The BanditPAM Algorithm: Getting to Production

Thomas Cheung
Whole Foods Market
cccheung@stanford.edu

Ben Hora
Department of Computer Science
Stanford University
bhora@stanford.edu

Mo Tiwari
Department of Computer Science
Stanford University
motiwari@stanford.edu

Abstract

The k-medoids approach to unsupervised clustering tasks defines k different medoids as the points in a dataset whose dissimilarities with all other points associated with each medoid are at a minimum. Modern theoretical advances in unsupervised machine learning, specifically in regards to the k-medoids approach, have pushed the state-of-the-art implementation of this algorithm forward. Until recently, the state-of-the-art approach to k-medoids - a greedy search algorithm named Partitioning Around Medoids (PAM) - was quadratic in time complexity where, for each iteration over n datapoints, updating all k medoids took $O(kn^2)$ time. A new randomized k-medoids algorithm named BanditPAM inspired by techniques from multi-armed bandits has now been able to reduce the complexity of each iteration from $O(kn^2)$ to $O(kn \log n)$ while still returning the same results as the previous state-of-the-art PAM approach. We work with the author of the original BanditPAM paper to take the theoretical findings presented and implement them in an open-source C++ repository available for other developers to access.

1 Introduction

Unsupervised machine learning algorithms remain extremely important in modern data science applications. The k-means algorithm is one of the most commonly used approaches in industry and academia for tasks that require the clustering of large datasets [18]. The goal of k-means is rather straightforward: group similar data points together and discover underlying patterns [5, 27].

In an attempt to achieve this objective, k-means starts by randomly choosing k centroids which are used as the starting points for every cluster. Then, in an iterative approach, a k-means algorithm will compute the least squared Euclidean distance between each data point and the current centroids, assigning a data point to its closest centroid in the process. The algorithm will subsequently recompute the centroids for the clusters by moving the current centroid to the location of the mean of the all data points that belong to it. This process of assigining every data point to a cluster and then updating the centroid based on the mean of its newly assigned points is repeated until convergence (i.e. the assignments of points to centroids no longer change).

K-medoids is another unsupervised machine learning algorithm that can be employed in similar situations to those for which k-means is often used (i.e. tasks that require the clustering of large datasets) [11]. Both of these algorithms work to partition a dataset into k clusters and attempt to minimize a distance function between points labeled to be in a cluster. In contrast to the k-means algorithm however, k-medoids chooses actual data points from the provided dataset as centers, thereby implicitly picking an exemplar data point for each medoid produced [10]. Furthermore, k-medoids

can be used with arbitrary distance functions whereas k-means requires the use of the Euclidean distance function, thereby limiting the use cases for k-means to problems which can be defined using the Euclidean distance metric [4, 21].

We believe that the k-medoid algorithm provides some key advantages over k-means, namely its ability to provide an exemplar data value for each cluster and its ability to minimize the sum of dissimilarities between points labeled in a cluster through methods other than solely Euclidean distance. The main inhibitor to wider adoption of k-medoids, however, is the time complexity of this algorithm. Whereas k-means can run in $O(kn)$ time [9] where n is the number of datapoints and k is the number of clusters, the previous k-medoids state-of-the-art implementation - called the PAM algorithm - has time commplexity of $O(kn^2)$ [19].

Thus, when n is large, which is the case for many modern applications of clustering algorithms that attempt to identify patterns in massive datasets, the k-medoids algorithm (implemented with PAM) performs significantly slower iterations than the state-of-the-art k-means implementation. This substantial difference in runtime complexity is a main impetus for large-scale use of k-means instead of k-medoids in many data clustering applications despite the numerous benefits that the k-medoids approach provides [1, 26, 25].

Our implementation of the BanitPAM algorithm aims to make the k-medoids approach to clustering more accessable and advantageous for developers working on a vast variety of unsupervised learning tasks. The BanditPAM algorithm has decreased the runtime of k-medoids from $O(kn^2)$ to $O(kn\log(n))$, making it a viable alternative to k-means even in cases when the size of the data set is large. We believe that due to the inherent benefits of k-medoids in comparision to k-means as well as the vast improvement in runtime efficiency of k-medoids due to the implementation of a multi-armed bandits approach, BanditPAM has the potential to become an industry standard for unsupervised clustering tasks.

