**K-Means Clustering Algorithm**

Jupyter Notebook:

* K-Means\_Algorithm.ipynb
* kmeans-clustering-Scratch(behind the scene).ipynb

References:

* <https://towardsdatascience.com/k-means-clustering-algorithm-applications-evaluation-methods-and-drawbacks-aa03e644b48a>
* Impo. <https://blog.bioturing.com/2018/10/17/k-means-clustering-algorithm-and-example/>
* <https://www.kaggle.com/tonzowonzo/simple-k-means-clustering-on-the-iris-dataset>

# **Clustering**

* Clustering is one of the most common exploratory data analysis techniques used to get an intuition about the structure of the data. It can be defined as the task of identifying subgroups in the data such that data points in the same subgroup (cluster) are very similar while data points in different clusters are very different.
* In other words, we try to find homogeneous subgroups within the data such that data points in each cluster are as similar as possible according to a similarity measure such as euclidean-based distance or correlation-based distance. The decision of which similarity measure to use is application-specific.
* Clustering analysis can be done on the basis of features where we try to find subgroups of samples based on features or on the basis of samples where we try to find subgroups of features based on samples. We’ll cover here clustering based on features. Clustering is used in market segmentation; where we try to find customers that are similar to each other whether in terms of behaviors or attributes, image segmentation/compression; where we try to group similar regions together, document clustering based on topics, etc.
* Unlike supervised learning, clustering is considered an unsupervised learning method since we don’t have the ground truth to compare the output of the clustering algorithm to the true labels to evaluate its performance. We only want to try to investigate the structure of the data by grouping the data points into distinct subgroups.

# **Kmeans Algorithm**

Kmeans algorithm is an iterative algorithm that tries to partition the dataset into Kpre-defined distinct non-overlapping subgroups (clusters) where each data point belongs to only one group. It tries to make the inter-cluster data points as similar as possible while also keeping the clusters as different (far) as possible.

It assigns data points to a cluster such that the sum of the squared distance between the data points and the cluster’s centroid (arithmetic mean of all the data points that belong to that cluster) is at the minimum. The less variation we have within clusters, the more homogeneous (similar) the data points are within the same cluster.

The way the K-means algorithm works is as follows:

1. Specify the number of clusters K.
2. Initialize centroids by first shuffling the dataset and then randomly selecting K data points for the centroids without replacement.
3. Keep iterating until there is no change to the centroids. i.e assignment of data points to clusters isn’t changing.

* Compute the sum of the squared distance between data points and all centroids.
* Assign each data point to the closest cluster (centroid).
* Compute the centroids for the clusters by taking the average of all data points that belong to each cluster.

The approach kmeans follow to solve the problem is called Expectation-Maximization. The E-step is assigning the data points to the closest cluster. The M-step is computing the centroid of each cluster

**Few things to note here:**

* Since clustering algorithms including kmeans use distance-based measurements to determine the similarity between data points, it’s recommended to standardize the data to have a mean of zero and a standard deviation of one since almost always the features in any dataset would have different units of measurements such as age vs income.
* Given kmeans iterative nature and the random initialization of centroids at the start of the algorithm, different initializations may lead to different clusters since kmeans algorithm may be stuck in a local optimum and may not converge to the global optimum. Therefore, it’s recommended to run the algorithm using different initializations of centroids and pick the results of the run that that yielded the lower sum of squared distance.

# **Elbow Method**

**Elbow** method gives us an idea of what a good *k* number of clusters would be based on the sum of squared distance (SSE) between data points and their assigned clusters’ centroids. We pick *k* at the spot where SSE starts to flatten out and forming an elbow. We’ll use the geyser dataset and evaluate SSE for different values of *k* and see where the curve might form an elbow and flatten out. - Check Jupiter notebook for code example

**Below are the main takeaways:**

* Scale/standardize the data when applying the kmeans algorithm.
* Elbow method in selecting a number of clusters doesn’t usually work because the error function is monotonically decreasing for all *k*s.
* Kmeans gives more weight to the bigger clusters.
* Kmeans assumes spherical shapes of clusters (with a radius equal to the distance between the centroid and the furthest data point) and doesn’t work well when clusters are in different shapes such as elliptical clusters.
* If there is overlapping between clusters, kmeans doesn’t have an intrinsic measure for uncertainty for the examples belong to the overlapping region in order to determine for which cluster to assign each data point.
* Kmeans may still cluster the data even if it can’t be clustered such as data that comes from *uniform distributions*.

### **Disadvantages of K-Means**

* The biggest disadvantage is that K-Means requires you to pre-specify the number of clusters (k).
* K-Means is a lazy learner where generalization of the training data is delayed until a query is made to the system. This means K-Means starts working only when you trigger it to, thus lazy learning methods can construct a different approximation or result to the target function for each encountered query. It is a good method for online learning, but it requires a possibly large amount of memory to store the data, and each request involves starting the identification of a local model from scratch.

### **Algorithm**

Our algorithm works as follows, assuming we have inputs

x\_1, x\_2, x\_3, ..., x\_n and value of K.

**Step 1** - Pick K random points as cluster centers called centroids.

**Step 2** - Assign each X\_i to the nearest cluster by calculating its distance to each centroid.

**Step 3** - Find a new cluster center by taking the average of the assigned points.

**Step 4** - Repeat Step 2 and 3 until none of the cluster assignments change.

