MACHINE LEARNING BASICS

Contents

Why this doc

The purpose of this doc is to help peoples with no machine learning background to better understand machine learning basics.

It may seem weird to describe how to calculate a standard deviation, a variation, a Euclidean distance, an argmin, … but this is required to fully understand how k-means clustering works.

What is machine learning

Machine Learning is the science of getting computers to learn from data to make decisions or predictions. Machine learning is about teaching computers how to learn from data to make decisions or predictions.

True machine learning use algorithms to build a model based on a training set in order to make predictions or decisions without being explicitly programmed to perform the task

Supervised learning

Learning with a teacher.

The machine learning algorithm learns on a labeled dataset

Unsupervised learning

Learning without a teacher.

The machine learning uses unlabeled dataset.

The advantage of using an unsupervised technique is that we do not need to have labeled data, i.e., we do not need to create a training dataset that contains examples of outliers.

k-means clustering and DBSCAN are unsupervised clustering machine learning algorithms.

They group the data that has not been previously labelled, classified or categorized.

Machine learning model

This is the output generated when you train your machine learning algorithm with your training data-set.

The machine learning model is what you get when you run the machine learning algorithm over your training data.

Once The machine learning model is built, it can be used to classify new data points.

k-Fold Cross-Validation (CV)

CV can be used to test a model. It helps to estimate the model performance. It gives an indication of how well the model generalizes to unseen data.

CV uses a single parameter called k.

It works like this:

it splits the dataset into k groups.

For each unique group:

- Take the group as a test data set

- Take the remaining groups as a training data set

- Use the on the training set to build the model, and then use the test set and evaluate

Example:

A dataset 6 datapoints: [0.1, 0.2, 0.3, 0.4, 0.5, 0.6]

The first step is to pick a value for k in order to determine the number of folds used to split the dataset.

Here, we will use a value of k=3. so we split the dataset into 3 groups. each group will have an equal number of 2 observations.

For example:

Fold1: [0.5, 0.2]

Fold2: [0.1, 0.3]

Fold3: [0.4, 0.6]

Three models are built and evaluated.

Model1: Trained on Fold1 + Fold2, Tested on Fold3

Model2: Trained on Fold2 + Fold3, Tested on Fold1

Model3: Trained on Fold1 + Fold3, Tested on Fold2

Signal vs. Noise

The "signal" is the true underlying pattern that you wish to learn from the data.

"Noise", on the other hand, refers to the irrelevant information in a dataset.

A well-functioning ML algorithm will separate the signal from the noise.

But the algorithm can end up "memorizing the noise" instead of finding the signal. The model will then make predictions based on that noise. So it will perform poorly on new/unseen data.

Model fitting

What is Model Fitting

Fitting is a measure of how well a machine learning model generalizes to similar data to that on which it was trained.

A model that is well-fitted produces more accurate outcomes, a model that is overfitted matches the data too closely, and a model that is underfitted doesn't match closely enough.

Goodness of fit of a model

it tells you if the sample data used to build the model represents the data you would expect to find in the actual population.

It measures the discrepancy between observed values and expected values.

Overfitting

A model that has learned the noise instead of the signal is considered "overfit"

This overfit model will then make predictions based on that noise. It will perform poorly on new/unseen data.

The overfit model doesn't generalize well from the training data to unseen data.

How to Detect Overfitting

we can't know how well a model will perform on new data until we actually test it.

To address this, we can split our initial dataset into separate training and test subsets.

- The training sets are used to build the models.
- The test sets are put aside as "unseen" data to evaluate the models.

This method will help to know of how well the model will perform on new data (i.e to estimate of our model's performance)

k-Fold Cross-Validation (CV) and overfitting

CV gives an indication of how well the model generalizes to unseen data.

CV does not prevent overfitting in itself, but it may help in identifying a case of overfitting.

It estimates the model on unseen data, using all the different parts of the training set as validation sets.

How to Prevent Overfitting

Detecting overfitting is useful, but it doesn’t solve the problem.

To prevent overfitting, train your algorithm with more data. It won’t work every time, but training with more data can help algorithms detect the signal and the noise better. Of course, that’s not always the case. If we just add more noisy data, this technique won’t help. That’s why you should always ensure your data is clean and relevant.