2 Related Work

The k-medoids approach to clustering has existed for some time. The notion of searching for k representative objects that are actual samples from the dataset itself was first theorized by Kaufman and Rousseeuw [12] as they searched for an alternative to the commonly used "sum of squares" similarity function used in k-means. Kaufman and Rousseeuw showed that k-medoids could instead use the L_1 distance function to group objects with high degrees of similarity.

Shortly after the initial publication of the k-medoids approach to clustering, researchers Jain and Dubes were able to show that solving the central goal of k-medoids - finding a set of k medoids from n datapoints that minimize the overall distance of points to their closest medoids - is actually an NP-hard problem [8]. However, over time, various heuristics have been developed that still produce meaningful clustering results using a k-medoids approach. The most common of these heuristic algorithm is the Partitioning Around Medoids (PAM) approach which was also first proposed by Kaufman and Rousseeuw [11, 10].

Other algorithmic approaches to k-medoids include Clustering Large Applications (CLARA) [17, 29], which samples a subset of data, applies PAM to choose medoids in the subset, repeats this process multiple times, and finally approximates the medoids for the entire set based on these results. FastPAM, another k-medoids algorithm implementation [23], also embodies a similar structure to that of PAM while being able to improve runtime complexity by $O(k)$ time (this factor improvement however becomes less significant as $k \ll n$ in most modern applications).

But not all k-medoids implementations make use of the PAM approach. The Clustering Large Applications based upon Randomized Search (CLARANS) algorithm [20, 22], for instance, chooses an entirely different approach to the process of selecting medoids. CLARANS defines the medoid choosing process as akin to searching through a graph, where each edge corresponds to swapping a medoid and non-medoid pair during an iterative medoid selection process.

In addition to new algorithmic approaches to k-medoids, researchers have published findings regarding how to better explain these algorithms [23], better scale them to larger data sets [15, 30], and better understand the relationships among them [24]. Our paper builds on these research endeavors by taking a new approach to k-medoids that employs multi-armed bandits to reduce algorithmic time complexity from $O(kn^2)$ to $O(kn\log n)$ while still maintaining the same level of clustering loss.

3 The PAM Algorithm

Given a set of n data points $\mathcal{X} = \{x_1, \ldots, x_n\}$ and a user-defined distance function $d(\cdot, \cdot)$, the k-medoids algorithm aims at finding k points from the data set called medoids, $\mathcal{M} = \{m_1, \ldots, m_n\} \subset \mathcal{X}$, to minimize the overall distance of all points in the data set from their closest medoids.

$$\mathcal{L}(\mathcal{M}) = \sum_{i=1}^{n} \min_{m \in \mathcal{M}} d(m, x_i) \tag{1}$$

The PAM algorithm [11, 12] is the current state-of-the-art approach for finding the medoids. It consists of two steps: first it processes the BUILD step to initialize a set of k medoids from the data set. Then it repeatedly perform the SWAP step to reduce the loss $\mathcal{L}$ until convergence.

BUILD: Given a set of n points, PAM initializes the k medoids by greedily assigning points as medoids through minimizing the loss $\mathcal{L}$. Given the current set of l medoids $\mathcal{M}_l$, the next point m^* to add to the medoids set is

$$\text{BUILD:} \quad m^* = \operatorname*{arg\,min}_{x \in \mathcal{X} \setminus \mathcal{M}_l} \frac{1}{n} \sum_{j=1}^{n} \left[d(x, x_j) \wedge \min_{m' \in \mathcal{M}_l} d(m', x_j) \right] \tag{2}$$

SWAP: PAM then considers all the $k(n-k)$ medoid-nonmedoid pairs and choose the pair which would reduce the loss $\mathcal{L}$ most by swapping the roles of medoid and nonmedoid. The SWAP step is then repeated until convergence. Let M be the current set of k medoids, then the best medoid-nonmedoid (m^*, x^*) to swap is

$$\text{SWAP:} \quad (m^*, x^*) = \operatorname*{arg\,min}_{(m,x) \in \mathcal{M} \times (\mathcal{X} \setminus \mathcal{M})} \frac{1}{n} \sum_{j=1}^{n} \left[d(x, x_j) \wedge \min_{m' \in \mathcal{M} \setminus \{m\}} d(m', x_j) \right] \tag{3}$$

