Machine Learning, Data Science

Clustering: What Is It and When To use it?

A comprehensive guide to K-Means, K-Means++, and DBSCAN.

Clustering is a Machine Learning technique whose aim is to group the data points having **similar properties and/or features**, while data points in different groups should have highly offbeat properties and/or features.

Table of Content

K-Means

Introduction to K-means

K-means come from a family of **unsupervised learning** algorithms, where the input is unlabeled unlike that in supervised learning algorithms.

The end goal of K-means is **clustering**, let's dive deep into clustering.

Sometimes we just want to organize the data, and that's where clustering comes into play. It can be used for both labeled as well as unlabeled data.

Photo by [Thibault Penin](#) on [Unsplash](#)

Everybody has heard about Netflix and its never-ending content compilation.

The content is well organized in different genres such as comedy, drama, thriller, etc.

Now suppose one day you log in to Netflix and archive is cluttered and vague. How cumbersome that would be.

This is the concept of Clustering, grouping all the collateral data point into a cluster for a better and cataloged experience.

This is exactly how K-means works.

Clustering is often found in realms of *data analysis, customer segmentation, recommendation systems, search engines, semi-*

supervised learning, dimensionality reduction, and more.

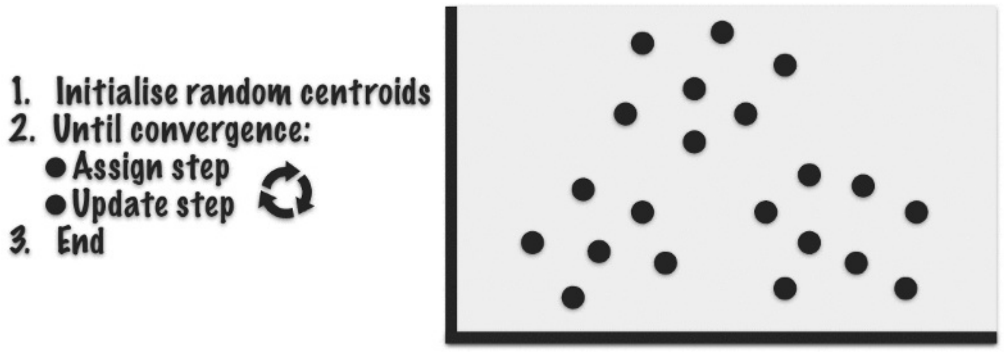
K-means algorithm is a part of **hard clustering**, that corresponds that every point belongs only to one cluster.

How K-means work?

The *“K”* in K-Means denotes the **number of clusters**. This algorithm is bound to **converge** to a solution after some iterations.

Goal: Partition data among some “K” number of clusters.

1. Initialize K-points.
2. Categorize each item to its closest mean.
3. Update coordinates of mean that is average of items categorized in mean so far.
4. Repeat the above steps until our algorithm converges.



The cost function is :

$$J = \sum_{i=1}^{m} \sum_{k=1}^{K} w_{ik} \|x^i - \mu_k\|^2 \qquad (1)$$

where m = all points

K = all clusters

wik=1 for data point if ith belongs to cluster *k*;

otherwise, wik=0.

To minimize the loss, we implement **coordinate descent**. The loss encountered in K-means **isn’t convex function** therefore there can be multiple local minima.

It’s a minimization problem of two parts:

We first minimize J w.r.t. wik and treat μk fixed. Then we minimize J w.r.t. μk and treat wik fixed.

1. **E-step**: We differentiate J w.r.t. wik first and update cluster assignments (*E-step*).

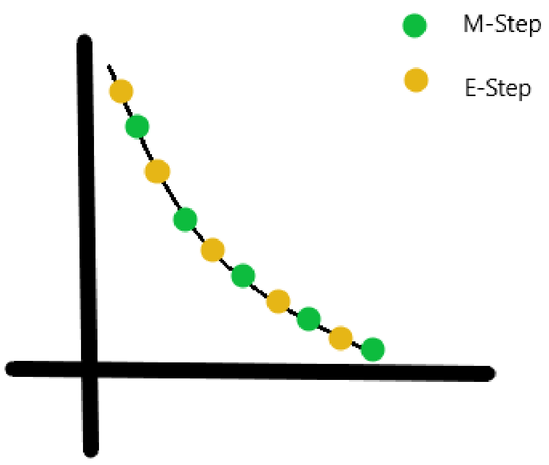
$$\frac{\partial J}{\partial w_{ik}} = \sum_{i=1}^{m} \sum_{k=1}^{K} \|x^i - \mu_k\|^2$$
$$\Rightarrow w_{ik} = \begin{cases} 1 & \text{if } k = argmin_j \ \|x^i - \mu_j\|^2 \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

We are assigning the data point xi to the nearest cluster assessed by its Euclidean distance from the cluster's centroid.

