CSE 847 Project

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 $O(nkid)$, where n is the number of data points, k is the number of desired clusters, i is the number of iterations, and d is the number of dimensions. 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 [9, 6, 5] and on approximation [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 [4].

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 of 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* [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 1,000 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 [4].

2.1 Yinyang K-means

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 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. DATA

The main claim of the authors 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 (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 average 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 Gaussian distributions. We first generate an N_C-sized set of means. An overview of this process is shown in Figure 1. 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. We use this mean and the covariance as arguments for the **mvnrnd** MATLAB function to generate the data with N_S samples.

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

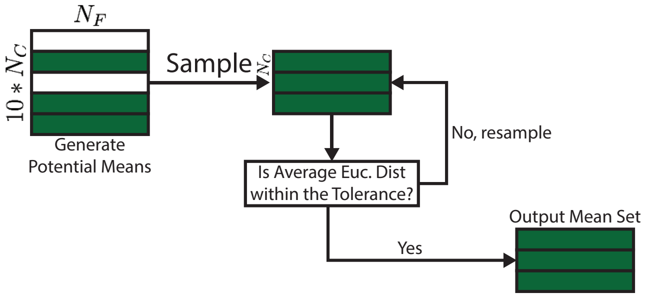
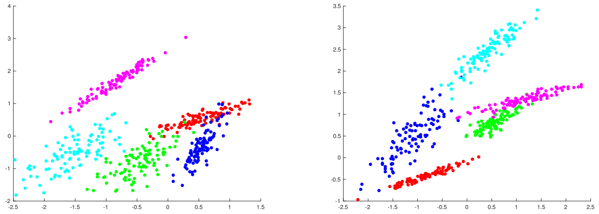


Figure 1: Overview of the mean generation step. A N_C-sized set of means is sampled from the $10 * N_C$-sized set of potential means. If the sample has an average Euclidean distance within the tolerance it is selected. If not, a new sample is selected.



(a) Example Data 1

(b) Example Data 2

Figure 2: Examples of generated synthetic data in two dimensions. Each cluster is represented with different colors.

4. YINYANG K-MEANS

4.1 Properties

4.1.1 *Definitions*

We define the parameters used in the Yinyang K-Means method here. Since Yinyang K-Means is an iterative method, we consider what happens between iterations i and $i{+}1$ until the convergence condition is met. Let $\mathcal{C}$ be the set of clusters centers at iteration i and $\mathcal{C}'$ be the set of clusters at iteration $i+1$. Let c be a single cluster in $\mathcal{C}$ at iteration i and c' be a single cluster in $\mathcal{C}'$ at iteration $i+1$. Let $b(x)$ be the cluster to which x is assigned at iteration i and $b'(x)$ be the cluster to which x is assigned at iteration $i+1$. Let $\delta(c) = d(c, c')$, that is, the distance a cluster center c travels between iteration i and iteration $i+1$. Let $ub(x) \geq d(x, b(x))$, that is, $ub(x)$ is an upper bound on the distance between x and its assigned cluster. Let $lb(x) \leq d(x, c)\ \forall\ c \in \{\mathcal{C} - b(x)\}$, that is, $lb(x)$ is a lower bound on the distance between x and the closest cluster to which it is not assigned.

4.1.2 *Triangle Inequality*

The Triangle Inequality is a well-known property used in many Computer Science fields. We document its definition and proof here for completeness.

Theorem: Let $d(a, b)$ represents the distance (in Euclidean space) between a point a and b. If x is data point and b &

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

c are cluster centers, then

$$|d(x,b) - d(b,c)| \leq d(x,c) \leq d(x,b) + d(b,c) \qquad (1)$$

Proof:

$$\begin{aligned} d(x,c) &= \|x - c\|_2 \\ &= \|x - b + b - c\|_2 \\ &\leq \|x - b\|_2 + \|b - c\|_2 \quad \text{Cauchy–Schwarz Inequality} \\ &= d(x,b) + d(b,c) \qquad (2) \end{aligned}$$

Suppose that the left-hand side of Equation 1 is not true, that is

$$d(x,c) < |d(x,b) - d(b,c)|$$

Without loss of generality, let $d(x,b) > d(b,c)$. Then,

$$d(x,b) - d(b,c) > d(x,c)$$

From Equation 2, we know

$$\begin{aligned} d(b,c) &\leq d(b,x) + d(x,c) \\ &= d(x,b) + d(x,c) \end{aligned}$$

Thus,

$$\begin{aligned} d(x,c) &< d(x,b) - d(b,c) \\ &\leq d(x,b) - (d(x,b) + d(x,c)) \\ &= -d(x,c) \end{aligned}$$

