Technical
University
of Munich

Elements of Machine Learning

“K-means Algorithm and K-means++ Initialization”

Raúl San Miguel Peñas

Engineering Risk Analysis Group
Department of Civil, Geo and Environmental Engineering
Technische Universität München

Munich, January 2023

ABSTRACT

Many problems involving data require the grouping of data points according to their common properties or their differences with other data points. To accomplish this, machine learning, and specifically the unsupervised category of clustering, can help reach conclusions using just input data. In this project, the K-means algorithm as well as the method of K-means++ initialization are explained and then applied to a dataset in order to explore their performance and carry out an appropriate a parametric study.

Keywords: machine learning, clustering, unsupervised, k-means, k-means++

CONTENTS

LIST OF FIGURES

1. INTRODUCTION

1.1. MOTIVATION

Machine learning is a term that is too often thrown around in everyday life, from the news to conversations with your neighbors, often to the point where it begins to lose meaning. To try and get an introductory-level understanding of what machine learning actually is, its many different types, its fundamentals, and its applications, a course such as the one this project is a part of is a great tool for engineering students. The block course "Elements of Machine Learning", offered by the Engineering Risk Analysis group of the Department of Civil and Environmental Engineering, aims to introduce said students to the world of machine learning and and allow them to gain some hands-on experience by focusing on a specific algorithm, method or topic. This specific project focuses on the the K-means clustering algorithm as well as the K-means++ initialization method.

1.2. OBJECTIVES

The objectives of this project are as follows:

- To explain the K-means algorithm and the method of K-means++ initialization.
- To describe the dataset chosen for the use of the K-means algorithm.
- To describe the methodology and tools used for the application of the K-means algorithm on the dataset.
- To investigate the performance of the K-means algorithm through appropriate parametric study.

1.3. STRUCTURE OF REPORT

This report follows a structure along the lines of the objectives just laid out. First, the main benefits and drawbacks of the K-means algorithm are discussed, followed by the description of the steps the method entails. Then, K-means++ initialization is explained and illustrated with the use of an example. After this part, the key components of the dataset used are summarized, and then the programming tools that make the analysis of this dataset possible are laid out. Finally, several results obtained from the application of K-means to the dataset are displayed and discussed, focusing on the key parameters of the algorithm.

2. K-MEANS ALGORITHM

2.1. PROPERTIES

The K-means algorithm is part of so-called unsupervised learning. Datasets are unlabeled, and the attributes of the input data is the only information available to discover any possible interpretations. K-means is used for clustering, a process through which the data is assigned to a number of groupings called clusters based on their similarities *and* differences, both of which have to be as clear as possible and have some kind of real-world meaning. Specifically, K-means encompasses centroid-based clustering, through which the centers of these clusters are identified and data points are assigned to each based on a certain measure of distance.

K-means' implementation, which will be seen in the following section, is not overly complex, relying on the iteration of a few basic concepts from statistics and linear algebra. It guarantees convergence of the objective function, but not that it will be a global minimum. This shortcoming is illustrated on Figure 2.1. The two clusters, one blue and one red, will remain constant as long as the the chosen centroids, indicated by the triangular icons, are moved along the black discontinuous line while as they remain equidistant to the green discontinuous line. It is evident, however, that the much better clusters, with much lower distances between member points, would be the ones created by the black-discontinuous line; one on the left and one on the right. K-means, unfortunately, is dependent on the initial choice of centroid position; an initialization like K-means++ aims to alleviate this drawback.

Closely linked to the problem of local minima is the choice of the number of centroids that best suits the dataset. At the moment, the process is mostly heuristic. Using a scree plot to display how the number of clusters relates to the evolution of the the objective function can be helpful in determining this number, but it nevertheless necessitates applying the K-means algorithm repeatedly before a proper decision is reached, which might not always makes in terms of time or computational cost. At any rate, there is a risk in increasing the number of clusters excessively, as eventually, when the number of clusters equals the number of data points, the objective function would be zero, AKA its best possible value, but no conclusions could be taken from the data[1].

Note also that the number of operations involved in K-means is the following:

$$O(\#iterations \cdot \#clusters \cdot \#instances \cdot \#dimensions) \tag{2.1}$$

The number of instances is the number of data points while the number of dimensions is the number of attributes corresponding to each data point.

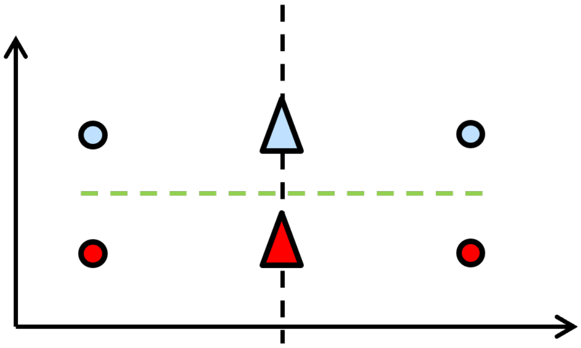


Fig. 2.1. Local minimum of simple dataset.

2.2. STEPS

The K-means algorithm as explained in this report is composed of a total of five steps, three of which are preliminary steps needed for the last two, which are the ones used in iteration procedures [2].

INITIAL STEPS/MEASURES:

1. Define k (number of clusters) initial cluster centroids $\{\boldsymbol{\mu}_k, k = 1, \dots, K\}$ where $\boldsymbol{\mu}_k \in \mathbb{R}^D$. These points can be random data points from the dataset, random points, or points chosen through another method (like K-means++). As seen in the previous section, this step is paramount for the good performance of the algorithm.