2. M-step: Then we differentiate J w.r.t. μk and re-compute the centroids after the cluster assignments from the previous step.

$$\frac{\partial J}{\partial \mu_k} = 2\sum_{i=1}^{m} w_{ik}(x^i - \mu_k) = 0$$
$$\Rightarrow \mu_k = \frac{\sum_{i=1}^{m} w_{ik} x^i}{\sum_{i=1}^{m} w_{ik}} \tag{3}$$

In the nutshell, first, we'll get wik using E-step and it will classify the point as either 0 or 1. If wik =1 then we will shift to M-step and using μk we get mean of all points to get updated cluster center.



Sci-kit implementation of K-means

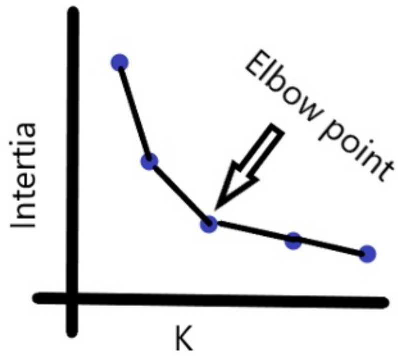
```
from sklearn.cluster import KMeans
import numpy as np
from matplotlib import pyplot as plt
from sklearn.dataset import make_blobs
x,y = make_blobs(n_sample=500, n_features = 2, center = 5, random_state
km = KMeans(n_clusters=8)
km.fit(x)
centers = km.cluster_centers_
plt.scatter(x[:,0], x[:,1])
plt.scatter(center[:,0], center[:,1], markers = "*")
plt.show()
pred = km.labels_   #show prediction for each data point
```

kmeans hosted with ❤ by **GitHub** **view raw**

To specify the number of clusters, there are two methods:

1. **Direct method:** Just plot the data points and see if it gives you a hint.
2. **Value of inertia:** The idea behind good clustering is having a small value of inertia, and a small number of clusters.

The value of inertia is inversely proportional to the number of clusters. So, its a trade-off here. Rule of thumb: The elbow point in the inertia graph is an optimal choice because after that the change in the value of inertia isn't relevant.



Pros and Cons of K-means

Pros:

1. Easy to implement.
2. Scalable for large data
3. Assure convergence.

Cons:

1. Sensitive to outliers.
2. Picking the number of clusters is a tedious job.
3. Initialization is random.
4. Not suitable for non-linear data.

K-means++

Introduction to K-means++

K-means++ is an **extended variation** of K-means. The drawback of K-means was, it used a **random initialization technique** that often leads to a dysfunctional algorithm, as, once centroids are randomly chosen there is a high risk of being struck into **local minima.**

K-means++ avoids that hindrance by choosing centroids that are **statistically close** to real centers.

> Sci-kit learn uses k-means++ by default.

Sci-kit implementation for K-means++

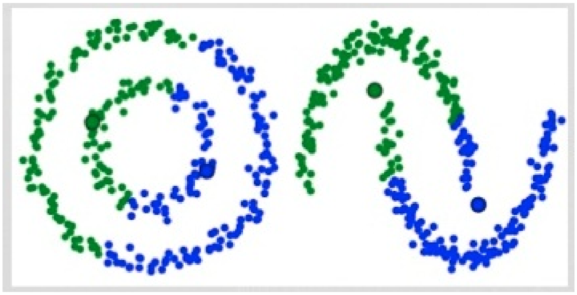
```
from sklearn.cluster import KMeans
import numpy as np
from matplotlib import pyplot as plt
from sklearn.dataset import make_blobs
x,y = make_blobs(n_sample=500, n_features = 2, center = 5, random_state
km = KMeans(n_clusters=8, init = kmeans++)
km.fit(x)
centers = km.cluster_centers_
plt.scatter(x[:,0], x[:,1])
plt.scatter(center[:,0], center[:,1], markers = "*")
plt.show()
pred = km.labels_   #show prediction for each data point
```

But, K-means++ is still not suitable for non linear data points.

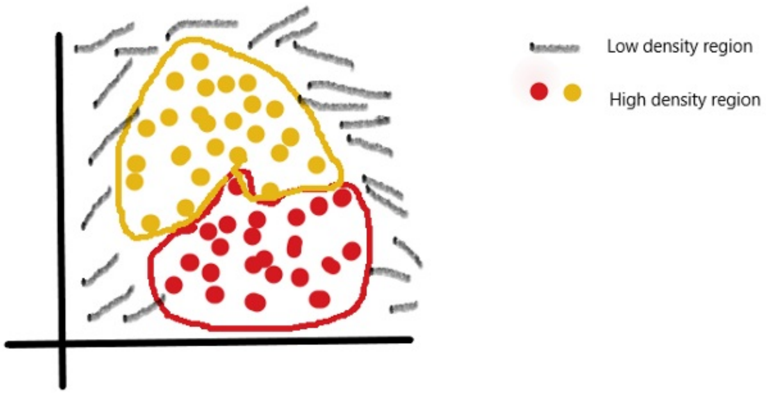
DBSCAN (Density-Based Spatial Clustering of Application with Noise)