Note that the second terms in (2), namely $\min_{m' \in \mathcal{M}_l} d(m', x_j)$ can be determined by caching the smallest distance between each point to the previous set of medoids. Hence, the term only needs to be computed once and the complexity of the BUILD step is $O(kn^2)$. Similarly, the second term of (3), namely $\min_{m' \in \mathcal{M} \setminus \{m\}} d(m', x_j)$ can be determined by caching the smallest and the second smallest distances from each point to the previous set of medoids. Thus, the complexity of the SWAP step is also $O(kn^2)$.

4 The BanditPAM Algorithm

Note that the BUILD step and the SWAP step in PAM algorithm can be written in a similar structure:

$$\text{Shared Problem:} \quad \operatorname*{arg\,min}_{x \in S_{\text{tar}}} \frac{1}{|S_{\text{ref}}|} \sum_{x_j \in S_{\text{ref}}} g_x(x_j) \tag{4}$$

where S_{tar} is the set of target points, S_{ref} is the set of reference points and $g_x(\cdot)$ is the objective function that depends on the target point x. In particular, we can rewrite the BUILD and SWAP steps in terms of Problem (4) as follows:

$$\text{BUILD: } \mathcal{S}_{\text{tar}} = \mathcal{X} \setminus \mathcal{M}_l,\ \mathcal{S}_{\text{ref}} = \mathcal{X},\ g_x(x_j) = \left(d(x, x_j) - \min_{m' \in \mathcal{M}_l} d(m', x_j) \right) \wedge 0, \tag{5}$$

$$\text{SWAP: } \mathcal{S}_{\text{tar}} = \mathcal{M} \times (\mathcal{X} \setminus \mathcal{M}),\ \mathcal{S}_{\text{ref}} = \mathcal{X},\ g_x(x_j) = \left(d(x, x_j) - \min_{m' \in \mathcal{M} \setminus \{m\}} d(m', x_j) \right) \wedge 0. \tag{6}$$

where each medoid-nonmedoid pair in the SWAP step is regarded as a target point.

The shared problem (4) can be viewed as a best-arm identification problem from the multi-armed bandits (MAB) literature [2, 7]. In a typical MAB porblem, there are m arms and at each time $t = 0, 1, \ldots$, we pull an arm and receive a reward. The goal is to identify the best arm which has the largest expected reward with high probability. In the shared problem (4), we view each $x \in S_{\text{tar}}$ as an arm in best-arm identification problem, and pulling an arm corresponds to computing the loss $g_x(x_j)$ on a

random data point x_j. Specifically, we consider the objective function $\mu_x = \frac{1}{|S_{\text{ref}}|} \sum_{x_j \in S_{\text{ref}}} g_x(x_j)$ and estimate μ_x by drawing n' independent samples $S_1, \ldots, S_{n'}$ each of size B uniformly with replacement from S_{ref} so that μ_x can be estimated by $\hat{\mu_x} = \frac{1}{n'} \sum_{i=1}^{n'} g(x_{S_j})$. The best arm can then be estimated by the Upper Confidence Bound (UCB) algorithm [13] with successive elimination [7]. The complexity of the whole BanditPAM algorithm is $O(kn \log n)$.

5 Data

In order to train and test the BanditPAM algorithm, two different approaches will be taken. The first will include using a Gaussian Mixture Model (GMM) for synthetic data generation. The idea of using a GMM for adding additional samples to a dataset has been studied and implemented before. In a 2005 paper, researcher and data scientist Bashar Awwad adapted the sequential EM algorithm in a GMM to generate synthetic data for an application in adaption failure detection [3]. More recently, a 2019 study by Qun Liu used a GMM to augment a small dataset of route choices of travelers traversing an urban area [16].

