**K-Means Clustering**

**1. Introduction**

Most of the machine learning algorithms widely used, such as Linear Regression, Logistic Regression, Decision Trees, and others are useful for making predictions from labeled data, that is, each input comprises feature values with a label value associated. That is what is called **Supervised Learning**.

However, often we have to deal with large sets of data with no label associated. Imagine a business that needs to understand the different groups of customers based on purchasing behavior, demographics, address, and other information, thus it can offer better services, products, and promotions.

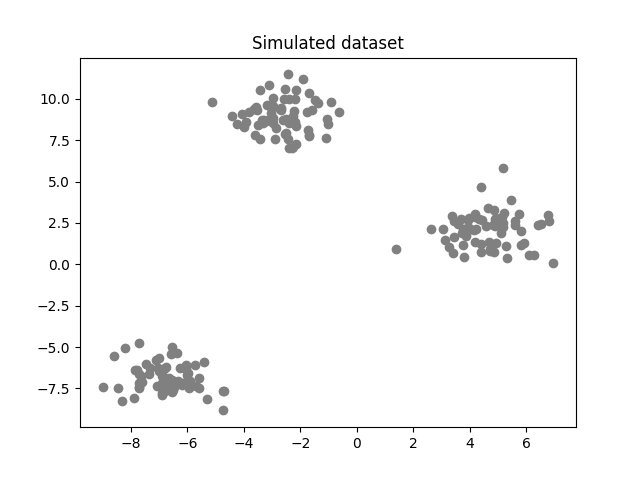
These types of problems can be addressed with the use of **Unsupervised Learning** techniques. The K-Means algorithm is a widely used unsupervised learning algorithm in Machine Learning. Its simple and elegant approach makes it possible to separate a dataset into a desired number of K distinct clusters, thus allowing one to learn patterns from unlabelled data.

**2. What Does the K-Means algorithm do?**

As said earlier, the K-Means algorithm seeks to partition data points into a given number of clusters. The points within each cluster are similar, while points in different clusters have considerable differences.

Having said that, one question arises: how do we define similarity or difference? In K-Means clustering, the Euclidean distance is the most common metric for measuring similarity.

In the figure below, we can clearly see 3 different groups. Hence, we could determine the centers of each group and each point would be associated with the closest center.

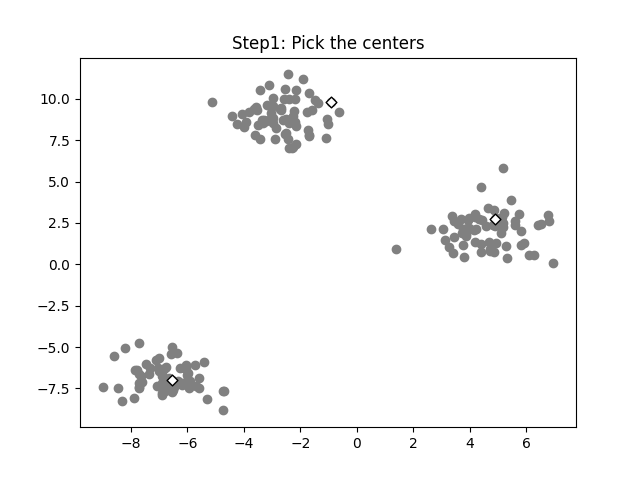
Simulated dataset with 200 observations (image by the author).

By doing that, mathematically speaking, the idea is to minimize the *within-cluster variance*, the measurement of similarity between each point and its closest center.

Performing the task in the example above was straightforward because the data was two-dimensional and the groups were clearly distinct. However, as the number of dimensions increases and different values of K are considered, we need an algorithm to handle the complexity.

**Step 1: Pick the initial centers (randomly)**

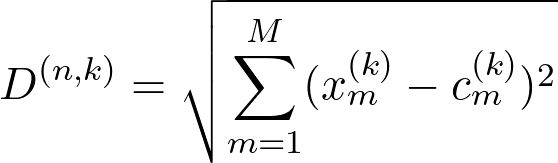
We need to seed the algorithm with initial center vectors that can be chosen randomly from the data or generate random vectors with the same dimensions as the original data. See the white diamonds in the image below.

Initial centers are randomly picked (image by the author).

**Step 2: Find the distances of each point to the centers**

Now, we’ll calculate the distance of each data point to the K centers. Then we associate each point with the center closest to that point.

Given a dataset with *N* entries and *M* features, the distances to the centers *c* can be given by the following equation:

Euclidean distance (image generated using codecogs.com).

where:

*k* varies from 1 to *K*;

*D* is the distance of a point n to the *k* center;

*x* is the point vector;

*c* is the center vector.

Hence, for each data point *n* we’ll have K distances, then we have to label the vector to the center with the smallest distance:

(image generated using codecogs.com)(image generated using codecogs.com)

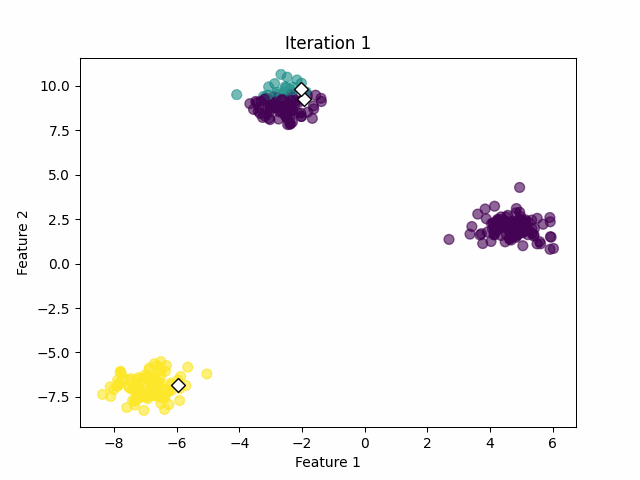
Where *D* is a vector with *K* distances.

