Bandit-PAM: Almost Linear Time k-Medoids Clustering via Multi-Armed Bandits

Anonymous Author(s)
Affiliation
Address
email

Abstract

Clustering is a ubiquitous task in data science. Compared to the commonly used k-means clustering algorithm, k-medoids clustering algorithms require the cluster centers to be actual data points and support arbitrary distance metrics, allowing for greater interpretability and the clustering of structured objects. Current state-of-the-art k-medoids clustering algorithms, such as Partitioning Around Medoids (PAM), are iterative and are quadratic in the dataset size, n, for each iteration, prohibitively expensive for large datasets. We propose Bandit-PAM, a randomized algorithm inspired by techniques from multi-armed bandits, that significantly improves the computational efficiency of PAM. We theoretically prove that Bandit-PAM reduces the complexity of each PAM iteration from $O(n^2)$ to $O(n \log n)$ and returns the same results with high probability, under assumptions on the data that often hold in practice. We empirically validate our results on several large-scale real-world datasets, including the a coding exercise submissions dataset from Code.org, the 10x Genomics 68k PBMC single-cell RNA sequencing dataset, and the MNIST handwritten digits dataset. We observe that Bandit-PAM provides returns the same results as PAM while performing up to 200x fewer distance computations. The improvements demonstrated by Bandit-PAM enable k-medoids clustering on a wide range of applications, including identifying cell types in large-scale single-cell data or providing scalable feedback for students learning computer science.

1 Introduction

Many modern data science applications require the clustering of very-large-scale data. Due to its computational efficiency, the k-means clustering algorithm [27, 25] has been one of the most widely-used clustering algorithms. k-means alternates between assigning points to their nearest cluster centers and recomputing those centers. Central to its success is the specific choice of the cluster center: for a set of points, k-means defines the cluster center as the point with the smallest average *squared Euclidean distance* to all other points in the set. Under such a definition, the cluster center is the arithmetic mean of the cluster's points and can be computed efficiently.

While commonly used in practice, k-means clustering suffers from several drawbacks. First, while one can efficiently compute the cluster centers under squared Euclidean distance, it is not straightforward to generalize to other distance metrics [32, 12, 5]. However, a different distance may be desirable in different applications. For example, l_1 and cosine distance are often used in sparse data, such as in recommendation systems [23] and single-cell RNA-seq analysis [31]; additional examples include string edit distance in text data [29], and graph metrics in social network data [28]. Second, the cluster center in k-means clustering is in general not a point in the dataset and may not be interpretable in many applications. This is especially problematic when the data is structured, such as parse trees in

Submitted to 34th Conference on Neural Information Processing Systems (NeurIPS 2020). Do not distribute.

context-free grammars, sparse data in recommendation systems [23], or images in computer vision where the mean image is visually random noise [23].

Alternatively, k-medoids clustering algorithms [16, 17] use the *medoid* to define the cluster center for a set of point, where for an arbitrary distance function, the medoid is the point *in the set* that minimizes the average distance to all the other points. Note that the distance metric can be arbitrary—indeed, it need not be a distance metric at all and could be an asymmetric dissimilarity measure—which addresses the first shortcoming of k-means outlined above. Moreover, unlike k-means, the cluster centers in k-medoids, i.e. the medoids, are restricted to be points in the dataset, thus addressing the second shortcoming of k-means clustering described above.

Despite its advantages, k-medoids clustering is less popular than k-means due to its computational efficiency. Indeed, state-of-art k-medoids clustering algorithms are iterative and are quadratic in the data size, whereas k-means is linear in dataset size in each iteration.

Mathematically, for n data points $\mathcal{X} = \{x_1, x_2, \cdots, x_n\}$ and a user-specified distance function $d(\cdot, \cdot)$, the k-medoids problem is to find a set of k medoids $\mathcal{M} = \{m_1, \cdots, m_k\} \subset \mathcal{X}$ to minimize the overall distance of points from their closest medoids:

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

This problem is, unfortunately, NP-hard in general [35]. Partitioning Around Medoids (PAM) [16, 17] is one of the most widely used heuristic algorithms for k-medoids clustering. PAM is split into two subroutines: BUILD and SWAP. First, in the BUILD step, PAM aims to find an initial set of k medoids by greedily and iteratively selecting points that minimize the k-medoids clustering loss as described in (1). Next, in the SWAP step, PAM considers all $k(n-k)$ possible pairs of medoid and non-medoid points and swaps the pair that reduces the loss the most. The SWAP step is repeated until no further improvements can be made.

PAM has been empirically shown to produce better results than other popular k-medoids clustering algorithms [34, 35]. However, the BUILD step and each of the SWAP steps require $O(kn^2)$ distance evaluations and can be prohibitively expensive to run, especially for large datasets or when the distance evaluations are themselves expensive (e.g. edit distance between two long strings).

Randomized algorithms like CLARA [17] and CLARANS [30] have been proposed to improve computational efficiency, but at the cost of deteriorated clustering quality. More recently, Schubert et al. [35] proposed a deterministic algorithm, dubbed FastPAM1, that guarantees the same output as PAM but improves the complexity to $O(n^2)$ when the cluster sizes are similar. However, the factor $O(k)$ improvement becomes less relevant when the sample size n is large and the number of medoids k is relatively small compared to n. Throughout the rest of this work, we treat k fixed.

Contributions: In this work, we propose a novel randomized k-medoids algorithm, called Bandit-PAM, that significantly improves the computational efficiency of PAM while returning the same result with high probability. We theoretically prove that Bandit-PAM reduces the complexity on the sample size n from $O(n^2)$ to $O(n \log n)$, both for the BUILD step and each SWAP step, under reasonable assumptions that hold in many real-world datasets. We empirically validate our results on several large-scale real-world datasets and observe that Bandit-PAM provides up to 200x reduction of distance computations while returning the same result as PAM. We also release a high-performance C++ of implementation Bandit-PAM, which brings a 3.2x wall-clock-time speedup over the state-of-the-art FastPAM implementation [36] on the full MNIST dataset without precomputing and caching the the n^2 pairwise distance matrix as in previous approaches.

Intuitively, Bandit-PAM works by recasting each step of PAM from a *deterministic computational problem* to a *statistical estimation problem*. In the BUILD step assignment of the lth medoid, for example, we need to choose the point amongst all $n - l$ non-medoids that will lead to the lowest overall loss (1) if chosen as the next medoid. Mathematically, we wish to find x that minimizes:

$$L(x; \mathcal{M}) = \sum_{j=1}^{n} \min_{m \in \mathcal{M} \cup \{x\}} d(m, x_j) \stackrel{\text{def}}{=} \sum_{j=1}^{n} g(x_j), \tag{2}$$

where $g(\cdot)$ is a function that depends on $\mathcal{M}$ and x. Eq. (2) shows that the loss of a new medoid assignment $L(\mathcal{M}; x)$ can be written as the summation of the value of the function $g(\cdot)$ evaluated

on all n points in the dataset. Though approaches such as PAM compute $L(\mathcal{M}; x)$ exactly for each x, Bandit-PAM *adaptively estimates* this quantity by sampling reference points x_j for the most promising candidates. Indeed, computing $L(\mathcal{M}; x)$ exactly for every x is not required; promising candidates can be estimated with higher accuracy (more reference point x_j's) and less promising ones can be discarded early without requiring further unnecessary computation.

To design the adaptive sampling strategy, we show that the BUILD step and each SWAP iteration can be formulated as a best-arm identification problem from the multi-armed bandits (MAB) literature [1, 10, 13, 14]. In the typical version of the best-arm identification problem, we have m arms. At each time step $t = 0, 1, \cdots$, we decide to pull an arm $A_t \in \{1, \cdots, m\}$, and receive a reward R_t with $E[R_t] = \mu_{A_t}$. The goal is to identify the arm with the largest expected reward with high probability while expending the fewest number of total arm pulls. In the BUILD step, we view each candidate medoid x as an arm in a best-arm identification problem. The arm parameter corresponds $E[g]$, and by pulling an arm, we observe the loss evaluated on a randomly sampled data point x_j. Using this reduction, the best candidate medoid can be estimated using existing best-arm algorithms like the Upper Confidence Bound (UCB) algorithm [21].

