CS229 Final Project: k-Means Algorithm

Colin Wei and Alfred Xue SUNet ID: colinwei axue

December 11, 2014

1 Notes

This project was done in conjuction with a similar project that explored k-means in CS 264. Relevant parts of our paper are shared between the two. However, the majority of the two papers are different – the CS 264 paper focuses on theoritical research that has been done on k-means, while this project focuses on the differences between $k-means$ and single-link++, with a more practical approach.

2 Introduction

The k-means algorithm is an algorithm used commonly for clustering points in $\mathbb{R}^n$. While this algorithm works quite well in practice, there are two aspects of this algorithm that are hard to grasp theoretically.

First, it has been hard to prove any meaningful upper bound on the running time of this algorithm. The worst case running time of k-means has been shown to be $2^{\Omega(n)}$ [7], although in practice, k-means takes a sublinear number of iterations to converge for real datasets. Smoothed analysis has given polynomial-time bounds to this problem, but even these bounds are much higher than what has been observed empirically. We will summarize some of the smoothed analysis work on k-means.

Second, it has been hard to quantify the accuracy of the solution that k-means converges to. Although k-means has guaranteed convergence because each step of the algorithm performs coordinate descent on the k-means objective, the algorithm rarely converges to the exact optimal solution because it mostly gets stuck at local minima. Modifications of the k-means algorithm with different centroid initialization procedures, such as k-means++, have improved the accuracy of convergence.

We are interested to see if k-means performance correlated with notions of stability we discussed in class. We were interested in γ-perturbation stability in particular. Unfortunately, the proof of the (c, ϵ)-stability discussed in [1] does not extend to γ pertubation stability. Upon intital testing, we found little indication that *gamma*-stability increased k-means performance. We also noticed that k-means seemed to get stuck on local minimum more when the pertubation stability factor increased. To explore this, we implemented k-means and compared its performance to single-link++.

3 Definitions

The k-means clustering problem is as follows: given a set P of n points $p_1, ..., p_n \in \mathbb{R}^d$, choose a set C of k centers $c_1, ..., c_k \in \mathbb{R}^d$ to minimize the objective function

$$\phi_C(P) = \sum_{p \in P} \min_{c \in C} ||p - c||^2$$

The k-means algorithm for this problem works as follows:

1. Randomly initialize cluster centroids $c_1, ..., c_k \in \mathbb{R}^d$.
2. Until convergence, repeat:
   i. For every point in the data set p_i, let $w_i = \arg\min_j ||p_i - c_j||$.
   ii. For every cluster centroid c_j, set $c_j = \frac{\sum_{i=1}^n 1\{w_i = j\} p_i}{\sum_{i=1}^n 1\{w_i = j\}}$.

The single-link++ clustering method is implmeneted as follows:

1. Construct a tree as follows
   i. Place every point in its own tree.
   ii. Select the two points not already connected with the smallest distance between them.
   iii. Connect the heads of the two trees that corespond to these points. (that is, we construct a binary tree)
   iv. Repeat step ii. and iii. until there is only one tree remaining.
2. Prune the tree by removing k links starting from the head of the tree, where a link can only be broken from a head, that maximize the objective function

$$\phi_C(P) = \sum_{p \in P} \min_{c \in C} ||p - c||^2$$

   where the center of each cluster is the most optimal point for said cluster.

4 k-Means vs. Single-Link++

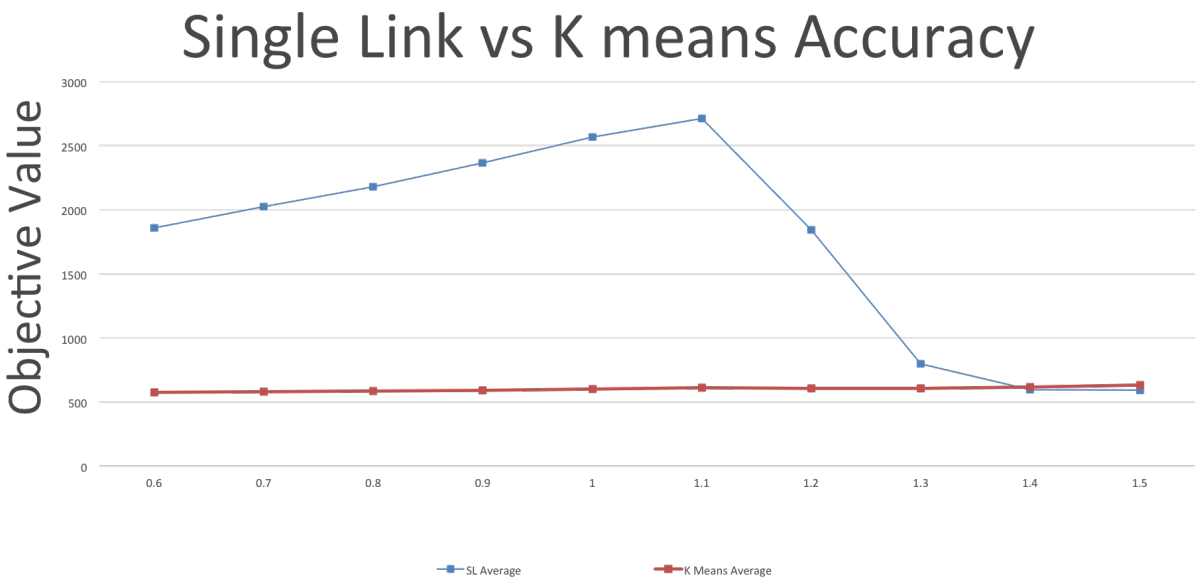
We wanted to test the performance of k-means and single-link++, due to [8] on data satisfying γ-perturbation stability to see how the two algorithms would compare. Furthermore, we wanted to see if γ-perturbation stability was a particularly useful notion for k-means at all, as it seemed like something that applied much better to hierarchical clustering algorithms. We generated our own data to satisfy γ-perturbation stability as follows:

1. We let $k = \gamma + 1$, and we placed cluster centers in $\mathbb{R}^3$ at $(k, 0, 0)$, $(0, k, 0)$, $(0, 0, 0)$, $(k, k, 0)$, $(k/2, k/2, k/\sqrt{2})$, $(k/2, k/2, -k/\sqrt{2})$.