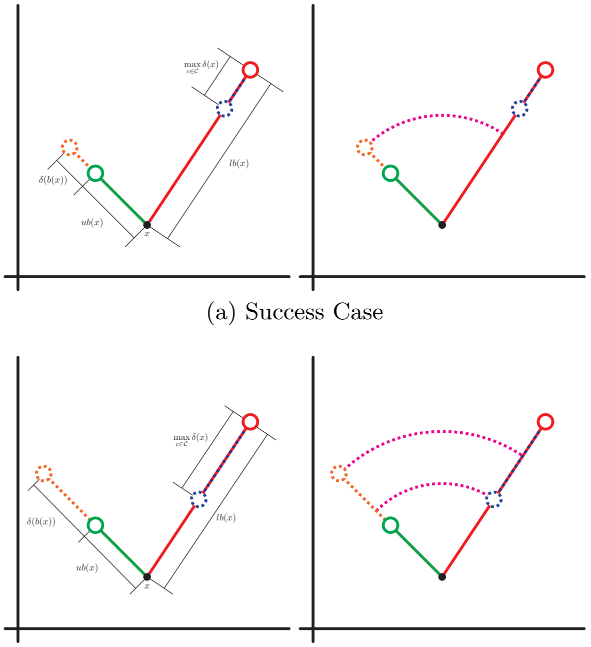
This is a contradiction. Thus, $d(x,c) \geq |d(x,b) - d(b,c)|$. □

4.1.3 Global-Filtering Condition

Theorem: A point x in the cluster defined by the cluster center $b(x)$ does not change its cluster after a center update if

$$lb(x) - \max_{c \in \mathcal{C}} \delta(c) \geq ub(x) + \delta(b(x)) \qquad (3)$$

Intuition: We know that $lb(x)$ is the distance between x and the closest cluster center to which x is not assigned (i.e. closest cluster center other than $b(x)$). The $\max_{c \in \mathcal{C}} \delta(c)$ term will yield the distance of the farthest-moving cluster from iteration i to iteration $i+1$. We know that $ub(x)$ is the distance between x and its assigned cluster $b(x)$. The $\delta(b(x))$ term will yield the distance x's assigned cluster at iteration i moves between iteration i and iteration $i+1$. The left-hand side of Equation 3, $lb(x) - \max_{c \in \mathcal{C}} \delta(c)$, indicates how close a cluster other than $b(x)$ could possibly be between iteration i and $i+1$. The right-hand side of Equation 3, $ub(x) + \delta(b(x))$, tell us the maximum possible distance between x and the center of $b(x)$ at iteration $i+1$. Thus, $b'(x) = b(x)$ if Equation 3 holds true, otherwise we need to compare the distances between x and all $c \in \mathcal{C}$ to find $b'(x)$. Figure 3 illustrates this concept.


$\max_{c \in \mathcal{C}} \delta(x)$
$lb(x)$
$\delta(b(x))$
$ub(x)$
x
$\max_{c \in \mathcal{C}} \delta(x)$
$lb(x)$
$\delta(b(x))$
$ub(x)$
x

(a) Success Case

(b) Failure Case

Figure 3: The Global-Filtering Condition tests whether we need to compare x to all the new centroid update or leave it assigned to the same cluster. If the test is successful (i.e. Equation 3 holds), then $b'(x) = b(x)$. If the test fails (i.e. Equation 3 does not hold), then we must calculate the distance between x and all $c \in \mathcal{C}$ to find $b'(x)$.

Proof: Let $b(x)$ be x's cluster in iteration i and b' is the same cluster in iteration $i+1$. Let c be any cluster other than $b(x)$ and c' is the same cluster in iteration $i+1$. From the triangle inequality, we have

$$d(x, c') \geq d(x,c) - d(c, c') = d(x,c) - \delta(c)$$

We also know that $\delta(c) \geq \max_{p \in \mathcal{C}} \delta(p)$. Thus,

$$d(x, c') \geq d(x,c) - \max_{p \in \mathcal{C}} \delta(p)$$

We use the fact that $lb(x) \leq d(x,c)$ to acheive the inequality

$$d(x, c') \geq lb(x) - \max_{p \in \mathcal{C}} \delta(p)$$

We also know that

$$d(x, b') \leq d(x, b(x)) - d(b(x), b') = d(x, b(x)) - \delta(b(x))$$

By definition, $d(x, b(x)) \leq ub(x)$. Thus,

$$d(x, b') \leq d(x, b(x)) - \delta(b(x)) \leq ub(x) - \delta(b(x))$$

Therefore, b' remains the closest cluster provided that

$$lb(x) - \max_{c \in \mathcal{C}} \delta(c) \geq ub(x) + \delta(b(x))$$
$$d\big(x, c'\big) \geq d\big(x, b'\big) \qquad \square$$

Potential Problem: The Global-Filtering Condition is sensitive to "*big-movers*". That is, if $\max_{c \in \mathcal{C}} \delta(c)$ is large, then Equation 3 will be rarely satisfied. Therefore, we will be forced to calculate the distance between x and *all* of the cluster centers. The speed-up gained by a reduction in the number of distance calculations will be lost. Classical K-means will yield better performance since the work is the same but Global-Filtering Condition will incur additional overhead in maintain the upper and lower bounds for each point.

Generalization to Group Filtering: To combat the "*big-movers*" problem, the authors propose a generalization of the Global-Filtering Condition. Instead of applying the Global-Filter to the initial k clusters, the data is split into t groups (this is based on the initial clustering, which is obtained with 5 iterations of classic K-means). The Global-Filtering condition will be applied to each group rather than each cluster.

