Medoids in Almost-Linear Time via Multi-Armed Bandits

Vivek Bagaria* Stanford University
Govinda M. Kamath* Stanford University
Vasilis Ntranos* UC, Berkeley
Martin J. Zhang* Stanford University
David N. Tse Stanford University

Abstract

Computing the medoid of a large number of points in high-dimensional space is an increasingly common operation in many data science problems. We present an algorithm `Med-dit` to compute the medoid with high probability, which uses $O(n \log n)$ distance evaluations. `Med-dit` is based on a connection with the multi-armed bandit problem. We evaluate the performance of `Med-dit` empirically on the Netflix-prize and single-cell RNA-Seq datasets, containing hundreds of thousands of points living in tens of thousands of dimensions, and observe a 5-10x improvement in performance over the current state of the art. We have released the code of `Med-dit` and our empirical results at `https://github.com/bagavi/Meddit`

1 INTRODUCTION

An important building block in many modern data analysis problems such as clustering is the efficient computation of a representative point for a large set of points in high dimension. A commonly used representative point is the *centroid* of the set, the point which has the smallest average distance from all other points in the set. For example, k-means clustering [Steinhaus, 1956, MacQueen et al., 1967, Lloyd, 1982] computes the centroid of each cluster under the squared Euclidean distance. In this case, the centroid is the arithmetic mean of the points in the cluster, which can be computed very efficiently. Efficient computation of centroid is central to the success of k-means, as this computation has to be repeated many times in the clustering process.

While commonly used, centroid suffers from several drawbacks. First, the centroid is in general *not* a point in the dataset and thus may not be interpretable in many applications. This is especially true when the data is structured like in images, or when it is sparse like in recommendation systems [Leskovec et al., 2014]. Second, the centroid is sensitive to outliers: several far away points will significantly affect the location of the centroid. Third, while the centroid can be efficiently computed under squared Euclidean distance, there are many applications where using this distance measure is not suitable. Some examples would be applications where the data is categorical like medical records [Sudlow et al., 2015]; or situations where the data points have different support sizes such as in recommendation systems [Leskovec et al., 2014]; or cases where the data points are on a high-dimensional probability simplex like in single cell RNA-Seq analysis [Ntranos et al., 2016]; or cases where the data lives in a space with no well known Euclidean space like while clustering on graphs from social networks.

An alternative to the centroid is the *medoid*; this is the point *in the set* that minimizes the average distance to all the other points. It is used for example in k-medoids clustering [Kaufman and Rousseeuw, 1987]. On the real line, the medoid is the median of the set of points. The medoid overcomes the first two drawbacks of the centroid: the medoid is by definition one of the points in the dataset, and it is less sensitive to outliers than the centroid. In addition, centroid algorithms are usually specific to the distance used to define the centroid. On the other hand, medoid algorithms usually work for arbitrary distances.

The naive method to compute the medoid would require computing all pairwise distances between points in the set. For a set with n points, this would require the computation of $\binom{n}{2}$ distances, which would be computationally prohibitive when there are hundreds of thousands of points and each point lives in a space with dimensions in tens of thousands.

In the one-dimensional case, the medoid problem reduces to the problem of finding the median, which can be solved in linear time through `Quick-select` [Hoare, 1961]. However, in higher dimensions, no linear-time

Proceedings of the 21^{st} International Conference on Artificial Intelligence and Statistics (AISTATS) 2018, Lanzarote, Spain. PMLR: Volume 84. Copyright 2018 by the author(s).

*Contributed equally and listed alphabetically.

algorithm is known. `RAND` [Eppstein and Wang, 2006] is an algorithm that estimates the average distance of each point to all the other points by sampling a random subset of other points. It takes a total of $O\left(\frac{n \log n}{\epsilon^2}\right)$ distance computations to approximate the medoid within a factor of $(1 + \epsilon\Delta)$ with high probability, where Δ is the maximum distance between two points in the dataset. We remark that this is an approximation algorithm, and moreover Δ may *not* be known apriori. `RAND` was leveraged by `TOPRANK` [Okamoto et al., 2008] which uses the estimates obtained by `RAND` to focus on a small subset of candidate points, evaluates the average distance of these points *exactly*, and picks the minimum of those. `TOPRANK` needs $O(n^{\frac{5}{3}} \log^{\frac{4}{3}} n)$ distance computations to find the *exact* medoid with high probability under a distributional assumption on the average distances. `trimed` [Newling and Fleuret, 2017] presents a clever algorithm to find the medoid with $O(n^{\frac{3}{2}} 2^{\Theta(d)})$ distance evaluations, which works very well when the points live in a space of dimensions less than 20 (i.e., $2^d \ll \sqrt{n}$) under a distributional assumption on the points (in particular on the distribution of points around the medoid). However, the exponential dependence on dimension makes it impractical when $d \geq 50$, which is the case we consider here. Note that their result requires the distance measure to satisfy the triangle inequality.