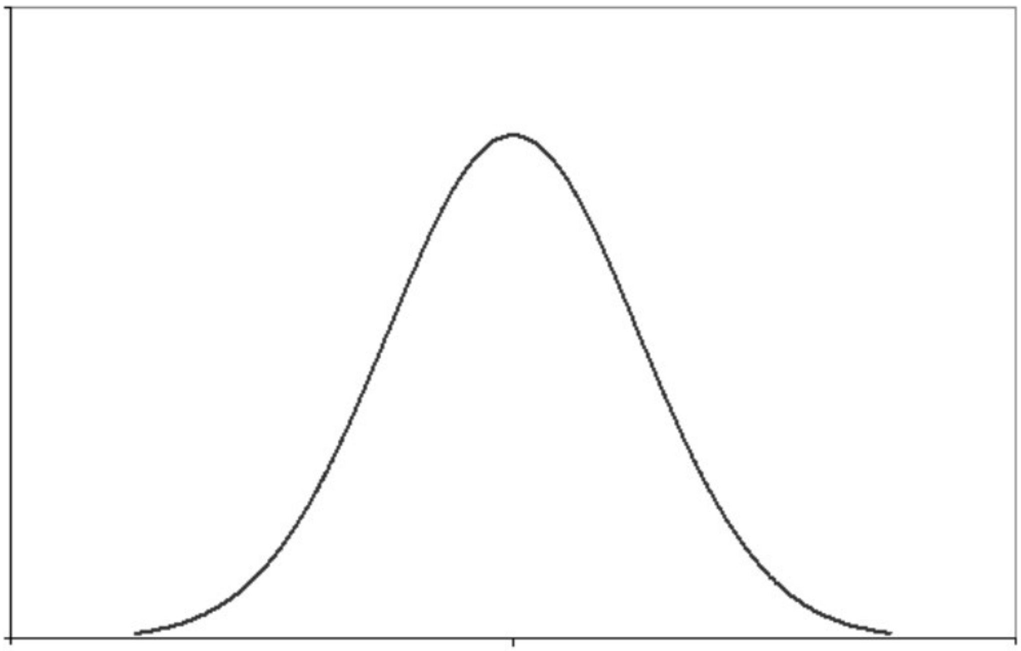
Normal distribution

Also called: Bell curve, Gaussian distribution, Gauss distribution, Laplace–Gauss distribution

A “normally distributed” data set has most of the data aggregates around its mean in a symmetric fashion. Values become less and less likely to occur the farther they are from the mean.

Think about a factory producing 1 kg bags of sugar. They won’t always make each exactly 1 kg. In reality, the bags are around 1 kg. Most of the time they will be very close to 1 kg, and very rarely far from 1 Kg.

Example of normally distributed data: height of adults



Standard deviation

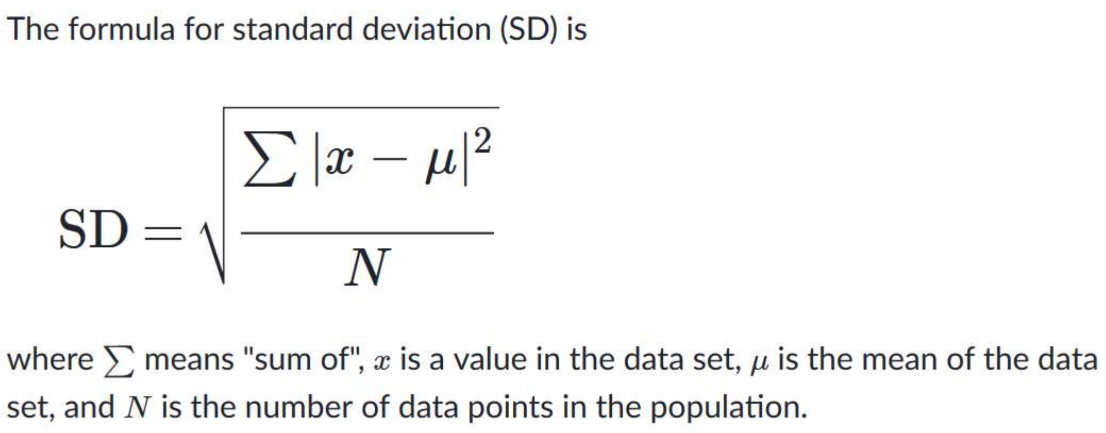
Overview

A measure that is used to quantify the amount of variation of a set of data values.

A low standard deviation indicates that the data points tend to be close to the mean of the set.

A high standard deviation indicates that the data points are spread out over a wider range of values.

Its symbol is **σ** (the greek letter sigma)



The formula for standard deviation (SD) is

$$\mathrm{SD} = \sqrt{\frac{\sum |x - \mu|^2}{N}}$$

where $\sum$ means "sum of", x is a value in the data set, μ is the mean of the data set, and N is the number of data points in the population.

Calculate it by hand

Data set = 101, 102, 106, 107

Mean = (101 +102 + 106 + 107)/4=104

((101-104)^2 + (102-104)^2 + (106-104)^2 + (107-104)^2)/4 = (9 + 4 + 4 + 9)/4 = 6.5

standard deviation = $\sqrt{6.5}$ = 2.549

Variance

Overview

As indicated above, the Standard Deviation is a measure of how spread out numbers are. Think about the average difference around the mean.

The standard deviation is the square root of the variance. As example: if SD = 3, variance = 9

The variance is the average of the squared differences from the Mean.

