Faster K-Means Cluster Estimation

Siddhesh Khandelwal SIDDHESH166@GMAIL.COM
Amit Awekar AMITAWEKAR@GMAIL.COM
Indian Institute of Technology Guwahati, Guwahati, Assam, 781039.

Abstract

K-means is a widely used iterative clustering algorithm. There has been considerable work on improving k-means in terms of mean squared error (MSE) and speed, both. However, most of the k-means variants tend to compute distance of each data point to each cluster centroid for every iteration. We propose two heuristics to overcome this bottleneck and speed up k-means. Our first heuristic predicts the candidate clusters for each data point by looking at nearby clusters after first iteration of k-means. Our second heuristic further reduces this candidate cluster list aggressively. We augment well known variants of k-means with our heuristics to demonstrate effectiveness of our heuristics. For various synthetic and real-world datasets, our heuristics achieve speed-up of up-to 10 times without significant increase in MSE.

1. Introduction

The objective of K-means is to partition a dataset D with n data points in d dimensions into k clusters such that the mean squared error (MSE) is minimized. The K-means problem is NP-hard. Polynomial time heuristics are commonly applied to obtain a local minimum.

One such popular heuristic is the Lloyd's algorithm(Lloyd, 1982) that selects certain initial centroids at random from the dataset. Each data point is assigned to the cluster corresponding to the closest centroid. Each centroid is then recomputed as mean of the points assigned to that cluster. This procedure is repeated until convergence. Each iteration involves $n*k$ distance computations. Our contribution is to reduce this cost to $n*k'$, where k' is significantly smaller than k.

Over the years, there has been significant work to improve Lloyd's algorithm both in terms of MSE and speed. These improvements can be classified into the following broad domains: Improving initial seed selection(Likas et al., 2003; Arthur & Vassilvitskii, 2007), Selecting ideal value for number of clusters(Pham et al., 2005), and Estimating bounds for distance between data point and cluster centroid(Elkan, 2003; Fahim et al., 2006). Our work does not compete with these variants of K-means. Rather, we augment these variants of K-means with our work to demonstrate effectiveness of our work.

We propose two heuristics that significantly reduce the number of data point to cluster centroid distance computations. Our first heuristic considers only a subset of nearby cluster as candidates for deciding membership for a data point. This heuristic has advantage of speeding up K-means clustering with marginal increase in MSE at the cost of memory overhead. Candidate cluster list for a data point can be further reduced aggressively depending upon how close the data point is to the centroid of its current cluster. As compared to first heuristic, second heuristic gives two benefits: reduced number of distance computations and reduced memory footprint. However, second heuristic results in slightly higher value of MSE.

2. Our Work

Our main contribution is in defining two heuristics that can be used as augmentation to current variants of k-means for faster cluster estimation. Let algorithm V be a variant of k-means and algorithm V' be the same variant augmented with our heuristics. Let T be the time required for V to converge to MSE value of E. Similarly, T' is the time required for V' to converge to MSE value of E'. We should satisfy following two conditions when we compare V with V':

- Condition 1: T' is smaller than T, and
- Condition 2: E' is not significantly higher than E.

Appearing in Proceedings of the 2^{nd} Indian Workshop on Machine Learning, IIT Kanpur, India, 2016. Copyright 2016 by the author(s).

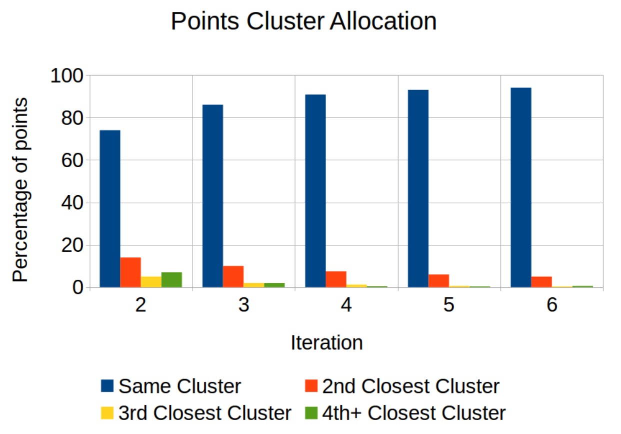

Figure 1. Point Distribution per Iteration - Compared to Cluster Assignment in Previous Iteration

In short, these conditions state that a K-means variant augmented with our heuristics should converge faster without significant increase in final MSE.