In this paper, we present `Med(oid)-(Ban)dit`, a sampling-based algorithm that finds the *exact* medoid with high probability. It takes $O(n \log n)$ distance evaluations under natural distributional assumptions on the distances, which are justified by the characteristics observed in real data (Section 2). In contrast to `trimed`, the number of distance computations is *independent* of the dimension d; moreover, there is no specific requirement for the distance measure, e.g. the triangle inequality or the symmetry. Thus, `Med-dit` is particularly suitable for high-dimensional data and general distance measures.

In Figure 1, we showcase the performance of `RAND` and `Med-dit` on the largest cluster of the single cell RNA-Seq gene expression dataset of 10xGenomics [2017]. This consists of $27,998$ gene-expressions of each of $109,140$ cells, *i.e* $109,140$ points in $27,998$ dimensions. We note that `Med-dit` evaluates around 10 times fewer distances than `RAND` to achieve similar performance. At this scale running `TOPRANK` and `trimed` is computationally prohibitive.

The main idea behind `Med-dit` is noting that the problem of computing the medoid can be posed as that of computing the best arm in a multi-armed bandit

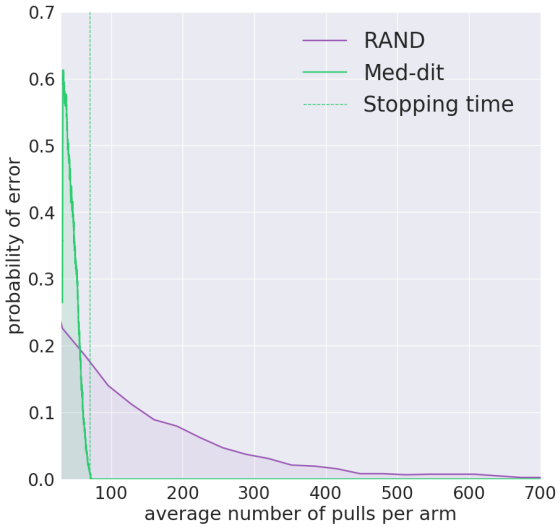

Figure 1: Large single cell dataset: We plot the probability that `RAND` and `Med-dit` do not return the true medoid as a function of the number of distance evaluations per point over 1000 monte-carlo trials. We note that `Med-dit` needs much fewer distance evaluations than `RAND` for comparable performance.[1]

(MAB) setting [Even-Dar et al., 2002, Jamieson and Nowak, 2014, Lai and Robbins, 1985]. One views each point in the medoid problem as an arm whose unknown parameter is its average distance to all the other points. Pulling an arm corresponds to evaluating the distance of that point to a randomly chosen point, which provides an estimate of the arm's unknown parameter. We leverage the extensive literature on multi-armed bandits to propose `Med-dit`, which is a variant of the Upper Confidence Bound (UCB) Algorithm [Lai and Robbins, 1985].

Like the other sampling based algorithms `RAND` and `TOPRANK`, `Med-dit` also aims to estimate the average distance of every point to the other points by evaluating its distance to a random subset of points. However, unlike these algorithms where each point receives a fixed amount of sampling decided *apriori*, the sampling in `Med-dit` is done *adaptively*. More specifically, `Med-dit` maintains confidence intervals for the average distances of all points and adaptively evaluates distances of only those points which could *potentially* be the medoid. Loosely speaking, points whose lower confidence bounds on the average distance are small form the set of points that could *potentially* be the medoid. As the algorithm proceeds, additional distance evaluations narrow the confidence intervals. This rapidly shrinks the set of points that could *potentially* be the medoid, enabling the algorithm to declare a medoid while evaluating a few distances.

[1] The computation of the true medoid here is also computationally prohibitive and is discussed in Section 4.

We mention that multi-armed bandits have been used to obtain randomised algorithms to reduce computational costs in problems with no inherent stochasticity before; for instance in the context of searching large state-spaces in Markovian Decision Problems like the Monte-Carlo tree search of Kocsis and Szepesvári [2006].

In Section 2 we delve more into the problem formulation and assumptions. In Section 3 we present `Med-dit` and analyze its performance. We extensively validate the performance of `Med-dit` empirically on two large-scale datasets: a single cell RNA-Seq gene expression dataset of 10xGenomics [2017], and the Netflix-prize dataset of Bennett et al. [2007] in Section 4.