Calculate it by hand

Data set = 101, 102, 106, 107

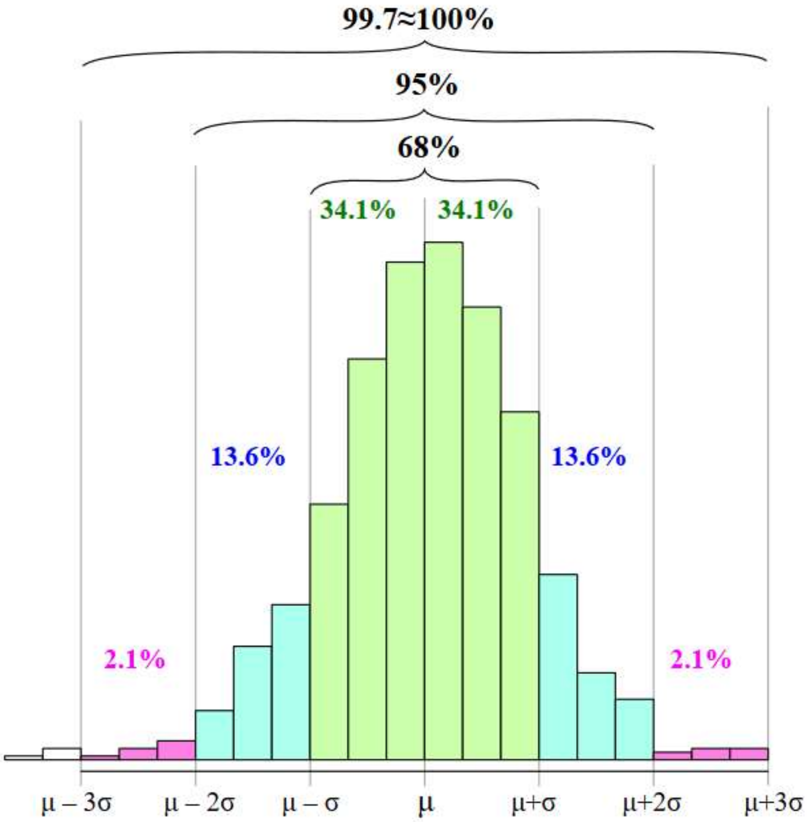
Mean = (101 +102 + 106 + 107)/4=104

Var = ((101-104)^2 + (102-104)^2 + (106-104)^2 + (107-104)^2)/4 = (9 + 4 + 4 + 9)/4 = 6.5

68–95–99.7 rule

With a normal data set (normally distributed) (as example: height of adults):

- 68.27% of the values of the data set are in a band of two standard deviations around the mean (mean - 1 standard deviation <-> mean + 1 standard deviation)
- 95% of the values of the data set are in a band of four standard deviations around the mean (mean - 2 standard deviations <-> mean + 2 standard deviations)
- 99.7% of the values of the data set are in a band of six standard deviations around the mean (mean - 3 standard deviations <-> mean + 3 standard deviations)



sigma is the greek letter σ. This is also the Standard Deviation symbol

Calculate it by hand

Data points = 101, 102, 106, 107

Mean = (101 +102 + 106 + 107)/4=104

((101-104)^2 + (102-104)^2 + (106-104)^2 + (107-104)^2)/4 = (9 + 4 + 4 + 9)/4 = 6.5

standard deviation = $\sqrt{6.5}$ = 2.54

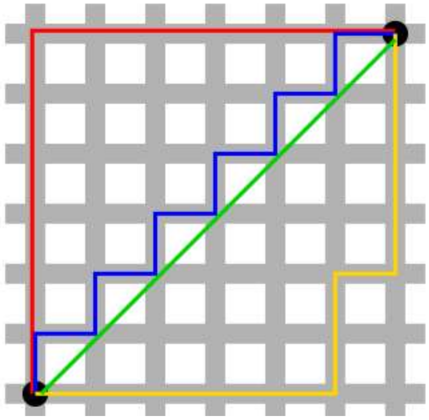
3 * standard deviation = 7.62

mean – 3 * standard deviation = 96.38

mean + 3 * standard deviation = 111.62

Euclidean distance vs Manhattan distance

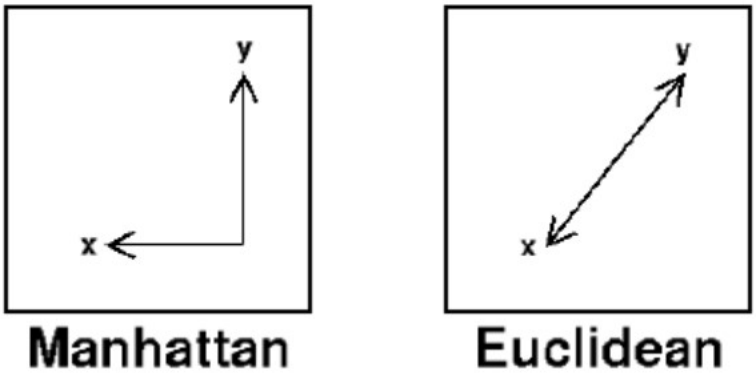
Overview



In green: Euclidean distance between X and Y. This is the shortest path between X and Y. crow flies.

In Red, Yellow, Blue: Manhattan distance between X and Y. This is the shortest path a car could take between X and Y in Manhattan

Note: Red, Yellow, Blue paths have the same length.