2. Define variables $r_{nk} \in \{0, 1\}$ to determine to which cluster each datapoint is assigned to. If the point x_n is closest to cluster k, then the variable r_{nk} has a value of 1, and if that is not the case then the point belongs to a different cluster j and the variable r_{nj} has a value of 0. AKA:

$$r_{nk} = 1 \text{ and } r_{nj} = 0 \text{ for } j \neq k \tag{2.2}$$

3. Define the objective function known as the distortion measure. This function measures the sum the of the squared Euclidean distances between the cluster centroids and each point belonging to that cluster. The objective of the algorithm is to minimize this measure through iteration of the value of the centroids and the data point assignment variables.

$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} \left\| \mathbf{x}_n - \boldsymbol{\mu}_k \right\|^2 \tag{2.3}$$

ITERATION STEPS:

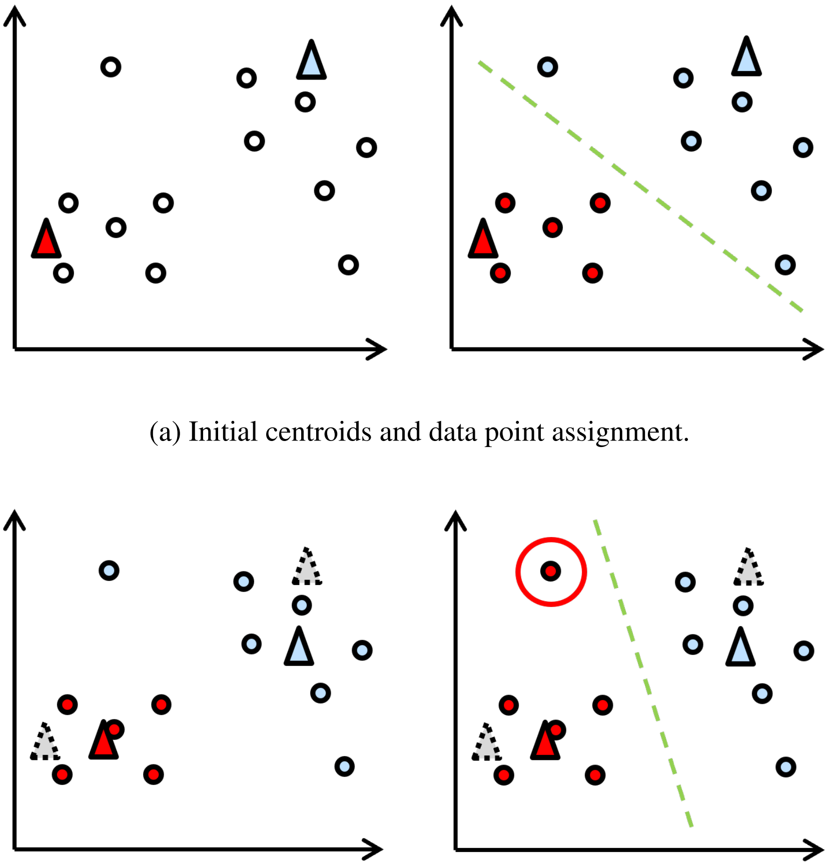
4. Fix centroids and assign data points to the centroid to which they are closest:

$$r_{nk} = \begin{cases} 1 & \text{if } k = \arg\min_j \left\| \mathbf{x}_n - \boldsymbol{\mu}_j \right\|^2 \\ 0 & \text{otherwise} \end{cases} \tag{2.4}$$

5. Fix data point assignment variables and recalculate then centroid values as the mean of the data points belonging to that cluster. This measure corresponds to the partial derivative of the distortion measure with respect to the centroids:

$$\nabla_{\tilde{\mu}_k} J = \mathbf{0} \Rightarrow \boldsymbol{\mu}_k = \frac{\sum_{n=1}^{N} r_{nk}\mathbf{x}_n}{\sum_{n=1}^{N} r_{nk}} \tag{2.5}$$

Steps 4 and 5 are iterated until convergence is reached. Convergence in this context means that the distortion measure, that is to say Equation 2.3, does not change value or that it changes very little, implying that no or very few points are changing cluster assignment after each iteration. These five steps are fairly straightforward, but they are impossible to visualize for a dataspace beyond three dimensions. Figure 2.2 helps visualize the algorithm for a 2D dataspace.



(a) Initial centroids and data point assignment.

(b) Recalculation of data points and re-assignment of data points.