2. For 1000 total points, we chose one of the 6 points and inserted a random point in a sphere of radius one around that center point. These 1000 random points formed our dataset.

Since our data was randomly generated, it might not have satisfied γ-perturbation stability perfectly if one of the actual centers was off, but we believed it approximated this well enough for our purposes. For each of the 10 values of γ from $0.6, 0.7, ..., 1.4, 1.5$, we generated 40 such data sets and ran single-link++ and k-means on both data sets. (Even though γ-perturbation stability is not a valid notion for $\gamma < 1$, we still used it as a parameter for controlling how separated our clusters were. The following chart shows the average value of the objective function for those data sets:

γ	SL++ Avg.	k-Means Avg.	SL++ Std. Dev.	k-Means Std. Dev.
0.6	1860.34	575.0438	35.58476	18.11837
0.7	2027.64	583.94	37.38439	6.852872
0.8	2180.089	587.9218	37.96719	9.915998
0.9	2364.864	593.7948	31.50132	9.957035
1.0	2569.541	602.3708	60.71504	40.21661
1.1	2714.904	613.0033	155.5746	76.3374
1.2	1845.208	606.7595	565.664	54.38815
1.3	798.727	606.5338	235.5272	62.24314
1.4	597.335	618.976	6.602786	96.25676
1.5	594.659	631.018	7.134685	130.2924



We found it really interesting that the performance of single-link++ dropped so quickly; we were surprised to see such a dramatic change. The other interesting thing we noticed about our results was how k-means became more and more erratic as perturbation stability increased, which seemed like a counterintuitive idea at first. The standard deviation of the k-means objective increased significantly from $\gamma = 0.6$ to $\gamma = 1.5$. However, this makes

sense because as data becomes more and more well-separated, local maxima to the k-means objective become worse relative to the optimal solution.

Before we ran these tests, we believed that stability notions would have some positive impact on either k-means runtime or approximation ratio. However, our tests showed no correlation between perturbation stability and the number of iterations k-means took and also negative results for approximation in terms of increasing standard deviation. Granted, our results could have been different had we chosen to implement k-means++ instead of k-means. However, it seems like in general, k-means is not very tractable even under stability assumptions. This makes sense, as k-means is used a lot for real world data that isn't well-separated.

5 Conclusion

k-means does well through all data. As far as time efficency goes, k-means is far superior, as even in the worst case of its run times only had 40 iterations. Since our dataset was large, even if k-means was run 25 different times, k-means would still be superior in time efficency. We notice that for low values of γ, the mminimum objective value of sl++ steadily increases until it reaches around 1.4. This can be explained by the idea that the number of misses remain constant, but each miss is punished more due to the larger pertubation factor. it is likely the case that for pertubation factors above 1.4, sl++ returns the optimal value, but has yet to be proven to be the case. The larger standard deviations as the pertubation factor increases for k-means could either indicate that it fails more as the pertubation factor increases, or simply that k-means is getting punished more for failed pertubations.

In conclusion, k-means seems to be the optimal method in practice, even for data with large pertubation factors. SL++ remains a popular tool, however, for theoritical purposes and bound exploration.

6 Future Research

Primary future research should focus on divising theory to support our researchw. Other areas include exploring SL++ and (c, ϵ) stability.

7 References

[1] Agarwal, Manu, Ragesh Jaiswal, and Arindam Pal. "k-means++ under Approximation Stability." Theory and Applications of Models of Computation: 84.

[2] Arthur, David, Bodo Manthey, and H. Roglin. "k-Means has polynomial smoothed complexity." Foundations of Computer Science, 2009. FOCS'09. 50th Annual IEEE Symposium on. IEEE, 2009.

[3] Arthur, David, and Sergei Vassilvitskii. "Worst-case and smoothed analysis of the ICP algorithm, with an application to the k-means method." Foundations of Computer Science, 2006. FOCS'06. 47th Annual IEEE Symposium on. IEEE, 2006.

[4] Ragesh Jaiswal and Nitin Garg. Analysis of k-means++ for separable data. In Proceedings of the 16th International Workshop on Randomization and Computation, pp. 591?602, 2012.

[5] David Arthur and Sergei Vassilvitskii. k-means++: the advantages of careful seeding. In Proceedings of the 18th annual ACM-SIAM symposium on Discrete Algorithms (SODA?07), pp. 1027-1035, 2007

[6] Ostrovsky, Rafail, et al. "The effectiveness of Lloyd-type methods for the k-means problem." Journal of the ACM (JACM) 59.6 (2012): 28.

[7] Andrea Vattani. k-means requires exponentially many iterations even in the plane. In Proc. of the 25th ACM Symp. on Computational Geometry (SoCG), pages 324-332, 2009.

[8] Awasthi, Pranjal, Avrim Blum, and Or Sheffet. "Center-based clustering under perturbation stability." Information Processing Letters 112.1 (2012): 49-54.

[9] Balcan, Maria-Florina, Avrim Blum, and Anupam Gupta. "Approximate clustering without the approximation." Proceedings of the twentieth Annual ACM-SIAM Symposium on Discrete Algorithms. Society for Industrial and Applied Mathematics, 2009.