2 PROBLEM FORMULATION

Consider n points $x_1, x_2, \cdots, x_n$ lying in some space $\mathcal{U}$ equipped with the distance function $d : \mathcal{U} \times \mathcal{U} \mapsto \mathbb{R}_+$. Note that we do not assume the triangle inequality or the symmetry for the distance function d. Therefore the analysis here encompasses directed graphs, or distances like Bregman divergence and squared Euclidean distance. We use the standard notation of $[n]$ to refer to the set $\{1, 2, \cdots, n\}$. Let $d_{i,j} \triangleq d(x_i, x_j)$ and let the average distance of a point x_i be

$$\mu_i \triangleq \frac{1}{n-1} \sum_{j \in [n]-\{i\}} d_{i,j}.$$

The medoid problem can be defined as follows.

Definition 1. *For a set of points $\mathcal{X} = \{x_1, x_2, \cdots, x_n\}$, the medoid is the point in $\mathcal{X}$ with the smallest average distance to other points.* [2] *Let x_{i^*} be the medoid. The index i^* and the average distance μ^* of the medoid are given by*

$$i^* \triangleq \operatorname*{argmin}_{i \in [n]} \mu_i, \qquad \mu^* \triangleq \min_{i \in [n]} \mu_i.$$

In the worst case over distances and points, we would need to evaluate $O(n^2)$ distances to compute the medoid. Consider the following example.

Example 1. *Consider a setting where there are n points. We pick a point i uniformly at random from $[n]$ and set all distances to point i to 0. For every other point $j \in [n] - \{i\}$, we pick a point k uniformly at random from $[n] - \{i, j\}$ and set $d_{j,k} = d_{k,j} = n$. As a result, each point apart from i has a distance of n to at least one point. Thus picking the point with distance 0 would require at least $O(n^2)$ distance evaluations. We note that this distance does not satisfy the triangle inequality, and thus is not a metric.*

[2] There may be multiple medoids in a set of points, but we restrict ourselves to the case where the medoid is unique in this manuscript for simplicity of exposition.

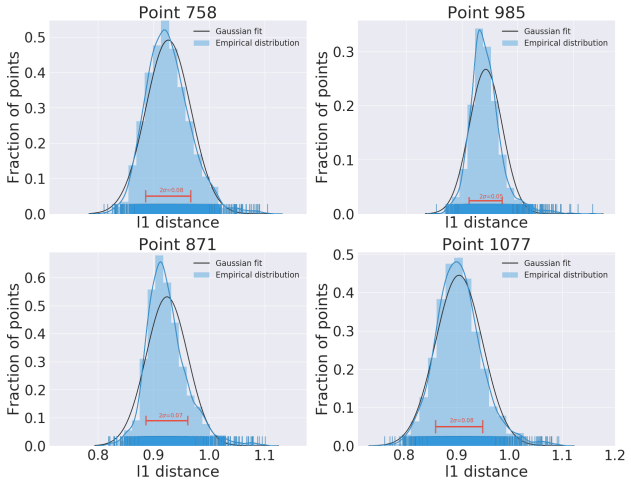

Figure 2: Histogram of ℓ_1 distance of 4 randomly chosen points (from 20,000 points). We note that the standard deviations are around 0.05.

The issue with Example 1 is that, the empirical distributions of distances of any point (apart from the true medoid) have heavy tails. However, such worse-case instances rarely arise in practice. Consider the following real-world example.

Example 2. *We consider the ℓ_1 distance between each pair of points in a cluster – 109,140 points in 27,998 dimensional probability simplex – taken from a single cell RNA-Seq dataset from 10xGenomics [2017]. Empirically the ℓ_1 distance for a given point follows a Gaussian-like distribution with a small variance, as shown in Figure 2. Modelling distances among high-dimensional points by a Gaussian distribution is also discussed in Linderman and Steinerberger [2017]. Clearly the distances do not follow a heavy-tailed distribution like Example 1.*

2.1 The Distance Sampling Model

Example 2 suggests that for any point x_i, we can estimate its average distance μ_i by the empirical mean of its distance from a subset of randomly sampled points. Let us formally state this. For any point x_i, let $\mathcal{D}_i = \{d_{i,j}, j \neq i\}$ be the set of distances associated with x_i. Define the random distance samples $D_{i,(1)}, D_{i,(2)}, \cdots, D_{i,(k)}$ to be i.i.d. *sampled with replacement* from $\mathcal{D}_i$. We note that

$$\mathbb{E}[D_{i,(j)}] = \mu_i, \ \forall i \in [n], j \in [k].$$

Hence the average distance μ_i can be estimated as

$$\hat{\mu}_i = \frac{1}{k} \sum_{i=1}^{k} D_{i,(k)}.$$

To capture the concentration property illustrated in Example 2, we assume the random variables $D_{i,(j)}$, i.e. sampling with replacement from $\mathcal{D}_i$, are σ-sub-Gaussian [3] for all i. While $D_{i,(j)}$ are trivially sub-gaussian because they are samples drawn from a finite set, we note that sub-Gaussianity parameter here would be

$$\sigma_i' = \frac{1}{2}\left(\max \mathcal{D}_i - \min \mathcal{D}_i\right).$$

Note that σ_i' could potentially scale with n. In fact in Example 1 it scales as $\Theta(n)$. However, we get around this by assuming that $D_{i,(j)}$ are σ-sub-Gaussian for some constant σ independent of n. We justify this by empirical observations on datasets such as that of Example 2. Further note that while practically implementing the algorithm, estimating σ_i' would require us to estimate the maximum distance of the ensemble. This problem seems to be as hard as the medoid problem.