**Step 3: Find the *K* centroids and iterate**

For each of the *K* clusters, recalculate the centroid. The new centroid is the mean of all data points assigned to that cluster. Then update the positions of the centroids to the newly calculated.

Check if the centroids have changed significantly from the previous iteration. This can be done by comparing the positions of the centroids in the current iteration with those in the last iteration.

If the centroids have changed significantly, go back to Step 2. If not, the algorithm has converged and the process stops. See the image below.

Convergence of the centroids (image by the author).

**How to choose the value of K?**

One of the most challenging tasks in this clustering algorithm is to choose the right values of k. What should be the right k-value? How to choose the k-value? Let us find the answer to these questions. If you are choosing the k values randomly, it might be correct or may be wrong. If you will choose the wrong value then it will directly affect your model performance. So there are two methods by which you can select the right value of k.

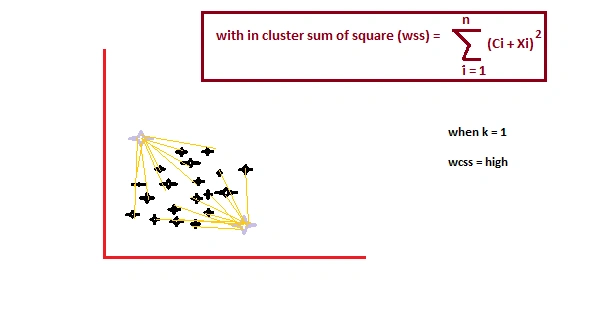
* Elbow Method.
* Silhouette Method.

Now, Let’s understand both the concept one by one in detail.

**Elbow Method**

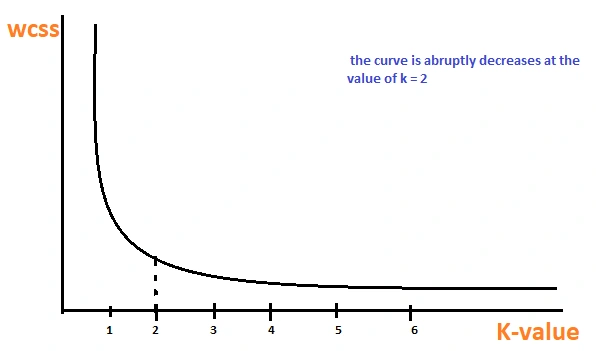
Elbow is one of the most famous methods by which you can select the right value of k and boost your model performance. We also perform the hyperparameter tuning to chose the best value of k. Let us see how this elbow method works.

It is an empirical method to find out the best value of k. it picks up the range of values and takes the best among them. It calculates the sum of the square of the points and calculates the average distance.



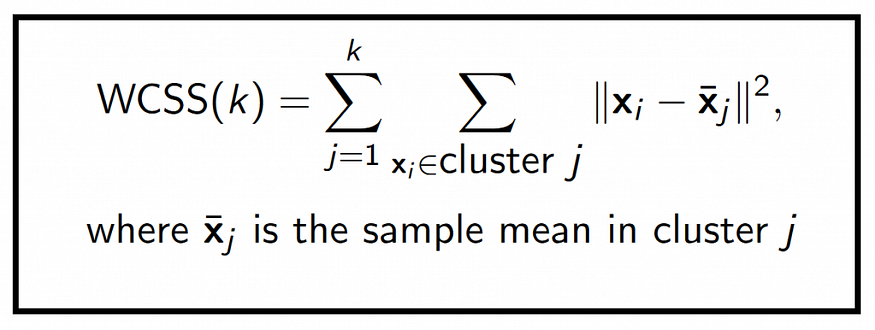
When the value of k is 1, the within-cluster sum of the square will be high. As the value of k increases, the within-cluster sum of square value will decrease.

Finally, we will plot a graph between k-values and the within-cluster sum of the square to get the k value. we will examine the graph carefully. At some point, our graph will decrease abruptly. That point will be considered as a value of k.



The Elbow method is one of the most popular ways to find the optimal number of clusters. This method uses the concept of WCSS value. WCSS stands for Within Cluster Sum of Squares, which defines the total variations within a cluster.