Related work: Many other k-medoids algorithms exist, in addition to CLARA, CLARANS, and FastPAM as described above. Park et al. [33] proposed a k-means-like algorithm that alternates between reassigning the points to their closest medoid and recomputing the medoid for each cluster until the k-medoids clustering loss can no longer be improved. Other proposals include optimizations for Euclidean space and tabu search heuristics [9]. Recent work has also focused on distributed PAM, where the dataset cannot fit on one machine [37]. All of these algorithms, however, scale quadratically in dataset size and could benefit from improvements in complexity.

The idea of algorithm acceleration by converting a computational problem into a statistical estimation problem and designing the adaptive sampling procedure via multi-armed bandits has witnessed a few successes [7, 18, 24, 15, 3, 39]. In the context of k-medoids clustering, previous work [2, 4] has considered finding the *single* medoid of a set points (i.e. the 1-medoid problem). In these works, the 1-medoid problem was also formulated as a best-arm identification problem, with each point being an arm and its average distance to other points being the arm parameter.

While the 1-medoid problem considered in prior work can be solved exactly, the k-medoids problem is NP-Hard and is therefore only tractable with heuristic solutions. Hence, this paper focuses on improving the computational efficiency of an existing heuristic solution, PAM, that has been empirically observed to be superior to other techniques. Moreover, instead of having a single best-arm identification problem, we reformulate PAM as a sequence of best-arm problems. We treat different objects as arms in different steps of PAM; in the BUILD step, each point corresponds to an arm, whereas in the SWAP step, each medoid-and-non-medoid pair corresponds to an arm. We further notice that the intrinsic difficulties of this sequence of best-arm problems are different, which can be exploited to further speed up the algorithm, as demonstrated in Section 5.

2 Preliminaries

For n data points $\mathcal{X} = \{x_1, x_2, \cdots, x_n\}$ and a user-specified distance function $d(\cdot, \cdot)$, the k-medoids problem aims to find a set of k medoids $\mathcal{M} = \{m_1, \cdots, m_k\} \subset \mathcal{X}$ to minimize the overall distance of points from their closest medoids:

$$L(\mathcal{M}) = \sum_{i=1}^{n} \min_{m \in \mathcal{M}} d(m, x_i) \tag{3}$$

Note that d does not need to satisfy symmetry, triangle inequality, or positivity. For the rest of the paper, we use $[n]$ to denote the set $\{1, \cdots, n\}$ and $|\mathcal{S}|$ to represent the cardinality of a set $\mathcal{S}$. For two scalars a, b, we let $a \wedge b = \min(a, b)$ and $a \vee b = \max(a, b)$.

2.1 Partitioning Around Medoids (PAM)

The original PAM algorithm [16, 17] first initializes the set of k medoids via the BUILD step and then repeatedly performs the SWAP step to improve the loss (3) until convergence.

BUILD: PAM initializes a set of k medoids by greedily assigning medoids one-by-one so as to minimize the overall loss (3). The first point added in this manner is the medoid of all n points. Given

the current set of l medoids $\mathcal{M}_l = \{m_1, \cdots, m_l\}$, the next point to add m^* can be written as

$$\text{BUILD:} \quad m^* = \underset{x \in \mathcal{X} \setminus \mathcal{M}_l}{\arg\min} \frac{1}{n} \sum_{j=1}^{n} \left[d(x, x_j) \wedge \min_{m' \in \mathcal{M}_l} d(m', x_j) \right]. \tag{4}$$

SWAP: PAM then swaps the medoid-nonmedoid pair that would reduce the loss (3) the most among all possible $k(n-k)$ such pairs. Let $\mathcal{M}$ be the current set of k medoids. Then the best pair to swap is

$$\text{SWAP:} \quad (m^*, x^*) = \underset{(m,x) \in \mathcal{M} \times (\mathcal{X} \setminus \mathcal{M})}{\arg\min} \frac{1}{n} \sum_{j=1}^{n} \left[d(x, x_j) \wedge \min_{m' \in \mathcal{M} \setminus \{m\}} d(m', x_j) \right]. \tag{5}$$

The second term in both (4) and (5), namely $\min_{m' \in \mathcal{M}_l} d(m', x_j)$ and $\min_{m' \in \mathcal{M} \setminus \{m\}} d(m', x_j)$, can be determined by caching the smallest and the second smallest distances from each point to the previous set of medoids, namely $\mathcal{M}_l$ in (4) and $\mathcal{M}$ in (5). Therefore, in both (4) and (5), we only need to compute the distance once for each summand. As a result, PAM needs $O(kn^2)$ distance computations for the k greedy searches in the entire BUILD step and $O(kn^2)$ distance computations for each SWAP iteration.

3 Bandit-PAM

At the core of the PAM algorithm is the $O(n^2)$ BUILD search (4), which is repeated k times for initialization, and the $O(kn^2)$ SWAP search (5), which is repeated until convergence. We first show that both searches share a similar mathematical structure, and then show such a structure can be optimized efficiently using a bandit-based randomized algorithm, thus giving rise to Bandit-PAM. Rewriting the BUILD search (4) and the SWAP search (5) in terms of the change in total loss yields

$$\text{BUILD:} \quad \underset{x \in \mathcal{X} \setminus \mathcal{M}_l}{\arg\min} \frac{1}{n} \sum_{j=1}^{n} \left[\left(d(x, x_j) - \min_{m' \in \mathcal{M}_l} d(m', x_j) \right) \wedge 0 \right], \tag{6}$$

$$\text{SWAP:} \quad \underset{(m,x) \in \mathcal{M} \times (\mathcal{X} \setminus \mathcal{M})}{\arg\min} \frac{1}{n} \sum_{j=1}^{n} \left[\left(d(x, x_j) - \min_{m' \in \mathcal{M} \setminus \{m\}} d(m', x_j) \right) \wedge 0 \right]. \tag{7}$$

One may notice that the above two problems share the following similarities. First, both are searching over a finite set of parameters: $n - l$ points in the BUILD search and $k(n-k)$ swaps in the SWAP search. Second, both objective functions have the form of an average of an $O(1)$ function evaluated over a finite set of reference points. We formally describe the shared structure:

$$\text{Shared Problem:} \quad \underset{x \in \mathcal{S}_{\text{tar}}}{\arg\min} \frac{1}{|\mathcal{S}_{\text{ref}}|} \sum_{x_j \in \mathcal{S}_{\text{ref}}} g_x(x_j), \tag{8}$$

for target points $\mathcal{S}_{\text{tar}}$, reference points $\mathcal{S}_{\text{ref}}$, and an objective function $g_x(\cdot)$ that dependents on the target point x. Then both the BUILD search and the SWAP search can be written as instances of Problem (8) with:

$$\text{BUILD: } \mathcal{S}_{\text{tar}} = \mathcal{X} \setminus \mathcal{M}_l,\ \mathcal{S}_{\text{ref}} = \mathcal{X},\ g_x(x_j) = \left(d(x, x_j) - \min_{m' \in \mathcal{M}_l} d(m', x_j) \right) \wedge 0, \tag{9}$$

$$\text{SWAP: } \mathcal{S}_{\text{tar}} = \mathcal{M} \times (\mathcal{X} \setminus \mathcal{M}),\ \mathcal{S}_{\text{ref}} = \mathcal{X},\ g_x(x_j) = \left(d(x, x_j) - \min_{m' \in \mathcal{M} \setminus \{m\}} d(m', x_j) \right) \wedge 0. \tag{10}$$

Crucially, in the SWAP search, each *pair* of medoid-and-non-medoid points (m, x) is treated as one target point in $\mathcal{S}_{\text{tar}}$ in this new formulation.