2.2 Connection with Best-Arm Problem

For points whose average distance is close to that of the medoid, we need an accurate estimate of their average distance, and for points whose average distance is far away, coarse estimates suffice. Thus the problem reduces to adaptively choosing the number of distance evaluations for each point to ensure that on one hand, we have a good enough estimate of the average distance to compute the true medoid, while on the other hand, we minimise the total number of distance evaluations.

This problem has been addressed as the best-arm identification problem in the multi-armed bandit (MAB) literature (see the review article of Jamieson and Nowak [2014] for instance). In a typical setting, we have n arms. At the time $t = 0, 1, \cdots$, we decide to pull an arm $A_t \in [n]$, and receive a reward R_t with $\mathbb{E}[R_t] = \mu_{A_t}$. The goal is to identify the arm with the largest expected reward with high probability while pulling as few arms as possible.

The medoid problem can be formulated as a best-arm problem as follows: Let each point in the ensemble be an arm and let the average distance μ_i be the loss of the i-th arm. Medoid corresponds to the arm with the *smallest* loss, i.e best-arm. At each iteration t, pulling arm i is equivalent of sampling (with replacement) from $\mathcal{D}_i$ with expected value μ_i. As stated previosuly, the goal is to find the best arm (medoid) with as few pulls (distance computations) as possible.

[3] X is a σ-sub-Gaussian if $P(X > t) \leq 2e^{-\frac{t^2}{\sigma^2}}$.

3 ALGORITHM

We have n points $x_1, \cdots, x_n$. Recall that the average distance for point i is $\mu_i = \frac{1}{n-1}\sum_{j\neq i} d_{i,j}$. At iteration t of the algorithm, we evaluate a distance D_t to point $A_t \in [n]$. Let $T_i(t)$ be the number of distances of point i evaluated up to time t. We use a variant of the Upper Confidence Bound (UCB) [Lai and Robbins, 1985] algorithm to sample distance of a point i with another point chosen uniformly at random.

We compute the empirical mean and use it as an estimate of μ_i at time t, in the initial stages of the algorithm. More concretely when $T_i(t) < n$,

$$\hat{\mu}_i(t) \triangleq \frac{1}{T_i(t)} \sum_{1\leq\tau\leq t, A_\tau = i} D_\tau.$$

Next we recall from Section 2 that the sampled distances are independent and σ-sub-Gaussian (as points are sampled with replacement). Thus for all i, for all $\delta \in (0,1)$, with probability at least $1-\delta$, $\mu_i \in [\hat{\mu}_i(t) - C_i(t), \hat{\mu}_i(t) + C_i(t)]$, where

$$C_i(t) = \sqrt{\frac{2\sigma^2 \log\frac{2}{\delta}}{T_i(t)}}. \tag{1}$$

When $T_i(t) \geq n$, we compute the exact average distance of point i and set the confidence interval to zero.

Algorithm 1 describes the `Med-dit` algorithm.

Theorem 1. *For $i \in [n]$, let $\Delta_i = \mu_i - \mu^*$. If we pick $\delta = \frac{2}{n^3}$ in Algorithm 1, then with probability $1 - o(1)$, it returns the true medoid with the number of distance evaluations M such that,*

$$M \leq \sum_{i\in[n]} \left[2n \wedge \left(\frac{24\sigma^2}{\Delta_i^2}\log n\right)\right]. \tag{2}$$

Note that the term in the sum corresponding to i^* is $2n$ even though Δ_{i^*} is zero and that $x \wedge y = \min(x, y)$.

The proof of theorem 1 is a simple adapatation of the fairly standard proof of the number of pulls needed by the UCB algorithm and is thus relegated to Appendix 3. We assume that σ is known in the proof, whereas we estimate them in the empirical evaluations. See Section 4 for details.

If $\frac{\Delta_i}{\sigma}$ is $\Theta(1)$, then Theorem 1 gives that `Med-dit` takes $O(n \log n)$ distance evaluations. In addition, if one assumed a Gaussian prior on the average distance of points (as one would expect from Figure 3), then `Med-dit` would also take $O(n \log n)$ distance evaluations in expectation over the prior. See Appendix 2.