Press enter or click to view image in full size

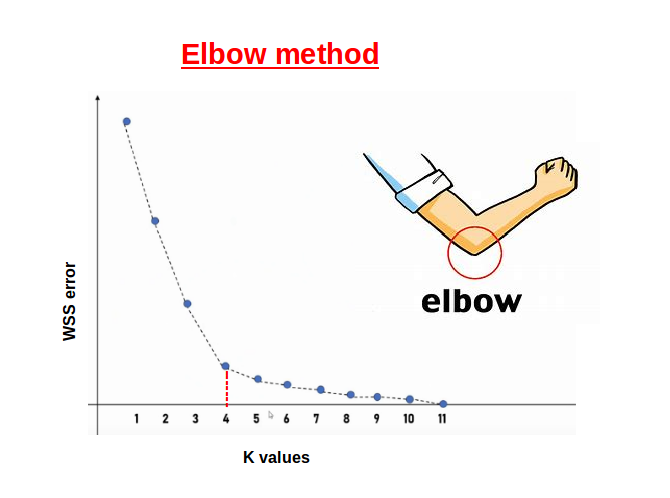


**How does it work?**

* Start with some k

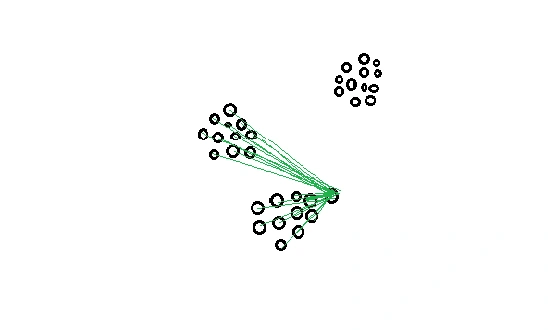
k=[2,3,4,5,6,…10]

* If k=2 apply k-mean
* It will find WCSS
* Then step repeat for different k values
* Plot a graph of k versus WCSS.
* Choose the k value after which the WCSS value is constant.



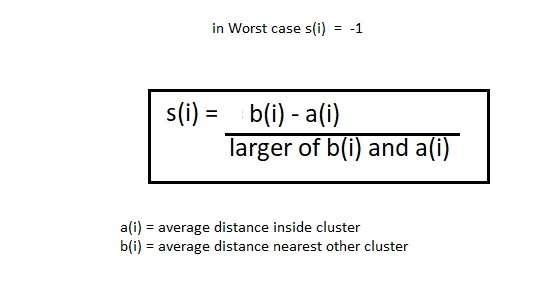
**Silhouette Method**

The silhouette method is somewhat different. The elbow method it also picks up the range of the k values and draws the silhouette graph. It calculates the silhouette coefficient of every point. It calculates the average distance of points within its cluster a (i) and the average distance of the points to its next closest cluster called b (i).



***Note****: The a (i)value must be less than the b (i) value, that is ai<<bi.*

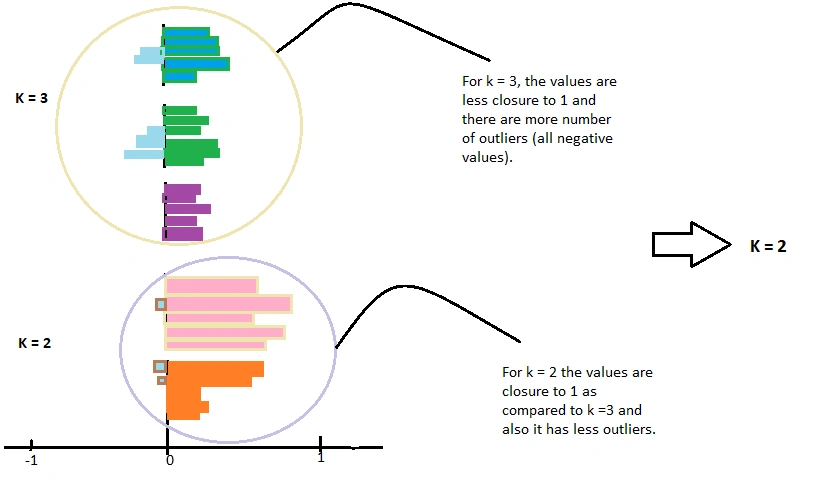
Now, we have the values of a (i) and b (i). we will calculate the silhouette coefficient by using the below formula.



Now, we can calculate the silhouette coefficient of all the points in the clusters and plot the silhouette graph. This plot will also helpful in detecting the outliers. The plot of the silhouette is between -1 to 1.

***Note****: that for silhouette coeficient equal to -1 is the worst case scenario.*

Observe the plot and check which of the k values is closer 1.



Also, check for the plot which has fewer outliers which means a less negative value. Then choose that value of k for your model to tune.

--------------

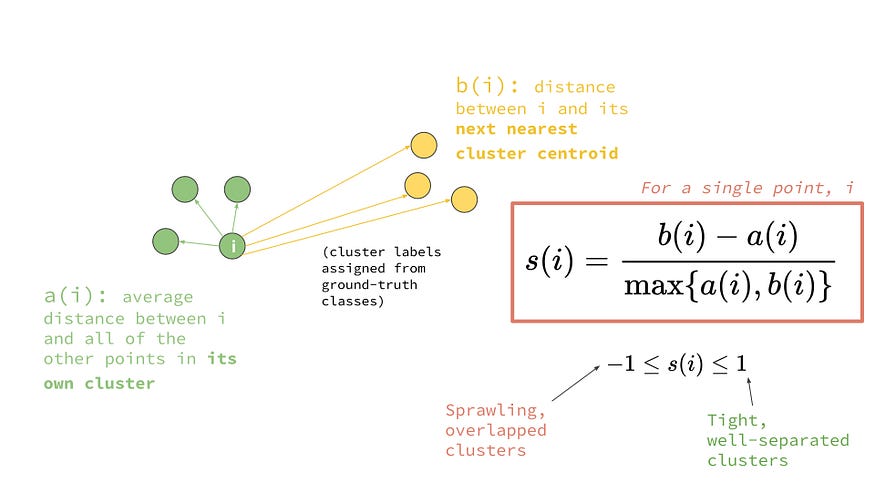
Silhouette Coefficient or silhouette score is a metric used to calculate the goodness of a clustering technique. Its value ranges from -1 to 1.