The process of using a GMM for generating new data is accomplished by first defining a Gaussian mixture that is comprised of k Gaussians where $k \in \{1, ..., K\}$ and K is the number of clusters that we will have for our dataset. Each Gaussian is parameterized by a mean μ that defines its cluster center, a covariance Σ that defines its width, and a mixing probability ϕ that defines the size of the Gaussian function. The mixing coefficients (i.e. ϕ_k) are themselves probabilities that must satisfy $\sum_{k=1}^{K} \phi_k = 1$. In order to use this type of Gaussian Mixture Model for synthetic data generation for the BanditPAM algorithm, we follow a two step algorithmic process that will repeat K times to generate our first K datapoints. Step 1 consists of picking a value k from categorical distribution parametrized by vector ϕ. Step 2 consists of drawing a single value from the k-th Gaussian distribution parametrized by μ_k and Σ_k. This two step process can subsequently be repeated arbitrarily many times until enough datapoints have been created to train and test the BanditPAM algorithm.

The second approach that will be taken for training consists of using the MNIST dataset and visualizing the results via calculating t-SNE embeddings. The MNIST dataset [14] consists of 70,000 black-and-white images of handwritten digits, where each digit is represented as a 784-dimensional vector. For the purposes of implementing BanditPAM, a distance function must be chosen which will be minimized during the medoid SWAP step. In order to allow for direct comparison to the k-means algorithm, which requires the use of Euclidean distance, we chose to also use the Euclidean distance metric during BanditPAM training and testing.

In order to visualize the results of BanditPAM on a high dimensional dataset like MNIST while still preserving as much significant structure and information present in the data as possible, we calculate t-Distributed stochastic neighbour embeddings (t-SNE). This specific type of embedding, originally created by Laurens van der Maaten and Geoffrey Hinton [28], calculates a similarity measure between datapoints in both high and low dimensional space before trying to optimize these measures in an iterative process. For our specific case, we use t-SNE embeddings on the MNIST dataset in order to project the original 784-dimensional data into much more interpretable 2-D data.

6 Empirical Results

The main goal of this project is to productionize the BanditPAM software. Throughout the length of the project, we worked in tandem with the first author of the original BanditPAM theoretical algorithm to build, implement, and refine the C++ source code of BanditPAM.

The central core component of BanditPAM's implementation is the use of multi-armed bandits, which occurs when we sample the reference points in the dataset, calculating both the empirical arm means as well as a corresponding confidence interval that we are 99 percent confident that the arm mean falls within. We then use this confidence interval to get a sense of how accurate our sample mean is (from the reference points we've computed the change in loss to) to the true mean (if we were to calculate this for all reference points). So each time we calculate the distance to some more reference points, we updated the running mean and the confidence interval. Then, with approximately 99 percent confidence, we can make the assertion that the true arm parameter (mean) lies within a

range (lcb, ucb) where lcb = sample mean - confidence interval length/2 and ucb = sample mean + confidence interval length / 2.

In addition to the development of the main BanditPAM algorithm, many additional features were added to the BanditPAM repo in order to ensure that other developers and researchers could easily access and understand the codebase. For example, we have verified that BanditPAM source code accepts either space-delimited or comma-delimited by testing with two files outlining matrices, one using space to separate values and the other using commas. We have also added the log file's name as a parameter to the executable so that the user can specify it. We have modified the source code to throw proper error messages whenever the input arguments are invalid, e.g. when the mandatory input like file path or number of medoids are missing. Moreover, to make sure the customized loss function is well-defined, we update the code to validate the name of the loss function and throw an error if it is not recognized.

We demonstrate the scaling of BanditPAM to be linear in n using a synthetic GMM dataset with $k = 3$ clusters with each cluster consists of 300 points. Random samples of size $100, 200, 400, 600$ are taken from the GMM dataset and the runtimes per iteration of BanditPAM clustering are recorded. To compute the runtime per iteration, we divide the total wall clock time by the number of SWAP iterations plus 1. Figure 1 shows the runtime per iteration versus different sample size n on a log-log plot. The slope of the best fit line is 1.03 which indicates the scaling of BanditPAM is linear in n. The reference lines demonstrate the expected scaling of PAM and FastPAM1. We can see that both have a steeper slope than BanditPAM as their scalings are nonlinear in n.

