

Assignment1

by Martin Pugsley

Submission date: 07-Jan-2022 08:36PM (UTC+0000)

Submission ID: 167916653

File name: 18449_Martin_Pugsley_Assignment1_393826_220564763.docx (586.38K)

Word count: 5270

Character count: 28719

Name: Martin Pugsley

Student ID: 1907620

Module Name: AI and Machine Learning (ACCA6011)

Search and ML Clustering

Table of Contents

Assignment: Search and ML Clustering	1
Table of Abbreviations	3
Table of Figures.....	3
1. Depth First Search Program.....	4
1.1. The DFS Algorithm Implemented.....	4
1.2. Variables and Functions	4
2. The K-Means Algorithm	8
2.1. Machine Learning Introduction	8
2.2. Introduction to K-Means.....	9
2.2.1. Number of Clusters (k) Selection	9
2.2.2. Initiate Starting Centroid Locations	12
2.2.3. Assign Observations to Centroids to Form Clusters	14
2.2.4. Centroid Recalculation	15
2.2.5. Ending K-Means	15
3. Calculated K-Means Example.....	16
3.1. Initial Observations and Centroids.....	16
3.2. Plotting the Starting Information.....	Error! Bookmark not defined.
3.3. Assign Observations to a Centroid	Error! Bookmark not defined.
3.4. Recalculate Centroids	Error! Bookmark not defined. 9
3.5. Re-Assign Observations to a Centroid	Error! Bookmark not defined. 9
3.6. Final Clustering.....	20
References	22
Appendices.....	23
Appendix A	24
Appendix B	25

Table of Abbreviations

Abbreviation	Meaning
DFS	Depth First Search

Table of Figures

Figure 1 – User Interface Menu	4
Figure 2 – Instructions when Option 1 Selected	4
Figure 3 - Instructions Prompting User to Load File First	5
Figure 4 – Depth First Search Initialised	5
Figure 5 – Depth First Search Encounters Sign ‘A’, pops it into stack.....	6
Figure 6 – Depth First Search’s Sign ‘A’, pushes back to Visited Vector	6
Figure 7– Depth First Search completed, Visited Vector Showing Directions Taken by Sign Value	7
Figure 8 – Silhouette Coefficient for an Observation	9
Figure 9 – Average Silhouette Calculation	10
Figure 10 – Silhouette Diagrams Depicting Results for Numerous (k) [10]	10
Figure 11 – Silhouette Formation [20]	11
Figure 12 – Elbow Chart with Inertia Scores [7].....	11
Figure 13 – Sorting by composite value and sharding [14].....	13
Figure 14 – Shard attribute means become centroid axis value [14]	13
Figure 15 – Resulting Naïve Sharding Starting Centroids as Yellow Circles [14]]	13
Figure 16 – Euclidean Distance Metric [16]	14
Figure 17 – Manhattan Distance Metric [16].....	14
Figure 18 – Chebychev Distance Metric [16]	14
Figure 19 – K-Means Centroid Calculation.....	15
Figure 20 – K-Means Flow Diagram	16
Figure 21 – Observation Dimensions	16
Figure 22 – Given Starting Centroid Dimensions.....	16
Figure 23 – Scatter-Graph Showing Starting Centroid and Observation Locations, Labelled with Name.....	17
Figure 24 – Observation Distance Measures from Centroids and Centroid Assignment by Distance Measuring Metric.....	18
Figure 25 – Scatter-Graph Showing Starting Centroid and Observations, Labelled with Assigned Centroid	18
Figure 26 – Current Centroid Locations and Observation Distance and Centroid Number Assigned ..	19
Figure 27 – Calculation to Find New Location for Centroid 1	19
Figure 28 – New Observation Distance Measures Based on Re-Calculated Centroids and Centroid Number Assigned	20
Figure 29 – Iterations for Centroid Re-Location, Observation Distance Measuring and Centroid Assignments	21
Figure 30 – Final Scatter-Graph Showing Final Centroid Locations and Observation Centroid/Cluster Assignment	21

1. Depth First Search Program

1.1 The DFS Algorithm Implemented

The DFS algorithm shows the agent traverses left until a wall is hit, recording signs in the 'stack' and 'visited' vectors on the way, then does the same in the right-direction, the down-direction and the up-direction. Once the up-direction wall is hit, the process loops to start again until there is nowhere else to go. Conditional-statements control what the agent can do by either leaving a 'trail' to say the area has been visited or not [Pseudocode Appendix A].

1.2 Variables and Functions

When program is initially loaded, the user is face with a menu, listing options 1 to 4, which can be selected [Figure 1].

```
1: Load a File
2: View Map
3: Carry out Depth First Search
4: End program
Please enter a choice (1 to 3):
```

Figure 1 – User Interface Menu

The option selected is assigned to integer variable 'x'.

Conditional-Statements control functions based on the user selection. If user selects 1, instructions to load a file are shown [Figure 2]. Once a file is selected, a function called 'loadFile' loads the file into memory by reading the text-file line-by-line and integer variable 'fileLoaded' is assigned 1. When the user selects 2, the 'printArray' function is run and displays the text-file in memory. If user selects 2 or 3 before 1 then variable 'fileLoaded' is still zero and conditional-statements address this, prompting instructions to load a file [Figure 3].

```
1: Load a File
2: View Map
3: Carry out Depth First Search
4: End program
Please enter a choice (1 to 3): 1

Enter file name you want to carry out a depth first search on

Hint: The file that works with this program is called layout.txt,
if you want to use another file please move it to the programs location
```

Figure 2 – Instructions when Option 1 Selected

```
1: Load a File
2: View Map
3: Carry out Depth First Search
4: End program
Please enter a choice (1 to 3): 3

*** Please Load a File First ***

1: Load a File
2: View Map
3: Carry out Depth First Search
4: End program
Please enter a choice (1 to 3):
```

