CSE 847 Project Proposal

Yinyang K-means

Ben Frey
freybenj@msu.edu

Thomas Swearingen
swearin3@msu.edu

1. PROBLEM DESCRIPTION

K-means is a popular machine learning algorithm for clustering. As the amount of data has grown ever larger, the limitation of the classic K-mean algorithm has become more apparent. Specifically, the K-means algorithm is linear in data set size—the number of distance calculations is nki, where n is the number of data points, k is the number of desired clusters, and i is the number of iterations. This linearity reduces the usability of the algorithm with large datasets. However, Ding *et al.* propose Yinyang K-means [4] which seeks to solve this problem. The authors assert Yinyang K-means can be used in place of classical K-means with no extra conditions or requirements while simultaneously achieving an order of magnitude higher performance. We intend to explore the theoretical properties of this proposed method and verify the authors' claim of a significant speed up.

2. INTRODUCTION

K-means is a venerable clustering algorithm which has gained the trust of researchers through years of use. However, when the dimensionality, data set size, or number of desired clusters is large, k-means becomes prohibitively expensive. Attempts at increasing the performance of k-means mainly fall into two categories: working on improving the core algorithm or improving the performance through some other means (e.g. K-means++ [1]). This work focuses on the former method, which includes approximation and optimization. Within this realm, work has been done previously on structural or incremental optimization by [9], [6], and [5] and on approximation by several groups ([7], [3]), [10], [11], [12]). While this previous work has been of high quality, none of the innovations have gotten much traction or widespread use.

In the first category, *Elkan* [6] uses two lemmas to find a lower bound using the triangle inequality:

1. For a point x and centers b and c, if $d(b,c) \geq 2d(x,b)$ then $d(x,c) \geq d(x,b)$.
2. For a point x and centers b and c, $d(x,c) \geq \max\{0, d(x,b) - d(b,c)\}$.

The first lemma is used to avoid calculating distances to centers in cases where $\frac{1}{2}d(c,c') \geq d(x,c)$. The second lemma lets the algorithm use a lower bound $l \leq d(x,b)$, where x is any data point and b is any center. As long as center b does not move too much in an iteration, then l can be used as $d(x,b)$ without actually calculating the distance. Empirically, Elkan's algorithm speeds up k-means by factors from 1.50 to 351. The algorithm performs better with higher k and lower dimensionality - of the four public datasets used in evaluation, performance increased as k varied from 3 to 20 to 100 and dimensionality decreased from 784 to 2.

In one work from the approximation tack, Sculley [11] uses mini batches to greatly improve on the accuracy of the online stochastic gradient descent method demonstrated by Battou and Bengio in [2]. In this approach, Sculley first finds the closest centers to each data point in the mini batch. An array v tracks the number of data points associated with a given center c. The main loop iterates over the data points:

1. Retrieve the center associated with this point.
2. Update $v[c]$.
3. Calculate the center's learning rate as $\eta = \frac{1}{v[c]}$.
4. Perform the gradient step for center c.

Sculley's experiments showed that this mini-batch gradient descent with batch size 1000 converged orders of magnitude faster than classic k-means while achieving much better accuracy than Bottou and Bengio's method. (Here, the classic algorithm is treated as the ground truth for calculating accuracy.)

In *Drake and Hamerly* [5], the authors keep b lower bounds where $b < k$, k being the number of bounds kept in *Elkan* [6]. These bounds are the distances to the b nearest neighbors of the point in question, and are kept by tracking a point's center and the $1 \leq z \leq b$ closest centers. Of particular note, *Drake and Hamerly* tune b adaptively as follows: start at $b = \frac{k}{4}$; after each iteration, b becomes the number of useful bounds while staying at least $\frac{k}{8}$. Tests that Drake and Hamerly performed on their algorithm show that for a medium range of dimensions (25-125), it out-performs algorithms by Elkan [6], Hamerly [8], and the traditional k-means.

It has been hypothesized that in order to gain popularity in practical use, a replacement for k-means must meet three requirements: equivalent clustering to traditional k-means, consistent and significant performance gains, and simple to use.

The proposed work aims to satisfy those three requirements. It utilizes the triangle inequality in a novel way to keep track of two bounds: the upper bound on the distance from a given point to its assigned cluster center and the lower bound on the distance from the point to any other center. These two bounds act to reduce the number of distance calculations that need to be performed during the assignment step of k-means. This is achieved through two