The number of groups is an important question. The authors suggest $t = \frac{k}{10}$ which they found to give the best overall performance in their experiments. When $t = k$ (i.e. the number of clusters per group is 1), then Group Filtering reduces to Global Filtering. The parameter t makes the algorithm more elastic. That is, if t is large then there are fewer distance calculations but more overhead (to maintain lower bounds). If t is small, then there are more distance calculation but less overhead. Thus, the algorithm is adjustable to fit the needs of the application it will be deployed in.

The grouping is sensitive to a poor initialization. The authors claim that random groups will lead to poor performance. This is because the cluster center in this case are essentially meaningless in the beginning as there is no structure to the groupings. However, if we make an educated guess about the initial groups, then we stand to reduce the chance of a bad initialization. For this reason, the authors suggest grouping based on the initial cluster sets.

4.1.4 Local-Filtering Condition

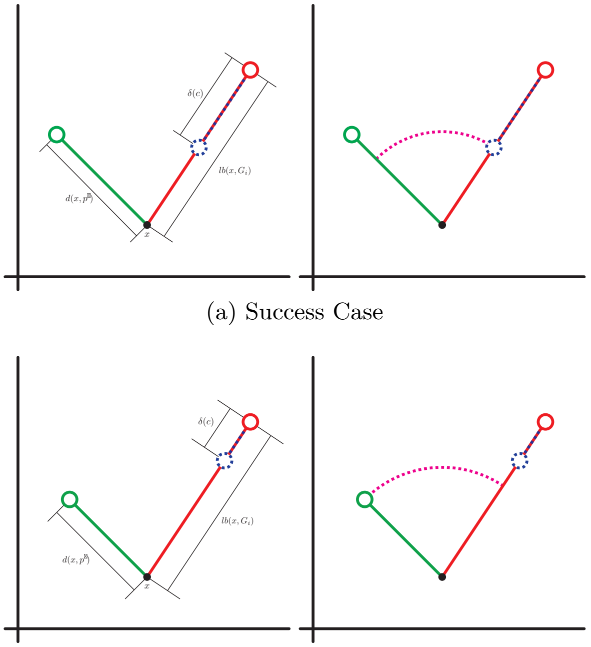
Theorem: A cluster center $c' \in G'_i$ cannot be the closest center to the point x if there is a center $p' \neq c'$ where $p' \in \big\{G'_i \cup (G'_i)^{\mathrm{C}}\big\}$ such that

$$d\big(x, p'\big) < lb(x, G_i) - \delta(c) \qquad (4)$$

Intuition: We know that $lb(x, G_i)$ represents a lower bound on the distance between x and the closest cluster center in G_i to which x is not assigned. The $\delta(c)$ term is the distance the cluster c moves between iteration i and iteration i and $i+1$. Thus, the right-hand side of Equation 4 represents the smallest possible distance between x and c'. Therefore, if there c' cannot be the closest cluster if there is a point with a distance smaller than the right-hand side of Equation 4. Figure 4 illustrates this concept.

Proof: From the triangle inequality, we have

$$d\big(x, c'\big) \geq d(x, c) - d\big(c, c'\big)$$



(a) Success Case

(b) Failure Case

Figure 4: The Local-Filtering Condition tests whether c' has the potential to be $b'(x)$. If the test is successful (i.e. Equation 4 holds), then c' has the potential to be $b'(x)$. If the test fails (i.e. Equation 4 does not hold), then c' cannot be $b'(x)$.

This theorem stems from the fact that $lb(x, G_i) - \delta(c)$ represents the smallest possible distance between x and c'. By definition, we know $d(x, c) \geq lb(x, G_i)$ and $d(c, c') = \delta(c)$. Thus,

$$d\big(x, c'\big) \geq d(x, c) - d\big(c, c'\big) \geq lb(x, G_i) + \delta(c)$$

Therefore c' cannot be the closest center, if there exists a $p' \neq c'$ such that

$$d\big(x, p'\big) < lb(x, G_i) - \delta(c) \qquad \square$$

4.2 Algorithm

5. IMPLEMENTATION

The authors generously provide their code free to use. It is implemented using the C++ library GraphLab. We attempted to build this tool and run the code on our lab's computing server. However, we were unable to successfully compile the code after several days of attempting.

In order to avoid any possible under-the-hood optimizations as well as to time the execution, we opted not to use Matlab's built-in K-means. Instead, we wrote our own version which allowed us to seed the initial centers as well as record the time per iteration and the number of iterations required to converge.

Due to these difficulties, we chose to attempt to implement the algorithm ourselves in Matlab. This was an enlightening process for us. We found that the algorithm as described is not precise enough for easy implementation; there were even some errors in the paper's description. We consulted one of the authors for assistance and she was kind enough to answer several questions we had while working on our implementation.

Despite this assistance, we were unable to successfully implement the algorithm. Our code runs and does cluster, but it does not achieve the exact same cluster assignments as traditional K-means, and it is also slower, both in number of iterations required to converge and time per iteration.

6. CONTRIBUTIONS

7. 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.