Figure 3 – Instructions Prompting User to Load File First

When 3 is selected, the DFS is initiated, the floor-layout is printed showing: signs represented by letters, spaces represented with ' ' and walls represented with '*' characters.

```
*****  
*S   A *  
**** *****  
  
*C*      *  
* **** *  
*  *    *  
* **   *  
  
* B     D*  
* **** *  
  
*E      F*  
*****
```

Surroundings Count = 3
Current Stack = ***
Visted = S

*** INSTRUCTIONS ***

*** Step 1 - Traverse left with 'a' key until '* Wall' hit ***
*** Step 2 - Traverse right with 'd' key until '* Wall' hit ***
*** Step 3 - Traverse Down with 's' key until '* Wall' hit ***
*** Step 4 - Traverse Up with 'w' key until '* Wall' hit ***
*** Step 5 - Go Back to Step 1 and Repeat Process until Completed Instrcutions Appear ***

Figure 4 – Depth First Search Initialised

The agent starting-point is 'S', and is already visited, so is placed in the 'Visited' vector. Figure 4 shows this, also showing a 'Surroundings Count' and 'Current Stack'.

The 'Surroundings Count' is an integer variable 'count' value. Adjacent spaces to the current space are tested to see what value they hold. Character variables called 'test1' to 'test4' are used to capture the value of adjacent positions, for left, right, down and up respectively. Each test returns either '*', ' ' or letter (wall, space or sign) values and each '*' the tests find increase the 'count' variable by 1, which is used in conditional-statements to determine what the agent can do. This test is looped throughout the traversal process.

The 'Current Stack' is the 4 test variables put together in an output statement and effectively shows in the surrounding 4 adjacent spaces what's found (wall, space or sign), whereas 'Visited' is a string created from a vector. The reason it is a string is C++ doesn't allow a vector to be amended, but a string can by using the 'push.back' function, so when the agent moves to an adjacent square which contains a sign, it is removed from the 'Current Stack' and the value is then 'pushed back' to the 'Visited' string [Figure 5].

```

*****
***** QA *
*****
*C*      *
* *****
* *      *
* *      *
* * *****
* *B      D*
* *****
*E        F*
*****

Surroundings Count = 2
Current Stack = A **
Visted = S

*** INSTRUCTIONS ***

*** Step 1 - Traverse left with 'a' key until '* Wall' hit ***
*** Step 2 - Traverse right with 'd' key until '* Wall' hit ***
*** Step 3 - Traverse Down with 's' key until '* Wall' hit ***
*** Step 4 - Traverse Up with 'w' key until '* Wall' hit ***
*** Step 5 - Go Back to Step 1 and Repeat Process until Completed Instrcutions Appear ***

```

Figure 5 – Depth First Search Encounters Sign 'A', pops it into stack

```

*****
***** Q *
*****
*C*      *
* *****
* *      *
* *      *
* * *****
* *B      D*
* *****
*E        F*
*****

Surroundings Count = 2
Current Stack = **
Visted = SA

*** INSTRUCTIONS ***

*** Step 1 - Traverse left with 'a' key until '* Wall' hit ***
*** Step 2 - Traverse right with 'd' key until '* Wall' hit ***
*** Step 3 - Traverse Down with 's' key until '* Wall' hit ***
*** Step 4 - Traverse Up with 'w' key until '* Wall' hit ***
*** Step 5 - Go Back to Step 1 and Repeat Process until Completed Instrcutions Appear ***

```

Figure 6 – Depth First Search's Sign 'A', pushes back to Visited Vector

The user traverses the file by pressing the 'a', 'd', 's', or 'w' keys (case-sensitive) to go left, right, down or up respectively [Instructions - Appendix B], the switch function is used to carry out movements based on user input within the loop.

Appendix A displays the pseudocode used in traversal but to summarise, for all directions except for down, if the 'Surroundings Count' is equal to 3, move to the new space and replace with avatar, old space is replaced with '*' so cannot go back to it, but if the 'Surroundings Count' is less 3, move to the new space and replace with avatar, old space is replaced with ' ' so can go back there. For moving down, every move results in the old space being replaced with a '*'. Adjacent values of spaces are placed into the stack and once the square with a sign is landed on, the value gets added to the 'Visited' vector. When 'Surroundings Count' is equal to 4, then there is nowhere else to go, and the search is ended, the 'visited' vector displays the directions taken [Figure 7].

```
*****
*****
*****
*Q*****
*****
*****
*****
*****
*****
*****
*****

Surroundings Count = 4

*** SEARCH ENDED - No where else to visit, please press 'z' to end ***

Visted = SABDFEC
```

Figure 7 – Depth First Search completed, Visited Vector Showing Directions Taken by Sign Value

2. The K-Means Algorithm

2.1 Machine Learning Introduction

The term “Machine Learning” was first used in 1959 within a journal by Arthur Samuel [1]. Samuel’s research described how a computer could be programmed to learn a game of checkers and successfully beat a human and states *“Programming computers to learn from experience should eventually eliminate the need for much of this detailed programming effort”* [1]. What I believe Arthur meant by this is a Machine Learning algorithm can be used to continuously learn from observations and improve its accuracy over time without the need to change or upgrade any code, essentially imitating the way humans learn.

Arthur’s work is a major milestone in the Machine Learning world but since then the field has advanced massively due to increased processing powers and technological advancements. This has resulted in many different Machine Learning types, but Geron (2019) summarises they can be classified into broad categories based on their characteristics, the first category noted is *“Whether or not they are trained with human supervision (supervised, unsupervised, semisupervised, and Reinforcement Learning)”* [2]. In supervised learning, the algorithm is trained with data which has been labelled, meaning the data is accompanied with the desired solution, but additionally may contain extra pieces of information, called predictors (variables etc), which relate to the solution. Geron (2019), provides us with an example of supervised learning with the task of predicting car prices based on data which contains the value of the car (the label) but also additional information around mileage, brand, age etc (the predictors) [3]. To train the algorithm in predicting the car price, when the introduction of new data containing no label is made, many examples of car data containing both labels and predictors must be given to the algorithm, this is called the training data. Once the algorithm has the training data it can predict a label for the new data based on its predictors by cross-referencing for similarities in the training data.