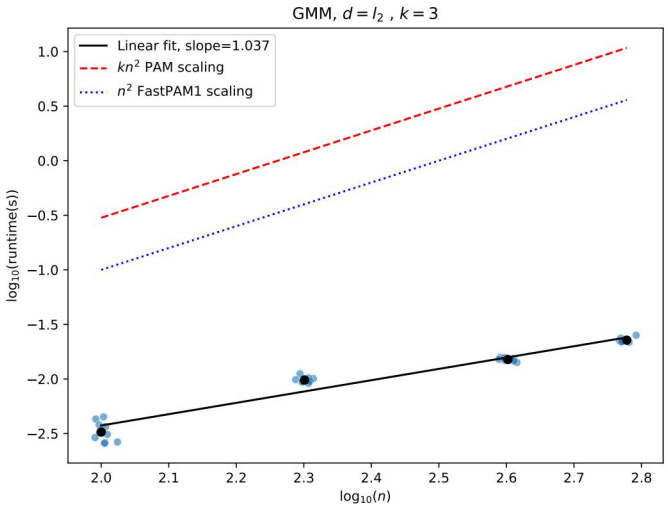

Figure 1: Average runtime per iteration vs. sample size n for GMM with l_2 distance and $k = 3$ on a log-log scale. The black line is the best fit line with slope = 1.03. The red and blue lines are reference lines demonstrating the expected scaling of PAM and FastPAM1 respectively.

7 Future Work

Work implementing and productionizing the BanditPAM still has many places to improve and continue growing. For the next step, we are going to allow users to input custom distance metrics into our open-source BanditPAM project. This will hugely increase the flexibility of the cluster algorithm and provide freedom for users to use their metrics suitable for their problems. We will test the algorithm on classifying similar solutions of online coding problems from Code.org [6]. The HOC4 dataset from Code.org consists of 3,360 unique solutions to a block-based programming exercise. The solutions are represented as abstract syntax trees (ASTs). Our goal is to allow for custom defined metrics like tree edit distance [31] to be used by our BanditPAM implementation to quantify similarity among points in a provided dataset.

8 Contributions

Thomas: Thomas has worked on Sections 3, 4, 6, and 7 of the proposal. This included summarizing key mathematical concepts related to PAM and BanditPAM, writing a discussion of multi-armed bandits/its implementation, outlining the empirical results of our work, sharing where we will be taking the project from this point forward.

Ben: Ben has worked on Sections 1, 2, 5, and 6 of the proposal. This included summarizing key algorithms/papers relevant to the project, providing context to the relevancy of our work to the field of unsupervised machine learning, sharing information regarding the datasets generated and used, and describing some key features of BanditPAM's open-source implementation.

9 Code

Link to GitHub Repository: https://github.com/ThrunGroup/BanditPAM

References

[1] Preeti Arora, Deepali, and Shipra Varshney. Analysis of k-means and k-medoids algorithm for big data. *Procedia Computer Science*, 78:507–512, 12 2016.

[2] Jean-Yves Audibert, Sébastien Bubeck, and Rémi Munos. Best arm identification in multi-armed bandits. In *International Conference on Learning Theory*, pages 41–53, 2010.

[3] Bashar Awwad Shiekh Hasan and John Gan. Sequential em for unsupervised adaptive gaussian mixture model based classifier. volume 5632, pages 96–106, 07 2009.

[4] Paul S. Bradley, Olvi L. Mangasarian, and W. N. Street. Clustering via concave minimization. In *Advances in Neural Information Processing Systems*, pages 368–374, 1997.

[5] Peter Bryant. On characterizing optimization-based clustering methods. In *Journal of Classification*, volume 5, pages 81–84, 1988.

[6] Code.org. Research at code.org. In *https://code.org/research*, 2013.

[7] Eyal Even-Dar, Shie Mannor, and Yishay Mansour. Pac bounds for multi-armed bandit and markov decision processes. In *International Conference on Computational Learning Theory*, pages 255–270, 2002.

[8] Anil K. Jain and Richard C. Dubes. *Algorithms for clustering data*. Prentice-Hall, 1988.