3.1 Adaptive Search for the Shared Problem

Recall that the computation of $g(x_j)$ is $O(1)$. A naive, explicit method would require $O(|\mathcal{S}_{\text{tar}}||\mathcal{S}_{\text{ref}}|)$ computations of $g(x_j)$ to solve Problem (8). However, as shown in previous works [2, 3], a randomized search would return the correct result with high confidence in $O(|\mathcal{S}_{\text{tar}}| \log |\mathcal{S}_{\text{ref}}|)$ computations

Algorithm 1 `Adaptive-Search` ($\mathcal{S}_{\text{tar}}, \mathcal{S}_{\text{ref}}, g_x(\cdot), \text{batchsize}, \delta, \sigma_x$)

$\mathcal{S}_{\text{solution}} \leftarrow \mathcal{S}_{\text{tar}}$ ▷ Set of potential solutions to Problem (8)
$n_{\text{used\_ref}} \leftarrow 0$ ▷ Number of reference points evaluated
For all $x \in \mathcal{S}_{\text{tar}}$, set $\hat{\mu}_x \leftarrow 0, C_x \leftarrow \infty$ ▷ Initial mean and confidence interval for each arm
while $n_{\text{used\_ref}} < |\mathcal{S}_{\text{ref}}|$ and $|\mathcal{S}_{\text{solution}}| > 1$ **do**
  Draw a batch of batchsize samples with replacement from reference $\mathcal{S}_{\text{ref\_batch}} \subset \mathcal{S}_{\text{ref}}$
  for all $x \in \mathcal{S}_{\text{solution}}$ **do**
    $\hat{\mu}_x \leftarrow \frac{n_{\text{used\_ref}}\hat{\mu}_x + \sum_{y \in \mathcal{S}_{\text{ref\_batch}}} g_x(y)}{n_{\text{used\_ref}} + \text{batchsize}}$ ▷ Update running mean
    $C_x \leftarrow \sigma_x \sqrt{\frac{2\log(\frac{1}{\delta})}{n_{\text{used\_ref}} + \text{batchsize}}}$ ▷ Update confidence interval
  $\mathcal{S}_{\text{solution}} \leftarrow \{x : \hat{\mu}_x - C_x \leq \min_y(\hat{\mu}_y + C_y)\}$ ▷ Remove points that can no longer be solution
  $n_{\text{used\_ref}} \leftarrow n_{\text{used\_ref}} + \text{batchsize}$
if $|\mathcal{S}_{\text{solution}}| = 1$ **then**
  return $x^* \in \mathcal{S}_{\text{solution}}$
else
  Compute μ_x exactly for all $x \in \mathcal{S}_{\text{solution}}$
  return $x^* = \arg\min_{x \in \mathcal{S}_{\text{solution}}} \mu_x$

of $g(x_j)$. Specifically, for each target x in Problem (8), let $\mu_x = \frac{1}{|\mathcal{S}_{\text{ref}}|}\sum_{x_j \in \mathcal{S}_{\text{ref}}} g_x(x_j)$ denote its objective function. Computing μ_x exactly takes $O(|\mathcal{S}_{\text{ref}}|)$ computations of $g(x_j)$, but we can instead estimate μ_x with fewer computations by drawing $J_1, J_2, ..., J_{n'}$ independent samples uniformly with replacement from $[|\mathcal{S}_{\text{ref}}|]$. Then, $E[g(x_{J_i})] = \mu_x$ and μ_x can be estimated as $\hat{\mu}_x = \frac{1}{n'}\sum_{i=1}^{n'} g(x_{J_i})$, where n' determines the estimation accuracy. To estimate the solution to Problem (8) with high confidence, we can then choose to sample different targets in $\mathcal{S}_{\text{tar}}$ to different degrees of accuracy. Intuitively, promising targets with small values of μ_x should be estimated with high accuracy, while less promising ones can be discarded without being evaluated on too many reference points.

The specific adaptive estimation procedure is described in Algorithm 1. It can be viewed as a batch version of the conventional UCB algorithm [21, 39] and is easier to implement. The algorithm uses the set $\mathcal{S}_{\text{solution}}$ to track all potential solutions to Problem (8); $\mathcal{S}_{\text{solution}}$ is initialized as the set of all target points $\mathcal{S}_{\text{tar}}$. For each potential solution $x \in \mathcal{S}_{\text{solution}}$, the algorithm maintains its mean objective estimate $\hat{\mu}_x$ as well as a confidence interval C_x, where the latter depends on the exclusion probability δ as well as the dispersion parameter σ_x.

In each iteration, a new batch of reference points $\mathcal{S}_{\text{ref\_batch}}$ is evaluated for all potential solutions in $\mathcal{S}_{\text{solution}}$, making the estimate of $\hat{\mu}_x$ more accurate. Based on the current estimate, if a target's lower confidence bound $\hat{\mu}_x - C_x$ is still greater than the upper confidence bound of the most promising target $\min_y(\hat{\mu}_y + C_y)$, we remove it from the set of possible solutions $\mathcal{S}_{\text{solution}}$. This process continues until there is only one point in $\mathcal{S}_{\text{solution}}$ or until we have sampled more reference points than in the whole reference set. In the latter case, we know that the difference between the remaining targets in $\mathcal{S}_{\text{solution}}$ is so subtle that an exact computation is more efficient. We then compute those targets' objectives exactly and return the best target in the set.

3.2 Algorithmic details

Estimation of each σ_x: Bandit-PAM uses Algorithm 1 in both the BUILD step and each SWAP iteration, with input parameters specified in (9) and (10). In practice, σ_x is not known *a priori* and we estimate σ_x for each $x \in |\mathcal{S}_{\text{tar}}|$ from the data. In the first batch of sampled reference points in Algorithm 1, we estimate each σ_x as:

$$\sigma_x = \text{STD}_{y \in \mathcal{S}_{\text{ref\_batch}}} g_x(y) \tag{11}$$

where STD denotes standard deviation. Intuitively, this allows for smaller confidence intervals in later iterations, especially in the BUILD step, when we expect the average arm returns to become smaller as we add more medoids (since we are taking the minimum over a larger set on the RHS of Eq. (4)). We also allow for arm-dependent σ_x, as opposed to a fixed global σ, which allows for narrower confidence intervals for arms whose returns are heavily concentrated (e.g., distant outliers). Empirically, this results in significant speedups and results in fewer arms being computed exactly

(Line 14 in Algorithm 1). In all experiments, the batchsize is set to 100 and the error probability δ is set to $\delta = \frac{1}{1000|\mathcal{S}_{\text{tar}}|}$ in Algorithm 1. Empirically, this value of batch size and this setting of δ are such that Bandit-PAM recovers the same results in PAM in almost all cases.

Combination with FastPAM1: We also combine Bandit-PAM with the FastPAM1 optimization [35]. We discuss this optimization in Appendix 1.2.

4 Analysis of the Algorithm

The goal of Bandit-PAM is to track the optimization trajectory of the standard PAM algorithm, ultimately identifying the same set of k medoids with high probability. In this section, we formalize this statement and provide bounds on the number of distance computations required by Bandit-PAM.

We will assume that both PAM and Bandit-PAM place a hard constraint T on the maximum number of SWAP steps that are allowed. Notice that, as long as Bandit-PAM finds the correct solution to the search problem (6) at each BUILD step and to the search problem (7) at each SWAP step, it will reproduce the sequence of BUILD and SWAP steps of PAM identically, returning the same set of k medoids in the end. The hard constraint T guarantees that, even if the trajectories of PAM and Bandit-PAM deviate from each other, at most $k + T$ calls to Algorithm 1 will be performed.