Calculate Euclidean distance by hand

We generally use a “double vertical line” notation for Euclidean distance. So the Euclidean distance between X and Y is d(X,Y) = || Y – X ||

<u>Euclidean distance formula:</u>

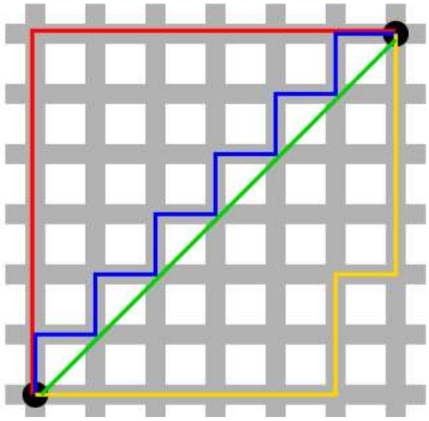
$$\sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

If X and Y are defined with 2 variables, let's say X = (X_1, X_2) and Y = (Y_1,Y_2), the Euclidean distance between X and Y is equal to: $\sqrt{}((X_1-Y_1)^2+(X_2-Y_2)^2)$

Just remember Pythagorean theorem! That's it.

If X = (0,0) and Y = (6,6), the Euclidean distance between X and Y is equal to $\sqrt{(36+36)}$ = 8.36

See the green path to better understand the above example



Calculate Manhattan distance by hand

Manhattan distance formula:

If X and Y have n dimensions:

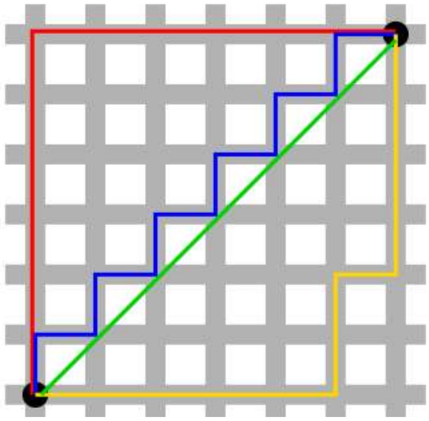
$$\sum_{i=1}^{n} |x_i - y_i|$$

If X and Y are defined with 2 variables, let's say X = (X_1, X_2) and Y = (Y_1, Y_2), the Manhattan distance between X and Y is equal to:

$|Y_1 – X_1| + |Y_2 – X_2|$

If X = (0,0) and Y = (6,6), the Manhattan distance between X and Y is equal to 12

See the red/blue/yellow paths to better understand the above example.



argmax

Also called arg max.

$$\arg\max_x f(x)$$

In mathematics, this is the inputs, or arguments, at which the output of the function f is maximized

example:

f(x) = −|x|

f(x) value is maximized with x = 0

argmin

Also called arg min.

$$\arg\min_x f(x)$$

This is the value of x for which f(x) attains it's minimum.

example:

f(x) = 2 * x

f(x) value is minimized with x = 0

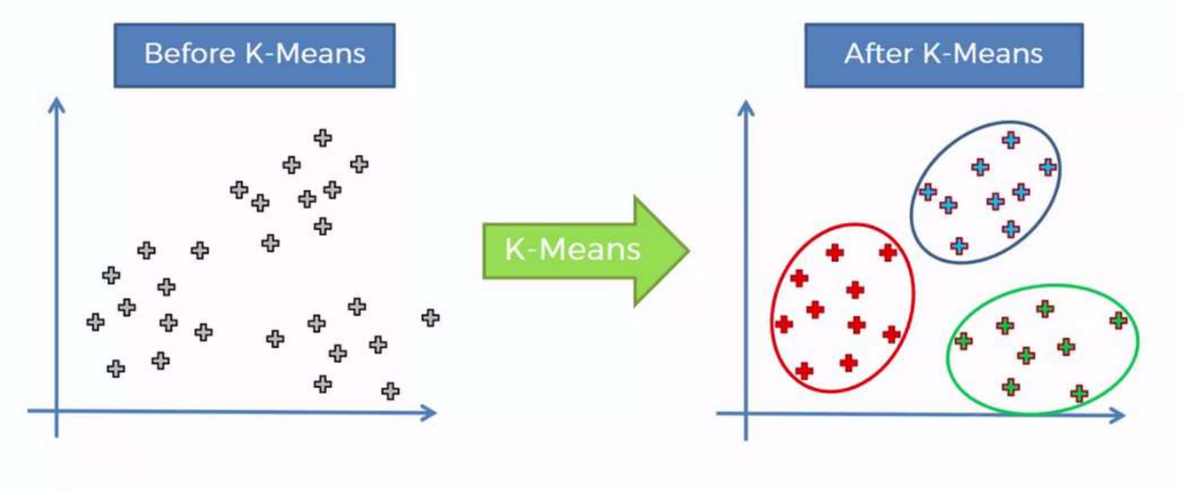
k-means clustering

Overview

k-means clustering splits N data points into K groups (called clusters).

You need to specify how many clusters you want.