If we use the same assumption as `TOP-RANK`, *i.e.* the μ_i are i.i.d. Uniform $[c_0, c_1]$, then the number of distance evaluations needed by `Med-dit` is $O(n^{\frac{3}{2}} \log^{\frac{1}{2}} n)$,

Algorithm 1 `Med-dit`

Evaluate distances of each point to a randomly chosen point and build a $(1-\delta)$-confidence interval for the mean distance of each point i: $[\hat{\mu}_i(1) - C_i(1),\ \hat{\mu}_i(1) + C_i(1)]$.
while true do
  At iteration t, pick point A_t that minimises $\hat{\mu}_i(t-1) - C_i(t-1)$.
  if distances of point A_t are evaluated less than $n-1$ times **then**
    evaluate the distance of A_t to a randomly picked point and update the confidence interval of A_t.
  else
    Set $\hat{\mu}_i(t)$ to be the empirical mean of distances of point A_t by computing its distance to all $(n-1)$ other points and set $C_{A_t}(t) = 0$.
  end if
  if there exists a point x_{i^*} such that $\forall i \neq i^*$, $\hat{\mu}_{i^*}(t) + C_{i^*}(t) < \hat{\mu}_i(t) - C_i(t)$ **then**
    return x_{i^*}.
  end if
end while

compared to $O(n^{\frac{5}{3}} \log^{\frac{4}{3}} n)$ for `TOP-RANK`. Under these assumptions `RAND` takes $O(n^2)$ distance evaluations to return the true medoid.

Remark 1. *With a small modification to* `Med-dit`*, the sub-Gaussian assumption in Section 2 can be further relaxed to a finite variance assumption [Bubeck et al., 2013]. One the other hand, one could theoretically adapt* `Med-dit` *to other best-arm algorithms to find the medoid with $O(n \log\log n)$ distance computations at the sacrifice of a large constant [Karnin et al., 2013]. Refer to Appendix 4 for details.*

4 EMPIRICAL RESULTS

We empirically evaluate the performance of `Med-dit` on two real-world large-scale high-dimensional datasets: the Netflix-prize dataset by Bennett et al. [2007], and 10x Single Cell RNA-Seq dataset [10xGenomics, 2017].

We picked these large datasets because they have sparse high-dimensional feature vectors, and are at the throes of active efforts to cluster using non-Euclidean distances. In both these datasets, we use 1000 randomly chosen points to estimate the sub-Gaussianity parameter by cross-validation. We set $\delta = 1e\text{-}3$.

Running `trimed` and `TOPRANK` was computationally prohibitive due to the large size and high dimensionality ($\sim 20,000$) of the above two datasets. Therefore, we compare `Med-dit` only to `RAND`. We also ran `Med-dit` on datasets tested in Newling and Fleuret [2017] to compare it to `trimed` and `TOPRANK`.

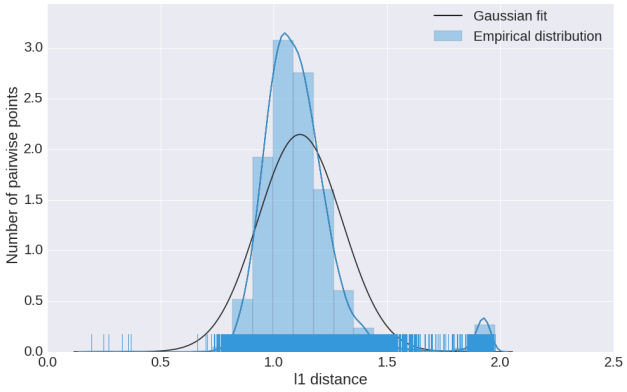

Figure 3: The distribution of mean distances of all the points in the 20,000-cell RNA-Seq dataset.

4.1 Single Cell RNA-Seq

Rapid advances in sequencing technology has enabled us to sequence DNA/RNA at a cell-by-cell basis and the single cell RNA-Seq datasets can contain up to a million cells [10xGenomics, 2017, Klein et al., 2015, Macosko et al., 2015, Zheng et al., 2017].

This technology involves sequencing an ensemble of single cells from a tissue, and obtaining the gene expressions corresponding to each cell. These gene expressions are subsequently used to cluster these cells in order to discover subclasses of cells which would have been hard to physically isolate and study separately. This technology therefore allows biologists to infer the diversity in a given tissue.

We note that tens of thousands gene expressions are measured in each cell, and million of cells are sequenced in each dataset. Therefore single cell RNA-Seq has presented us with a very important large-scale clustering problem with high dimensional data [Zheng et al., 2017]. Moreover, the distance metric of interest here would be ℓ_1 distance as the features of each point are probability distribution, Euclidean (or squared euclidean) distances do not capture the subtleties as discussed in Batu et al. [2000], Ntranos et al. [2016]. Further, very few genes are expressed by most cells and the data is thus sparse.