Consider one such call to Algorithm 1 and suppose x^* is the optimal target point (i.e., the one with minimum μ_x). For another target point $x \in \mathcal{S}_{\text{tar}}$, let $\Delta_x = \mu_x - \mu_{x^*}$. To state the next result, we will assume that, for a randomly sampled reference point, say x_J, the random variable $Y = g_x(x_J)$ is σ-sub-Gaussian; i.e., that $\Pr(|Y - E[Y]| > t) < 2\exp\left(-t^2/(2\sigma^2)\right)$, for some known parameter σ. In addition, we assume that the data is generated in such a distribution that the mean rewarded μ_i's follow a sub-Gaussian distribution (see Sec. 6 for a discussion).

Theorem 1. *If Bandit-PAM is run on a dataset $\mathcal{X}$ with $\delta = n^{-3}$, then it returns the same set of k medoids as PAM with probability $1 - 2(k+T)/n$. Furthermore, the total number of distance computations M_{total} required satisfies*

$$E[M_{\text{total}}] = O\left((k+T)n\log n\right).$$

When the number of desired medoids k is a constant and the number of allowed SWAP steps is small (which is often sufficient in practice as discussed in Sec. 6). Theorem 1 implies that only $O(n\log n)$ distance computations are necessary to reproduce the results of PAM with high probability.

In order to prove Theorem 1, we prove a more detailed result for each call that Bandit-PAM makes to Algorithm 1. For this more specific case, we assume that, for target point x, $g_x(x_J)$ is σ_x-sub-Gaussian, where σ_x is a parameter specific to x (and can change across different calls to Algorithm 1). As it turns out, in practice one can estimate each σ_x by performing a small number of distance computations. Allowing σ_x to be estimated separately for each arm is beneficial in practice, as discussed in Sec. 6. The following theorem is proved in Appendix 3.

Theorem 2. *For $\delta = n^{-3}$, with probability at least $1 - \frac{2}{n}$, Algorithm 1 returns the correct solution to* (6) *(for a BUILD step) or* (7) *(for a SWAP step), using a total of M distance computations, where*

$$E[M] \leq 4n + \sum_{x\in\mathcal{X}} \min\left[\frac{24}{\Delta_x^2}\left(\sigma_x + \sigma_{x^*}\right)^2 \log n + \text{batchsize}, 2n\right].$$

While the assumption that σ_x is known for every x may seem excessive, it is worth pointing out that Algorithm 1 does not need to know all σ_xs exactly and an upper bound is sufficient. Notice that, if a random variable is σ-sub-Gaussian, it is also σ'-sub-Gaussian for $\sigma' > \sigma$. Hence, if we have a universal upper bound $\sigma_{\text{ub}} > \sigma_x$ for all x, the algorithm can be run with σ_{ub} replacing each σ_x. In that case, a direct consequence of Theorem 2 is that the total number of distance computations per call to Algorithm 1 satisfies

$$E[M] \leq 4n + \sum_{x\in\mathcal{X}} 96\frac{\sigma_{\text{ub}}^2}{\Delta_x^2}\log n + \text{batchsize} \leq 4n + 96\left(\frac{\sigma_{\text{ub}}}{\min_x \Delta_x}\right)^2 n\log n. \tag{12}$$

Furthermore, as proved in Appendix 2 of Bagaria et al. [2], such an instance-wise bound converts to an $O(n\log n)$ bound when μ_i's follow a Sub-Gaussan distribution. Moreover, from Theorem 2, the

Table 1: Description of Datasets

Name	Size	Distance Metric	Dimensionality
MNIST	70000	l_2, cosine distance	784
scRNA-seq	40000	l_1	10170
HOC4	3,360	Tree edit distance	∞

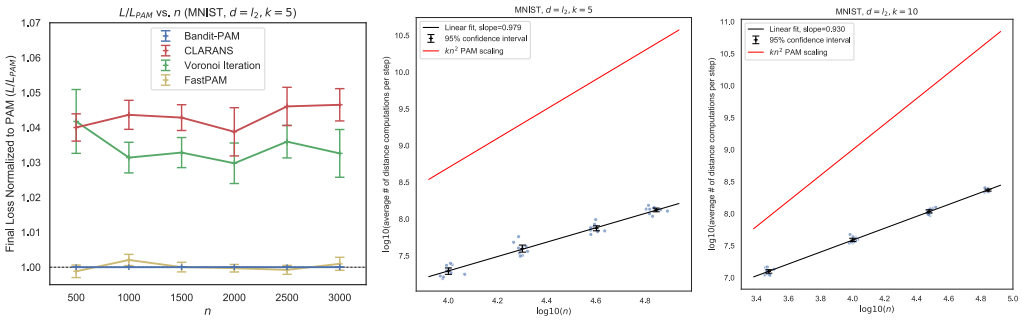
probability that Algorithm 1 does not return the target point x with the smallest value of μ_x is at most $2/n$. By the union bound, the probability that Bandit-PAM does not return the same set of k medoids as PAM is at most $2(k+T)/n$. Moreover, since at most $k+T$ calls to Algorithm 1 are made, from (12) we see that the total number of distance computations M_{total} required by Bandit-PAM satisfies $E[M_{\text{total}}] = O((k+T)n \log n)$. This proves Theorem 1.

5 Empirical Results

We run experiments on three real-world datasets to validate the expected behavior of Bandit-PAM: the MNIST hand-written digits dataset [22], the 10x Genomics 68k PBMCs scRNA-seq dataset [40], and the Code.org Hour Of Code #4 (HOC4) coding exercise submission dataset (`code.org`), all of which are publicly available.

Datasets. The MNIST dataset [22] consists of 70,000 black-and-white images of handwritten digits, where each digit is represented as a 784 dimensional vector. We consider two distance metrics, namely l_2 distance and cosine distance. The scRNA-seq dataset contains the gene expression levels of 10,170 different genes in each of 40,000 cells after standard filtering. We consider l_1 distance that is popular for clustering scRNA-seq data [31] has often processed scRNA-seq datasets with l_1 distance. The HOC4 dataset from Code.org [8] consists of 3,360 unique solutions to a block-based programming exercise on Code.org. Solutions to the programming exercise are represented as abstract syntax trees (ASTs), and we consider the tree edit distance to quantify the similar between solutions. See Table 1 for a summary.

Setup. In Subsec. 5.1, we show that Bandit-PAM returns the same results as PAM. We also compare the clustering results with other popular k-medoids clustering algorithms in terms of the clustering loss (3), including FastPAM [35], CLARANS [30], and Voronoi Iteration [33]. In Subsec. 5.2, we demonstrate that Bandit-PAM scales linearly in the number of samples n for all datasets and all metrics considered. Each parameter setting was repeated 10 times with data subsampled from the original dataset. 95% confidence intervals are provided.