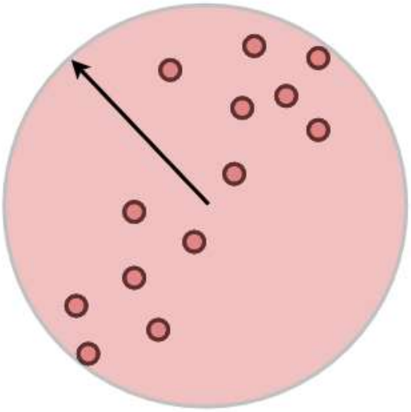
k ≤ n.



A cluster is a group of data points.

Each cluster has a center, called the centroid. A cluster centroid is the mean of a cluster (average across all the data points in the cluster).

The radius of a cluster is the maximum distance between all the points and the centroid.



Distance between clusters = distance between centroids.

k-means clustering is more difficult than three-sigma rule. k-means looks difficult, but it is not so difficult. It uses a basic iterative process. We will run it later on in this doc.

k-means clustering splits N data points into K clusters. Each data point will belong to a cluster. This is based on the nearest centroid. This uses Euclidean distance.

The objective is to find the most compact partitioning of the data set into k partitions. k-means makes compacts clusters. It minimizes the radius of clusters.

The objective is to minimize the variance within each cluster.

Clusters are well separated from each other. It maximizes the average inter-cluster distance.

Formula

Given a set of data points ($\mathbf{x}_1$, $\mathbf{x}_2$, …, $\mathbf{x}_n$), k-means partitions the n data points into k (≤ n) clusters **S** = {S_1, S_2, …, S_k} while minimizing variance within each cluster (i.e while minimizing the Euclidean distance between each data point and the centroid of the cluster it belongs to)

The objective is to find:

$$\underset{\mathbf{S}}{\arg\min} \sum_{i=1}^{k} \sum_{\mathbf{x}_j \in S_i} \left\| \mathbf{x}_j - \boldsymbol{\mu}_i \right\|^2$$

where $\boldsymbol{\mu}_i$ is the mean of points in S_i.

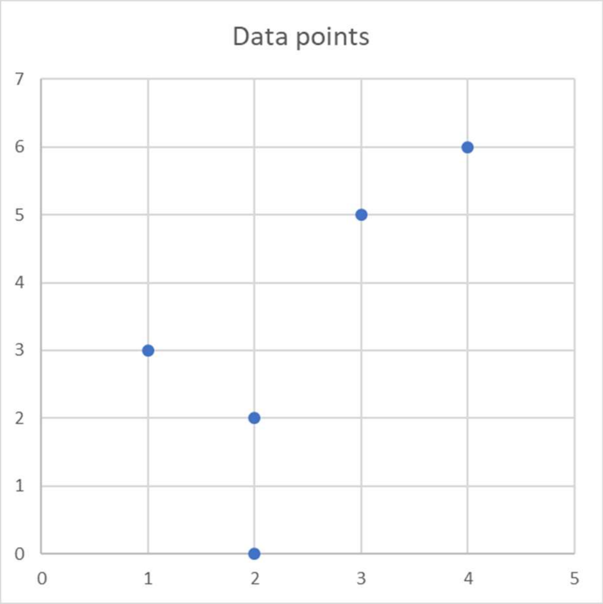
Calculate it by hand

As indicated, k-means looks difficult, but it is not so difficult. It uses a basic iterative process. Let’s run it.

let’s run k-means, with k=2, with the below data points.

In this example each data point is defined with 2 variables.

Data points	Variable 1	Variable 2
D1	2	0
D2	1	3
D3	3	5
D4	2	2
D5	4	6



So:

we have 5 data points (D1, D2, D3, D4, D5).

We want 2 clusters (cluster 1 and cluster 2).

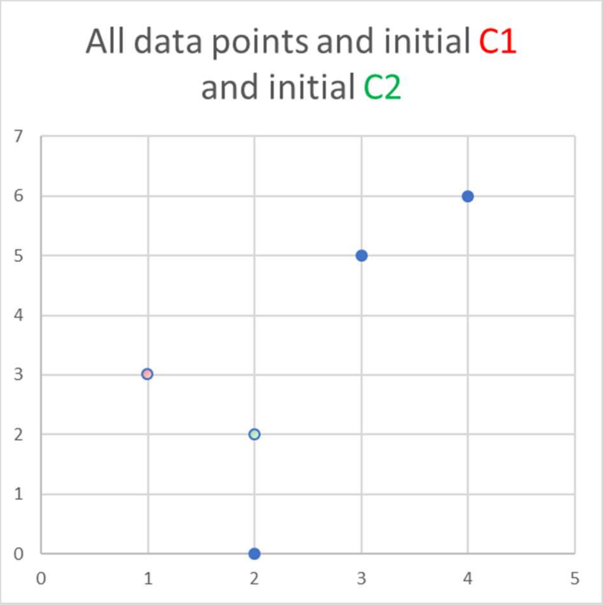
We will calculate the centroid for each of the 2 clusters. Let's call the centroids C1 and C2. C1 is the centroid for Cluster 1 and C2 is the centroid for Cluster 2.