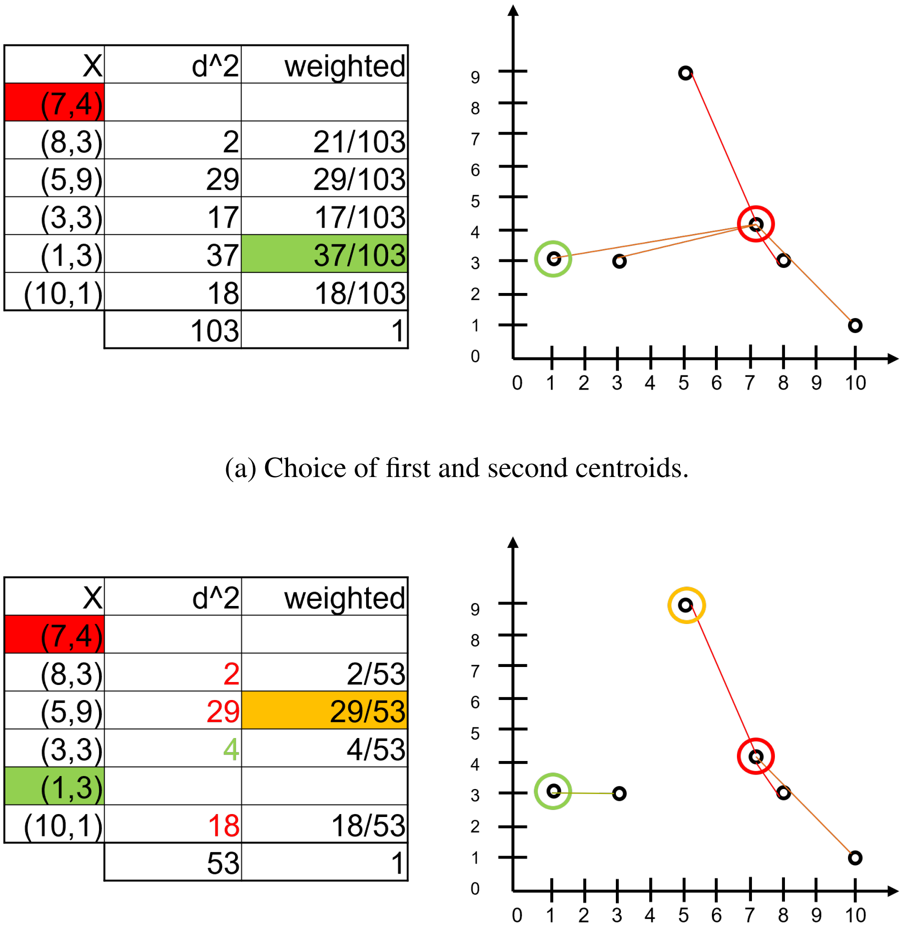
Fig. 2.2. Visual example of K-means iteration.

2.3. K-MEANS++ INITIALIZATION

The K-mean++ initialization attempts to improve the rest of the K-means algorithm by making sure that the initial centroid choices are as likely as possible to belong to different clusters. To do so, it makes sure that the centroids are as far from each other as possible. It consists of three steps:

1. Choose one initial centroid from the data points $\{\mathbf{x}_n, n = 1, \ldots, N\}$.
2. Choose the next centroid as the farthest data point from the first centroid.
3. Choose next centroid as the farthest data point from the previous centroids, with distance being defined as $\arg\min_j \left\| x_n - \mu_j \right\|^2$, which is to say, distance from a certain data point to the closest centroid. The distance corresponding to each data point is weighted by the sum of all distances, and the data point with the highest weighted distances is chosen as the next centroid. This last step is repeated until the desired number of cluster centroids is reached.

The example in Figure 2.3, corresponding to dataset X = [(7,4), (8,3), (5,9), (3,3), (1,3), (10,1)] and $k = 3$, illustrates these steps. In part (a), point (7,4) (in red) is chosen at random as the first cluster center. Then, the squared Euclidean distances to this center are calculated, followed by the weighting based on the total sum of 103. This leads to point (1,3), with a proportion of 37/103, to be chosen as the second cluster center (in green). In part (b), the distances are once again calculated. Since there are now two cluster centers, point (3,3) is closer to centroid (1,3), meaning that the distance between the two is recorded. Once again, the distances are weighted by the sum of the distances and point (5,9) is chosen as the third and final cluster center (in orange) [3].



X	d^2	weighted
(7,4)		
(8,3)	2	21/103
(5,9)	29	29/103
(3,3)	17	17/103
(1,3)	37	37/103
(10,1)	18	18/103
	103	1

(a) Choice of first and second centroids.

X	d^2	weighted
(7,4)		
(8,3)	2	2/53
(5,9)	29	29/53
(3,3)	4	4/53
(1,3)		
(10,1)	18	18/53
	53	1

(b) Calculation of squared distances and choice of third centroid.

Fig. 2.3. Example of K-means++ initialization for $k = 3$.

3. DATASET AND METHODOLOGY

3.1. DATASET

The dataset used for the application of the K-means algorithm and K-means initialization is called "seeds" and comes from various Polish institutions: John Paul II Catholic University of Lublin, Cracow University of Technology, and the Polish Academy of Sciences. This dataset, as well as many others used for all types of machine learning operations, can be found in SOURCE at the Machine Learning Repository of UC Irvine [4].

It contains data pertaining to three different varieties of wheat: Kama, Rosa and Canadian. These names are not used in the rest of the report as they are deemed irrelevant to the purpose of this project, which is to study the K-means algorithm itself rather than the dataset. There are a total of 210 instances, 70 corresponding to each type of wheat. Each data point is 7-dimensional, with the attributes being purely geometric (there is no information about, say, nutrition content): Area, Perimeter, Compactness (defined as C = 4*pi* A/P^2, Length of Kernel, Width of Kernel, Asymmetry Coefficient, and Length of Kernel Groove.

The data points are, in fact, labeled for the purpose of classification, which is a supervised machine learning procedure. Even though not necessary for clustering, the labeling provides a so-to-say "cheat sheet" for the method and can help determine the accuracy of K-means when it comes to determining the different clusters. At any rate, the dataset used in the results contained in this report has had the label attribute removed and contains the seven attributes previously mentioned. These have been normalized to have a value between 0 and 1.