In contrast to supervised learning is unsupervised learning. IBM describe unsupervised learning algorithms as ways to *“discover hidden patterns or data groupings without the need for human intervention”* [4]. The unsupervised learning algorithm is trained with data, which is unlabelled, so the data is not accompanied by a desired solution, and it is the job of the algorithm to try and learn without being taught. Norvig and Russell (2016), state the most common supervised learning task is clustering [5], where the algorithm will detect potentially useful clusters within a given dataset by calculating similarities in the data attributes. Many clustering algorithms exist but one of the most popular if not the most popular is K-Means.

2.2 Introduction to K-Means

An unsupervised learning algorithm, K-Means is a very common approach used for clustering data-points, commonly known as observations, into groups based on similarities in the data and this will display underlying patterns. On paper, the K-Means algorithm looks like a simple step-by-step process:

1. Choose the number of clusters to sort observations into
2. Select starting centroid (cluster centre) locations for the clusters
3. Assign all observations to the closest cluster centroid
4. Re-calculate centroid locations base on the newly formed clusters
5. Repeat steps 3 and 4 until a stopping-criteria is met

But as you may be aware, entire process isn't fully automated, the user still needs to make decisions to instruct the algorithm how to operate. The decisions the user will take are better known as hyperparameters.

The following documentation will explain and evaluate each step in the K-Means algorithm, including how the user decides on the number of clusters, initial cluster centroid locations and what equations to use in distance measuring. The documentation approaches the K-Means algorithm in a two-dimensional manner, but K-Means can be used in data which contains more.

2.2.1 Number of Clusters (k) Selection

The first hyperparameter the user decides on is the number of clusters to use. As previously mentioned, k-means is an unsupervised method, so the data has no labels, therefore the user doesn't know how many clusters there are to begin with, making the selection process quite complex, but there are methods to aid the user in the decision process, the two most common being the Silhouette and Elbow methods.

The Silhouette method measures the quality of a clustering by calculating how well each observation lies within a cluster. This method provides a test, which is carried out on numerous cluster amounts by first calculating the average distance between a single observation and all other observations in the same cluster, this is known as the mean intra-cluster distance, then secondly the average distance between the single observation and all other observations in the next nearest cluster is calculated, known as the mean nearest-cluster distance [6].

With these measurements it is possible to calculate a silhouette coefficient [Figure 8] of an observation $S(i)$, where the mean intra-cluster distance is $a(i)$ and the mean nearest-cluster distance is $b(i)$.

$$S(i) = \frac{b(i) - a(i)}{\max \{a(i), b(i)\}}$$

Figure 8 – Silhouette Coefficient for an Observation

Once this Silhouette coefficient has been calculated for all observations, an average value of a silhouette is calculated by using the mean of the silhouette coefficients in each cluster.

$$\text{AverageSilhouette} = \text{mean}\{S(i)\}$$

Figure 9 – Average Silhouette Calculation

Results of calculations are then displayed in a visualisation, depicting a silhouette coefficient score from -1 to 1 on the x-axis and the number of clusters (k) on the y-axis. The silhouette diagrams [Figure 10] depict results for tests on 3 to 6 clusters, where the average silhouette score is represented by the vertical dashed line and the silhouette coefficients are represented by the bars. To determine good clustering, all the bars should fall to the right of the average silhouette score line, as figure 3 shows for tests run on 4 and 5 clusters, whereas some bars fall short on tests run for 3 and 6 clusters depicting bad clustering.

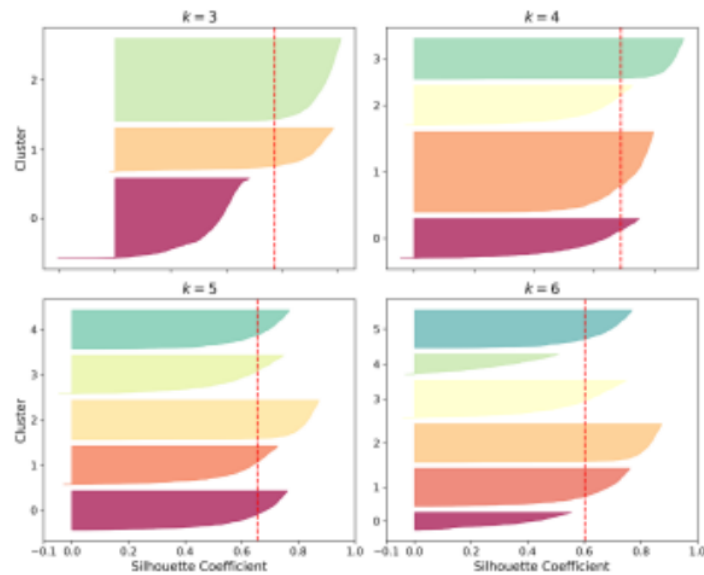


Figure 10 – Silhouette Diagrams Depicting Results for Numerous (k) [10]

Bar thickness is also an indicator to determine good clustering, for instance one of the bars in the 4-cluster diagram is thicker than others, this shows there are more observations below the average silhouette score than above. Rousseeuw (1986), provides an example [Figure 11] of how silhouettes are formed, where observation silhouette coefficients are represented by dots scored against the x-axis metrics, showing the thicker the bar, the more observations in the cluster [20].

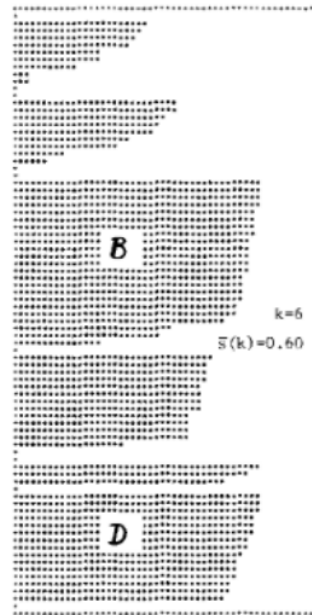


Figure 11 – Silhouette Formation [20]

With this knowledge, using 5 clusters seems the optimal choice from the diagram at figure 10, as the cluster bars are of similar size and are all above the average silhouette score, this represents good clustering with even volumes of observations, therefore the user would have a clear visual to select 5 clusters as the first hyperparameter.

