**ML – K-means algorithm,beginning last part of unsupervised learning**

Links:

* <https://www.analyticsvidhya.com/blog/2021/01/a-simple-guide-to-centroid-based-clustering-with-python-code/>
* Slides on classroom for graphs and stuff

We’ve imposed the objective of gaining variance from data, this changes with **clustering**

We would like to derive a partition of the data in groups (clusters), with each group sharing at least one characteristic that sets it apart from other groups

**DEF:** Centroid:

* mean vector of a cluster of data points in a high-dimensional space

It’s not clear how the **number** of clusters is chosen yet, making clustering **ill-posed**

The core idea behind the algorithm is to find k centroids followed by finding k sets of points which are grouped based on the proximity to the centroid such that the squared distances of the points in the cluster to the centroid are minimized

**It aims to keep the centroids as small as possible while allocating data points to nearest clusters**

The k-means algorithm is NP-Hard, so it’s not optimal; we need to find:

* A set of centroids
* A set of assignment labels that create a mapping from data points to centroids

***Naïve k-means algorithm***

Given an arbitrarily given number of clusters to find (namely, *k*), and a random sample of the available centroids (using heuristic-like guessing):

1. **Select k points at random as centroids/cluster centers**
2. **Assign data points to the closest cluster** based on Euclidean distance Calculate centroid of all points within the cluster (i.e. minimize within-cluster variance, while maximizing inter-cluster variance), and update mean (μj)
3. Text, letter

   Description automatically generated **Repeat iteratively till convergence** (i.e. same points are assigned to the clusters in consecutive iterations)

Intuitively, each cluster k can be thought of as a group of points that lie close to each other around its mean/centroid μj (1 is the indicator function)

***Lloyd’s algorithm (Voronoi iteration)***

Lloyd’s algo initially chooses *k* datapoints as centroids, randomly (as seen in Naïve k-means)

it repeatedly finds the centroid of each set in the partition and then re-partitions the input according to which of these centroids is closest

In this setting, the mean operation is an integral over a region of space, and the nearest centroid operation results in Voronoi diagrams