For each cluster we will find out the centroid and which data points belongs to the cluster.

Let's start:

k-means generates randomly 2 initial centroids (or select randomly 2 existing data points as the initial centroids).

In this demo, we will select 2 existing data points (D2 and D4) as the initial centroids. Let's say initial C1 = D2 and initial C2 = D4 so initial C1 is (1,3) and initial C2 is (2,2)



Iteration 1

Let’s calculate the Euclidean Distance between each data points and the current centroids.

Current centroids are: C1 (1,3) and C2 (2,2)

Details for d1 and d2: $\sqrt{}((2-1)^2+(3-0)^2) = \sqrt{}((1)^2+(3)^2) = \sqrt{}(1+9) = \sqrt{}10 = 3.16227766$

Details for d1 and d4: $\sqrt{}((2-2)^2+(2-0)^2) = \sqrt{}((0)^2+(2)^2) = \sqrt{}(0+4) = \sqrt{}4 = 2$

After calculating the Euclidean distance between all data points each data points and the current centroids, we get:

Data points	Euclidean distance to current C1	Euclidean distance to current C2
D1	3.16227766	2
D2	0	1.41421356
D3	2.82842712	3.16227766
D4	1.41421356	0
D5	4.24264069	4.47213595

Let’s group the data points in clusters, based on the closest centroid (Euclidean distance)

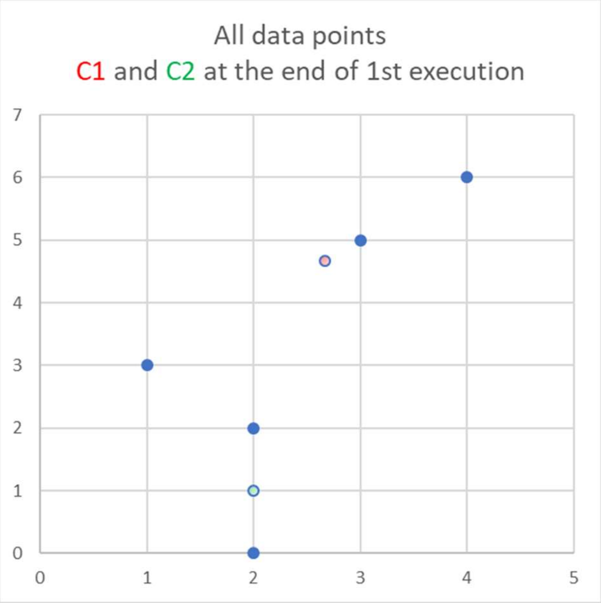
Cluster 1: D2, D3, D5.

Cluster 2: D1, D4

For each cluster, let’s calculate the mean for Variable 1 and Variable 2. This will be the new centroids.

Cluster	Data points	Data points Mean (Variable 1)	Data points Mean (Variable 2)
Cluster 1	D2, D3, D5	2.67	4.67
Cluster 2	D1, D4	2	1

The new centroid for Cluster 1 is (2.67, 4.67) and the new centroid for Cluster 2 is (2,1).



1st iteration is done.

Let’s repeat these steps until either k-means converges (i.e new centroids don’t change anymore) or until we hit the max number of iterations.

2nd iteration.

Let’s calculate the Euclidean Distance between each data points and the current centroids.

Current centroids are: C1 (2.67, 4.67) and C2 (2,1).

Data points	Euclidean distance to current C1	Euclidean distance to current C2
D1	4.71781729	1
D2	2.36173665	2.23606798
D3	0.46669048	4.12310563
D4	2.75278041	1
D5	1.88090404	5.38516481

Let’s group the data points in clusters, based on the closest centroid (Euclidean distance)

Cluster 1: D3, D5

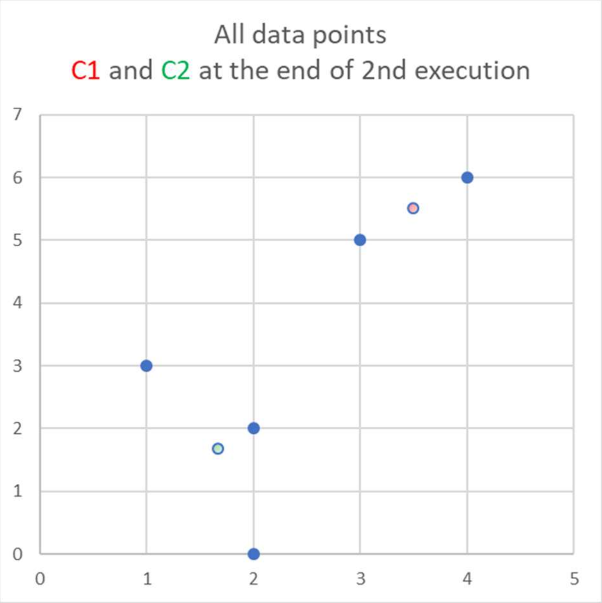
Cluster 2: D1, D2, D4

For each cluster, let’s calculate the mean for Variable 1 and Variable 2. This will be the new centroids.