4.1.1 10xGenomics Mice Dataset

We test the performance of `Med-dit` on the single cell RNA-Seq dataset 10xGenomics [2017]. This dataset from 10xGenomics consists of 27,998 genes-expression of 1.3 million neurons cells from the cortex, hippocampus, and subventricular zone of a mouse brain. These

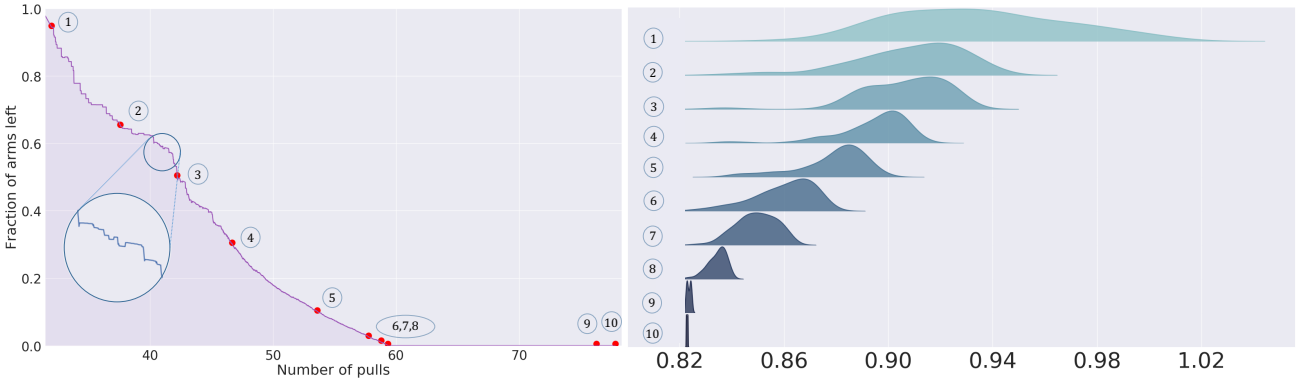

Figure 4: In the left panel, we show the number of points *under consideration* at different iterations in `Med-dit`. We note that though the number of points *under consideration* show a decreasing trend, they do not decrease monotonically. In the right panel, we show the distribution of the true distances among the arms that are *under consideration* at various snapshots of the algorithm. We note that both the mean and the variance of the distributions keeps decreasing. The last snapshot shows only the distance of the point declared a medoid.

are clustered into 60 clusters. Around 93% of the gene expressions in this dataset are zero.

We test `Med-dit` on subsets of this dataset of two sizes:

- small dataset - 20,000 cells (randomly chosen): We use this as we can compute the true medoid on this dataset by brute force and thus can compare the performance of `Med-dit` and `RAND`.
- large dataset - 109,140 cells (cluster 1) is the largest cluster in this dataset. We use the most commonly returned point as the true medoid for comparision. We note that this point is the same for both `Med-dit` and `RAND`, and has the smallest distance among the top 100 points returned in 1000 trials of both.

Performance Evaluation : We compare the performance of `RAND` and `Med-dit` in Figures 1 and 8 on the large and the small dataset respectively. We note that in both cases, `RAND` needs 7-10 times more distance evaluations to achieve 2% error rate than what `Med-dit` stops at (without an error in 1000 trials).

On the 109,140 cell dataset, we note that `Med-dit` stopped within 140 distance evaluations per arm in each of the 1000 trials (with 120 distance evaluations per arm on average) and never returned the wrong answer. `RAND` needs around 700 distance evaluations per point to obtain 2% error rate. The $20k$ user dataset is discussed in Appendix 1.

4.2 Recommendation Systems

With the explosion of e-commerce, recommending products to users has been an important avenue. Research into this took off with Netflix releasing the Netflix-prize dataset [Bennett et al., 2007].

Usually in the datasets involved here, we have either rating given by users to different products (movies, books, etc), or the items bought by different users. The task at hand is to recommend products to a user based on behaviour of similar users. Thus a common approach is to cluster similar users and to use trends in a cluster to recommend products.

We note that in such datasets, we typically have the behaviour of millions of users and tens of thousands of items. Thus recommendation systems present us with an important large-scale clustering problem in high dimensions [Daruru et al., 2009]. Further, most users buy (or rate) very few items on the inventory and the data is thus sparse. Moreover, the number of items bought (or rated) by users vary significantly and hence distance metrics that take this into account, like cosine distance and Jaccard distance, are of interest while clustering as discussed in Leskovec et al. [2014, Chapter 9].

4.2.1 Netflix-prize Dataset

We test the performance of `Med-dit` on the Netflix-prize dataset of Bennett et al. [2007]. This dataset from Netflix consists of ratings of 17,769 movies by 480,000 Netflix users. Only 0.21% of the entries in the matrix are non-zero. As discussed in Leskovec et al. [2014, Chapter 9], cosine distance is a popular metric considered here.

