G SuryaDev Reddy
14CO214
8123997429
gouru.surya@gmail.com

An Efficient k-Means Clustering Algorithm: Analysis and Implementation

By Tapas Kanungo, David, Nathan, Christine and Angela

1. **Essence of the Paper:**
   - K-means clustering algorithm partitions n observations into k clusters in which each observation belongs to the cluster with the nearest center. Although finding an exact solution to the k-means problem for arbitrary input is NP-hard, the standard approach to finding an nearest solution often called *Lloyd's k-means* algorithm is used widely as it finds reasonable solutions quickly
   - However Llyod's algorithm has several shortcomings like the worst case running time is polynomial. This is mainly due to the cost of computing the nearest centers from the data points.
   - This paper uses kd-tree to speed up the k-means stages by *filtering* or *pruning* the centers at every level of kd-tree. Hence this algorithm is named as Filtering algorithm
   - The experiments have shown that the filtering algorithm significantly outperforms the brute force and kd-center algorithm in all the cases provided the dimensions are moderate between 1 to 20

2. **Work Described:**
   - Presented a simple and efficient implementation of Llyod's k-means clustering algorithm called *Filtering algorithm* by using kd-tree as a major data structure to store the data points
   - They have shown that as the cluster separation increases, the running time of the algorithm decreases.
   - They also compared the performances of three algorithms *Filtering*, *Brute Force* and *kd-center* for both synthetically generated data and data used in real applications.

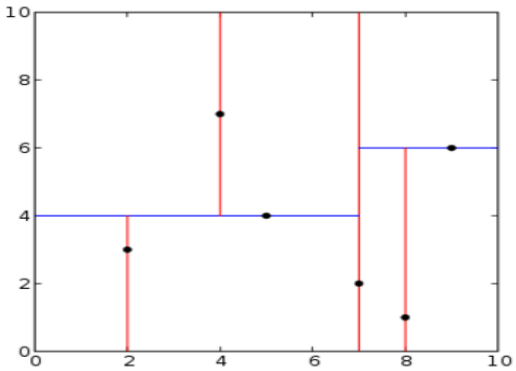
3. **Importance of work in Pattern Analysis and Machine Intelligence**
   - K-means Clustering is used in many different applications, such as data mining and knowledge discovery ,data compression and vector quantization , and pattern recognition and pattern classification
   - A straightforward implementation of Lloyd's algorithm can be quite slow. So this paper provides an efficient implementation of Lloyd's k-means algorithm
   - The Algorithm proposed differs from existing approaches only in how nearest centers are computed, so it could be applied to many variants of Llyod's algorithm.

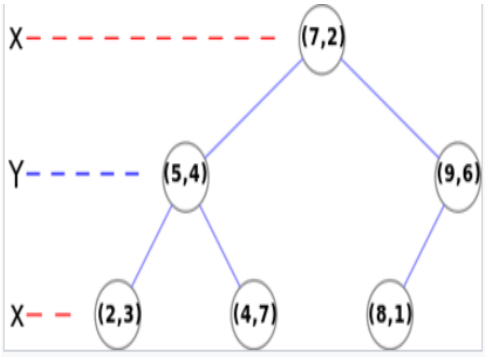
4. **Methodology:**

K-Dimensional tree:

K-d tree decomposition for the point set

(2,3) , (5,4) , (9,6) , (4,7) , (8,1) , (7,2)





Algorithm stores the multi-dimensional data points in a kd-tree

- Box is an axis-aligned hyper-rectangle
- Bounding Box of a point set is a smallest box containing all the points

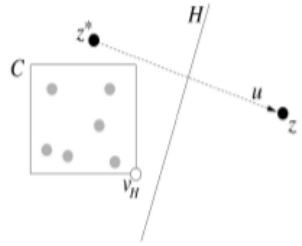
Each Node of the kd-tree is associated with a closed box called a cell C

- Root node's cell C is a bounding box of the given point set

If the cell contains at most one point, then it is a *leaf*
Otherwise split into two hyper rectangles by an axis-orthogonal hyperplane passing through the median of the associated data points

Pruning/ Filtering:
For each node of the kd-tree , we maintain a set of candidate centers Z.
Z → subset of center points that might serve as nearest neighbor for some point lying within the associated cell
z* → closest point in Z to C's midpoint
For each (z belongs to Z \ {z*})

- If no part of C is closer to z than it is to z* , then **z** can be **pruned** or **removesd from the Z**



Determining if z is closer to any point in C than from z* to that point:

- If cell lies entirely on one side of the bisecting plane H then the center z lying on the other side is pruned or removed from the set of candidate centers.
- If cell lies partially on the both sides of bisecting hyper plane H, then the both centers lying on either sides are retained. No pruning takes place**.**

For each *internal node u* in the tree,

u.count	←	associated data points	**z, z*** - centers
u.wgtCent	←	vector sum of all the associated points	**C**- cell
Actual Centroid	←	u.wgtCent/u.count.	**H** - Hyperplane

- For each stage of Lloyd's algorithm, for each of the k centers, we need to compute the centroid of the set of data points for which this center is closest.
- We then move this center z* to the computed centroid and proceed to the next stage.

5. Pros:

- This *Filtering* algorithm significantly outperformed the *Brute Force and kd-center* algorithms in all the cases for both synthetic data and real data.
- *Filtering* algorithm uses kd-tree to store all the *data points* and since data points do not vary throughout the computation, the kd-tree data structure does not need to be recomputed at each stage. But in case of *kd-center* algorithm**,** it operates by building kd-tree with respect to *center points* and hence kd-tree is rebuilt at the start of each stage because centers tend to change after every stage.

6. Cons:

- Algorithm might encounter scenarios in which it degenerates to brute-force search. For example, if the center points are all located on a unit sphere centered at the origin and the data points are clustered tightly around the origin. Because the centers are nearly equidistant to any subset of data points, very little pruning takes place and the algorithm degenerates to a brute force O(kn) search.
- Filtering algorithm's running time increases exponentially with dimension. Thus, as the dimension increases, the speed advantage would tend to diminish.

7. Future Work Proposed:

In the later stages of Llyod's algorithm as the centers are converging to their final positions, the most number of the data points have the same closest center from one stage to the next but this algorithm does not exploit this coherence to improve the running time.

A Kinetic method was proposed along these lines but this algorithm is quite complex and does not have faster running time in practice. Hence the Future work would be to combine the best elements of kinetic and filtering approaches.

8. Work planning to take up:

I'm planning to implement this efficient k-means Lloyd's Filtering algorithm using kd-tree data structure to store the multi-dimensional data points. Hence Faster running time can be achieved as the data structure does not have to be recomputed at each stage and also because of filtering or pruning the centers as and when the tree is propagated down the levels.