# A Critique of An Efficient K-means Clustering Algorithm - Kanungo et. All.

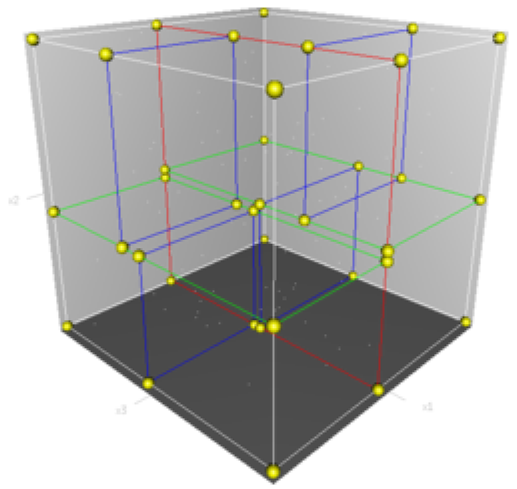
## A SHORT SUMMARY & CRITIQUE

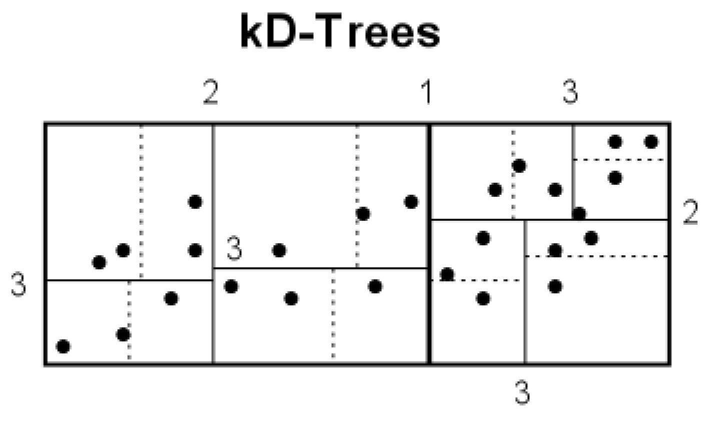
The k-means algorithm is a clustering algorithm that recognizes the patterns in data and groups them together statistically. For example, the nose of all human beings is very similar and hence would appear closer to each other graphically as compared to a set of ears. Similarly, the numbers 1 and 7 are quite similar in shape and if we visualize them graphically, we would observe that their clusters are very close to each other or are conjoined. This is very evident through visualizing of the MNIST dataset using PCA.

The k-means algorithm provides an unsupervised form of learning where the algorithm groups and forms these clusters as mentioned above. The simplest implementation of the k-means is the Lloyd's Algorithm. The algorithm is as below:

1. Initialize k centers randomly or through any other feasible means.
2. Compute the closest center to the data points and assign the data points to the respective clusters.
3. Now with the clusters of points formed, it is obvious that the cluster must have center point. This need not coincide necessarily with the initial centers chosen. Hence, we move the respective centroids to their cluster centers.
4. We repeat steps 1, 2 & 3 until we obtain a minima (In k-means, the algorithms does not necessarily converge to a global minima. Hence it is important to run the algorithm many with different center initializations).

Although the algorithm converges to minima, it greatest con that must be observed from this algorithm is that it computes distance for the farthest of the points which mostly might never appear in a center's cluster. This just adds to the computation costs! Hence, a much more sensible way would to calculate the distances of points that are in proximity a center or in other words selecting only the subspace that seems more relevant to be computed distances with. This can be done using the kd-tree, which is essentially a Binary Search Tree with d dimension and k centers.





Split longer dimension near data median

The kd-tree as can be seen in the above two figures, splits the longest dimension at its median (Any other method can also be used to split). The points belonging to the respective regions after the split are run down the tree and this process is repeated until log(n) depth, where n is the number of points.

Now the all we have to do is prune the generated centers after running them down to their respective nodes. This paper provides an implementation for the same which they call "Filtering Algorithm". The filtering algorithm is as follows:

1. Find the closest point $z*$ to the cell node $u$.
2. To check whether this point $z*$ is truly the closest to all parts of the cell and not another candidate $z$, we define a bisecting hyperplane to the vector $\vec{u} = z - z^*$.
3. Choose the closest point of the cell to the hyperplane. If the distance of this farthest point from $z \geq$ distance of this farthest point from $z^*$. Then, prune $z$.
4. Now with $z^*$ update it to the subspace's centroid of the points i.e., weighted center of the cell / the number of points in the respective cell.
5. Repeat this process until the candidates localize to a minima.

This algorithm presents a significant improve in both the run time and the computation cost. But with the scale of dimensionality, the time complexity increases exponentially due to the presence of the kd-tree. This algorithm also seems to perform very efficiently when the clusters are more distanced to each other.

## SOME OF THE BROADER THOUGHTS & QUESTIONS…

1. This is essentially the gist of Neural Networks where it is trying to fit a randomly initialized curve to the labelled clusters. At the end of the day, both of these are performing the same function! Recognizing patterns and fitting into the data.
2. This leads us to the next question, whether it is possible to efficiently form these clusters unsupervisedly as at the end of the day, these are multidimensional groups of patterns that are very similar and hence are in proximity to each other!