Similar to Section 4.1.1, we use a small and a large subset of the dataset of size 20,000 and size 100,000 respectively picked at random. For the former we compute the true medoid by brute force, while for the lat-

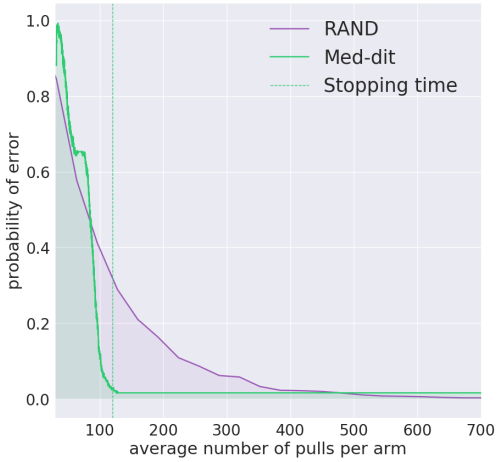

Figure 5: Large Netflix-prize dataset: The y-axis shows the probability that the estimated medoid does *not* correspond to the true medoid as a function of the number of pulls per arm. We note that `Med-dit` has a stopping condition while `RAND` does not. However we ignore the stopping condition for `Med-dit` here. `Med-dit` stops after 500 distance evaluations per point with 1% probability of error, while `RAND` takes around 500 distance evaluations to reach a 2% probability of error. The computation of the true medoid here is computationally prohibitive and for comparision we use the most commonly returned point as the true medoid (which was same for both Med-dit and RAND).

ter we use the most commonly returned point as the true medoid for comparision. We note that this point is the same for both `Med-dit` and `RAND`, and has the smallest distance among the top 100 points returned in 1000 trials of both.

Performance Evaluation: On the 100, 000 users on Netflix-prize dataset (Figure 5), we note that `Med-dit` stopped within 150 distance evaluations per arm in each of the 1000 trials (with 120 distance evaluations per arm on average) and returned the wrong answer only once. RAND needs around 500 distance evaluations per point to obtain 2% error rate. The $20k$ user dataset is discussed in Appendix 1.

4.3 Performance on Previous Datasets

We show the performance of `Med-dit` on datasets used for performance evaluation in prior works such as Newling and Fleuret [2017] — `BIRCH1`, `BIRCH2`, `BIRCH3` from Zhang et al. [1997]; `Europe` from Fränti et al. [2014]; `U-Sensor` and `D-Sensor` from Newling and Fleuret [2017]; `Eur-rail` from Meuser [2016]; Road network of Pennsylvania (`Pen-road`) from Leskovec and Krevl [2014]; `Gnutella` from Ripeanu et al. [2002]; `MNIST` from LeCun [1998]— in Table 1.

Dataset	n, d	`TR`	`tm`	`meddit` ($P_{success}$)
Birch 1	$100k, 2$	58k	2181	1436(.99)
Birch 2	$100k, 2$	66k	2195	2140(1.0)
Birch 3	$100k, 2$	61k	1495	1737(1.0)
Europe	$160k, 2$	176k	2862	3514(1.0)
U-sensor	$100k, G$	8314	1372	177(.64)
D-sensor	$100k, DG$	3659	862	205(.74)
Euro-rail	$46k, G$	35k	539	142(.96)
Pen-road	$1M, G$	216k	2633	120(.70)
Gnutella	$6.3k, G$	7043	6328	83(.99)
MNIST	$6.7k, 784$	7472	6514	91(1.0)

Table 1: ($d = \{G$: graph, DG: digraph $\}$, `TR`: `TOPRANK`, `tm`: `trimed`) This table shows the performance on real-world datasets picked from Newling and Fleuret [2017]. We note that the performance of `Med-dit` is not as good as `trimed` in low-dimensional datasets. For high-dimensional datasets `Med-dit` performs much better.

The number of distance evaluations for `TOPRANK` and `trimed` use the implementation of Newling [2017]. The number of distance evaluations for `Med-dit` is computed by averaging over 100 trials. The probability of success of `Med-dit` is also calculated over them and reported.

We note that `Med-dit` has quite low probability of success on graph datasets (like `U-Sensor` , `D-Sensor` and `Pen-road`), worse than that of `trimed`. This is due to the fact that the sub-Gaussianity assumption is violated in these datasets. On datasets of points in low-dimensional metric spaces (like `Europe` and the three `BIRCH` datasets), the performance of `Med-dit` and `trimed` are comparable. On datasets of points in high-dimensional metric spaces (like `MNIST`), `Med-dit` performs much better. We conjecture that this is due to the fact the distances in high dimensions are average of functions (over each dimension) and hence have Gaussian-like behaviour due to the Central Limit Theorem (as well as the delta method [Van der Vaart, 1998, Chapter 3]).