3.2. METHODOLOGY

The results that will be discussed in the following section are all based on MATLAB's "kmeans" function, which offers a variety of input options and fairly complete output data. Among other options, the number of clusters can be specified as well as the type of initialization (K-means++ by default) and distance (squared Euclidean by default). The function can return the cluster indices, which indicate to which cluster each point belongs, the centroid locations, and the distortion measure. The number of iterations as well as the evolution of the distortion measure for each iteration step can also be shown in the console.

Beyond MATLAB, other options were explored in Python. ScikitLearn's "sklearn. cluster.Kmeans" performs the same operations as MATLAB's "kmeans", also allows for the detailed modification of input options and offers a range of output variables.

4. RESULTS

4.1. SCREE PLOTS OF ENTIRE DATASET

As mentioned in Section 2.1, a scree plot can be a useful tool to help determine how many clusters are most optimal for a certain dataset. Simply put, this type of plots displays the evolution of the distortion measure as the number of clusters *k* increases for a K-means procedure on the same exact dataset. Figures 4.1 and 4.2 are scree plots for the "seeds" dataset created for 2 to 30 clusters and using *all* instance dimensions. The number in Figure 4.1 are obtained using K-means++ initialization while in Figure 4.2 a random sample initialization is used. This method simply chooses *k* data points from the dataset at random to serve as the initial cluster centroids.

Both scree plots have very similar shapes and values, as would be expected since, regardless of the type of initialization, convergence is reached every time. Looking at the first few number of clusters, it is evident that the biggest drop in distortion measure happens between 2 and 3 clusters. This drop gives a good basis for choosing 3 as the most appropriate number of clusters. There is, however, not a clear elbow at 3, as the values for the distortion measure don't begin to really flatten out until 8 clusters or so. Therefore, even though it is known, thanks to the labels in the given dataset, that 3 is the accurate number of clusters, choosing 4 or 5 clusters could be justified.

The figures also display the time the whole process took as well as the average number of iterations for each K-means run. It is interesting to see that K-means++ took 0.184 seconds as opposed to the 0.118 seconds taken using a random sample. This difference makes theoretical sense, as K-means++ initialization can be considered an "offline" cost before the actual K-means algorithm springs into action, which will in all likelihood extend the total runtime. This statement is specially true when the number of clusters is high and the iterative steps highlighted in Section 2.3 need to be repeated 20 or 30 times.

The benefits of choosing proper (or more likely to be proper) centroids is indicated by the average number of iterations. While random sample initialization required a bit over 9 iterations to reach convergence, K-means++ was closer to 7. Furthermore, random sample needed, for certain number of clusters, up to 18 iterations to each convergence while K-means++ only ever needed 12 at most. These values, precisely because of the random nature of random sample initialization, depend on luck, so to say, meaning that the best method of guaranteeing quick convergence is using K-means++, as random sample could, in theory, select such bad centroids that the total runtime could even surpass the one for K-means++. This idea probably becomes more pronounced when the number of clusters is low and K-means++ necessitates fewer iterations.

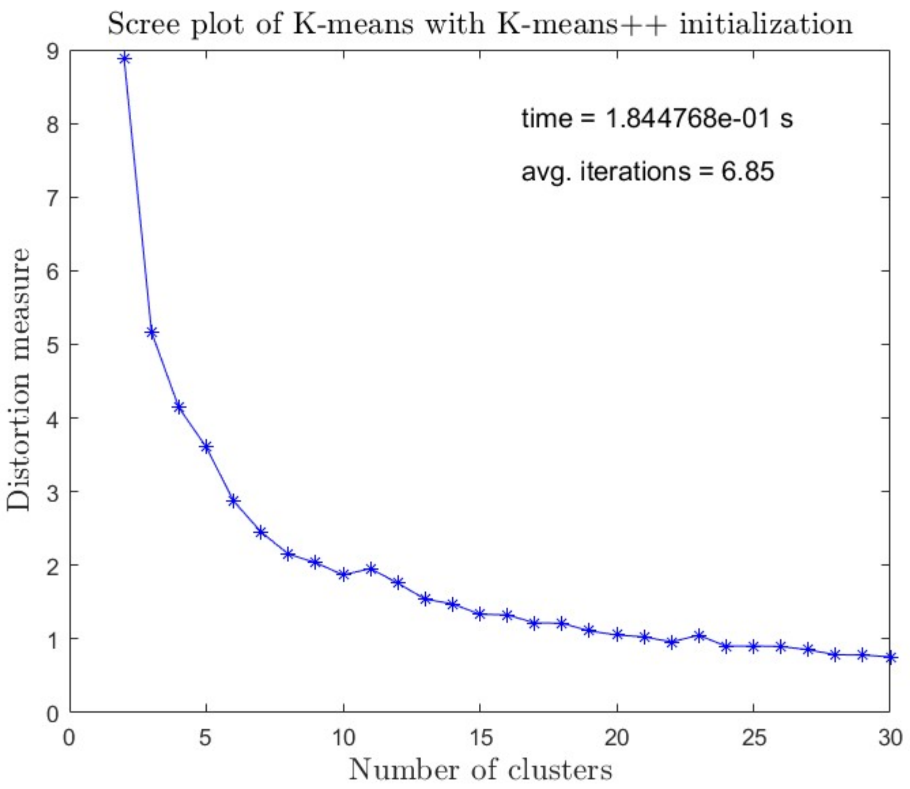


Fig. 4.1. Scree plot for 2 to 30 clusters using K-means++ initialization.

