K-Means Preliminaries

K-Means Clustering is not only the most popular algorithm for clustering, but quite possibly the most popular algorithm within Unsupervised Learning.

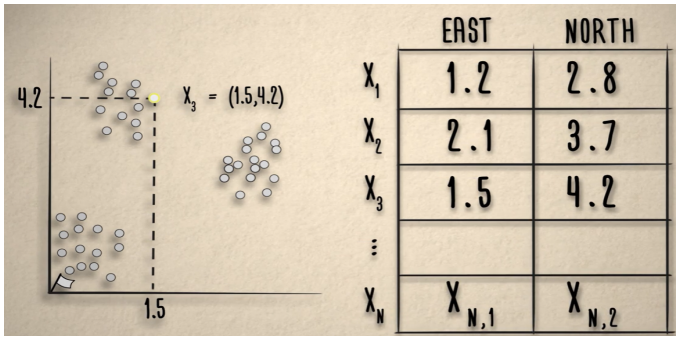
The reasons why K-Means is popular are:

- It's fast. The steps of the algorithm are simple and quick to run.
- The algorithm can easily take advantage of parallel computing by running calculations simultaneously across multiple cores or processors. So, we get even further speed up the running time.
- It's fast in programmer time. The algorithm is straightforward to understand and to code. Also, it doesn't require large numbers of parameter tweaks like some other algorithms.

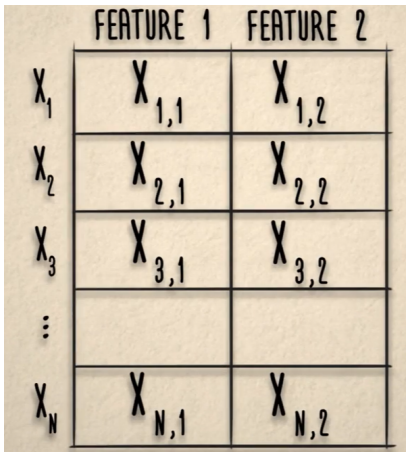
The Set-Up and Assumption of the K-Means Clustering Problem:

Assumption:

- The K-Means algorithm assumes that we know the K-value in advance.
- It also assumes we can express any data point as a list(vector) of continuous values.



Example: From the previous archaeology example, we can write the location of an artifact as a list or vector with two elements, which are the distance in east(in meters) from a marker near the site and the distance in north from the marker.

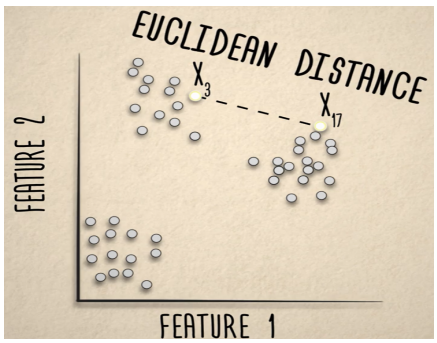


For example, in the figure the datapoint X3 is 1.5 meters east and 4.2 meters north from the marker.
We have one vector for each data point. So, the dataset is the list of all these vectors.

Here, we have N data points. In general, the information that we collect for any data point does not have to be a distance relative to some marker in an archaeological dig.
Usually we just call each component of the vector a feature.
In general, we might have any number of features.

Now we know that clustering is all about grouping the data according to similarity. We have defined what a data point is for the K-Means Clustering problem. Now we will see more elaborately what **Similarity** means in K-Means clustering.

It's equivalent to defining dissimilarity for two data points instead of similarity.
Here we say, the dissimilarity between two data points in the archaeology example, is the physical distance that is the length of the line connecting them. This length is sometimes called the **Euclidean Distance**.



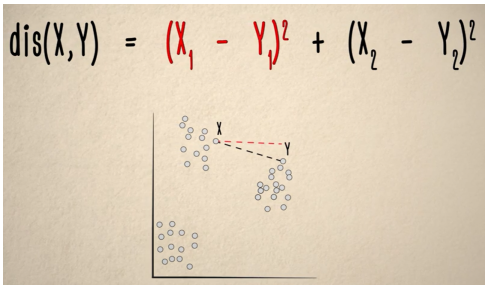
- The Euclidean Distance between two points (X1, Y1) and (X2, Y2) is calculated as:

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

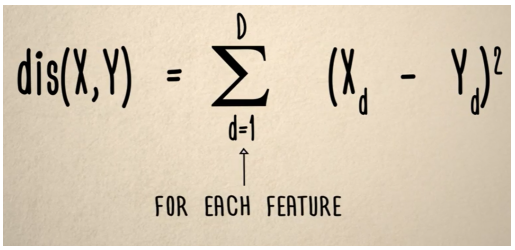
One pair of data points that is farther in Euclidean Distance than another pair, will also be farther away in the Squared Euclidean Distance.

In this case, it turns out that we get the same result from K-Means Clustering if we define dissimilarity as **Squared Euclidean Distance.**

- The Squared Euclidean Distance is just the square of Euclidean Distance.So, for two data points X(X1, X2) and Y(Y1, Y2) it is given by:



- A more convenient way to write the formula is using summation:



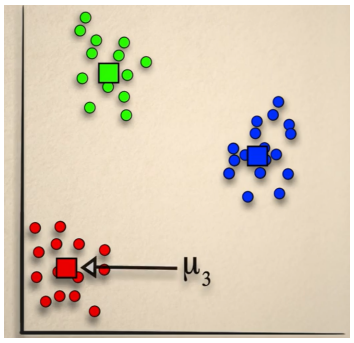
Now, we have our data, we have also defined "**Dissimilarity**". Now we need to group the data.

The question now is: **What is the output we expect from our algorithm?**

One thing to keep in mind here is that, the “K” in the K-Means refers to the number of clusters.

Like, in the archaeology example, K is equal to 3.

- Now for each of the clusters, we plan to get out a description of the cluster that is what the clusters looks like, which data points belong to it, etc. To get a clear picture, here we get a **Cluster Center** for each cluster.
- Suppose, the cluster center for the below cluster(in red) is μ3, and now we want to know which data points are assigned to each cluster.
- Let S_3, be the set of data points assigned to the cluster with cluster center μ_3.



A Few Points to Note :

- In clustering each data point has to be assigned to exactly one cluster.
- The K-Means Clustering problem is more specific than clustering in general. We assume that the data points can be expressed as continuous numbers, which is not really true if we have a “Yes or No” vector. Even if we encode “Yes” as 1 and “No” as 0 we can’t get any value other than those two in our vector.
- We have also assumed that there are exactly K clusters, that is the value of K is known in advance, but this is not always true in applications.