The Elbow method alternatively measures the ‘SSE’ (Sum of Square Error), sometimes known as the ‘WCSS’ (Within-Cluster Sum of Square), for various cluster numbers. This simply means the squared distance between each observation and the cluster centroid is calculated and all values are summed to form a grand total, this value is known as the ‘inertia’. Once inertia values have been calculated for when using various numbers of clusters, they are plotted into a chart, where the values decrease with the increase in number of clusters [7].

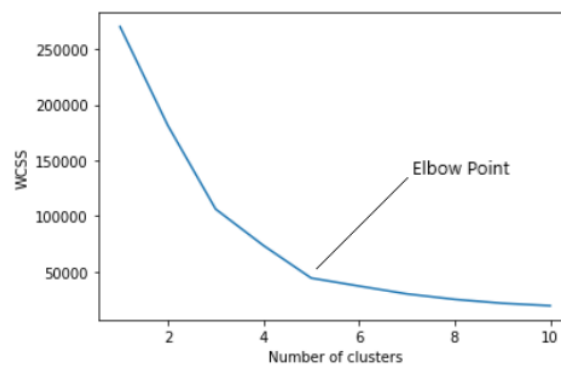


Figure 12 – Elbow Chart with Inertia Scores [7]

Analysing the chart [Figure 12], it would be easy to say the lower the inertia value the better the model but that is not the case. Geron (2019) states '*the inertia is not a good performance measure*' because it will keep on decreasing with the more number of clusters [8], in fact it is the 'elbow' of the chart – the point where the chart starts to move almost parallel to the x-axis [7], which the user should use as the indicator for the number of clusters to use, in the above example the number of clusters to use would be seen as 5.

Overall, both Silhouette and Elbow methods seem to be the most common tools to help decide on the number of clusters, but Charrad (2015) released the R Programming language package 'NbClust', claiming to contain 30 different methods to assist in deciding on the number of clusters to use [9], each having different mathematical equations to measure distances of observations from cluster centroids, but ultimately there isn't a 'go-to' method for giving an absolute cluster number value, the user will still have to use their judgement in deciding, although Geron (2019) recommends the Silhouette above the Elbow Method, quoting it as '*more precise approach*' [10], but my opinion is each is valuable and can be used in conjunction with each other in deciding on the number of clusters (k).

2.2.2 Initiate Starting Centroid Locations

When the first hyperparameter (k) is decided on, locations for an equal number of cluster centroids must be initialised. Numerous tutorials can be searched on k -means and most instructions just simply state 'select starting centroids at random'. Although this can be completed to a certain level of accuracy, by plotting the observations into a scatter-graph and use judgement to select appropriate looking starting centroids, there are other approaches to take, such as k -means++ and naïve-sharding.

Ali (2021) explains to start the K -Means++ process, an original observation is selected at random as the first cluster starting centroid, the method then uses the Euclidean distance metric (explained in 2.2.3) to calculate the distances between the first cluster starting centroid and each of the remaining observations. The observation calculated to be furthest away from the first cluster starting centroid is selected as the next cluster centroid and once this is done clusters are formed from the remaining observations by assigning them to the nearest centroid. If more centroids need to be plotted the distance of each observation is again calculated against their assigned centroid and the observation with the greatest distance is chosen as the next centroid, the process is repeated until k has been reached [11].

The Naïve-Sharding approach calculates starting centroids through a composition of the observations value attributes. Mayo (2017) provides an overview [Figure 13], where a dataset is displayed in a table, where each row is a single observation (Mayo uses the word instance) and the relating value attributes to the observation are in the columns. A summarising column is then added to the table which totals the values of each observation's attributes and now that each row has a summed total value, the table is sorted from highest to lowest based on the summarised column. The observations are then divided into k equal groups, known as shards and then for each shard, the summarised column is averaged to form a value, this is used as one of the centroids axis. To get the other axis, the total value of each attribute column is calculated to form a new row in the shard (the summarised total column completed in the previous step is not included here), the average of the calculated column values is then used as the centroids second axis [12].

2.2.3 Assign Observations to Centroids to Form Clusters

Now the starting centroid locations have been established, each observation is then assigned to a centroid to form the clusters. The distance between each observation and the different centroids is calculated and the observation is then assigned to the nearest centroid. Again, the user must decide which mathematical approach is taken to measure the distances between the observations and centroids. Singh, Yadav and Rana (2013) illustrate the k-means algorithm using three different distance measuring metrics: the Euclidean, Manhattan and Chebychev [13].

The Euclidean distance metric derives from the Pythagoras theorem which measures the distance between two points or is known as the straight-line distance. The metric calculates the root of square difference between the co-ordinates of a pair of points [13] [Figure 16]. This metric is generally classed as the standard approach with calculating distances within k-means [14]

$$\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

Figure 16 – Euclidean Distance Metric [16]

The Manhattan distance metric, also known as the taxicab or city block distance measure, is generally used in measuring distances when it is not possible to travel in a straight line, for example, if a wall or building was between 2 location points. The metric calculates the absolute difference between the x and y-axis of 2 points and adds them together to provide a distance measure [15].

$$|x_1 - x_2| + |y_1 - y_2|$$

Figure 17 – Manhattan Distance Metric [16]

The Chebychev distance metric, also known as the Maximum Value Distance, is similar to the Manhattan distance metric, but instead of adding the differences in values together, simply takes the maximum value of the two as the distance measure.

$$\max(|x_1 - x_2|, |y_1 - y_2|)$$

Figure 18 – Chebychev Distance Metric [16]

Grootendorst (2021) writes there are up to 9 different distance measures which can be used [17] but Singh, Yadav and Rana (2013) tested the k-means algorithm with these three metrics and concluded the Euclidean distance metric worked best within the examined dataset, due to it being two-dimensional and the straight-line [13], however I believe it may be beneficial to test a few metrics to see if variances occur, especially if there is more than two dimensions.