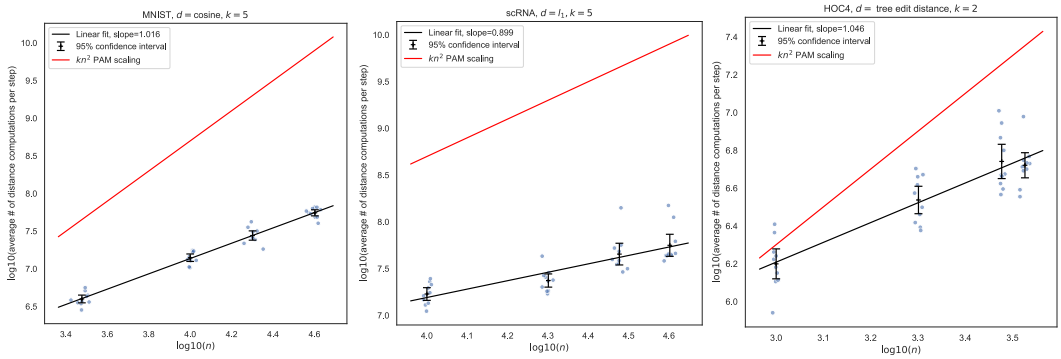
(a) (b) (c)

Figure 1: (a) Clustering loss relative to the PAM loss. Data is subsampled from MNIST, sample size n varies from 500 to 3000, $k = 5$ and 95% confidence intervals are provided. Bandit-PAM always returns the same solution as PAM and hence has loss ratio 1. FastPAM has a comparable performance, while the other two algorithms are significantly worse. (b-c) Average number of distance calls per iteration vs sample size n for MNIST and l_2 distance with (b) $k = 5$ and (c) $k = 10$. The plot is shown on a log-log scale. Lines of best fit (black) are plotted, as are reference lines demonstrating the expected scaling of PAM (red).

5.1 Clustering/loss quality

Figure 1 (a) shows the relative losses of algorithms with respect to the loss of PAM. Bandit-PAM and three other baselines, namely FastPAM [35], CLARANS [30], and Voronoi Iteration [33], are considered. We note that FastPAM is different from FastPAM1 mentioned before; it takes $O(n^2)$ for each SWAP step but does not guarantee the same solution as PAM. Bandit-PAM always returns the same solution as PAM and hence has loss ratio 1. FastPAM has a comparable performance, while the other two algorithms are significantly worse.

5.2 Scaling with n for different datasets, distance metric, and k values



(a) (b) (c)

Figure 2: Average number of distance calls per iteration vs sample size n, for (a) MNIST and cosine distance, (b) scRNA-seq and l_1 distance, and (c) HOC4 and tree edit distance. The plot is shown on a log-log scale. Lines of best fit (black) are plotted, as are reference lines demonstrating the expected scaling of PAM (red).

We next consider the number of distance calls per iteration as the sample size increases. The number of distance calls per iteration is defined as the total distance calls divided by the number of SWAP steps plus 1, where the 1 corresponds to the BUILD step. We choose to look at this quantity to account for different number of SWAPs for different runs, in order to provide a fair comparison.

If the complexity is linear, then the slope would be 1 in the log-log plot. Indeed, as shown in Figure 1 (b-c), the slope for $k = 5$ and $k = 10$ are 0.979 and 0.930, respectively, indicating the scaling is linear in n for different values of k.

In addition, as shown in Figure 2, the slopes of the log-log plot are 1.018, 0.899 and 1.046 for MNIST with cosine distance, scRNA-seq with l_1 distance, HOC4 with tree edit distance, respectively, validating our theory that Bandit-PAM takes almost linear number of distance evaluations per iteration for different datasets and different distance metrics.

6 Discussion

In all experiments, we have observed that the numbers of SWAPs are very small, typically fewer than 10, justifying the assumption of having an upper limit on the PAM SWAP step prior to running the algorithm in Sec. 4.

We observe that for all datasets, the randomly sampled distances have an empirical distribution similar to Gaussian distribution (Appendix Figures 4-5), justifying the Sub-Gaussian assumption in Sec. 4. In addition, we observe that the the Sub-Gaussian parameters are different for different steps and different points (Appendix Figures 2), justifying the adaptive estimation of the sub-Gaussianity parameters in SubSec. 3.2.

In addition, the distribution of the true arm parameters also mostly do have a heavy-tailed distribution (Appendix Figure 3), justifying the distributional assumption of μ_i's in Sec. 4.

Broader Impact

In this work, we proposed an algorithm that accelerated finding solutions to the k-medoids problem while producing comparable – and usually equivalent – final cluster assignments. Our work enables the discovery of high-quality medoid assignments in very large datasets, including some on which prior algorithms were prohibitively expensive. A potential negative consequence of this is that practitioners may be incentivized to gather and store larger amounts of data now that it can be meaningfully processed, in a phenomenon more generally described as induced demand [11]. This incentive realignment could potentially result in negative externalities such as an increase in energy consumption and carbon footprints.

We also anticipate, however, that Bandit-PAM will enable several beneficial applications in biomedicine, education, and fairness. For example, the evolutionary pathways of infectious diseases could possibly be constructed from the medoids of genetic sequences available at a given point in time, if prior temporal information about these sequences' histories is not available. Similarly, the medoids of patients infected in a disease outbreak may elucidate the origins of outbreaks, as did prior analyses of cholera outbreaks using Voronoi Iteration [6]. Our application to the HOC4 dataset also suggests a method for scaling personalized feedback to individual students in online courses. If limited resources are available, instructors can choose to provide feedback on just the *medoids* of submitted solutions instead of exhaustively providing feedback on *every* unique solution, of which there may be several thousand. Instructors can then refer individual students to the feedback provided for their closest medoid. We anticipate that this approach can be applied generally for students of Massive Open Online Courses (MOOCs), thereby enabling more equitable access to education and personalized feedback for students.. In particular, especially with recent interest in online learning, we hope that our work will improve the quality of learning for students worldwide.

References

[1] Jean-Yves Audibert and Sébastien Bubeck. Best arm identification in multi-armed bandits. 2010.

[2] Vivek Bagaria, Govinda Kamath, Vasilis Ntranos, Martin Zhang, and David Tse. Medoids in almost-linear time via multi-armed bandits. In *International Conference on Artificial Intelligence and Statistics*, pages 500–509, 2018.

[3] Vivek Bagaria, Govinda M Kamath, and David N Tse. Adaptive monte-carlo optimization. *arXiv preprint arXiv:1805.08321*, 2018.

[4] Tavor Baharav and David Tse. Ultra fast medoid identification via correlated sequential halving. In *Advances in Neural Information Processing Systems*, pages 3650–3659, 2019.

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

[6] Donald Cameron and Ian Jones. John snow, the broad street pump and modern epidemiology. In *International Journal of Epidemiology*, volume 12, page 393–396, 2020.

[7] Hyeong Soo Chang, Michael C Fu, Jiaqiao Hu, and Steven I Marcus. An adaptive sampling algorithm for solving markov decision processes. *Operations Research*, 53(1):126–139, 2005.

[8] Code.org. Research at code.org. 2013.

[9] Vladimir Estivill-Castro and Michael E Houle. Robust distance-based clustering with applications to spatial data mining. *Algorithmica*, 30(2):216–242, 2001.

[10] 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. Springer, 2002.

[11] Kent Hymel, Kenneth Small, and Kurt Van Dender. Induced demand and reboundeffects in road transport. In *Transportation Research B, Methodological*, volume 44, page 1220–1241, 2020.

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

[13] Kevin Jamieson, Matthew Malloy, Robert Nowak, and Sébastien Bubeck. lil'ucb: An optimal exploration algorithm for multi-armed bandits. In *Conference on Learning Theory*, pages 423–439, 2014.

[14] Kevin Jamieson and Robert Nowak. Best-arm identification algorithms for multi-armed bandits in the fixed confidence setting. In *2014 48th Annual Conference on Information Sciences and Systems (CISS)*, pages 1–6. IEEE, 2014.

[15] Kevin Jamieson and Ameet Talwalkar. Non-stochastic best arm identification and hyperparameter optimization. In *Artificial Intelligence and Statistics*, pages 240–248, 2016.

[16] Leonard Kaufman and Peter J Rousseeuw. Clustering by means of medoids. statistical data analysis based on the l1 norm. *Y. Dodge, Ed*, pages 405–416, 1987.

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

[18] Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. In *European conference on machine learning*, pages 282–293. Springer, 2006.

[19] Branislav Kveton, Csaba Szepesvari, and Mohammad Ghavamzadeh. Perturbed-history exploration in stochastic multi-armed bandits. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*.