2.1. Heuristic 1: Candidate cluster list for each data point

This heuristic leverages a list of candidate clusters for each data point to reduce distance computations. Let size of this list be k'. We assume that k' is significantly smaller than k. We build this candidate cluster list based on top k' nearest clusters to the data point after first iteration of K-means. Now each iteration of K-means will perform only $n * k'$ distance computations.

Motivation for this approach comes from the observation that data points have tendency to go to clusters that were closer in the previous iteration. Figure 1 shows an example execution of k-means for a synthetic dataset of 100,000 points in 10 dimensions which needs to be partitioned into 100 clusters. X axis represents iteration of the algorithm and Y axis represents percentage of points that get assigned to a particular cluster. As we progress through the algorithm, we observe from the figure that more points get assigned to same cluster or clusters that were close in previous iteration.

Consider a data point p_1 and various cluster centroids represented as $c_1, c_2 ..., c_k$. Initially, all centroids are chosen randomly or using one of the seed selection algorithms mentioned in Section 3. Let us assume that k'= 4 and after first iteration c_8, c_5, c_6, and c_1 are the top four closest centroids to p_1 in the increasing order of distance. If we run K-means for second iteration, p_1 will compute distance to all k centroids. After second iteration, top four closest centroid list might change in two ways:

- Members of the list do not change but only ranking changes among the members. For example, top four closest centroid list for p_1 might change to c_1, c_6, c_8, and c_5 in the increasing order of distance.

- Some of the centroids in the previous list are replaced with other centroids which were not in the list. For example, top four closest list for p1 might change to c_5, c_2, c_9, and c_8 in the increasing order of distance

However, we have observed that for synthetic as well as real world datasets, second case occurs rarely. Therefore, top k' closest centroid list after first iteration for a data point is a good enough estimate for the closest cluster for that data point after K-means converges. Our heuristic involves memory overhead of $O(n * k')$ to maintain candidate cluster list for each point and one time computation of candidate cluster list.

2.2. Heuristic 2: Reduction in size of candidate cluster list

Heuristic 2 dynamically reduces size of candidate cluster list for some of the data points if they are closer to centroid of current cluster. This heuristic is based on observation that points closer to centroid almost never change their cluster membership. Please refer to Figure 1. During later iterations, more than 90% of the data points maintain their current cluster membership. Computing distance of such data points to all members of candidate cluster list is redundant.

We define closeness of a data point to its current cluster as ratio of cluster radius to distance of that data point from cluster centroid. If closeness of data point is more than parameter α then we assume that the data point will not change its membership in further iterations and reduce the size of candidate cluster list to just one. This has two advantages: reduction in memory overhead to maintain candidate cluster lists and reduction in distance computations. We have observed that closeness value after second iteration is a good enough estimate of closeness value for later iterations. Closeness computation is done only once and candidate cluster list is truncated for points having closeness more than parameter α. It is possible that closeness for a point might decrease in further iterations. But we have observed that it happens rarely and avoid re-computation of closeness value for all data points. This aggressive truncation of candidate cluster list reduces running time at the cost of slight increase in the MSE.

3. Related Work

In last three decades, there has been significant work on improving Lloyd's algorithm (Lloyd, 1982) both in terms of reducing MSE and increasing speed. Arthur et. al.(Arthur & Vassilvitskii, 2007) provided a better method for seed selection based on a probability distribution over closest cluster centroid distances for each data point. Likas et. al.(Likas et al., 2003) proposed the Global k-means method for selecting one seed at a time to reduce final mean squared error. Pham et. al.(Pham et al., 2005) designed a novel function to evaluate goodness of clustering for various potential values of number of clusters. Elkan(Elkan, 2003) use triangle inequality to avoid redundant computations of distance between data points and cluster centroids. Fahim et. al.(Fahim et al., 2006) proposed a method that avoids redundant calculations if a data point moves closer to centroid of its current cluster.