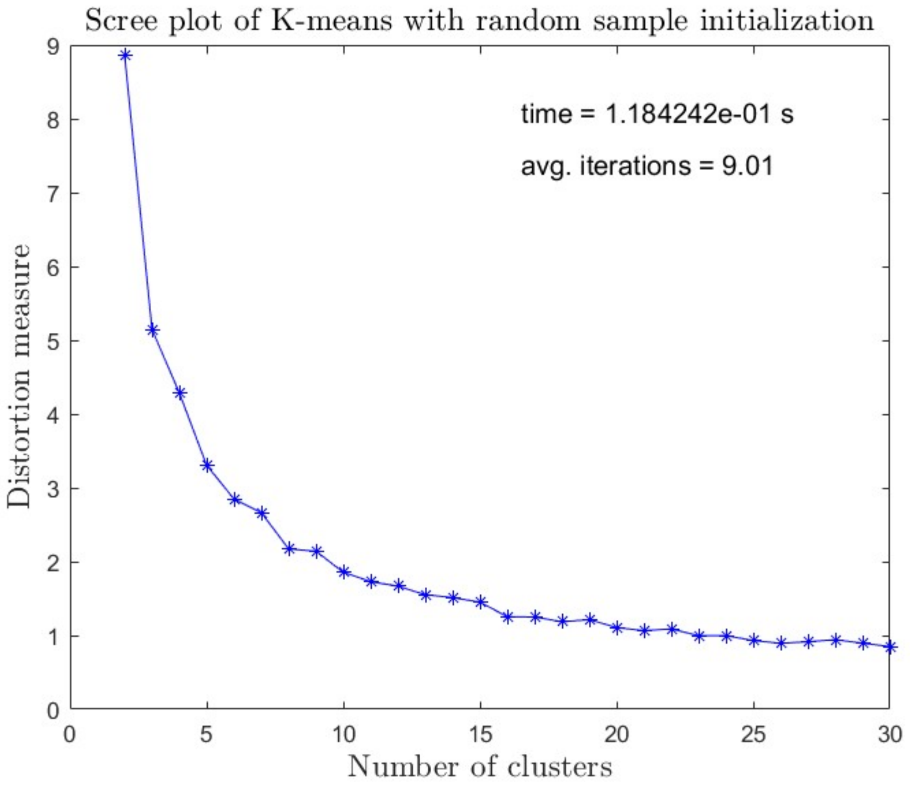


Fig. 4.2. Scree plot for 2 to 30 clusters using random sample initialization.

4.2. K-MEANS OF ENTIRE DATASET

The benefit of having a dataset *with* labels is that the performance of the K-means algorithm can be easily measured by simply tallying up the number of data points it correctly assigned. For instance, running K-means with K-means++ on the entire dataset results in a 0.89 accuracy index. In this report, "accuracy index" refers to to the ratio of correctly assigned data points to the total number of data points (210).

To allow for the easy visualization of the results of K-means, which cannot be done in 7D space, it is beneficial to choose an adequate 2D data space. In order to do so, it was deemed useful to apply K-means with K-means++ and 3 clusters to all 21 possible combinations of 2-variable cluster spaces. This would allow for the visualization of how each variable relates to the others, and, with the addition of the accuracy index, would indicate which pairs of variables lead to the most accurate clustering results.

The result of this process is depicted in Figure 4.3. The colors in each of the scatter plots correspond to the clusters created by the K-means algorithm (they do not indicate the true clusters). On the top left corner of every plot lies the accuracy index for that combination. Some attributes, like the perimeter and the area of the kernel or the perimeter and the length of the kernel, have a clear linear relationship, as would be expected. In general, when there is this type of relationship between variables, K-means is able to cluster the different data points fairly well, with accuracy indices usually above 0.80. When the data points are stacked in a more shapeless cloud along the vertical or horizontal axis, K-means has a much harder time finding the proper clusters. For instance, the scatter plot in the bottom right corner, which shows the relationship between the length of the groove and the asymmetry, has an accuracy index of just 0.56.

For the next section, Section 4.3, the chosen 2D data space is the area-asymmetry data space, which holds the highest accuracy index at 0.90. This data space is interesting as well because it shows no clear linear relationship and yet leads to good results. Other data spaces which could provide good visualization opportunities are perimeter-asymmetry or perimeter-compactness data spaces.