[20] Branislav Kveton, Csaba Szepesvari, Sharan Vaswani, Zheng Wen, Mohammad Ghavamzadeh, and Tor Lattimore. Garbage in, reward out: Bootstrapping exploration in multi-armed bandits. In *International Conference on Machine Learning*, page 3601–3610, 2019.

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

[22] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[23] Jure Leskovec, Anand Rajaraman, and Jeffrey David Ullman. *Mining of massive data sets*. Cambridge university press, 2020.

[24] Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. *arXiv preprint arXiv:1603.06560*, 2016.

[25] Stuart Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982.

[26] Malte Luecken and Fabian Theis. Current best practices in single-cell rna-seq analysis: A tutorial. *Molecular Systems Biology*, 155:e8746(6), 2019.

[27] James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA, 1967.

[28] Nina Mishra, Robert Schreiber, Isabelle Stanton, and Robert E Tarjan. Clustering social networks. In *International Workshop on Algorithms and Models for the Web-Graph*, pages 56–67. Springer, 2007.

[29] Gonzalo Navarro. A guided tour to approximate string matching. *ACM computing surveys (CSUR)*, 33(1):31–88, 2001.

[30] 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.

[31] Vasilis Ntranos, Govinda M Kamath, Jesse M Zhang, Lior Pachter, and N Tse David. Fast and accurate single-cell rna-seq analysis by clustering of transcript-compatibility counts. *Genome biology*, 17(1):112, 2016.

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

[33] Hae-Sang Park and Chi-Hyuck Jun. A simple and fast algorithm for k-medoids clustering. *Expert systems with applications*, 36(2):3336–3341, 2009.

[34] Alan P Reynolds, Graeme Richards, Beatriz de la Iglesia, and Victor J Rayward-Smith. Clustering rules: a comparison of partitioning and hierarchical clustering algorithms. *Journal of Mathematical Modelling and Algorithms*, 5(4):475–504, 2006.

[35] 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.

[36] Erich Schubert and Arthur Zimek. Elki: A large open-source library for data analysis-elki release 0.7. 5" heidelberg". *arXiv preprint arXiv:1902.03616*, 2019.

[37] Hwanjun Song, Jae-Gil Lee, and Wook-shin Han. Pamae: Parallel k -medoids clustering with high accuracy and efficiency. In *Proc. 23 ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining*, volume 1, pages 281–297. Oakland, CA, USA, 1967.

[38] Chi-Hua Wang, Yang Yu, Botao Hao, and Guang Cheng. Residual bootstrap exploration for bandit algorithms. 2020.

[39] Martin Zhang, James Zou, and David Tse. Adaptive monte carlo multiple testing via multi-armed bandits. In *International Conference on Machine Learning*, pages 7512–7522, 2019.

[40] Grace XY Zheng, Jessica M Terry, Phillip Belgrader, Paul Ryvkin, Zachary W Bent, Ryan Wilson, Solongo B Ziraldo, Tobias D Wheeler, Geoff P McDermott, Junjie Zhu, et al. Massively parallel digital transcriptional profiling of single cells. *Nature communications*, 8(1):1–12, 2017.

Appendix

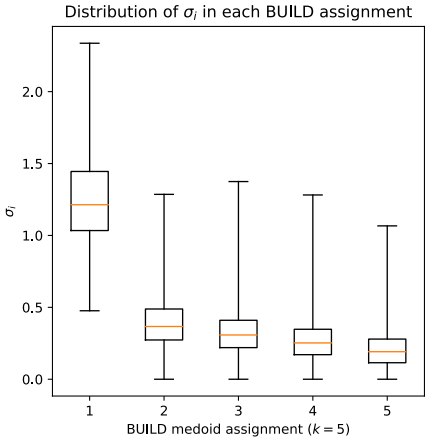
1.1 FastPAM1 Optimization

Algorithm 1 can also be combined with the FastPAM1 optimization from [35] to reduce the number of computations in each SWAP iteration. For a given candidate swap (m, x), we rewrite $g_{(m,x)}(x_j)$ from Eq. (10) as:

$$g_{m,x}(x_j) = -d_1(x_j) + \mathbb{1}_{x_j \notin \mathcal{C}_m} \min[d_1(x_j), d(x, x_j)] + \mathbb{1}_{x_j \in \mathcal{C}_m} \min[d_2(x_j), d(x, x_j)] \quad (13)$$

where $\mathcal{C}_m$ denotes the set of points whose closest medoid is m. Also, $d_1(x_j)$ and $d_2(x_j)$ are the distance from x_j to its nearest and second nearest medoid, respectively, before the swap is performed. We cache the values $d_1(x_j), d_2(x_j)$, and the cluster assignments $\mathcal{C}_m$ so that Eq. (13) no longer depends on m and instead depend only on $\mathbb{1}_{\{x_j \in \mathcal{C}_m\}}$, which is cached. This allows for an $O(k)$ speedup in each SWAP iteration.

1.2 Value of Re-estimating each σ_x



Appendix Figure 2: Boxplot showing the min, max, and each quartile for the set of all σ_x estimates for the full MNIST dataset.

The theoretical results in Section 4 and empirical results in Section 5 suggest that Bandit-PAM scales almost linearly in dataset size for a variety of real-world datasets and commonly used metrics. One may also ask if Lines 7-8 of Algorithm 1, in which we re-estimate each σ_i from the data, are necessary. In some sense, we treat the set of $\{\sigma_i\}$ as adaptive in two different ways: σ_i is calculated on a *per-arm* basis (hence the subscript i), as well recalculated in each BUILD and SWAP iteration. In practice, we observe that re-estimating each σ_x for each sequential call to Algorithm 1 significantly improves the performance of our algorithm. Figure 2 describes the distribution of estimate σ_x for the MNIST data at different stages of the BUILD step. The median σ_x drops dramatically after the first medoid has been assigned and then steadily decreases, as indicated by the orange lines, and suggests that each σ_x should be recalculated at every assignment step. Furthermore, the whiskers demonstrate significant variation amongst the σ_x in a given assignment step and suggest that having arm-dependent σ_x parameters is necessary. Without these modifications to our algorithm, we find that the confidence intervals used by Bandit-PAM (Line 8) are unnecessarily large and cause computation to be expended needlessly as it becomes harder to identify good arms. Intuitively, this is due to the much larger confidence intervals that make it harder to distinguish between arms' mean returns. For a more detailed discussion of the distribution of σ_x and examples where the assumptions of Theorem 1 are violated, we refer the reader to Appendix 1.3.

1.3 Violation of Distributional Assumptions

In this section, we investigate the robustness of Bandit-PAM to violations of the assumptions in 1 on an example dataset and provide intuitive insights into the degradation of scaling. We create a new