1: Means clusters are well apart from each other and clearly distinguished.

0: Means clusters are indifferent, or we can say that the distance between clusters is not significant.

* 1: Means clusters are assigned in the wrong way.

Press enter or click to view image in full size



**Important Points:**

* It will use a Distance measure.
* Scaling is very important
* Handling outliers is also Important.

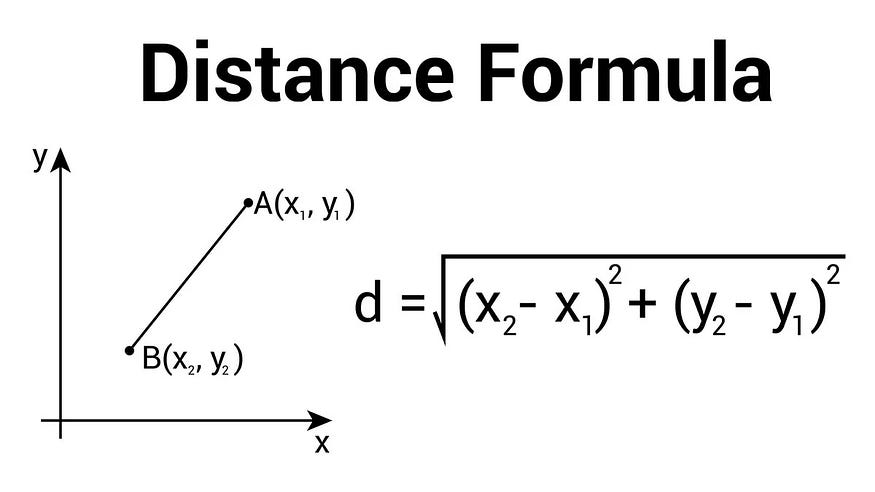
**Measuring Distance:**

* **Euclidean Distance Measure:**

The most common case is determining the distance between two points. If we have point P and point Q, the Euclidean distance is an ordinary straight line. It is the distance between the two points in Euclidean space.

The formula for the distance between two points is shown below:

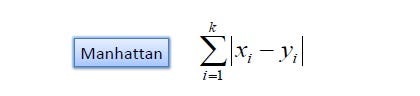
Press enter or click to view image in full size



* **Manhattan Distance Measure:**

The Manhattan distance is the simple sum of the horizontal and vertical components or the distance between two points measured along axes at right angles.

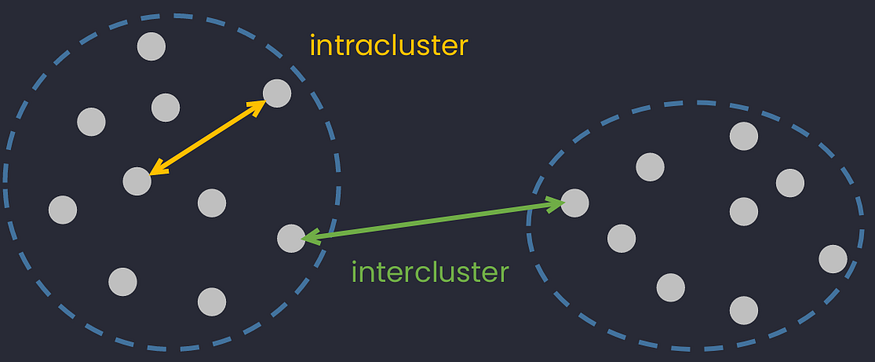
The formula is shown below:



**Main points:**

* **Intercluster distance should be high:**The distance between observations in two clusters should be High.
* **Intracluster Distance should be Very Less:**The distance of observation within the cluster should be very less.

Press enter or click to view image in full size



Read more - <https://www.analyticsvidhya.com/blog/2019/08/comprehensive-guide-k-means-clustering/>

**3. Implementation in Python**

Now that we know the fundamental concepts of the K-Means algorithm, it’s time to implement a Python class. The packages used were Numpy for mathematical calculations, Matplotlib for visualization, and the Make\_blobs package from Sklearn for simulated data.

# import required packages

import numpy as np

import matplotlib.pyplot as plt

from sklearn.datasets import make\_blobs

The class will have the following methods:

* **Init method**

A constructor method to initialize the basic parameters of the algorithm: the value *k* of clusters, the maximum number of iterations \_max*iter,* and the tolerance *tol* value to interrupt the optimization when there is no significant improvement.

* **Helper functions**

These methods aim to assist the optimization process during training, such as calculating the Euclidean distance, randomly choosing the initial centroids, assigning the closest centroid to each point, updating the centroids’ values, and verifying whether the optimization converged.

* **Fit and predict method**

As mentioned earlier, the K-Means algorithm is an unsupervised learning technique, meaning it does not require labeled data during the training process. That way, it’s necessary a single method to fit the data and predict to which cluster each data point belongs.

* **Total error method**

A method to evaluate the quality of the optimization by calculating the *total squared error* of the optimization. That will be explored in the next section.