2.2.4 Centroid Recalculation

The initial clusters have been formed at this stage, but the starting centroid locations are just that, so now the k-means algorithm re-calculates the centroid locations. It does this by calculating the mean of the observation dimensions currently assigned to the cluster [18], hence the name K-Means. For example, if we have a starting centroid which was calculated to be location 6, 4 (x, y-axis) and 2 observations were assigned to it at locations 4, 4 and 8, 6 respectively, then the new location of the centroid is calculated to be at location 6, 5 [Figure 19].

$$\begin{array}{rclcl} & & X & & Y \\ 4 & + & 8 & = & 12 \\ \hline & & 2 & & 2 \\ & = & 6 & & = 5 \end{array}$$

Figure 19 – K-Means Centroid Calculation

This calculation is completed for all centroids and because of the re-location of the centroids after the calculation completion, some observations may now be closer to a different centroid, so the distance of the observations against all centroids is calculated again, using the same metric measure decided, when assigning observations to centroids in the previous step.

2.2.5 Ending K-Means

Due to this re-calculation of the centroids and observation cluster assignment, the reader may start to think at what point does the algorithm stop? Sharma (2019) describes 3 main stopping criteria to end the K-Means algorithm, these being:

1. There is no change in position after calculated centroid re-location
2. Observations remain in the same cluster
3. Maximum number of calculation iterations is reached

If either stopping criteria 1 or is met, this known as the convergence and the k-means algorithm is complete. Then the final clusters can be seen, with centroid locations and the observations assigned to each [19].

In summary, this is a simple explanation of k-means, but the volume of observations may be in the thousands or more, so to do this by hand becomes a very complex process. Luckily there are tools available such as programming languages, for example Python, which offers packages, such as sklearn, where the k-means mathematical processes have been made into functions for the user to call on, this simplifies the k-means algorithm when dealing with large datasets.

3. Calculated K-Means Example

3.1 Initial Observations and Centroids

The following section will show an example of calculating clusters using the k-means algorithm.

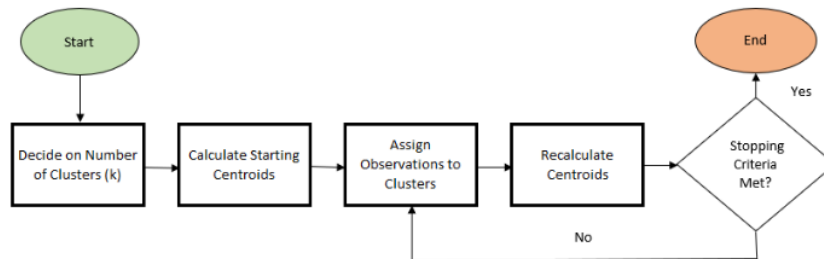


Figure 20 – K-Means Flow Diagram

For this example, the observations needed to be assigned to clusters are shown in figure 21 and the first two steps of the flow diagram can be skipped because the number of clusters (2) and initial starting centroid positions (Centroid 1 = 4.32, 4.92, Centroid 2 = 1.15, 0.33) have already been given [Figure 22].

DataPointNo	X	Y
1	5	5
2	0	5
3	2	3
4	2	2
5	5	1
6	3	3
7	1	4
8	5	3
9	1	2
10	4	2

Figure 21 – Observation Dimensions

ID	StudentNo	X C1	Y C1	X C2	Y C2
20	1907620/1	4.32	4.92	1.15	0.33

Figure 22 – Given Starting Centroid Dimensions

3.2 Plotting the Starting Information

With this information, it is possible to plot the values into a scatter-graph. This will show an initial view of the starting centroid positions and observations. Figure 23 shows this view and labels each observation and centroid with their name.

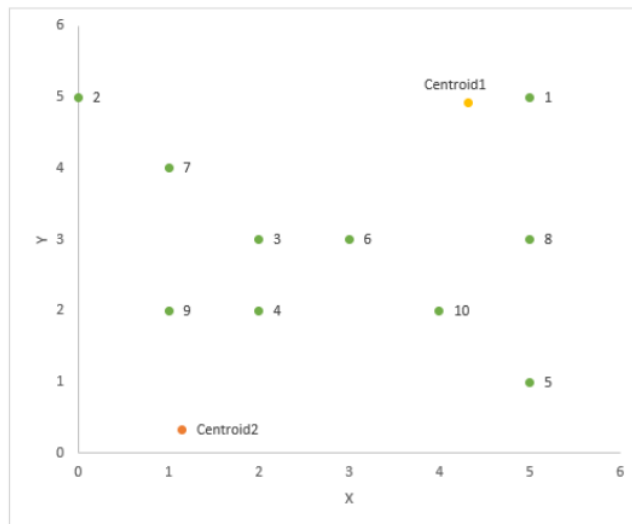


Figure 23 – Scatter-Graph Showing Starting Centroid and Observation Locations, Labelled with Name

3.3 Assign Observations to a Centroid

The observations are then assigned to a cluster, but before doing so, a decision on the distance measuring metric is required.

Following the approach of Singh, Yadav and Rana (2013), the three main distance measuring metrics are examined (Euclidean, Manhattan and Chebychev). Figure 17, columns F to K shows the distances for each observation against each centroid, using the three distance measuring metrics and as shown, some differences occur in the resulting measures. Columns L to N then show the centroid number assigned to the observation, based on shortest distance in the measuring metrics results. Figure 24 also shows the calculations used for the different metrics when calculating the first observation's distance against centroid 1.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1														
2														
3														
4														
5														
6														
7														
8														
9														
10														
11														
12														
13														
14														
15														
16														
17														
18														
19														
20														
21														
22														
23														
24														
25														
26														
27														
28														
29														
30														
31														
32														
33														
34														
35														
36														
37														
38														
39														
40														
41														
42														
43														
44														
45														
46														
47														
48														
49														
50														
51														
52														
53														
54														
55														
56														
57														
58														
59														
60														
61														
62														
63														
64														
65														
66														
67														
68														
69														
70														
71														
72														
73														
74														
75														
76														
77														
78														
79														
80														
81														

Figure 24 – Observation Distance Measures from Centroids and Centroid Assignment by Distance Measuring Metric

Due to the differences in distance measures a decision must be made on which to use in measuring distances going forward. The most logical approach is to use the Euclidean distance metric because the data is two-dimensional and for this geometry, the shortest distance is indeed a straight line, so for this example the Euclidean distance metric has been used.