Introduction to DBSCAN

DBSCAN is a clustering solution for **non-linear data points.**



It is based on the idea that a cluster is a **high-density area** that is surrounded by **low-density regions.**

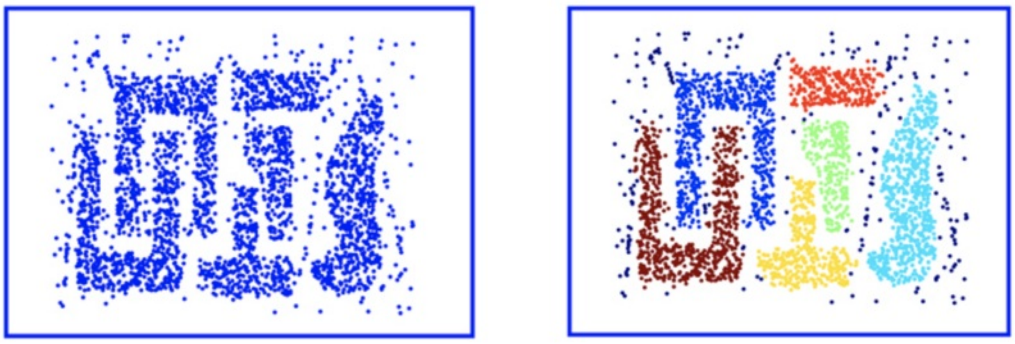


It starts by exploring the small area if the density of that area is “decent enough”, it is considered as part of a cluster and explores neighbor to increase the spatial area of the cluster.

It works on one rule: if the **distance of neighbor < threshold distance**, then it is added to the family.

Pseudocode for DBSCAN

1. Find all the neighbor points within eps and for each core unassigned point, create a new cluster.
2. Find recursively all its density connected points and add them to the same cluster as the core point(centroid).
3. Repeat the process for an unassigned neighbor.



K-means vs DBSCAN Clustering

Sci-kit implementation

```python
from sklearn.cluster import DBSCAN
from matplotlib import pyplot as plt
dbs = DBSCAN (eps = 0.1, min_sample = 5)
dbs.fit(x)                    #training
ypred = dbs.fit_predict(x)
print(ypred)                  #printing class for each data point
plt.scatter(x[:,0], x[:,1], c=ypred)
plt.show()
```

Parameters accepted:

eps: It decides how close points should be to each other, to be considered a part of a cluster. It acts as a threshold value.

minPoints: the minimum number of points to form a dense region e.g. if we set the minPoints parameter as 8, then we need at least 8 points to form a dense region(cluster).

Pros and Cons of DBSCAN

Pros of DBSCAN:

- Is great at separating clusters of high density versus clusters of low density within a given dataset.
- Less sensitive to outliers.

Cons of DBSCAN:

- DBSCAN struggles with clusters of similar density.

- DBSCAN is not efficient with high dimensionality data.
- Slower than K-Means.

Conclusion

Hopefully, this article will help you to understand clustering in the best way and also assist you to its practical usage.

As always, thanks so much for reading, and please share this article if you found it useful!

Feel free to connect:

> *LinkedIn ~ https://www.linkedin.com/in/dakshtrehan/*
>
> *Instagram ~ https://www.instagram.com/_daksh_trehan_/*
>
> *Github ~ https://github.com/dakshtrehan*

Follow for further Machine Learning/ Deep Learning blogs.

> *Medium ~ https://medium.com/@dakshtrehan*

Want to learn more?

The inescapable AI algorithm: TikTok
*Describing a progressive recommendation system used by TikTok to keep its users hooked!*towardsdatascience.com
Why are YOU responsible for George Floyd's murder & Delhi Communal Riots!!
*An ML enthusiast's approach to change the world.*medium.com
Detecting COVID-19 using Deep Learning
*A practical approach to help medical practitioners helping us in the battle against COVID-19*towardsdatascience.com
Start-off your ML journey with K-Nearest Neighbors!
*Detailed theoretical explanation and scikit-learn implementation with example!*medium.com
Things you never knew about Naive Bayes!!
*A quick guide to Naive Bayes, that will help you to develop a spam filtering system!*medium.com
Activation Functions Explained
*Step, Sigmoid, Hyperbolic Tangent, Softmax, ReLU, Leaky ReLU Explained*medium.com
Parameters Optimization Explained
*A brief yet descriptive guide to Gradient Descent, ADAM, ADAGRAD, RMSProp*towardsdatascience.com
Gradient Descent Explained
*A comprehensive guide to Gradient Descent*towardsdatascience.com
Logistic Regression Explained
*Explaining Logistic Regression as easy as it could be.*towardsdatascience.com
Linear Regression Explained
*Explaining Linear Regression as easy as it could be.*medium.com

> Cheers!

By Daksh Trehan on June 28, 2020.

Canonical link

Exported from Medium on July 15, 2020.