Here it goes the full code:

# helper function for calculating Euclidean distance

def euclidean\_distance(a,b):

d = np.sqrt(np.sum((a - b)\*\*2))

return d

class Kmeans:

# construct method for hyperparameter initialization

def \_\_init\_\_(self, k=3, max\_iter=100, tol=1e-06):

self.k = k

self.max\_iter = max\_iter

self.tol = tol

# randomly picks the initial centroids from the input data

def pick\_centers(self, X):

centers\_idxs = np.random.choice(self.n\_samples, self.k)

return X[centers\_idxs]

# finds the closest centroid for each data point

def get\_closest\_centroid(self, x, centroids):

distances = [euclidean\_distance(x, centroid) for centroid in centroids]

return np.argmin(distances)

# creates a list with lists containing the idxs of each cluster

def create\_clusters(self, centroids, X):

clusters = [[] for \_ in range(self.k)]

labels = np.empty(self.n\_samples)

for i, x in enumerate(X):

centroid\_idx = self.get\_closest\_centroid(x, centroids)

clusters[centroid\_idx].append(i)

labels[i] = centroid\_idx

return clusters, labels

# calculates the centroids for each cluster using the mean value

def compute\_centroids(self, clusters, X):

centroids = np.empty((self.k, self.n\_features))

for i, cluster in enumerate(clusters):

centroids[i] = np.mean(X[cluster], axis=0)

return centroids

# helper function to verify if the centroids changed significantly

def is\_converged(self, old\_centroids, new\_centroids):

distances = [euclidean\_distance(old\_centroids[i], new\_centroids[i]) for i in range(self.k)]

return (sum(distances) < self.tol)

# method to train the data, find the optimized centroids and label each data point according to its cluster

def fit\_predict(self, X):

self.n\_samples, self.n\_features = X.shape

self.centroids = self.pick\_centers(X)

for i in range(self.max\_iter):

self.clusters, self.labels = self.create\_clusters(self.centroids, X)

new\_centroids = self.compute\_centroids(self.clusters, X)

if self.is\_converged(self.centroids, new\_centroids):

break

self.centroids = new\_centroids

# method for evaluating the intracluster variance of the optimization

def clustering\_errors(self, X):

cluster\_values = [X[cluster] for cluster in self.clusters]

squared\_distances = []

# calculation of total squared Euclidean distance

for i, cluster\_array in enumerate(cluster\_values):

squared\_distances.append(np.sum((cluster\_array - self.centroids[i])\*\*2))

total\_error = np.sum(squared\_distances)

return total\_error

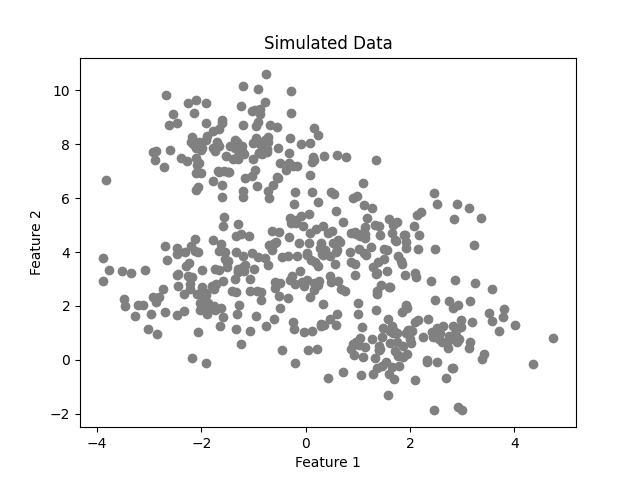
**4. Evaluation and Interpretation**

Now we’ll use the K-Means class to cluster simulated data. To do that, the make\_blobs package from the Sklearn library will be used. The data consists of 500 two-dimensional points with 4 fixed centers.

# create simulated data for examples

X, \_ = make\_blobs(n\_samples=500, n\_features=2, centers=4,

shuffle=False, random\_state=0)

Simulated data (image by the author).

After performing the training using four clusters, we achieve the following result.

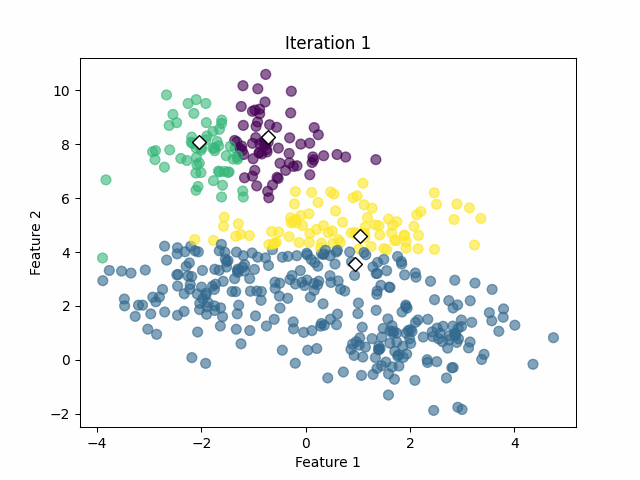
model = Kmeans(k=4)

model.fit\_predict(X)

labels = model.labels

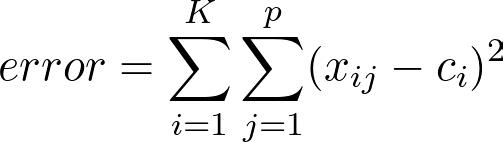
centroids =model.centroids

plot\_clusters(X, labels, centroids)

Clustering for k=4 (image by the author).

In that case, the algorithm was capable of calculating the clusters successfully with 18 iterations. However, we must keep in mind that we already know the optimal number of clusters from the simulated data. In real-world applications, we often don’t know that value.

