Appendix

1 Additional Discussions

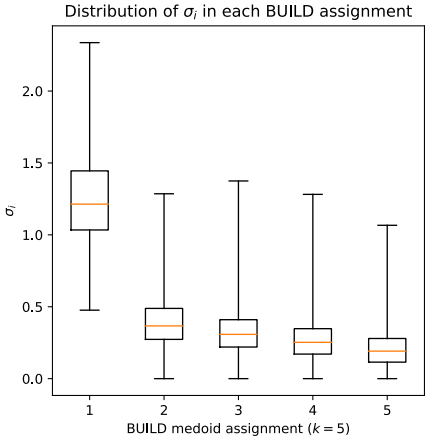
1.1 FastPAM1 optimization

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

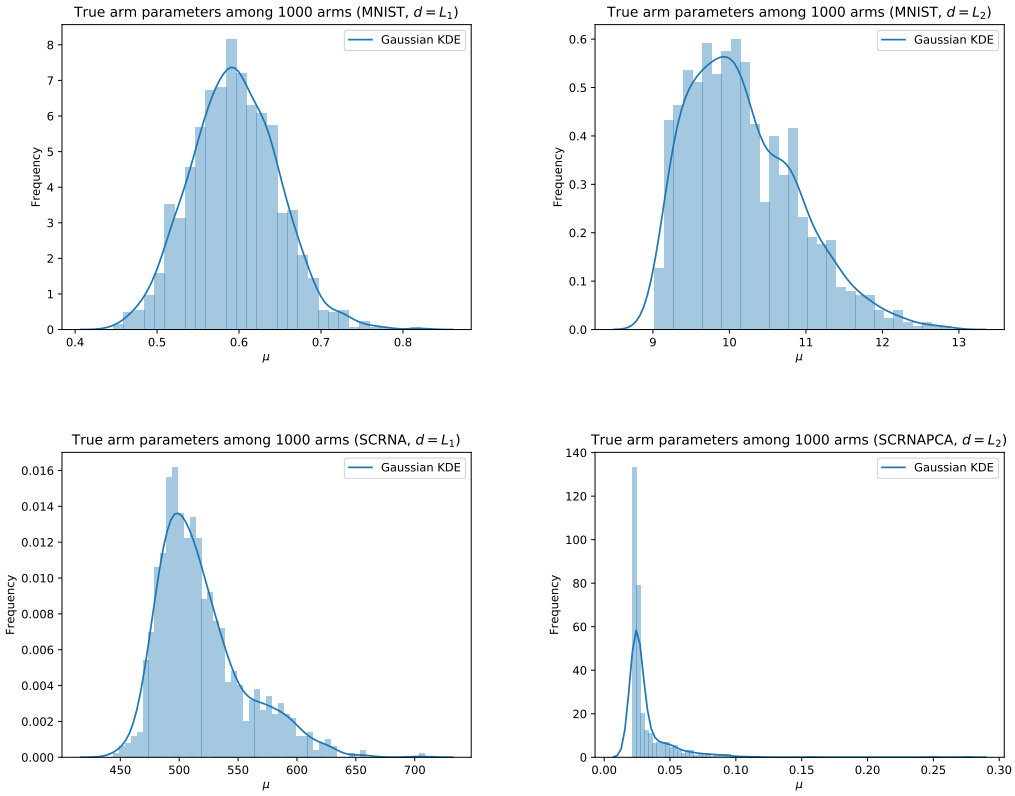
In this section, we investigate the robustness of Bandit-PAM to violations of the assumptions in 1 on an example dataset and provide intuitive insights into the degradation of scaling. We create a new dataset from the scRNA dataset by projecting each point onto the top 10 principal components of the 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.

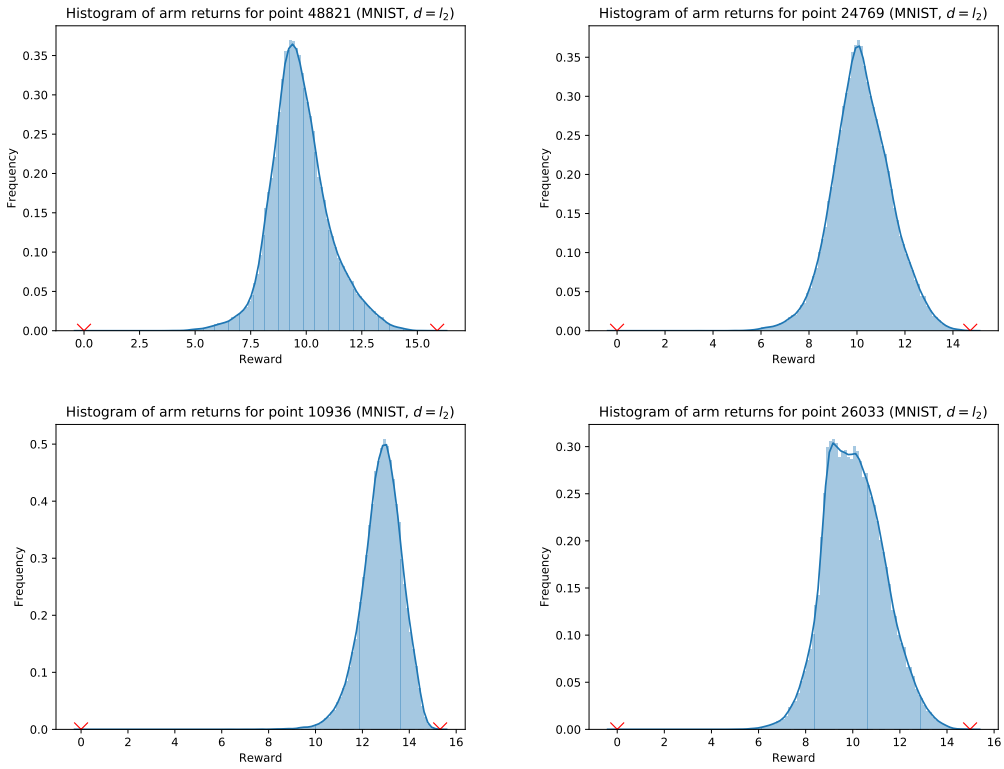
Figures 4 and 5 show the distribution of arm rewards for 4 arms (points) in MNIST and scRNA-PCA, respectively, in the first BUILD step. We note that the examples from scRNA-PCA display much larger tails, suggesting that their sub-Gaussianity parameters σ_x are very high.

