming languages. By evaluating the algorithm on a generated data set, it was found that MATLAB is generally faster than Python in terms of computation time. This is probably due to the optimized built-in functions and toolboxes in MATLAB, designed explicitly for numerical computations and data analysis tasks.

However, it is worth noting that Python offers a vast ecosystem of libraries and tools that can be used to implement the K-means algorithm and is more flexible in integrating with other systems.

In summary, both Python and MATLAB can be used to implement the K-means algorithm and give accurate clustering results. However, MATLAB may be a better choice for tasks requiring faster computation time, while for jobs requiring more flexibility and integration with other tools, Python may be a better choice.

Bibliography

1. M. MOHRI, A. ROSTAMIZADEH, and A. TALWALKAR, "Foundations of machine learning," p. 486, 2012, Accessed: Jan. 18, 2023. [Online].
2. C. BISHOP, "Neural networks for pattern recognition," 1995.
3. A. K. JAIN, M. N. MURTY, and P. J. FLYNN, "Data clustering," *ACM Computing Surveys (CSUR)*, vol. 31, no. 3, pp. 264–323, Sep. 1999, doi: 10.1145/331499.331504.
4. M. AHMED, R. SERAJ, and S. M. S. ISLAM, "The k-means Algorithm: A Comprehensive Survey and Performance Evaluation," *Electronics 2020, Vol. 9, Page 1295*, vol. 9, no. 8, p. 1295, Aug. 2020, doi: 10.3390/ELECTRONICS9081295.
5. X. WAN, "Application of K-means Algorithm in Image Compression," *IOP Conf Ser Mater Sci Eng*, vol. 563, no. 5, Aug. 2019, doi: 10.1088/1757-899X/563/5/052042.
6. Y. ZHAO and X. ZHOU, "K-means Clustering Algorithm and Its Improvement Research," *J Phys Conf Ser*, vol. 1873, no. 1, Apr. 2021, doi: 10.1088/1742-6596/1873/1/012074.
7. X. G. LI, M. F. YAO, and W. T. HUANG, "Speech recognition based on k-means clustering and neural network ensembles," *Proceedings - 2011 7th International Conference on Natural Computation, ICNC 2011*, vol. 2, pp. 614–617, 2011, doi: 10.1109/ICNC.2011.6022159.
8. H. LE MINH, T. SANG-TO, M. ABDEL WAHAB, and T. CUONG-LE, "A new metaheuristic optimization based on K-means clustering algorithm and its application to structural damage identification," *Knowl Based Syst*, vol. 251, p. 109189, Sep. 2022, doi: 10.1016/J.KNOSYS.2022.109189.
9. A. M. IKOTUN, A. E. EZUGWU, L. ABUALIGAH, B. ABUHAIJA, and J. HEMING, "K-means clustering algorithms: A comprehensive review, variants analysis, and advances in the era of big data," *Inf Sci (N Y)*, vol. 622, pp. 178–210, Apr. 2023, doi: 10.1016/J.INS.2022.11.139.
10. A. SINGH, A. YADAV, and A. RANA, "K-means with Three different Distance Metrics," *Int J Comput Appl*, vol. 67, no. 10, pp. 13–17, Apr. 2013, doi: 10.5120/11430-6785.
11. A. K. JAIN, "Data clustering: 50 years beyond K-means," *Pattern Recognit Lett*, vol. 31, no. 8, pp. 651–666, Jun. 2010, doi: 10.1016/J.PATREC.2009.09.011.
12. C. YUAN and H. YANG, "Research on K-Value Selection Method of K-Means Clustering Algorithm," *J 2019, Vol. 2, Pages 226-235*, vol. 2, no. 2, pp. 226–235, Jun. 2019, doi: 10.3390/J2020016.
13. "K-means/k-means.py at master · rezaahmadzadeh/K-means · GitHub." https://github.com/rezaahmadzadeh/K-means/blob/master/Python/k-means.py (accessed Jan. 19, 2023).