Introduction to K-means Clustering

Dileka Madushan Follow
Dec 1, 2017 · 4 min read

k-means is one of the simplest unsupervised learning algorithms that solve the clustering problems. The procedure follows a **simple** and easy way to classify a given data set through a certain number of clusters (assume **k** clusters). The main idea is to define **k** centers, one for each cluster.

K-Means is applied in

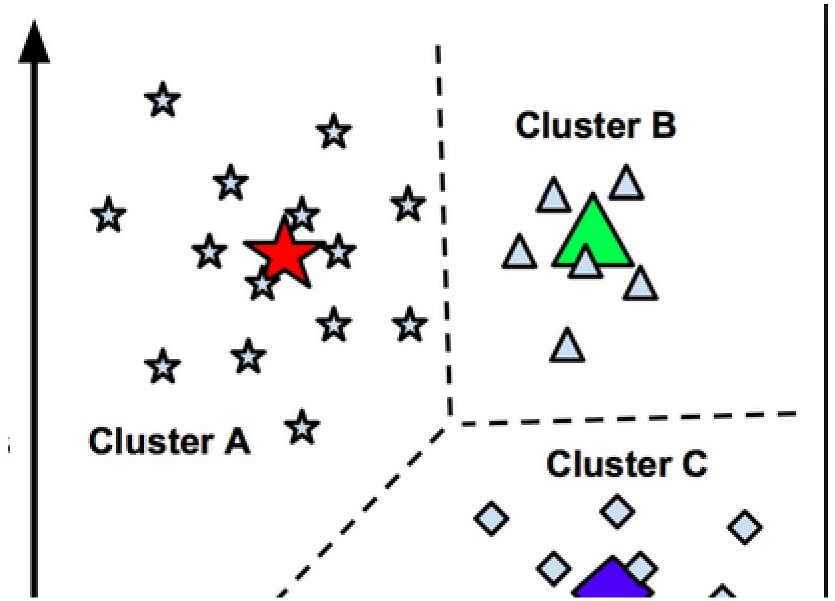
The *K*-means clustering algorithm is used to find groups which have not been explicitly labeled in the data and to find patterns and make better decisions.. Once the algorithm has been run and the groups are defined, any new data can be easily assigned to the most relevant group.

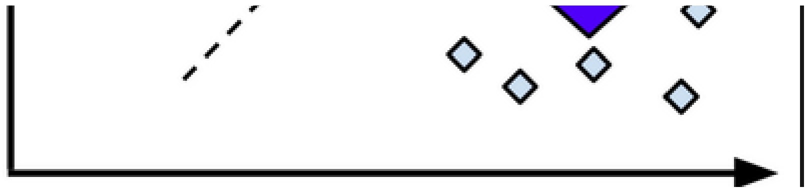
- Customer Profiling:

- market segmentation,
- computer vision
- Geo-statistics
- Astronomy

Algorithm

To start with k-means algorithm, you first have to randomly initialize points called the cluster centroids (**K**). K-means is an **iterative algorithm** and it does two steps: 1. **Cluster assignment** 2. **Move centroid step**.





Data points being clustered

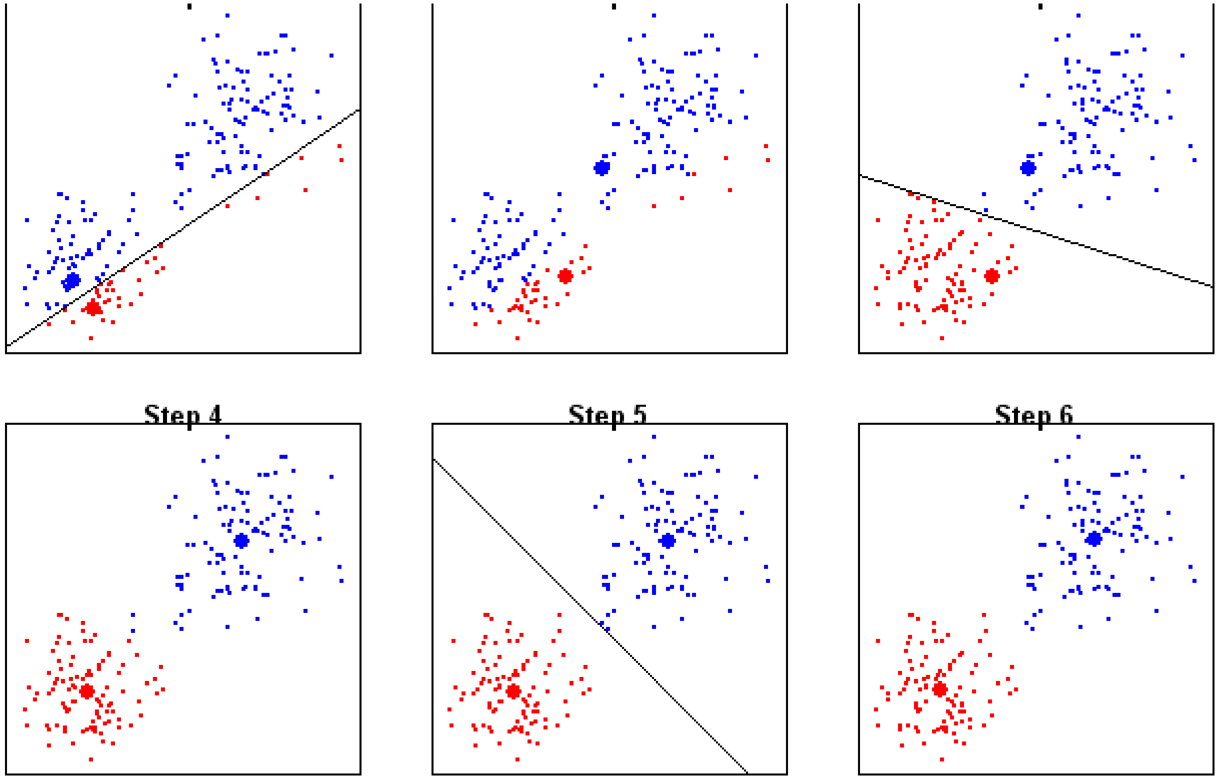
1. **Cluster assignment**

the algorithm goes through each of the data points and depending on which cluster is closer, It assigns the data points to one of the three cluster centroids.