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.

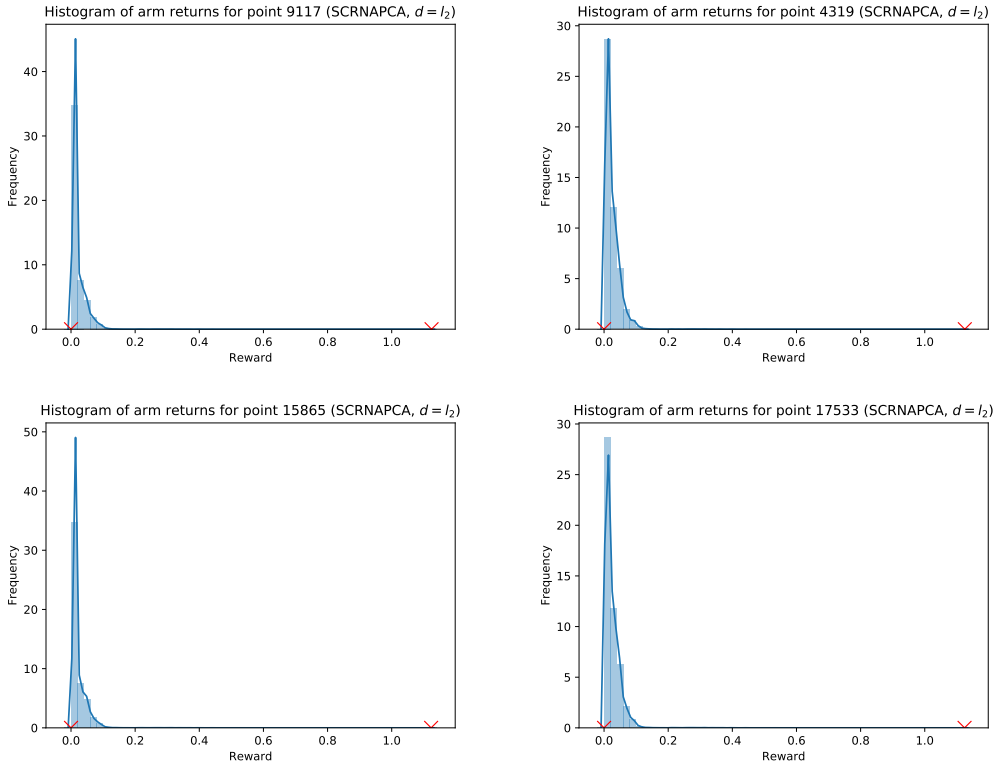
We note that, in general, it may be possible to characterize the distribution of arm returns μ_i at and the distribution of σ_x, the sub-Gaussianity parameter, at every step of Bandit-PAM from properties of the data-generating distribution, as done for several distributions in [2]. We leave this more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



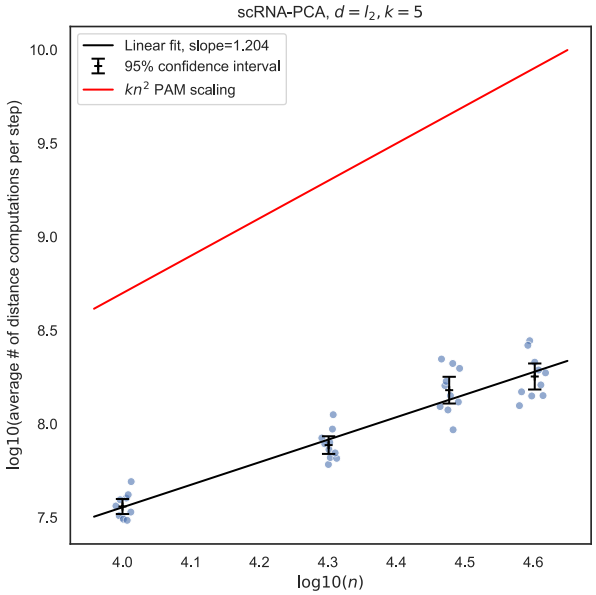
Appendix Figure 3: Histogram of true arm parameters, μ_i, for 1000 randomly sampled arms in the first BUILD step of various datasets. For scRNA-PCA with $d = l_2$ (bottom right), the arm returns are much more sharply peaked about the mininum than for the other datasets. In plots where the bin widths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



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).

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.

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 $2n$. □