As said earlier, the K-Means algorithm aims to make the *within-cluster variance* as small as possible. The metric used to calculate that variance is the *total squared Euclidean distance* given by:

Total squared Euclidean distance formula (image by the author using codecogs.com).

where:

p is the number of data points in a cluster;

c\_i is the centroid vector of a cluster;

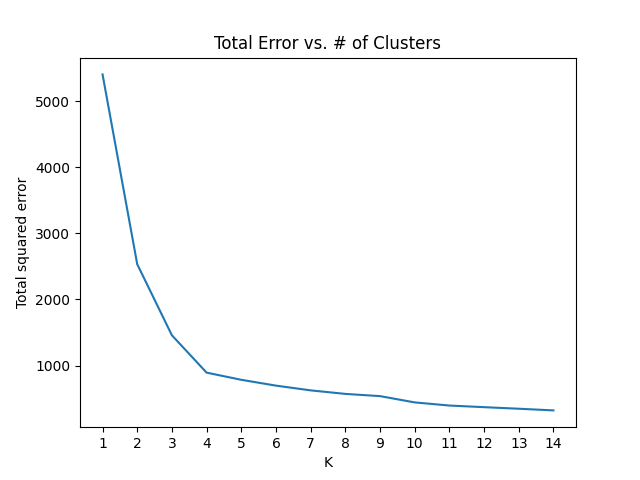
K is the number of clusters.

In words, the formula above adds up the distances of the data points to the nearest centroid. The error decreases as the number K increases.

In the extreme case of K =N, you have one cluster for each data point and this error will be zero.

Willmott, Paul (2019).

If we plot the error against the number of clusters and look at where the graph "bends", we’ll be able to find the optimal number of clusters.

Scree plot (image by the author).

As we can see, the plot has an "elbow shape" and it bends at K = 4, meaning that for greater values of K, the decrease in the total error will be less significant.

**1. What is K-Means Clustering?**

**Answer:**  
K-Means is an **unsupervised machine learning algorithm** used for clustering data into **K distinct, non-overlapping groups** based on similarity. It minimizes the **within-cluster sum of squares (WCSS)** by iteratively assigning points to the nearest cluster centroid and updating centroids.

**2. How does the K-Means algorithm work step-by-step?**

**Answer:**

1. Choose the number of clusters KKK.
2. Randomly initialize KKK centroids.
3. Assign each data point to the nearest centroid.
4. Recalculate the centroids as the mean of all points in the cluster.
5. Repeat steps 3–4 until centroids don’t change significantly or a max iteration limit is reached.

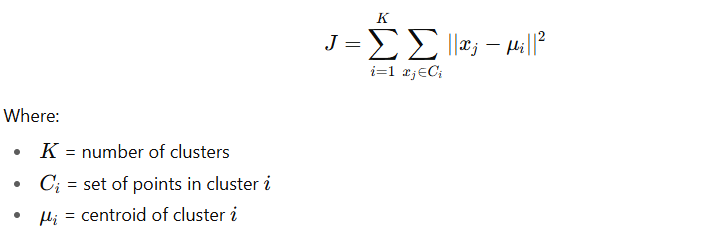
**3. How do you choose the value of K?**

**Answer:**

* **Elbow Method** – Plot WCSS vs. K, and choose the K where the curve forms an “elbow.”
* **Silhouette Score** – Measures how well points fit into their clusters.
* **Gap Statistics** – Compares WCSS with a reference null distribution.

**4. What is the objective function of K-Means?**

**Answer:**  
Minimize the **Within-Cluster Sum of Squares (WCSS)**:



**5. What are the advantages of K-Means?**

* Simple and easy to implement.
* Fast for large datasets (linear complexity).
* Works well when clusters are spherical and equally sized.

**6. What are the disadvantages of K-Means?**

* Requires predefining KKK.
* Sensitive to **outliers** and **noise**.
* Can converge to a **local minimum** depending on initialization.
* Struggles with **non-spherical** clusters.

**7. How can you improve K-Means results?**

* Use **K-Means++** initialization to avoid poor starting points.
* Normalize data before clustering.
* Run the algorithm multiple times with different seeds.
* Remove outliers before fitting.

**8. What is K-Means++?**

**Answer:**  
K-Means++ is an initialization method that spreads out initial centroids to improve accuracy and reduce the chances of poor local minima.

**9. How does K-Means differ from K-Medoids?**

**Answer:**

* **K-Means**: Uses mean as centroid, sensitive to outliers.
* **K-Medoids**: Uses actual data points (medoids) as centroids, more robust to outliers but slower.

**11. What distance metrics can K-Means use?**

* Default: **Euclidean distance**
* Variations: Manhattan, Cosine similarity (requires modification).

**12. What happens if you set K = 1 or K = N?**

* K=1K = 1K=1: All points fall into a single cluster.
* K=NK = NK=N: Each point becomes its own cluster.

**13. How does K-Means handle categorical data?**

**Answer:**  
K-Means doesn’t work well with categorical data. Alternatives:

* **K-Modes** (for categorical data).
* **K-Prototypes** (for mixed numeric and categorical data).

**14. Can K-Means guarantee the global optimum?**

**Answer:**  
No. K-Means is sensitive to centroid initialization and may converge to a **local optimum**.

**15. When should you not use K-Means?**

* When clusters are **not spherical**.
* When clusters have **vastly different sizes or densities**.
* When the dataset has many outliers.
* When data is categorical.

If you want, I can prepare **a table of tricky interview scenarios for K-Means** where they test you with “what-if” cases and you have to reason through them — this is very common in **ML system design interviews**.  
Would you like me to prepare that?

**You said:**

yes please

**ChatGPT said:**

Here’s a **set of K-Means Clustering interview questions with answers** — ranging from basic to advanced — so you can handle both theory and implementation-based discussions.