Seed selection based K-means variants differ from Lloyd's algorithm only for the method of seed selection. Our heuristics can be directly used in such algorithms. K-means variants that find appropriate number of clusters in data, evaluate goodness of clustering for various potential values of number of clusters. Such algorithms can use our heuristics while generating clustering for each potential value of k in K-means. K-means variants that compute exact distances only to few centroids for each data point, compute bounds on distances to rest of the centroids for each data point. Our heuristics can help such K-means variants to further reduce distance and bound calculations.

4. Experimental Results

Our heuristics can be used in augmentation with multiple variants of K-means mentioned in Section 3. However due to lack of space, we present results of augmenting our heuristics only with K-means with triangle inequality (KMT)(Elkan, 2003). Augmenting KMT with our heuristic 1 is referred as algorithm H1 and augmentation with heuristic 2 is referred as algorithm H2.

During each iteration KMT, a data point computes distance to the centroid of its current cluster. KMT uses triangle inequality to compute efficient lower bounds on distances to all other centroids. A data point will compute exact distance to any other centroid only when lower bound on such distance is smaller than distance to the centroid of its current cluster. During each iteration of H1, a data point will also compute distance to the centroid of its current cluster. However, H1 will compute lower bounds on distances to centroids only in candidate cluster list. A data point will compute exact distance to any other centroid in the candidate cluster list only when lower bound on such distance is smaller than distance to the centroid of its current cluster. Each iteration of H2 is similar to H1 with the difference that H2 will truncate candidate cluster list of some data points after second iteration.

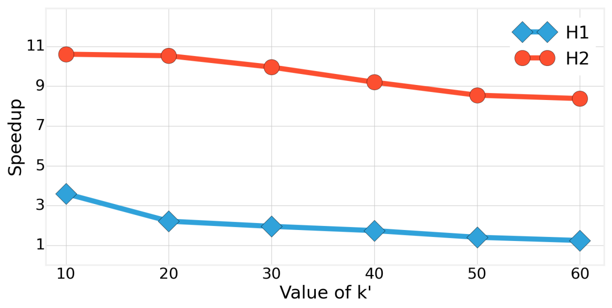

Figure 2. Speed Up Comparison

Experimental results presented here are done on a real world dataset called the "covertype" dataset (soil and forest cover measurements). A sample of 100000 points was used for our experiments. The dataset is in 54 dimensions and the value of the number of clusters (k) is set to 100. We chose this dataset because it was used by Elkan et. al.(Elkan, 2003) to demonstrate effectiveness of KMT. For H2, value of parameter α is set to two.

4.1. Comparison of T with T'

Please refer to Figure 2. X axis represents size of candidate cluster list (k'). Y axis represents speed up over KMT. Speed up of an algorithm is calculated as running time of KMT divided by running time of that algorithm. Running time of KMT is independent of value of k'. For small values of k', H1 provides good speed up over KMT. This is expected as for small value of k', H1 can avoid many redundant distance computations using small candidate cluster list. Speed up of H1 over KMT is not same as ratio k/k'. Reason for less speed up is that KMT also avoids some distance computations using its own filtering criteria of triangle inequality. As value of k' increases to 60, running time of H1 approaches running time of KMT. H2 performs significantly better than H1. Aggressive pruning of candidate cluster list pays off especially well for higher values of k'. This shows that our heuristics satisfy Condition 1 mentioned in Section2.