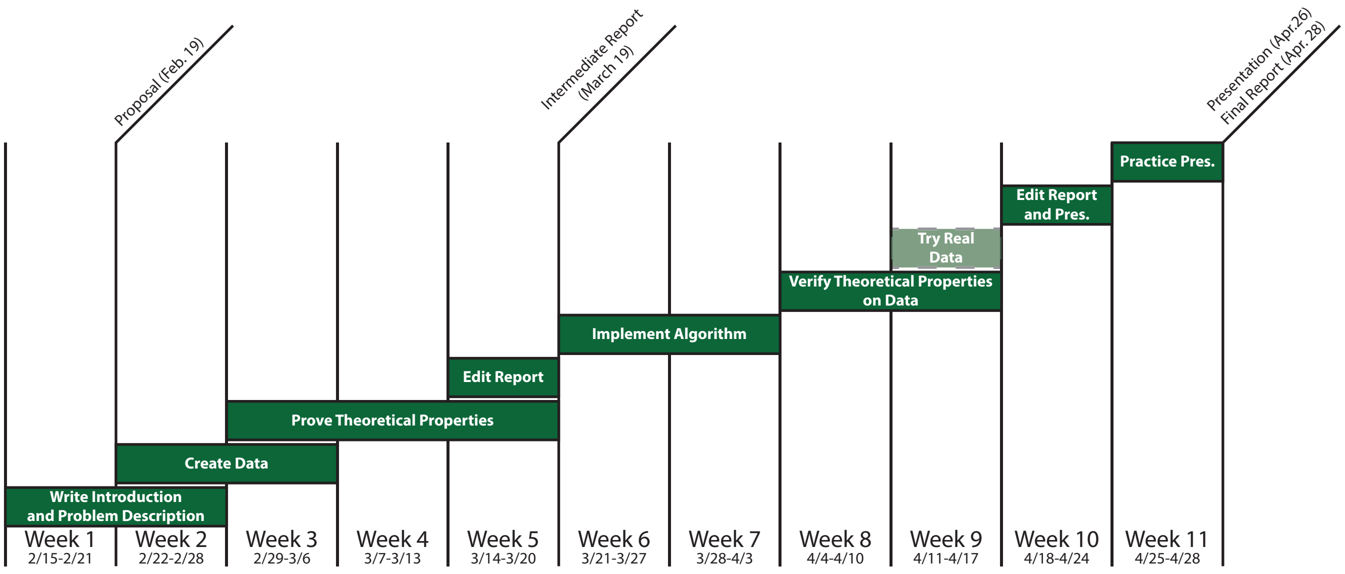


Figure 1: Project timeline showing key steps in the project. A solid color border indicates required step while a dotted line border indicates an optional step that will only be completed if there is time.

kinds of filtering: group/global filtering and local filtering. Global filtering works to determine if a point needs to be assigned to a different cluster based on the movement of the cluster centers. If centers change by a large amount, then it is more likely that points need their cluster assignment checked. Group filtering generalizes the global filtering by initially grouping the clusters into groups before the first iteration of k-means and applying the global filter to those groups. Local filtering is performed on any groups of cluster centers that make it through the group filter. Centers that get filtered by the local filter do not have their distances to the data points calculated.

Additionally, a method to optimize the second step of k-means, the center update step, is proposed. The new method updates the cluster centers by modifying them rather than calculating the average across all points contained in that cluster.

3. PRELIMINARY PLAN

The theory thrust of the project consists of 3 requirements and a 4th optional task:

- Show all details of the proof of correctness
- Implement the algorithm
- Test algorithm on synthetic dataset to verify theoretical properties
- *Test algorithm on a real-world dataset*

Figure 1 shows the overview of our project timeline. We have completed the introduction and problem description that will be included in the intermediate project report and the final project report. This document includes the a near-final draft of those sections. We will continue to refine the introduction and problem description as needed.

We have completed the code that generates our synthetic data

The next step is proving the theoretical properties of Yinyang K-means. This includes the Global-Filtering Condition and the Local-Filtering Condition. The Global-Filtering Condition asserts that cluster assignment of a point is only necessary if the cluster centroid changes drastically. The Local-Filtering Condition allows skipping distance calculations between a point and a centroid if that point is with some set distance to another centroid. We have made some progress on verifying these theoretical properties, but it is taking somewhat longer than expected. We will also investigate the authors' claim that Yinyang K-means achieves the same performance as classic K-means with an order of magnitude performance increase.

The last part of the project involves implementing the algorithm on a computer. Once it is implemented and verified to be correct, we can then test the algorithm on the synthetic data we generated earlier. The synthetic data will be created in a such a way that the theoretical properties of the algorithm will be tested. This will tell us that these properties actually hold in practice. If there is time, we also plan to try the algorithm on a real-world dataset which will hopefully see the same benefits as the synthetic data.

4. DATA

The main claim of the authors of the is the ability to apply the K-means clustering algorithm to data which is large. Thus, our synthetic dataset should attempt to verify this claim by including large amounts of data. It will also be interesting to document how the performance changes as the data size grows. The authors also claim that clustering results of Yinyang K-means is the same as standard K-means. So we vary the "difficulty" of clustering the data by varying the average euclidean distance of the mean of all of the clusters.