[9] T. Kanungo, D.M. Mount, N.S. Netanyahu, C.D. Piatko, R. Silverman, and A.Y. Wu. An efficient k-means clustering algorithm: analysis and implementation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):881–892, 2002.

[10] Leonard Kaufman and Peter J. Rousseeuw. *Finding groups in data: An introduction to cluster analysis*. Wiley New York.

[11] Leonard Kaufman and Peter J. Rousseeuw. Clustering by means of medoids. *Statistical Data Analysis based on the L1 Norm and Related Methods*, pages 405–416, 1987.

[12] Leonard Kaufman and Peter J. Rousseeuw. Partitioning around medoids (program pam). *Finding groups in data: an introduction to cluster analysis*, pages 68–125, 1990.

[13] Tze Leung Lai and Herbert Robbins. Asymptotically efficient adaptive allocation rules. *Advances in applied mathematics*, 6(1):4–22, 1985.

[14] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010.

[15] Jefrey Lijffijt, Panagiotis Papapetrou, and Kai Puolamäki. Size matters: choosing the most informative set of window lengths for mining patterns in event sequences. *Data Mining and Knowledge Discovery*, 29, 11 2015.

[16] Qun Liu, Supratik Mukhopadhyay, Yimin Zhu, Ravindra Gudishala, Sanaz Saeidi, and Alimire Nabijiang. Improving route choice models by incorporating contextual factors via knowledge distillation. 03 2019.

[17] C. Lucasius, A. Dane, and G. Kateman. On k-medoid clustering of large data sets with the aid of a genetic algorithm: background, feasiblity and comparison. *Analytica Chimica Acta*, 282:647–669, 1993.

[18] James MacQueen. Some methods for classification and analysis of multivariate observations. In *Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297, 1967.

[19] Hassan Mushtaq, Sajid Gul Khawaja, Muhammad Usman Akram, Amanullah Yasin, Muhammad Muzammal, Shehzad Khalid, and Shoab Ahmad Khan. A parallel architecture for the partitioning around medoids (pam) algorithm for scalable multi-core processor implementation with applications in healthcare. *Sensors*, 18(12):4129, 2018.

[20] Raymond T. Ng and Jiawei Han. Clarans: A method for clustering objects for spatial data mining. *IEEE transactions on knowledge and data engineering*, 14(5):1003–1016, 2002.

[21] Michael L Overton. A quadratically convergent method for minimizing a sum of euclidean norms. *Mathematical Programming*, 27(1):34–63, 1983.

[22] Anuradha Rani, Sushila Ratre, and Pranav More. Performance analysis of k-means and clarans clustering algorithm. 06 2018.

[23] Erich Schubert and Peter J Rousseeuw. Faster k-medoids clustering: improving the pam, clara, and clarans algorithms. In *International Conference on Similarity Search and Applications*, pages 171–187. Springer, 2019.

[24] Erich Schubert and Arthur Zimek. Elki: A large open-source library for data analysis-elki release 0.7.5 "heidelberg". *arXiv:1902.03616*, 2019.

[25] Saurabh Shah and Manmohan Singh. Comparison of a time efficient modified k-mean algorithm with k-mean and k-medoid algorithm. 05 2012.

[26] Kalpit Soni and Atul Patel. Comparative analysis of k-means and k-medoids algorithm on iris data. 13:899–906, 01 2017.

[27] Douglas Steinley. K-means clustering: a half-century synthesis. In *Br J Math Stat Psychol*, volume 56, pages 1–34, 2006.

[28] Laurens van der Maaten and Geoffrey Hinton. Viualizing data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605, 11 2008.

[29] Chih-Ping Wei, Yen-Hsien Lee, and Che-Ming Hsu. Empirical comparison of fast clustering algorithms for large data sets. pages 10 pp.–, 02 2000.

[30] Xianfeng Yang and Liming Lian. A new data mining algorithm based on mapreduce and hadoop. *International Journal of Signal Processing, Image Processing and Pattern Recognition*, 7:131–142, 04 2014.

[31] Kaizhong Zhang and Dennis Shasha. Simple fast algorithms for the editing distance between trees and related problems. *SIAM Journal of Computing*, 18(6):1245–1262, 1989.