Cluster	Data points	Data points Mean (Variable 1)	Data points Mean (Variable 2)
Cluster 1	D3, D5	3.5	5.5

Cluster 2	D1, D2, D4	1.67	1.67

The new centroid for Cluster 1 is (3.5, 5.5) and the new centroid for Cluster 2 is (1.67,1.67).



So the centroids changed, so the algorithm did not yet converged.

2nd iteration is done.

Let’s repeat these steps until either k-means converge (i.e new centroids don’t change) or we hit the max number of iterations.

<u>3rd iteration:</u>

Current centroids are: C1 (3.5, 5.5) and C2 (1.67,1.67)

Let’s calculate the Euclidean Distance between each data points and the current centroids.

Data points	Euclidean distance to current C1	Euclidean distance to current C2
D1	5.70087713	1.70229257
D2	3.53553391	1.48922799
D3	0.70710678	3.58577746
D4	3.80788655	0.46669048
D5	0.70710678	4.91709264

Let’s group the data points in clusters, based on the closest centroid (Euclidean distance)

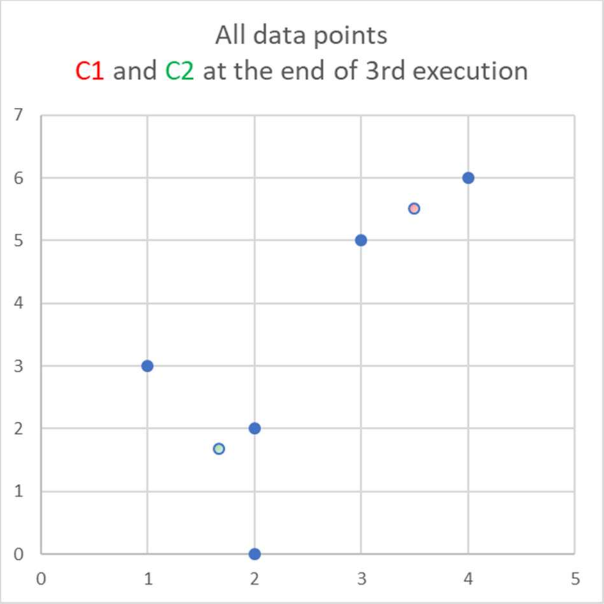
Cluster 2: D3, D5

Cluster 1: D1, D2, D4

For each cluster, let’s calculate the mean for Variable 1 and Variable 2. This will be the new centroids.

Cluster	Data points	Data points Mean (Variable 1)	Data points Mean (Variable 2)
Cluster 1	D3, D5	3.5	5.5
Cluster 2	D1, D2, D4	1.67	1.67

C1 (3.5, 5.5) and C2 (1.67,1.67)



C1 and C2 did not change with this new iteration!

K-means converged, in 3 iterations.

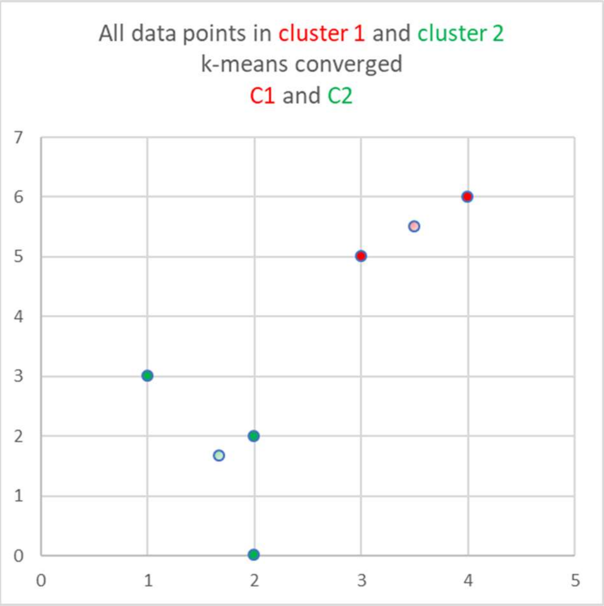
Done!