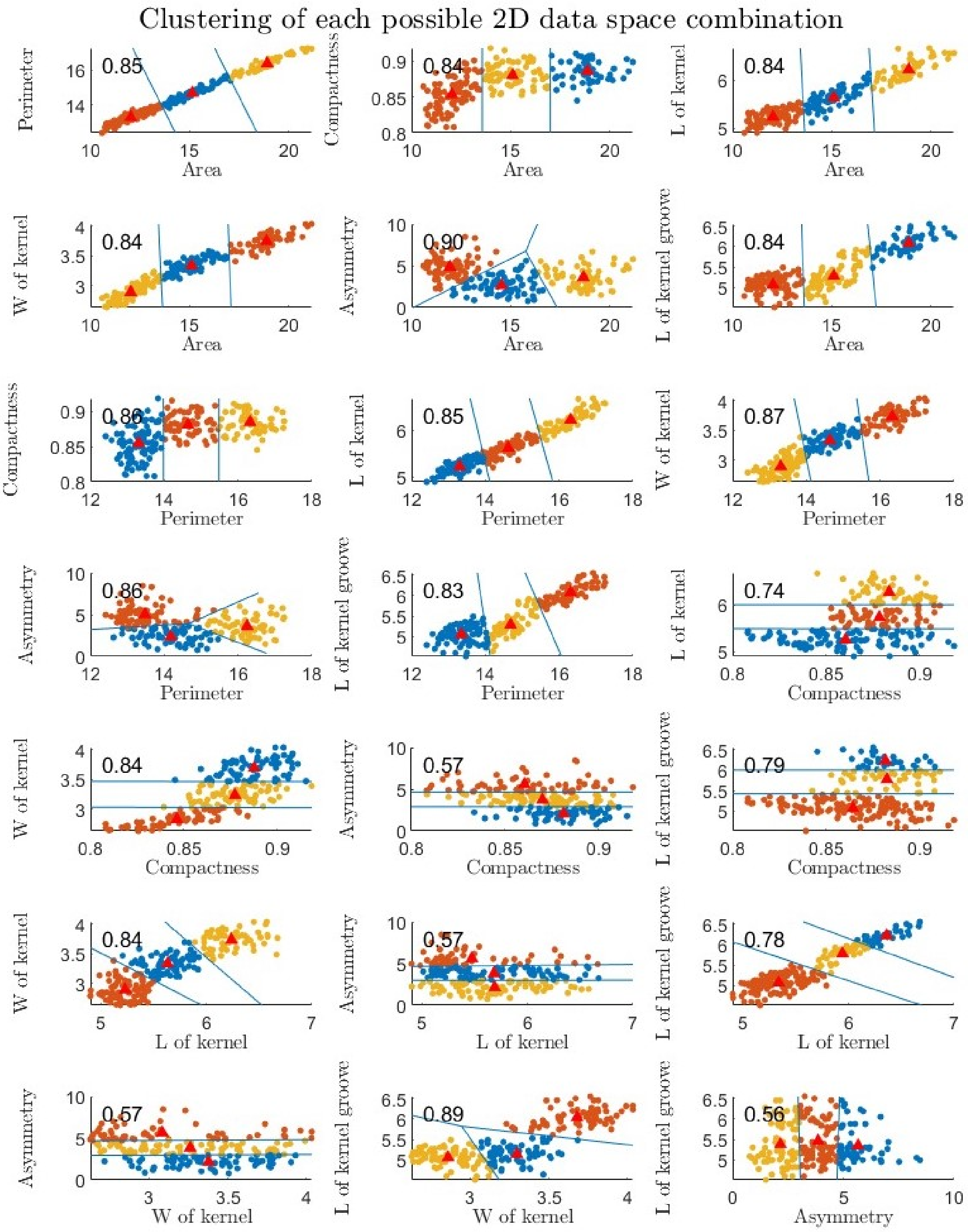


Fig. 4.3. Clustering of each possible 2D data space combination.

4.3. CLUSTERING OF AREA-SYMMETRY DATASPACE

This section contains three figures that aim to compare the accuracy of the K-means algorithm (with K-means++ initialization and 3 clusters) with the correct results given by the dataset's labels. Figure 4.4 plots the area-asymmetry data space and assigns colors to each data point based on the labels. The centroids of each section are also calculated so that the Voronoi diagram can also be displayed, all in green. Figure 4.5 contains the same data points, but this time their coloring is based on the clusters IDs from the K-means algorithm. Similarly, the centroids (in red) belong to the K-means clusters and the Voronoi diagram (in blue) is based on these as well.

Finally, Figure 4.6 overlays these two scatter plots on top of each other. Wherever the colors of the triangles and the circles match, K-means assigned the data points correctly. The confusion at the boundary between the yellow and blue clusters is interesting but not remarkable, as the blue cluster, mostly grouped around the bottom of the plot, has some surprising members around the middle-top of the plot. It is notable, however, that K-means mistakenly assigned the data points at the boundary of the orange and blue clusters, as those points seem to be much closer to the centroid of the orange cluster, as correctly indicated by the Voronoi diagram of Figure 4.4.

Figure 4.7 explores the ideas brought up in Section 4.1. K-means is able to find, as expected, more than three different clusters even though the dataset contains information for just three types of wheat. The colors in these plots correspond simply to the algorithm's own assignment. The scree plot in Figure 4.8, like in Section 4.1, seems to indicate that the more noticeable elbow appears at $k = 5$ rather than 3, even though the largest jump (not pictured in this figure) in the distortion measure appears between 2 and 3 clusters.

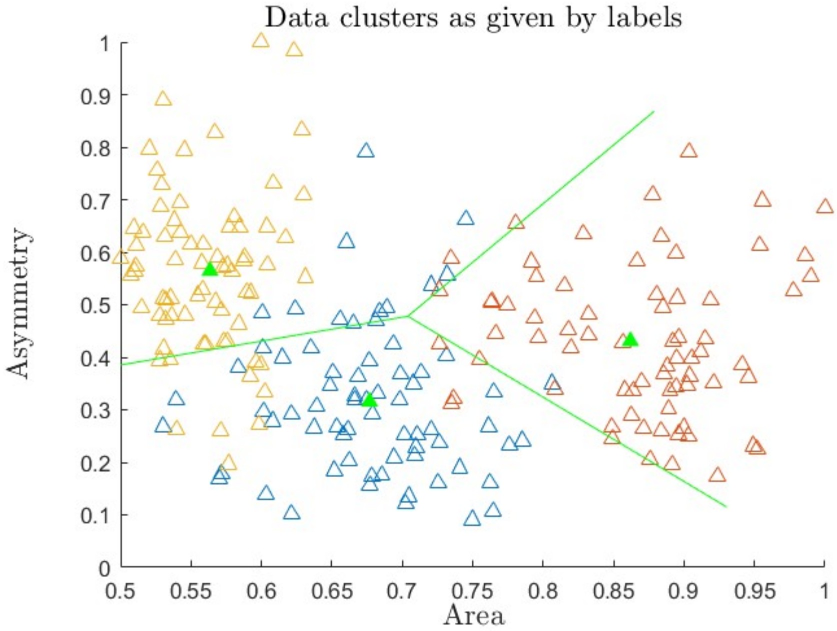


Fig. 4.4. Clusters, cluster centers and Voronoi diagram of data as given in labeled in dataset.