Figure 24 already shows the calculated distances, for each observation, using the Euclidean metric, and centroids have been assigned to each using the shortest distance value, so the assigned centroid numbers can now be assigned as the labels for each observation in the scatter-graph [Figure 25] to essential show the observations in their current cluster.

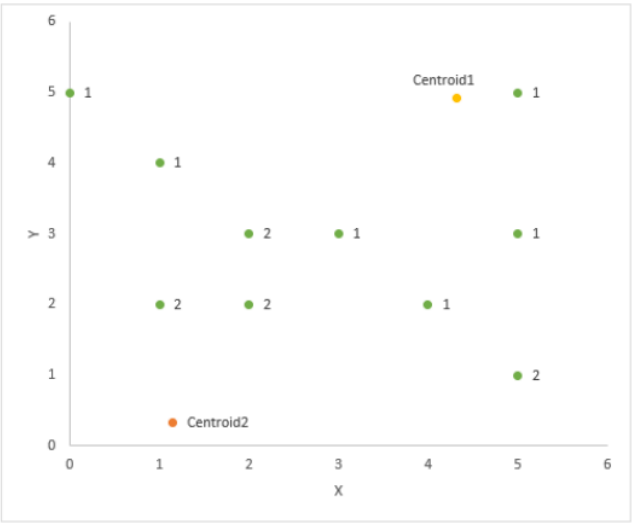


Figure 25 – Scatter-Graph Showing Starting Centroid and Observations, Labelled with Assigned Centroid

3.4 Recalculate Centroids

Now the observations have been assigned to a centroid, the re-calculation of centroid locations takes place. Figure 26 shows the current observations distance and the centroid/cluster number they have been assigned to.

Starting Point	DataPointNo	X	Y	Euclidean D1	Euclidean D2	Cluster (EUC)
	Centroid1	4.32	4.92			
	Centroid2	1.15	0.33			
	1	5	5	0.68	6.05	1
	2	0	5	4.32	4.81	1
	3	2	3	3.01	2.80	2
	4	2	2	3.73	1.87	2
	5	5	1	3.98	3.91	2
	6	3	3	2.33	3.25	1
	7	1	4	3.45	3.67	1
	8	5	3	2.04	4.69	1
	9	1	2	4.42	1.68	2
	10	4	2	2.94	3.30	1

Figure 26 – Current Centroid Locations and Observation Distance and Centroid Number Assigned

From here, the mean of the total observation values, for both X and Y against relevant centroid number is calculated, to give the new centroid location axis values. Figure 27 shows how the X and Y values for centroid 1 have been taken to form the new centroid location.

$$\begin{array}{rcl}
 \text{X} & & \text{Y} \\
 \hline
 5 + 0 + 3 + 1 + 5 + 4 = 18 & & 5 + 5 + 3 + 4 + 3 + 2 = 22 \\
 \text{Number of observations} = 6 & & \text{Number of observations} = 6 \\
 \hline
 \frac{18}{6} = 3.00 & & \frac{22}{6} = 3.67 \\
 \hline
 \text{New Centroid 1 Location} = \text{X (3.00), Y (3.67)}
 \end{array}$$

Figure 27 – Calculation to Find New Location for Centroid 1

3.5 Re-Assign Observations to a Centroid

As the new centroid locations have been re-calculated, the observation distances are re-measured. Figure 28 shows that centroid 1 originally assigned to observation 10 has now changed to centroid 2 because it is now closer to this centroid after its re-location.

Iteration 1	DataPointNo	X	Y	Euclidean D1	Euclidean D2	Cluster (EUC)	Compare	Converge
	Centroid1	3.00	3.67					
	Centroid2	2.50	2.00					
	1	5	5	2.40	3.91	1	TRUE	
	2	0	5	3.28	3.91	1	TRUE	
	3	2	3	1.20	1.12	2	TRUE	
	4	2	2	1.94	0.50	2	TRUE	
	5	5	1	3.33	2.69	2	TRUE	
	6	3	3	0.67	1.12	1	TRUE	
	7	1	4	2.03	2.50	1	TRUE	
	8	5	3	2.11	2.69	1	TRUE	
	9	1	2	2.60	1.50	2	TRUE	
	10	4	2	1.94	1.50	2	FALSE	FALSE

Figure 28 – New Observation Distance Measures Based on Re-Calculated Centroids and Centroid Number Assigned

It is because of the change in observation 10's centroid assignment, another iteration of centroid re-calculation must take place, as there is no convergence.

3.6 Final Clustering

The same calculation of centroid re-location and centroid assignment to observations at 3.4 and 3.5 were carried out, in fact it is not until iteration 3 that convergence is reached. Figure 29 shows the calculations of the different iterations (2-4) and finally at iteration 3, the centroid numbers have not changed from the previous iteration, but a fourth iteration was run to see if anything would change, but this resulted in no change to the centroid number assignment to observations, nor in the centroid locations X and Y values, therefore the stopping-criteria has been met and the k-means algorithm has ended.

We can now re-plot the scatter-graph [Figure 30] to view the final centroid locations, observation centroid assignments and clusters.