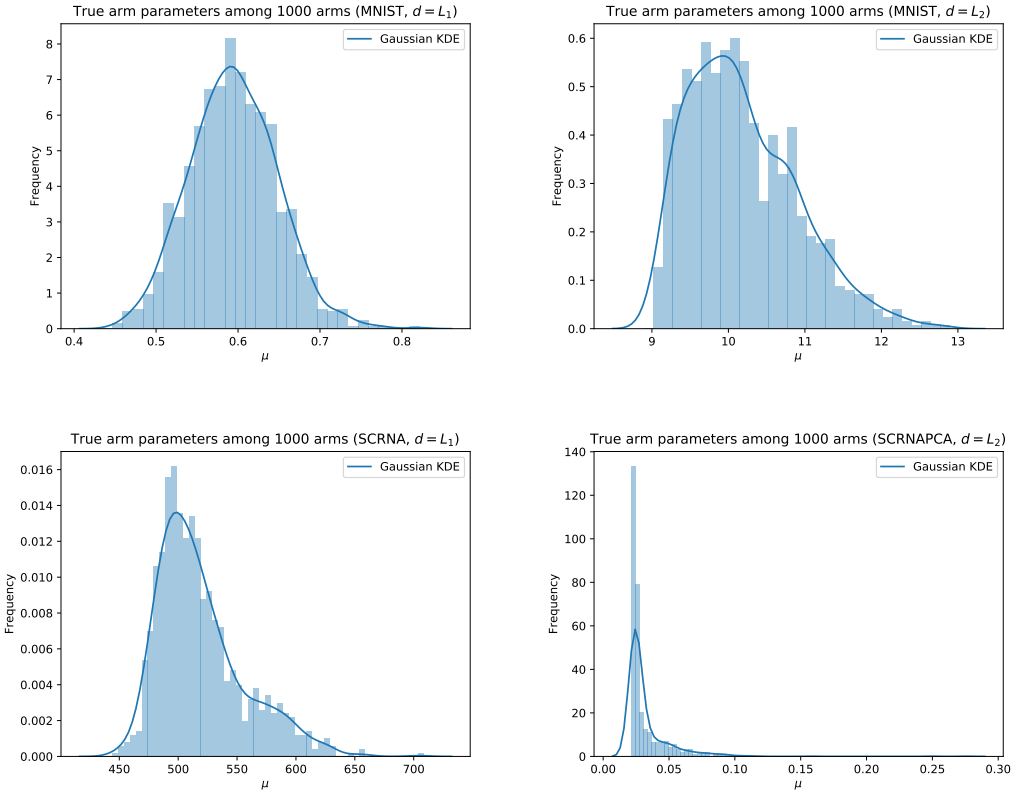
dataset from the scRNA dataset by projecting each point onto the top 10 principal components of the dataset; we call the dataset of projected points scRNA-PCA. Such a transformation is commonly used in prior work ; the most commonly used distance metric between points is the l_2 distance [26].

Figure 3 shows the distribution of arm parameters in for various (dataset, metric) pairs in the first BUILD step. In this step, the arm parameter corresponds to the mean distance from the point (the arm) to every other point. We note that the true arm parameters in scRNA-PCA are more heavily concentrated about the minimum than in the other datasets. Intuitively, we have projected the points from a 10,170 dimensional space into a 10 dimensional one and have lost significant information in the process. This makes many points appear "similar" in the projected space.

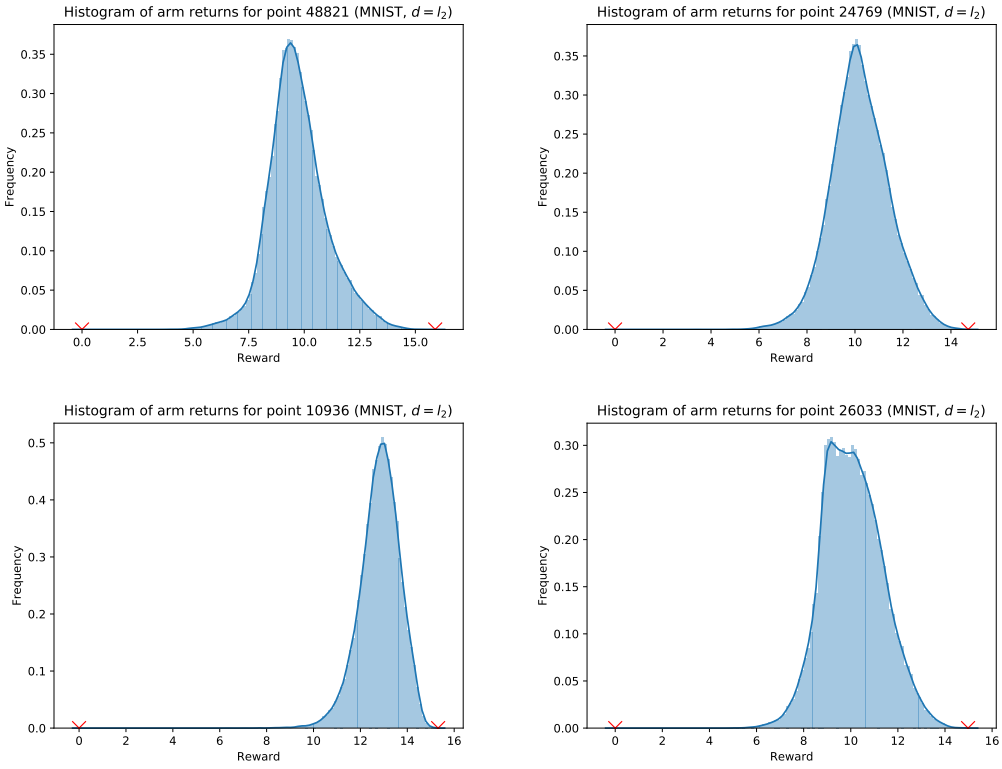
Figures 4 and 5 show the distribution of arm rewards for 4 arms (points) in MNIST and scRNA-PCA, respectively, in the first BUILD step. We note that the examples from scRNA-PCA display much larger tails, suggesting that their sub-Gaussianity parameters σ_x are very high.

Together, these observations suggest that the scRNA-PCA dataset may violate the assumptions of Theorems 1 and 2 and hurt the scaling of Bandit-PAM with n. Figure 6 demonstrates the scaling of Bandit-PAM with n on scRNA-PCA. The slope of the line of best fit is 1.204, suggesting that Bandit-PAM scales as $O(n^{1.2})$ in dataset size. We note that this is higher than the exponents suggested for other datasets by Figures 2 and 1, likely to the different distributional characteristics of the arm means and their spreads.

We note that, in general, it may be possible to characterize the distribution of arm returns μ_i at and the distribution of σ_x, the sub-Gaussianity parameter, at every step of Bandit-PAM from properties of the data-generating distribution, as done for several distributions in [2]. We leave this more general problem, as well as its implications for the complexity of our Bandit-PAM , to future work.



Appendix Figure 3: Histogram of true arm parameters, μ_i, for 1000 randomly sampled arms in the first BUILD step of various datasets. For scRNA-PCA with $d = l_2$ (bottom right), the arm returns are much more sharply peaked about the mininum than for the other datasets. In plots where the bin widths are less than 1, the frequencies can be greater than 1.



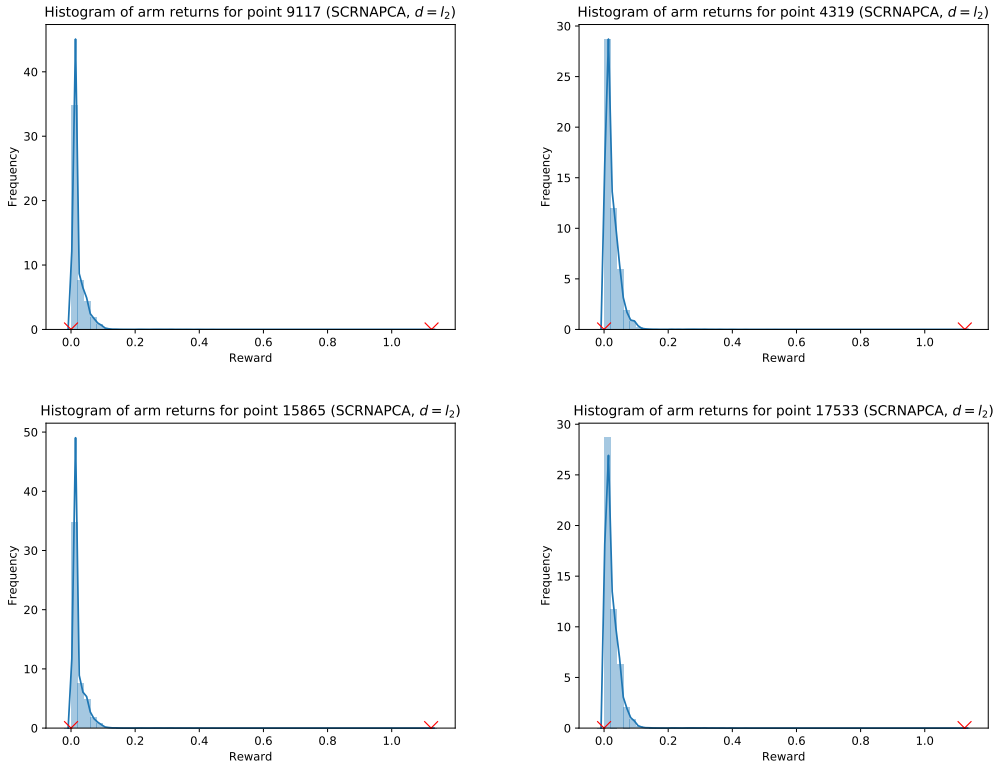
Appendix Figure 4: Example distribution of rewards for 4 points in MNIST in the first BUILD step. The minimums and maximums are indicated with red markers.