2. Move centroid

Here, K-means moves the centroids to the average of the points in a cluster. In other words, the algorithm calculates the average of all the points in a cluster and moves the centroid to that average location.

This process is repeated until there is no change in the clusters (or possibly until some other stopping condition is met). K is chosen randomly or by giving specific initial starting points by the user.



K-Means at work

How to select the best K…

The algorithm clusters the data into *k* clusters, even if *k* is not the right number of clusters to use. Therefore, when using k-means clustering, users need some way to determine whether they are using the right number of clusters.

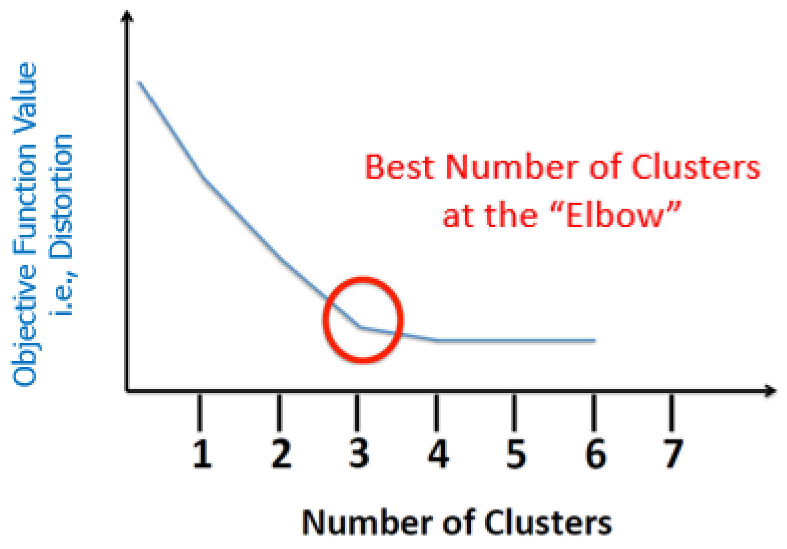
One method to validate the number of clusters is the ***elbow method***. The idea of the elbow method is to run k-means clustering on the dataset for a range of values of *k* (say, *k* from 1 to 10 in the examples above), and for each value of *k* calculate the sum of squared errors (SSE). Like this:

```
var SSE = {};
for (var i = 1; k <= MaxK; ++k) {
    SSE[k] = 0;
    clusters = kmeans(dataset, k);
    clusters.forEach(function(cluster) {
        mean = clusterMean(cluster);
        cluster.forEach(function(datapoint) {
            SSE[k] += Math.pow(datapoint - mean, 2);
        });
    });
}
```

Then, plot a line chart of the SSE for each value of *k*. If the line chart looks like an arm, then the "elbow" on the arm is the value of *k* that is the best. The idea is that we want a small SSE, but that the SSE tends to decrease toward 0 as we increase *k* (the SSE is 0 when *k* is equal to the number of data points in the dataset, because then each data point is its own cluster, and there is no error between it and the center of its cluster).

Elbow method

Elbow method



Let's run K-Means

Lets work on a sample program written in Python to get to know the *K*-means algorithm better. Python is a great tool to kick start your machine learning career. Here We implement K-Means on a set of words to identify the clusters. .

In our example, documents are simply text strings. But in your case, you might have to deal with terabytes of log data .

Here we import all the necessary **Python** libraries.

```
from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.cluster import KMeans

from sklearn.metrics import adjusted_rand_score
```

Here is our data..

```
documents = [

"Cricket is one of the most famous sport in the world","Football is
the most famous sport","Rugby is a popular sport among the young
","Lionel Messi is an icon in the world of football","Virat Kohli is
one of the best batsmen to play the game","Sporting makes you live
healthy"]
```

We use TF-IDF feature of our dataset to be used in K-Means

```
vectorizer = TfidfVectorizer(stop_words='english')
```

```
X = vectorizer.fit_transform(documents)
```

Here We initialize K to be 3

```
no_of_clusters = 3

model = KMeans(n_clusters=no_of_clusters, init='k-means++',
max_iter=100, n_init=1)

model.fit(X)
```

Lets find out which words belong to which cluster

```
order_centroids = model.cluster_centers_.argsort()[:, ::-1]

terms = vectorizer.get_feature_names()

for a in range(true_k):

 print("Cluster %d:" % a),

for ind in order_centroids[a, :10]:
 print(' %s' % terms[ind]),
```

Lets predict the clusters of new documents with our trained model 😃

```
Y = vectorizer.transform(["Steve Smith is one of the best captains in
Cricket"])

prediction = model.predict(Y)

print(prediction)

Y = vectorizer.transform(["England does very well in both Cricket and
Football"])

prediction = model.predict(Y)

print(prediction)
```

bravo ! You just mastered K-Means, but mind you there is always lot to learn in Machine Learning 🤪

```
print(prediction)
```

```
Y = vectorizer.transform(["England does very well in both Cricket and
Football"])

prediction = model.predict(Y)

print(prediction)
```

Machine Learning