**📌 Basic Level Questions**

**1. What is K-Means clustering?**

**Answer:**  
K-Means is an **unsupervised machine learning algorithm** used for clustering data into **K distinct, non-overlapping groups** based on feature similarity.  
It works by:

1. Choosing **K initial centroids** (randomly or by a method like k-means++).
2. Assigning each data point to the nearest centroid.
3. Updating centroids as the **mean** of points assigned to them.
4. Repeating steps 2–3 until centroids stabilize or a stopping criterion is met.

**2. Is K-Means supervised or unsupervised?**

**Answer:**  
It’s **unsupervised**, since it works without labeled data.

**3. How do you choose K in K-Means?**

**Answer:**  
Common methods include:

* **Elbow method** (plot inertia vs K and look for the “elbow” point).
* **Silhouette score** (measures cohesion vs separation).
* **Gap statistic**.

**4. What is inertia in K-Means?**

**Answer:**  
Inertia is the **sum of squared distances** of samples to their closest cluster center. Lower inertia means better clustering (but it always decreases as K increases).

**5. Why is K-Means sensitive to outliers?**

**Answer:**  
Because it uses **mean** to compute centroids, and means can be skewed by extreme values.

**📌 Intermediate Level Questions**

**6. What are the main limitations of K-Means?**

**Answer:**

* Sensitive to **initial centroid positions**.
* Requires **predefining K**.
* Assumes clusters are **spherical and equally sized**.
* Struggles with **categorical data** without preprocessing.
* Sensitive to **outliers**.

**7. What is the role of k-means++ initialization?**

**Answer:**  
It selects initial centroids in a smarter way:

* First centroid chosen randomly.
* Subsequent centroids are chosen **probabilistically proportional to their distance from the nearest existing centroid**.  
  This reduces poor initialization and improves convergence.

**8. What is the time complexity of K-Means?**

**Answer:**  
For **n** samples, **k** clusters, and **i** iterations:



It’s generally fast but can be expensive for large datasets.

**9. How do you handle categorical data in K-Means?**

**Answer:**

* Convert categories to numerical form using **One-Hot Encoding** or **Label Encoding** (but Label Encoding can mislead distance metrics).
* Alternatively, use **K-Modes** or **K-Prototypes** algorithms.

**10. What stopping criteria does K-Means use?**

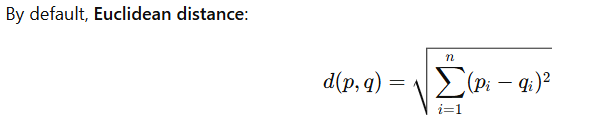
**Answer:**

* **Centroid stabilization** (no significant change in centroid positions).
* **Max iterations reached**.
* **Inertia improvement below a threshold**.

**📌 Advanced Level Questions**

**11. What distance metric does K-Means use?**

**Answer:**  
By default, **Euclidean distance**:



Modifications can use **Manhattan** or other distances.

**12. How does K-Means behave with non-spherical clusters?**

**Answer:**  
Poorly, because it assumes equal variance and spherical distribution — for complex shapes, algorithms like **DBSCAN** or **Spectral Clustering** perform better.

**13. Can K-Means converge to a local minimum?**

**Answer:**  
Yes, because initialization is random; that’s why multiple runs (n\_init > 1) are recommended.

**14. How do you evaluate K-Means clustering performance without labels?**

**Answer:**

* **Silhouette Score** (−1 to 1; higher is better).
* **Davies–Bouldin Index** (lower is better).
* **Calinski–Harabasz Index** (higher is better).

**15. How does mini-batch K-Means work?**

**Answer:**  
Instead of processing the entire dataset at once, it updates centroids using **small random batches** of data, making it faster for large datasets.

If you want, I can also prepare **20+ coding-based K-Means interview questions** where you implement Elbow method, silhouette score, mini-batch K-Means, and custom distance functions. That would give you practical hands-on readiness too.

Do you want me to prepare that next?

**You said:**

yes please go ahead

**ChatGPT said:**

Alright Rajeev — here’s a **K-Means Clustering** interview Q&A set that starts from basics and moves toward more advanced points so you’re ready for technical discussions.

**K-Means Clustering – Interview Questions & Answers**

**1. What is K-Means clustering?**

**Answer:**  
K-Means is an **unsupervised machine learning algorithm** used for clustering data into **K non-overlapping groups** based on feature similarity.  
It works by:

1. Selecting K initial centroids (randomly or using methods like K-Means++).
2. Assigning each data point to the nearest centroid.
3. Recomputing centroids as the mean of assigned points.
4. Repeating steps 2–3 until centroids stabilize or max iterations reached.

**2. Is K-Means supervised or unsupervised?**

**Answer:**  
It’s **unsupervised**, as it works without labeled data. The algorithm identifies patterns and groups solely based on the data's structure.

**3. What are the main assumptions of K-Means?**

**Answer:**

* Clusters are roughly spherical and equally sized.
* Data is continuous (not categorical).
* Each feature contributes equally to distance calculation (scaling is often required).

**4. How do you choose the value of K?**

**Answer:**

* **Elbow Method:** Plot Within-Cluster Sum of Squares (WCSS) vs. K and look for the "elbow" point.
* **Silhouette Score:** Measures cohesion and separation of clusters.
* **Gap Statistic:** Compares clustering performance against random uniform data.