Iteration 2	DataPointNo	X	Y	Euclidean D1	Euclidean D2	Cluster (EUC)	Compare	Converge
	Centroid1	2.80	4.00					
	Centroid2	2.80	2.00					
	1	5	5	2.42	3.72	1	TRUE	
	2	0	5	2.97	4.10	1	TRUE	
	3	2	3	1.28	1.28	2	TRUE	
	4	2	2	2.15	0.80	2	TRUE	
	5	5	1	3.72	2.42	2	TRUE	
	6	3	3	1.02	1.02	2	FALSE	
	7	1	4	1.80	2.69	1	TRUE	
	8	5	3	2.42	2.42	2	FALSE	
	9	1	2	2.69	1.80	2	TRUE	
	10	4	2	2.33	1.20	2	TRUE	FALSE
Iteration 3	DataPointNo	X	Y	Euclidean D1	Euclidean D2	Cluster (EUC)	Compare	Converge
	Centroid1	2.00	4.67					
	Centroid2	3.14	2.29					
	1	5	5	3.02	3.29	1	TRUE	
	2	0	5	2.03	4.15	1	TRUE	
	3	2	3	1.67	1.35	2	TRUE	
	4	2	2	2.67	1.18	2	TRUE	
	5	5	1	4.74	2.26	2	TRUE	
	6	3	3	1.94	0.73	2	TRUE	
	7	1	4	1.20	2.74	1	TRUE	
	8	5	3	3.43	1.99	2	TRUE	
	9	1	2	2.85	2.16	2	TRUE	
	10	4	2	3.33	0.90	2	TRUE	TRUE
Iteration 4	DataPointNo	X	Y	Euclidean D1	Euclidean D2	Cluster (EUC)	Compare	Converge
	Centroid1	2.00	4.67					
	Centroid2	3.14	2.29					
	1	5	5	3.02	3.29	1	TRUE	
	2	0	5	2.03	4.15	1	TRUE	
	3	2	3	1.67	1.35	2	TRUE	
	4	2	2	2.67	1.18	2	TRUE	
	5	5	1	4.74	2.26	2	TRUE	
	6	3	3	1.94	0.73	2	TRUE	
	7	1	4	1.20	2.74	1	TRUE	
	8	5	3	3.43	1.99	2	TRUE	
	9	1	2	2.85	2.16	2	TRUE	
	10	4	2	3.33	0.90	2	TRUE	TRUE

Figure 29 – Iterations for Centroid Re-Location, Observation Distance Measuring and Centroid Assignments

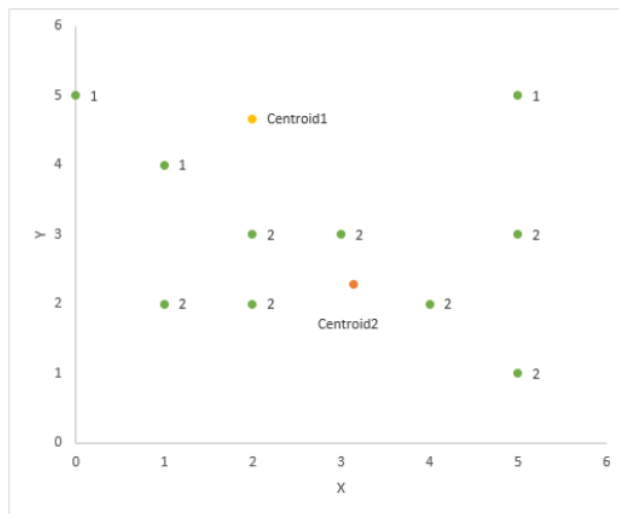


Figure 30 – Final Scatter-Graph Showing Final Centroid Locations and Observation Centroid/Cluster Assignment

References

- [1] A. Samuel [1959]: Samuel. A, "Some Studies in Machine Learning Using the Game of Checkers", IBM Journal, July, 1959
- [2] A. Geron. A [2019]: Geron. A, "Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow", 2nd ed, O'Reilly, 2019, pp. 8
- [3] A. Geron. A [2019]: Geron. A, "Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow", 2nd ed, O'Reilly, 2019, pp. 9
- [4] IBM Cloud Education [2020]: "Unsupervised Learning", lbn.com, September, 2020. [Online]. Available: <https://www.ibm.com/cloud/learn/unsupervised-learning> [Accessed: Nov. 29, 2021]
- [5] S. J. Russell & P. Norvig [2016]: Russell. S.J & Norvig. P, "Artificial intelligence a modern approach", 3rd ed. Boston, Mass: Pearson, 2016, pp. 695
- [6] A. Kumar [2020]: Kumar. A, "KMeans Silhouette Score Explained With Python Example", Dzone.com, September, 2020. [Online]. Available: <https://dzone.com/articles/kmeans-silhouette-score-explained-with-python-exam> [Accessed: Dec. 3, 2021]
- [7] B. Saji [2021]: Saji. B, "In-depth Intuition of K-Means Clustering Algorithm in Machine Learning", Analyticsvidhya.com, January, 2021. [Online]. Available: <https://www.analyticsvidhya.com/blog/2021/01/in-depth-intuition-of-k-means-clustering-algorithm-in-machine-learning/> [Accessed: Dec. 4, 2021]
- [8] A. Geron. A [2019]: Geron. A, "Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow", 2nd ed, O'Reilly, 2019, pp. 247
- [9] M. Charrad [2015]: Charrad. M, "Package 'NbClust'", Cran.R-Project.org, April, 2015. [Online]. Available: <https://cran.r-project.org/web/packages/NbClust/NbClust.pdf> [Accessed: Dec. 6, 2021]
- [10] A. Geron. A [2019]: Geron. A, "Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow", 2nd ed, O'Reilly, 2019, pp. 248
- [11] M. Ali [2021]: Ali. M, "Implementing K-Means Clustering with K-Means++ Initialization in Python.", Medium.com, April, 2021. [Online]. Available: <https://medium.com/geekculture/implementing-k-means-clustering-with-k-means-initialization-in-python-7ca5a859d63a> [Accessed: Dec. 6, 2021]
- [12] M. Mayo [2017]: Mayo. M, "The k-means Cluster Initialization Problem", Kdnuggets.com, March, 2017. [Online]. Available <https://www.kdnuggets.com/2017/03/naive-sharding-centroid-initialization-method.html> [Accessed: Dec. 7, 2021]
- [13] A. Singh, A. Yadav & A. Rana [2013]: Singh. A, Yadav. A & Rana. A, "K-means with Three different Distance Metrics", International Journal of Computer Applications, vol. 67, April, 2013
- [14] M. Mayo [2020]: Mayo. M, "k-means Clustering", Kdnuggets.com, June, 2020. [Online]. Available: <https://www.kdnuggets.com/2020/06/centroid-initialization-k-means-clustering.html> [Accessed: Dec. 7, 2021]