We have our 2 clusters: Cluster 1 and cluster 2

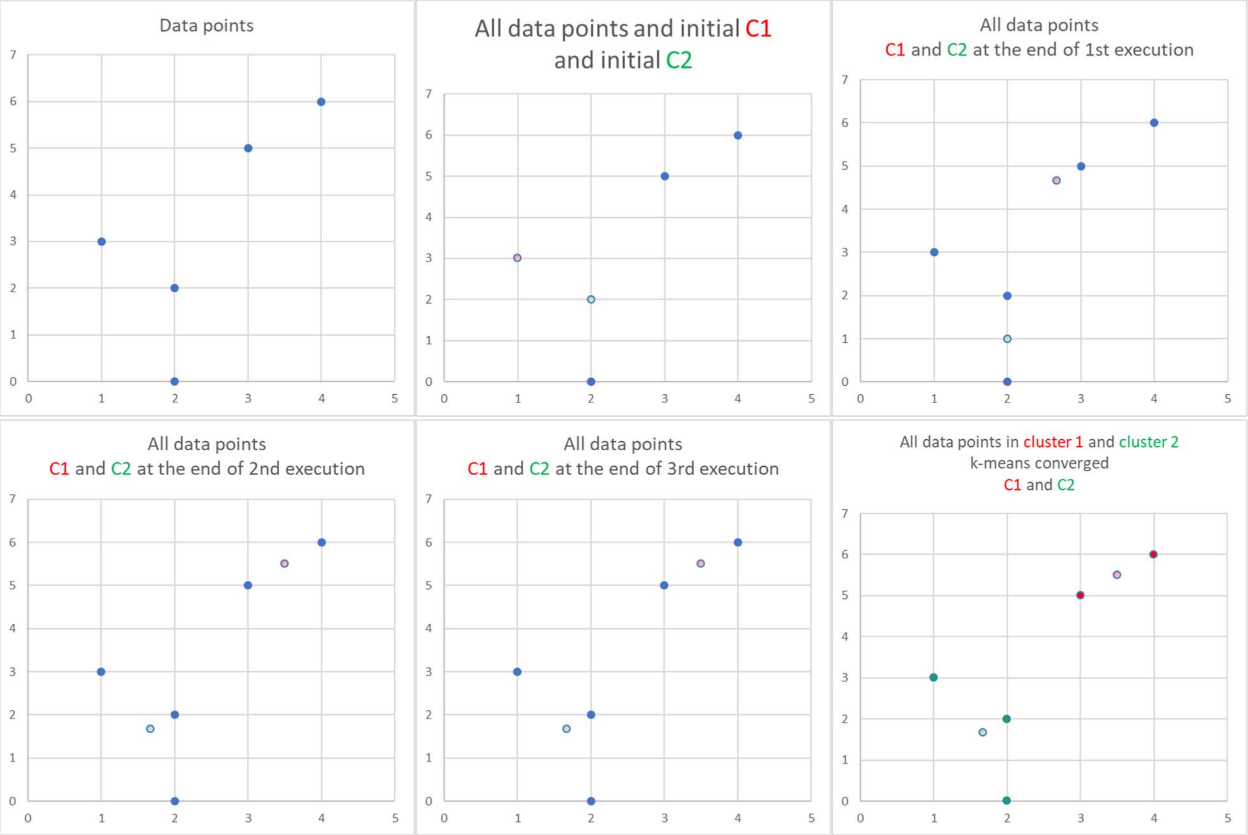
We have our 2 Centroids C1(3.5, 5.5) and C2(1.67, 1.67)

D3, D5 are in cluster 1.

D1, D2, D4 are in Cluster 2.



Quick visual recap:


Data points

All data points and initial C1 and initial C2

All data points
C1 and C2 at the end of 1st execution

All data points
C1 and C2 at the end of 2nd execution

All data points
C1 and C2 at the end of 3rd execution

All data points in cluster 1 and cluster 2
k-means converged
C1 and C2

Conclusion:

When a new data point will be added (D6) to the data set, k-means will classify it in cluster 1 or in cluster 2 based on the shortest Euclidean distance between D6 and C1 vs D6 and C2.

DBSCAN (Density-Based Spatial Clustering of Applications with Noise)

It is an unsupervised machine learning algorithm.

It is a density-based clustering algorithm.

It groups datapoints that are in regions with many nearby neighbors.

It groups datapoints in such a way that datapoints in the same cluster are more similar to each other than those in other clusters.

Clusters are dense groups of points. Clusters are dense regions in the data space, separated by regions of lower density

If a point belongs to a cluster, it should be near to lots of other points in that cluster.

It marks datapoints in lower density regions as outliers.

It uses time series data and detect outliers. We collect a window of data from a TSDB. This window is then passed to the DBSCAN algorithm, which returns the set of datapoints considered outliers.

It works like this:

First, we choose two parameters, a number epsilon (distance) and a number minPoints (minimum cluster size). epsilon is a letter of the Greek alphabet.

We then begin by picking an arbitrary point in our dataset.

If there are at least minPoints datapoints within a distance of epsilon from this datapoint, this is a high density region and a cluster is formed. i.e if there are more than minPoints points within a distance of epsilon from that point (including the original point itself), we consider all of them to be part of a "cluster".

We then expand that cluster by checking all of the new points and seeing if they too have more than minPoints points within a distance of epsilon, growing the cluster recursively if so.

Eventually, we run out of points to add to the cluster. We then pick a new arbitrary point and repeat the process.

Now, it's entirely possible that a point we pick has fewer than minPoints points in its epsilon range, and is also not a part of any other cluster: in that case, it's considered a "noise point" (outlier) not belonging to any cluster.

epsilon and minPoints remain the same while the algorithm is running.

For more information about this algorithm you can read https://www.naftaliharris.com/blog/visualizing-dbscan-clustering/

https://medium.com/netflix-techblog/tracking-down-the-villains-outlier-detection-at-netflix-40360b31732