There are four primary parameters that we vary in our synthetic dataset. The first parameter is the number of clusters (N_C) where we use 3 different values: 100, 1,000, and 10,000. The second parameter is the number of features

Table 1: Data Parameters

Num. of Clusters	Num. of Features	Num. of Samples Per Cluster	Avg. Euclidean Distance of Means
100	100	100	100
1,000	500	500	500
10,000	1,000	1,000	1,000

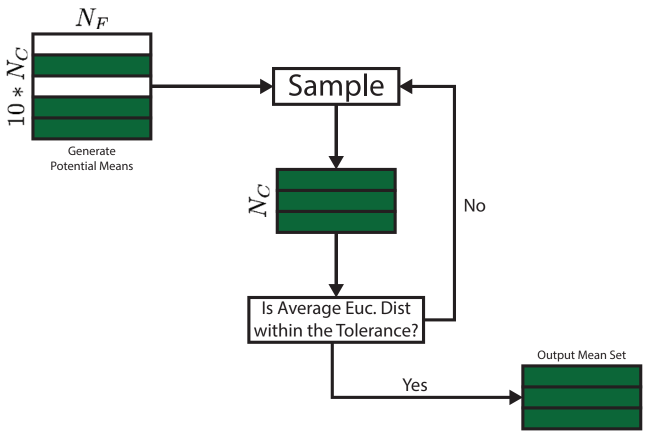
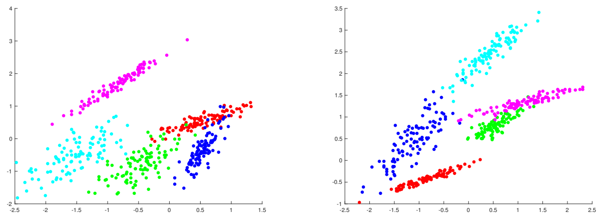


Figure 2: Overview of the mean generation step. A N_C-sized set of means is sampled from the $10 * N_C$-sized set of potential means. If the sample has an average euclidean distance within the tolerance it is selected. If not, a new sample is selected.



(a) Example Data 1 (b) Example Data 2

Figure 3: Examples of syntethic data generation. Each cluster is represented with different colors.

(N_F) where we use 3 different values: 100, 500, and 1,000. The third parameter is the number of samples per cluster (N_S) where we use 3 different values: 100, 500, and 1,000. The fourth and final parameter is the verage mean euclidean distance parameter (μ_{euc}) where we use 3 different values: 100, 500, and 1,000. There are 81 synthetic datasets that we will run both Yinyang K-means and standard K-means on. This information is summarized in Table 1.

We generate the data as multi-variate Gaussians. We first generate an N_C-sized set of means. An overview of this process is shown in Figure 2. We want to vary the average distance between clusters so we introduce an average mean euclidean distance parameter (μ_{euc}). We then use the `randn` MATLAB function to generate a set of potential means of size $10{*}N_C$ each with N_F features. We then sample from the set N_C means and calculate their average euclidean distance and compare it to μ_{euc}. If it is within the tolerance (δ), we use this as the set of N_C means for a single run.

We then generate a random matrix B and then generate a random positive semi-definite matrix $C = B^{\mathrm{T}}B$ that we use as the covariance. We do this for each mean in the run. Using the mean and the covariance we use the `mvnrnd` MATLAB to generate the data with N_S samples.

To demonstrate the data generation described above, we ran the above procedure with $N_C = 5$, $N_F = 2$, and $N_S = 100$. Since the data is two dimensional, we can visualize it. Two examples are shown in Figure 3.

5. REFERENCES

[1] D. Arthur and S. Vassilvitskii. K-means++: The advantages of careful seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '07, pages 1027–1035, Philadelphia, PA, USA, 2007. Society for Industrial and Applied Mathematics.

[2] L. Bottou and Y. Bengio. Convergence properties of the k-means algorithms. In *Advances in Neural Information Processing Systems 7*. Citeseer, 1995.

[3] A. Czumaj and C. Sohler. Sublinear-time approximation algorithms for clustering via random sampling. *Random Structures & Algorithms*, 30(1-2):226–256, 2007.

[4] Y. Ding, Y. Zhao, X. Shen, M. Musuvathi, and T. Mytkowicz. Yinyang k-means: A drop-in replacement of the classic k-means with consistent speedup. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, pages 579–587, 2015.

[5] J. Drake and G. Hamerly. Accelerated k-means with adaptive distance bounds. In *5th NIPS workshop on optimization for machine learning*, 2012.

[6] C. Elkan. Using the triangle inequality to accelerate k-means. In *ICML*, volume 3, pages 147–153, 2003.

[7] S. Guha, R. Rastogi, and K. Shim. Cure: an efficient clustering algorithm for large databases. In *ACM SIGMOD Record*, volume 27, pages 73–84. ACM, 1998.

[8] G. Hamerly. Making k-means even faster. In *SDM*, pages 130–140. SIAM, 2010.

[9] D. Pelleg and A. Moore. Accelerating exact k-means algorithms with geometric reasoning. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 277–281. ACM, 1999.

[10] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8. IEEE, 2007.

[11] D. Sculley. Web-scale k-means clustering. In *Proceedings of the 19th international conference on World wide web*, pages 1177–1178. ACM, 2010.

[12] J. Wang, J. Wang, Q. Ke, G. Zeng, and S. Li. Fast approximate k-means via cluster closures. In *Multimedia Data Mining and Analytics*, pages 373–395. Springer, 2015.