- [15] R. De Wet [2021]: Wet De. R, "Manhattan Distance Calculator", Omnicalculator.com, November, 2021. [Online]. Available: <https://www.omnicalculator.com/math/manhattan-distance#what-is-the-formula-for-the-manhattan-distance> [Accessed: Dec. 9, 2021]
- [16] A. Chatterjee: Chatterjee. A, "Euclidean vs Manhattan vs Chebyshev Distance", Iq.opengenus.org. [Online]. Available: <https://iq.opengenus.org/euclidean-vs-manhattan-vs-chebyshev-distance/> [Accessed: Dec. 10, 2021]
- [17] M. Grootendorst [2021]: Grootendorst .M, "9 Distance Measures in Data Science", Towardsdatascience.com, February, 2021. [Online]. Available: <https://towardsdatascience.com/9-distance-measures-in-data-science-918109d069fa> [Accessed: Dec. 13, 2021]
- [18] A. Azika [2018]: Azika. A, "K-Means Clustering: From A to Z", Towardsdatascience.com, September, 2018. [Online]. Available: <https://towardsdatascience.com/k-means-clustering-from-a-to-z-f6242a314e9a> [Accessed: Dec. 16, 2021]
- [19] P. Sharma [2019]: Sharma. P, "K-Means Clustering: From A to Z", Analyticsvidhya.com, August, 2019. [Online]. Available: <https://www.analyticsvidhya.com/blog/2019/08/comprehensive-guide-k-means-clustering/> [Accessed: Dec. 18, 2021]
- [20] J. Rouseeuw [1987]: Rouseeuw. J, "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis", Journal of Computational and Applied Mathematics, vol. 20, November, 1986

Appendices

Appendix A

WHEN TRAVERSING IN A PARTICULAR DIRECTION THAT IS NOT DOWN

PUT ADJACENT VALUES IN STACK BEFORE MOVING

IF WALL COUNT OF STACK IS == 3 AND SQUARE IN FRONT IS NOT A WALL (*)

THEN CHECK IF SQUARE IN FRONT IS NOT BLANK, IF IT ISN'T PUT IT IN VISTED VECTOR

WHEN MOVE OFF CURRENT SQUARE AND REPLACE 'Q' WITH '*' TO MARK AS VISITED

MOVE TO NEW SQUARE BY UPDATING IT WITH 'Q'

ELSE IF

PUT ADJACENT VALUES IN STACK BEFORE MOVING

IF WALL COUNT OF STACK IS < 3 AND SQUARE IN FRONT IS NOT A WALL (*)

THEN CHECK IF SQUARE IN FRONT IS NOT BLANK, IF IT ISN'T PUT IT IN VISTED VECTOR

WHEN MOVE OFF SQUARE REPLACE 'Q' (AVATAR) WITH BLANK TO GO BACK AT SOME POINT

MOVE TO NEW SQUARE BY UPDATING IT WITH 'Q'

BUT WHEN TARVERSING DOWN

PUT ADJACENT VALUES IN STACK BEFORE MOVING

IF WALL COUNT OF STACK IS == 3 AND SQUARE IN FRONT IS NOT A WALL (*)

THEN CHECK IF SQUARE IN FRONT IS NOT BLANK, IF IT ISN'T PUT IT IN VISTED VECTOR

WHEN MOVE OFF CURRENT SQUARE AND REPLACE 'Q' WITH '*' TO MARK AS VISITED

MOVE TO NEW SQUARE BY UPDATING IT WITH 'Q'

ELSE IF

PUT ADJACENT VALUES IN STACK BEFORE MOVING

IF WALL COUNT OF STACK IS < 3 AND SQUARE IN FRONT IS NOT A WALL (*)

THEN CHECK IF SQUARE IN FRONT IS NOT BLANK, IF IT ISN'T PUT IT IN VISTED VECTOR

WHEN MOVE OFF CURRENT SQUARE AND REPLACE 'Q' WITH '*' TO MARK AS VISITED

MOVE TO NEW SQUARE BY UPDATING IT WITH 'Q'

Appendix B

1. Keep pressing 'a' until a wall is hit
2. Keep pressing 'd' until a wall is hit
3. Keep pressing 's' until a wall is hit
4. Keep pressing 'w' until a wall is hit
5. Repeat steps 1 to 4 until instructed to end program

Assignment1

GRADEMARK REPORT

FINAL GRADE

64/100

GENERAL COMMENTS

Instructor

A valiant attempt of the problem.

Code runs, checks adjacencies and stacks nodes. A shame you didn't automate this in a loop. Nevertheless the essentials are there and in place. Demo provided and screenshots too.

Task 2:

An excellent review and analysis of k-means. distance measures and selection of the value of k. Well done. References good and used well.

Task 3 method and calculations correct. Algorithm completed.

Well done.

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5

PAGE 6

PAGE 7

PAGE 8

PAGE 9

PAGE 10

PAGE 11

PAGE 12

PAGE 13

PAGE 14

PAGE 15

PAGE 16

PAGE 17

PAGE 18

PAGE 19

PAGE 20

PAGE 21

PAGE 22

PAGE 23

PAGE 24

PAGE 25

GRADING FORM: SMU GENERIC_GJ

MARTIN PUGSLEY

0

RESEARCH-INFORMED LITERATURE

KNOWLEDGE AND UNDERSTANDING

ANALYSIS

PRACTICAL APPLICATION AND DEP.

SKILLS FOR PROFESSIONAL PRAC.