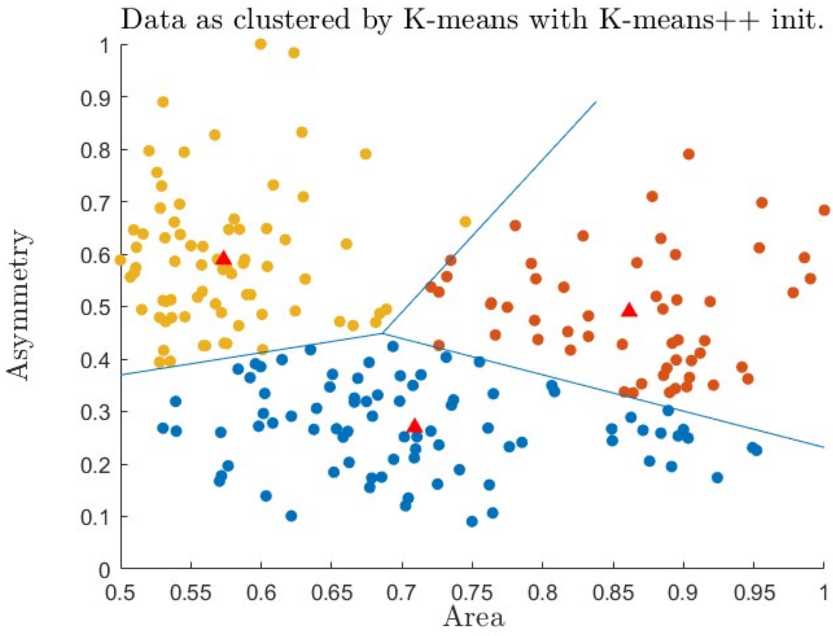


Fig. 4.5. Clusters, cluster centers and Voronoi diagram of data as given by K-means algorithm with K-means++ initialization.

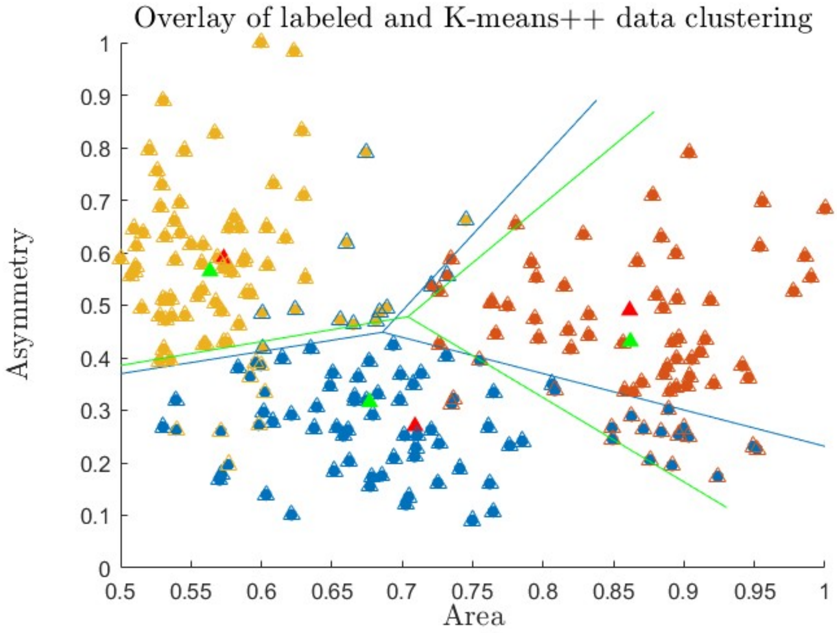


Fig. 4.6. Overlay of the Figures 4.4 and 4.5.

