DATA SCIENCE INTERVIEWS

How to Ace The K-Means Algorithm Interview Questions

Julien Kervizic Follow
Aug 13 · 8 min read ★

Photo by Billy Huynh on Unsplash

KMeans is one of the most common and important clustering algorithms to know for a data scientist. It is, however, often the case that experienced data scientists do not have a good grasp of this algorithm. This makes KMeans an excellent topic for interviews, to get a good grasp of the understanding of one of the most foundational machine learning algorithm.

There are a lot of questions that can touched-on when discussing the topic:

1. Description of the Algorithm
2. Big O Complexity & Optimization
3. Application of the algorithm
4. Comparison with other clustering algorithms
5. Advantages / Disadvantage of using K-Means

Description of Algorithm

Describing the inner working of the K-Means algorithm is typically the first step in an interview questions centered around clustering. It shows the interviewer whether you have grasped how the algorithm works.

It might sound fine just to apply a `KMeans().fit()` and let the library handle all the algorithm work. Still, in case you need to debug some behavior or understand if using KMeans would be fit for purpose, it starts with having a sound understanding of how an algorithm works.

High-Level Description

There are different aspects of K-means that are worth mentioning when describing the algorithm. The first one being that it is an **unsupervised learning** algorithm, aiming to group “records” based on their distances to a fixed number (i.e., k) of “centroids.” Centroids being defined as the means of the K-clusters.

Inner workings

Besides the high-level description provided above, it is also essential to be able to walk an interviewer through the inner workings of the algorithm. That is from initialization,

to the actual processing and the stop conditions.

Initialization: It is important to discuss that the initialization method determines the initial clusters' means. It would be expected from this point of view, to at least mention the problem of initialization, how it can lead to different cluster being created, the impact on the time it takes to obtain the different clusters, etc.. One of the key initialization method to mention is the "Forgy" initialization method.

Processing: I would expect a discussion on how the algorithm traverses the points, and iteratively assigns them to the nearest cluster. Great candidates would be able to go beyond that description and into a discussion over KMeans, minimizing the within-cluster variance and discuss Lloyd's algorithm.

Stop condition: The stop conditions for the algorithm needs to be mentioned. The typical stop conditions for the algorithm are usually based on the following

- (stability) Centroids of new cluster do not change
- (convergence) points stay in the same cluster
- (cap) Maximum number of iterations has been reached

Stop conditions are quite important to the algorithm, and I would expect a candidate, to at least mention the *stability* or *convergence* and the cap conditions. Another key point to highlight going through these stop conditions is articulating the importance of having a cap implemented *(see Big O complexity below)*.

Big O Complexity

It is important for candidates to understand the complexity of the algorithm, both from a training and prediction standpoint, and how the different variables impact the performance of the algorithm. This is why questions around the complexity of the KMeans are often asked, when deep-diving into the algorithm:

Training BigO

From a training perspective, the complexity is *(if using Lloyds' algorithm)*:

```
BigO(KmeansTraining) = K *I * N * M
```

Where:

- K: Number of clusters
- I: The number of iterations
- N: The sample size
- M: The number of variables

As it is possible to see, there can be a significant impact on capping the number of iterations.

Prediction BigO

K-means predictions have a different complexity:

```
BigO(KmeansPrediction) = K * N * M
```

KMeans prediction, only needs to have computed for each record, the distance (which complexity is based on the number of variables) to each cluster, and assign it to the nearest one.

Scaling KMeans

During an interview, you might be asked if there are any ways to make KMeans perform faster on larger datasets. This should be a trigger to discuss mini-batch KMeans.

Mini batch KMeans is an alternative to the traditional KMeans, that provides better performance for training on larger datasets. It leverages mini-batches of data, taken at random to update the clusters' mean with a decreasing learning rate. For each data bach, the points are all first assigned to a cluster and then means are then re-calculated. The clusters' centers are recalculated using gradient descent. The algorithm provides a faster convergence than the typical KMeans, but with a slightly different cluster output.

Applying K-means

Use cases

There are multiple use cases for leveraging the K-Means algorithm, from offering recommendations or offering some level of personalization on a website, to deep diving into potential cluster definitions from customer analysis and targeting.

Understanding what is expected from applying k-means also dictates how you should be applying it. Do you need to find the number of optimal number of K? or an arbitrary number given by the marketing department. Do you need to have interpretable variables, or is this something that would be better left for an algorithm to decide?

It is important to understand how particular K-Means use cases can impact its' implementations. Implementation specific questions, usually come up as follow-ups, such as:

> Let say, the marketing department asked you to providse them with user segments for an upcoming marketing campaign. What features would you look to feed into your model and what transformations woud you apply to provide them with these segments?

This type of followup question is very open-ended, can require further clarification, but does usually provide insights into whether or not the candidate understands how the results of the segmentation might be used.

