**Mini Batch K-means algorithm.**

[K-means](https://www.geeksforgeeks.org/k-means-clustering-introduction/) is one of the most popular clustering algorithms, mainly because of its good time performance. With the increasing size of the datasets being analyzed, the computation time of K-means increases because of its constraint of needing the whole dataset in main memory. For this reason, several methods have been proposed to reduce the temporal and spatial cost of the algorithm. A different approach is the ***Mini batch K-means algorithm***. **Mini Batch K-means algorithm**‘s main idea is to use small random batches of data of a fixed size, so they can be stored in memory. Each iteration a new random sample from the dataset is obtained and used to update the clusters and this is repeated until convergence. Each mini batch updates the clusters using a convex combination of the values of the prototypes and the data, applying a learning rate that decreases with the number of iterations. This learning rate is the inverse of the number of data assigned to a cluster during the process. As the number of iterations increases, the effect of new data is reduced, so convergence can be detected when no changes in the clusters occur in several consecutive iterations.

Mini-batch K-means is a variation of the traditional K-means clustering algorithm that is designed to handle large datasets. In traditional K-means, the algorithm processes the entire dataset in each iteration, which can be computationally expensive for large datasets.

Mini-batch K-means addresses this issue by processing only a small subset of the data, called a mini-batch, in each iteration. The mini-batch is randomly sampled from the dataset, and the algorithm updates the cluster centroids based on the data in the mini-batch. This allows the algorithm to converge faster and use less memory than traditional K-means.

The process of mini-batch K-means algorithm can be summarized as follows:

Initialize the cluster centroids randomly or using some other method.  
Repeat the following steps until convergence or a maximum number of iterations is reached:  
Select a mini-batch of data from the dataset.  
Assign each data point in the mini-batch to the closest cluster centroid.  
Update the cluster centroids based on the data points assigned to them.  
Return the final cluster centroids and the assignments of data points to clusters.

The algorithm takes small randomly chosen batches of the dataset for each iteration. Each data in the batch is assigned to the clusters, depending on the previous locations of the cluster centroids. It then updates the locations of cluster centroids based on the new points from the batch. The update is a gradient descent update, which is significantly faster than a normal *Batch K-Means update*.

As the number of clusters and the number of data increases, the relative saving in computational time also increases. The saving in computational time is more noticeable only when the number of clusters is very large. The effect of the batch size on the computational time is also more evident when the number of clusters is larger. It can be concluded that, increasing the number of clusters, decreases the similarity of the mini batch K-means solution to the K-means solution. Despite that the agreement between the partitions decreasing as the number of clusters increases, the objective function does not degrade at the same rate. It means that the final partitions are different, but closer in quality.