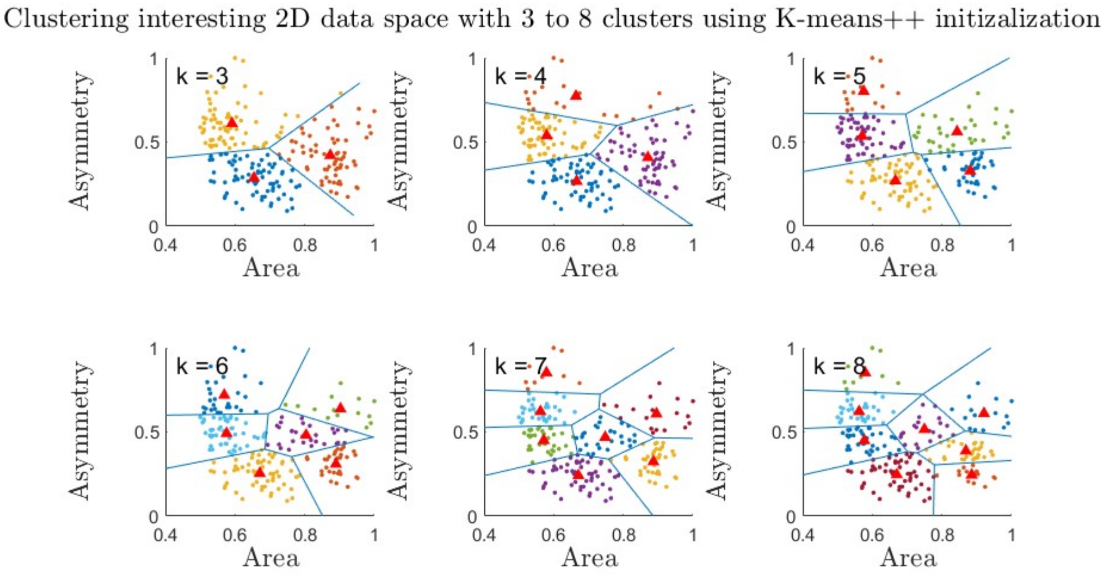


Fig. 4.7. Clustering using K-means with K-means++ as the number of clusters increases from 3 to 8.

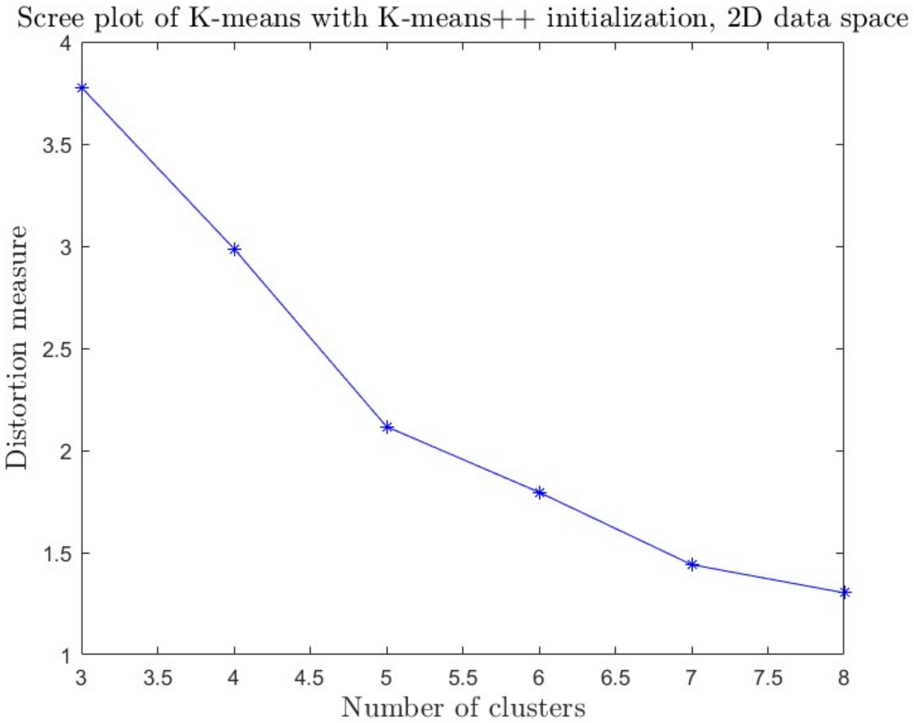


Fig. 4.8. Scree plot for 3 to 8 clusters using K-means++ initialization, area-symmetry data space

5. CONCLUSIONS AND DISCUSSION

Overall, this project was completed successfully. The K-means algorithm and the K-means++ initialization method were both explained with the help of formulas *and* visual examples. The dataset "seeds", its instances, and its attributes were described. The MATLAB tools used to apply K-means were indicated. Finally, the results of the application of K-means were displayed and discussed using a series of figures. These included scree plots of the entire data set using K-means++ and random sample initialization, K-means of all possible 2D data spaces, a detailed look at a specific 2D data space, a visual comparison with the correct clustering given by the labels in the dataset, and a quick visualization of the difference in clustering between different numbers of clusters.

For the purpose of generalization, the names of the wheat kernels in the data set were not mentioned in the results section, as the focus was on the performance of the algorithm rather than the data set itself. If this is the case, then perhaps the omission of the variable names would have provided a more complete generalization of the data set. At the same time, using the names of the wheat kernels rather than simply using the names of the cluster colors as in Section 4.3 could be interesting and definitely would be necessary if the purpose of the project was exclusively the study of the data set.

All in all, this project proved to be a valuable experience and provided a good introduction to the world of machine learning. The entire code and data for this project can be found in the following repository: https://github.com/R-SMP/RaulSMP_Kmeans [5].

BIBLIOGRAPHY

[1] V. Lavrenko, *K-means clustering: How it works*, 2014. [Online]. Available: https://www.youtube.com/watch?v=_aWzGGNrcic&list=PLBv09BD7ez_6cgkSUAqBXENXEhCkb_2wl&index=4.

[2] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[3] S. Jensen, *Kmeans++*, 2020. [Online]. Available: https://www.youtube.com/watch?v=HatwtJSsj5Q.

[4] J. N. M. Charytanowicz, *Seeds data set*, 2010. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/seeds.

[5] R. S. M. Peñas, *Raulsmp_kmeans*, 2023. [Online]. Available: https://github.com/R-SMP/RaulSMP_Kmeans.