Finding the optimal K

Understanding how to determine the number of K to use for KMeans often comes up as a followup question in the application of the algorithm.

There are different techniques to identify the optimal number of clusters to use with KMeans. Three different methods are used the Elbow method, the Silhouette method, and Gap statistics.

The Elbow method: is all about finding the point of inflection on a graph of % of variance explained to the number of K.

Silhouette method: The silhouette method, involves calculating for each point, a similarity/dissimilarity score between their assigned cluster, and the next best (i.e., nearest) cluster.

Gap statistics: The goal of the gap statistic is to compare the cluster assignments on the actual dataset against some randomly generated reference datasets. This comparison is done through the calculation of the intracluster variation, using the log of the sum of the pairwise distance between the clusters' points. Large gap statistics indicates that the cluster obtained on observed data, are very different from those obtained from the randomly generated reference data.

Input variables

When applying KMeans, it is crucial to understand what kind of data can be fed to the algorithm.

> For each user on our video streaming platform, you have been provided with their historical content views as well as their demographic data. How do you determine what to train the model on?

It is generally an excellent way to breach into the two subtopics of variable normalization and on the number of variables.

Normalization of variables

In order to work correctly, KMeans typically needs to have some form of normalization done of the datasets. K-means is sensitive to both means and variance in the datasets.

For numerical performing normalization using a StandardScaler is recommended, but depending on the specific cases, other techniques might be more suitable.

For pure categorical data, one hot encoding would likely be preferred, but worth being careful with the number of variables it ends up producing, both from an efficiency (BigO) standpoint and for managing KMeans' performance *(see below: Number of variables)*.

For mixed data types, it might be needed to pre-process the features beforehand. Techniques such as Principal Components Analysis (PCA) or Singular Value Decomposition (SVD) can, however, be used to transform the input data into a dataset that can be leveraged appropriately into KMeans.

Number of variables

The number of variables going into K-means has an impact on both the time/complexity it takes to train and apply the algorithm, but as well as an effect on how the algorithm behaves.

This due to the curse of dimensionality:

> So as the dimensionality increases, more and more examples become nearest neighbors of xt, until the choice of nearest neighbor (and therefore of class) is effectively random.
>
> https://homes.cs.washington.edu/~pedrod/papers/cacm12.pdf

A large number of dimensions has a direct impact on distance-based computations, a key component of KMeans:

> The distances between a data point and its nearest and farthest neighbours can become equidistant in high dimensions, potentially compromising the accuracy of some distance-based analysis tools.
> https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2238676/

Dimensionality reductions methods such as PCA, or feature selection techniques are things to bring up when reaching this topic.

Comparison with other Algorithm

Besides understanding the inner working of the KMeans algorithm, it is also important to know how it compares to other clustering algorithms.

There is a wide range of other algorithms out there, hierarchical clustering, mean shift clustering, Gaussian mixture models (GMM), DBScan, Affinity propagation (AP), K-Medoids/ PAM, …

> What other clustering methods do you know?
>
> How does Algorithm X, compares to K-Means?

Going through the list of algorithms, it is essential to at least know the different types of clustering methods: centroid/medoids (e.g., KMeans), hierarchical, density-based (e.g., MeanShift, DBSCAN). distribution-based (e.g., GMM) and Affinity propagation (Affinity Propagation)…

When doing these types of comparisons, it is important to list at least some K-Means alternatives, and showcasing some high-level knowledge of what the algorithm does and how it compares to K-Means.

You might be asked at this point to deep dive into one of the algorithms you previously mentioned, so be prepared to be able to explain how some of the other algorithm works, list their strengths and weakness compared to K-means and describe how the inner working of the algorithm differs from K-Means.

Advantages / Disadvantage of using K-Means

Going through any algorithms, it is important to know their advantage and disadvantage, it is not unsurprising that this is often asked during interviews.

Some of the key advantages of KMeans are:

1. It is simple to implement
2. Computational efficiency, both for training and prediction
3. Guaranteed convergence

While some of its disadvantages are:

1. The number of clusters needs to be provided as an input variable.
2. It is very dependent on the initialization process.
3. KMeans is good at clustering when dealing with spherical cluster shapes, but it performs poorly when dealing with more complicated shapes.
4. Due to leveraging the Euclidian distance function, it is sensitive to outliers.
5. Need pre-processing on mix data as it can't take advantages of alternative distance function such as Gower's distance

. . .

More from me on Hacking Analytics:

- SQL interview Questions For Aspiring Data Scientist — The Histogram
- Python Screening Interview questions for DataScientists
- ON Applying K-means Personalization to a website
- ON Coding K-Means in Vanilla Python
- How to Learn Data science from scratch

Data Data Science Machine Learning Statistics Interview