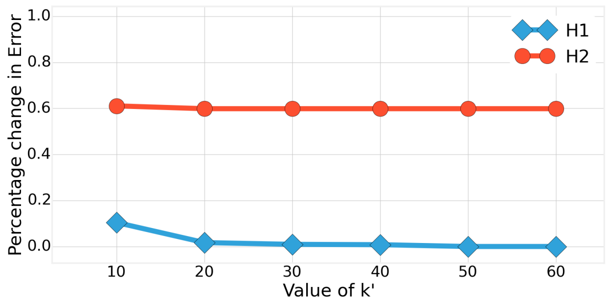

Figure 3. MSE Comparison

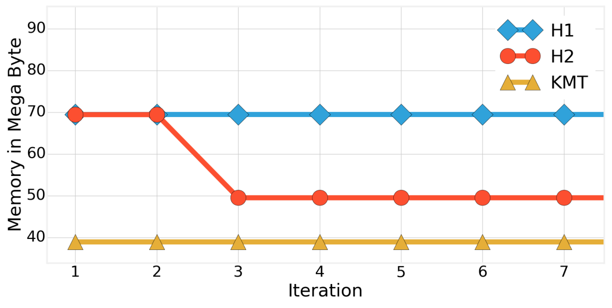

Figure 4. Memory Footprint Comparison

4.2. Comparison of E with E'

Please refer to Figure 3. X axis represents size of candidate cluster list (k'). Y axis represents percentage change in final MSE of an algorithm as compared to KMT. MSE of KMT is independent of k'. MSE of H1 is always slightly higher than KMT and it increases marginally as value of k' is reduced. MSE of H2 is slightly higher than H1 for all values of k' as H2 is aggressively reduces distance computations by pruning candidate cluster list for some data points. Even for small value of k', MSE of H2 is only 0.6 percent higher than KMT. This shows that our heuristics satisfy Condition 2 mentioned in Section 2.

4.3. Comparison of memory footprint

Please refer to Figure 4. X axis represents various iterations of a particular execution of an algorithm. Y axis represents memory footprint of the algorithm in megabytes. Value of k' is fixed to 40. Memory footprint of KMT remains constant through all iterations. Memory footprint of H1 is also constant but higher than KMT as H1 stores candidate cluster list for each data point. H2 starts with same memory footprint as H1, but quickly reduces its memory footprint by aggressively truncating candidate cluster list for many data points after second iteration. This shows effectiveness of H2 at reducing memory requirement.

5. Conclusion and Future Work

We presented two heuristics to attack the bottleneck of redundant distance computations in K-means. Our first heuristic limits distance computations for each data point to a small candidate cluster list. Our second heuristic further reduces this candidate cluster list to achieve higher speed up and lower memory footprint at the cost of slightly higher MSE. Our heuristics can be easily augmented with existing variants of K-means.

We believe that bottleneck of redundant distance computations is not unique to K-means. It will be interesting to see whether other iterative clustering algorithms have similar bottleneck and can we speed them up by avoiding redundant computations.

Dual Submission

Our work is currently under submission at the International Conference on Information and Knowledge Management (CIKM 2016). It will be held from October 24 to October 28, 2016 in Indianapolis, United States. The URL to the conference website is `http://cikm2016.cs.iupui.edu/`

References

Arthur, David and Vassilvitskii, Sergei. k-means++: The advantages of careful seeding. In *ACM-SIAM symposium on Discrete algorithms*, pp. 1027–1035, 2007.

Elkan, Charles. Using the triangle inequality to accelerate k-means. In *International Conference om Machine Learning*, pp. 147–153, 2003.

Fahim, AM, Salem, AM, Torkey, FA, and Ramadan, MA. An efficient enhanced k-means clustering algorithm. *Journal of Zhejiang University SCIENCE A*, 7(10):1626–1633, 2006.

Likas, Aristidis, Vlassis, Nikos, and Verbeek, Jakob J. The global k-means clustering algorithm. *Pattern recognition*, 36(2):451–461, 2003.

Lloyd, Stuart P. Least squares quantization in pcm. *Information Theory, IEEE Transactions on*, 28(2): 129–137, 1982.

Pham, Duc Truong, Dimov, Stefan S, and Nguyen, CD. Selection of k in k-means clustering. *Journal of Mechanical Engineering Science*, 219(1):103–119, 2005.