**5. What distance metric does K-Means use?**

**Answer:**  
By default, **Euclidean distance**.  
However, other variations can use Manhattan, cosine similarity, etc.

**6. What are the limitations of K-Means?**

**Answer:**

* Sensitive to initial centroid positions (can converge to local minima).
* Requires pre-defining K.
* Struggles with clusters of different densities, sizes, or non-spherical shapes.
* Sensitive to outliers.

**7. How does K-Means++ improve K-Means?**

**Answer:**  
Instead of choosing random initial centroids, **K-Means++** spreads out initial centroids:

1. First centroid picked randomly.
2. Next centroid chosen based on the probability proportional to the square of the distance from the nearest existing centroid.  
   This leads to faster convergence and better results.

**8. Can K-Means work with categorical data?**

**Answer:**  
Not directly, because it uses mean calculations. For categorical data:

* Use **K-Modes** or **K-Prototypes** (mixed data).
* Convert categories to numeric embeddings before applying K-Means.

**9. What is inertia in K-Means?**

**Answer:**  
Inertia = Sum of squared distances between data points and their assigned centroids.  
Lower inertia → tighter clusters (but too low may mean overfitting).

**10. How does K-Means converge?**

**Answer:**  
K-Means convergence is achieved when:

* Cluster assignments stop changing, OR
* Centroids move negligibly between iterations.

**11. How do you handle outliers in K-Means?**

**Answer:**

* Remove them before clustering (e.g., using IQR, Z-score).
* Use **K-Medoids** instead of K-Means.
* Scale data to reduce the influence of extreme values.

**12. What are some alternatives to K-Means?**

**Answer:**

* **K-Medoids** (more robust to outliers).
* **DBSCAN** (good for arbitrary-shaped clusters).
* **Mean Shift** (non-parametric).
* **Gaussian Mixture Models** (soft clustering).

**13. Can K-Means be used for image compression?**

**Answer:**  
Yes. It clusters pixel colors into K dominant colors, replacing all pixels in a cluster with their centroid color → reducing storage.

**14. How does scaling affect K-Means?**

**Answer:**  
K-Means is sensitive to feature scales. Features with larger ranges dominate distance calculations. Always use **StandardScaler** or **MinMaxScaler** before clustering.

**15. What’s the time complexity of K-Means?**

**Answer:**  
For **n** points, **k** clusters, and **i** iterations:  
**O(n × k × i × d)** where **d** = number of dimensions.

If you want, I can also prepare a **real-world scenario-based K-Means interview Q&A** — like “How would you cluster customers for targeted marketing?” — which helps in senior-level interviews.

Do you want me to prepare that next?

**You said:**

yes please only those which are not covered above

**ChatGPT said:**

Alright — here’s a fresh set of **K-Means clustering interview questions and answers** that **don’t overlap** with what I gave you earlier:

**Advanced & Less Common K-Means Interview Questions**

**1. How does K-Means behave when clusters have very different densities?**

**Answer:**  
K-Means assumes all clusters have similar density and size. If one cluster is dense and another is sparse, the algorithm may place centroids incorrectly, merging smaller dense clusters or splitting larger sparse clusters. In such cases, **DBSCAN** or **Gaussian Mixture Models** may perform better.

**2. What is the role of the “Voronoi diagram” in K-Means?**

**Answer:**  
A Voronoi diagram is a geometric representation of cluster boundaries. In K-Means, each cluster’s centroid defines a **Voronoi cell**, and all points inside a cell are closer to its centroid than to any other centroid.

**3. Why is K-Means not guaranteed to find the global optimum?**

**Answer:**  
Because it minimizes the within-cluster variance using an **iterative greedy approach**, it can get stuck in **local minima**. The result heavily depends on centroid initialization.

**4. How can you make K-Means clustering results more stable?**

**Answer:**

* Use **K-Means++** initialization.
* Run K-Means **multiple times** with different seeds and choose the clustering with the lowest inertia.
* Normalize or scale the data before clustering.
* Remove outliers.

**5. How do you handle categorical data in K-Means?**

**Answer:**  
K-Means works on numerical data. For categorical data:

* Use **One-Hot Encoding** (can increase dimensionality).
* Use **K-Modes** or **K-Prototypes** (extensions of K-Means for categorical/mixed data).

**6. What happens if all points are identical in K-Means?**

**Answer:**  
If all points have the same coordinates:

* All centroids may collapse to the same point.
* The algorithm will converge in **one iteration** with inertia = 0.

**7. How does K-Means behave with high-dimensional data?**

**Answer:**

* Distance metrics become less meaningful due to the **curse of dimensionality**.
* Clusters may become hard to separate visually.
* Dimensionality reduction (PCA, t-SNE, UMAP) is often applied before K-Means.

**8. What’s the relationship between K-Means and PCA?**

**Answer:**

* PCA reduces dimensionality while preserving variance.
* K-Means groups data points into clusters.  
  PCA is sometimes applied **before** K-Means to improve performance and reduce noise.

**9. Why might K-Means produce empty clusters?**

**Answer:**  
If no data points are closest to a particular centroid during an iteration, that cluster becomes **empty**. This can happen with:

* Poor initialization.
* Highly imbalanced cluster sizes.  
  One fix is to reinitialize the empty centroid to a random data point.

**10. Can K-Means be used for anomaly detection?**

**Answer:**  
Yes — anomalies can be detected by measuring a point’s **distance from its assigned cluster centroid**. If the distance exceeds a threshold, it can be flagged as an outlier.