2 Future Work

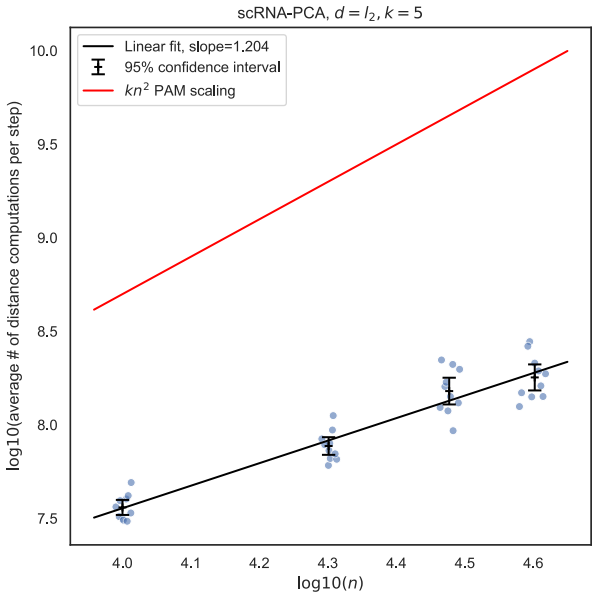
There are several ways in which Bandit-PAM could be improved or made more impactful. In this work, we chose to implement a UCB-based algorithm to find the medoids of a dataset. Other best-arm-identification approaches, however, could also be used for this problem. An alternate approach using bootstrap-based bandits could also be valuable, especially in relaxing the distributional assumptions on the data that the quantities of interest are σ-sub-Gaussian [38, 20, 19]. It may also be possible to generalize a recent single-medoid approach, Correlation-Based Sequential Halving [4], to more than 1 medoid. Though we do not have reason to suspect an algorithmic speedup (as measured by big-O), we may see constant factor improvements or improvements in wall clock time.

Throughout this work, we assumed that computing the distance between two points was an $O(1)$ operation. This obfuscates the dependence on the dimensionality of the data, d. If we consider computing the distance between two points as an $O(d)$ computation, the complexity of Bandit-PAM could be expressed as $O(dn\log n)$ in the BUILD step and each SWAP iteration. Recent work [3] suggests that this could be further improved; instead of computing the difference in each of the d coordinates, we may be able to adaptively sample which of the d coordinates to use in our distance computations and reduce the dependence on dimensionality from d to $O(\log d)$.

Finally, it may be possible to improve the theoretical bounds presented in Theorem 1. We also note that it may be possible to prove the optimality of Bandit-PAM in regards to algorithmic complexity, up to constant factors, using techniques from [2] that were developed for sample-efficiency guarantees in hypothesis testing.



Appendix Figure 5: Example distribution of rewards for 4 points in scRNA-PCA in the first BUILD step. The minimums and maximums are indicated with red markers. The distributions shown here are more heavy-tailed than in Figure 4. In plots where the bin widths are less than 1, the frequencies can be greater than 1.



Appendix Figure 6: Average number of distance calls per iteration vs n, for scRNA-PCA and l_2 distance on a log-log scale. The line of best fit (black) are plotted, as are reference lines demonstrating the expected scaling of PAM (red).

3 Proof of Theorem 2

Proof. First, we show that, with probability $1 - \frac{2}{n}$, all confidence intervals computed throughout the algorithm are true confidence intervals, in the sense that they contain the true parameter μ_x. To see this, notice that for a fixed x and a fixed iteration of the algorithm, $\hat{\mu}_x$ is the average of $n_{\text{used\_ref}}$ i.i.d. samples of a σ_x-sub-Gaussian distribution. From Hoeffding's inequality,

$$\Pr\left(|\mu_x - \hat{\mu}_x| > C_x\right) \leq 2\exp\left(-\frac{n_{\text{used\_ref}} C_x^2}{2\sigma_x^2}\right) = 2\delta.$$

Notice that there are at most $n^2/\text{batchsize} \leq n^2$ such confidence intervals computed across all target points (i.e., arms) and all steps of the algorithm. If we set $\delta = 1/n^3$, we see that $\mu_x \in [\hat{\mu}_x - C_x, \hat{\mu}_x + C_x]$ for every x and for every step of the algorithm with probability at least $1 - 2/n$, by the union bound.

Let $x^* = \arg\min_{x \in \mathcal{S}_{\text{tar}}} \mu_x$. Notice that if all confidence intervals throughout the algorithm are correct, it is impossible for x^* to be removed from the set of candidate target points. Moreover, it is clear that the main while loop in the algorithm can only run $n/\text{batchsize}$ times and that the algorithm must terminate. Hence, x^* (or some $y \in \mathcal{S}_{\text{tar}}$ with $\mu_y = \mu_{x^*}$) must be returned upon termination.

Let $n_{\text{used\_ref}}$ be the total number of arm pulls computed for each of the arms remaining in the set of candidate arms at some point in the algorithm. Notice that, for any suboptimal arm $x \neq x^*$ that has not left the set of candidate arms, we must have $C_x = \sigma_x \sqrt{2\log(\frac{1}{\delta})/n_{\text{used\_ref}}}$. Moreover, if $n_{\text{used\_ref}} > \frac{24}{\Delta_x^2}\left(\sigma_x + \sigma_{x^*}\right)^2 \log n$, we have that

$$2(C_x + C_{x^*}) = 2\left(\sigma_x + \sigma_{x^*}\right)\sqrt{2\log(n^3)/n_{\text{used\_ref}}} < \Delta_x = \mu_x - \mu_{x^*},$$

and $\hat{\mu}_x - C_x > \mu_x - 2C_x = \mu_{x^*} + \Delta_x - 2C_x \geq \mu_{x^*} + 2C_{x^*} > \hat{\mu}_{x^*} + C_{x^*}$, implying that x must be removed from the set of candidate arms at the end of that iteration. Hence, the number of distance computations M_x required for target point $x \neq x^*$ is at most

$$M_x \leq \min\left[\frac{24}{\Delta_x^2}\left(\sigma_x + \sigma_{x^*}\right)^2 \log n + \text{batchsize}, 2n\right].$$

Notice that this holds simultaneously for all $x \in \mathcal{S}_{\text{tar}}$ with probability $1 - \frac{2}{n}$. We conclude that the total number of distance computations M satisfies

$$\begin{aligned} E[M] &\leq E[M|\text{ all confidence intervals are correct}] + \frac{2}{n}(2n^2) \\ &\leq 4n + \sum_{x \in \mathcal{X}} \min\left[\frac{24}{\Delta_x^2}\left(\sigma_x + \sigma_{x^*}\right)^2 \log n + \text{batchsize}, 2n\right], \end{aligned}$$

where we used the fact that the maximum number of distance computations per target point is $2n$. □