4.4 Empirical Scaling of number of distance evaluations

We subsampled the $100k$ single-cell RNA-Seq datasets and computed the number of distance evaluations `Med-dit` needed to compute the medoids on these. As seen in Figure 6, the scaling here is almost linear.

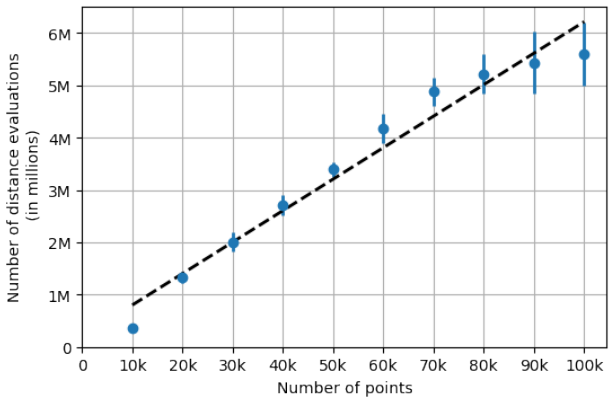

Figure 6: This shows the number of distance evaluations (shown in millions) for various random subsamplings of points from the 100,000 single cell RNA-Seq dataset over 30 trials for each sub-sampling. The scaling shows a reasonable linear fit.

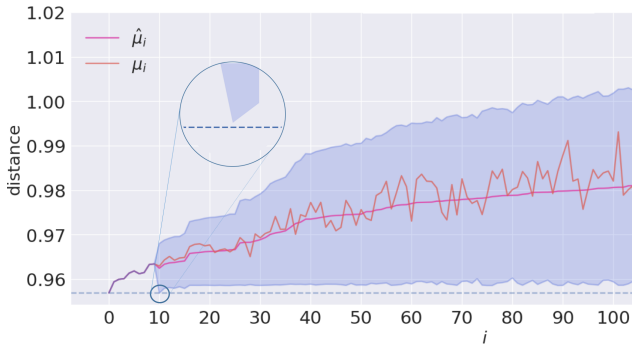

Figure 7: 99.99%-Confidence Intervals of the 150 points with smallest estimated mean distance in the 20,000 cell dataset at the termination of `Med-dit`. The true means are shown by the orange line, while the estimated means are shown by the pink line. We note that the distances of 9 points are computed to all other points. We also note that mean of the medoid is less than the 99.99%-lower confidence bound of all the other points.

4.5 Inner workings of `Med-dit`

Empirical running time: We computed the true mean distance μ_i, and variance σ, for all the points in the small dataset by brute force. Figure 3 shows the histogram of μ_i's. Using $\delta = 1e\text{-}3$ in Theorem 1, the theoretical average number of distance evaluations is 266. Empirically over 1000 experiments, the average number of distance computations is 80. This suggests that the theoretical analysis captures the essence of the algorithm.

Progress of `Med-dit`: We see that at any iteration of `Med-dit`, only the points whose lower confidence interval are smaller than the smallest upper confidence interval among all points have their distances evaluated. At any iteration, we say that such points are *under consideration*. We note that a point that goes out of consideration can be under consideration at a later iteration (this happens if the smallest upper confidence interval increases). We illustrate the fraction of points under consideration at different iterations (indexed by the number of distance evaluations per point) in the left panel of Figure 4. We pick 10 snapshots of the algorithm and illustrate the true distance distributions of all points *under consideration* at that epoch in the right panel of Figure 4. We note that both the mean and the standard deviation of these distributions decrease as the algorithm progresses.[4]

Confidence intervals at stopping time: `Med-dit` seems to compute the mean distance of a few points exactly. These are usually points with the mean distance close to that of the medoid. The points whose mean distances are farther away from that of the medoid have fewer distances evaluated to have the confidence intervals rule them out as the medoid. The 99.99% confidence interval of the top 150 points in a run of the experiment are shown in Figure 7 for the 20,000 cell RNA-Seq dataset. This allows the algorithm to save on distance computations while returning the correct answer.

4.6 Practice

Speed-accuracy tradeoff: Setting the value of δ between 0.01 to 0.1 will result in smaller confidence intervals around the estimates $\hat{\mu}_i$, which will ergo reduce the number of distance evaluations. One could also run `Med-dit` for a fixed number of iterations and return the point with the smallest mean estimate. Both methods improve the running time at the cost of accuracy. Practically, the first method has a better trade-off between accuracy and running time whereas the second method has a deterministic stopping time.

Computing σ: Overestimating the value of σ (likewise confidence intervals) increases running time whereas underestimating σ decreases the accuracy. In practice, σ_i can be estimated without any additional computational overhead by maintaining a (running) estimate of the mean and the second moment for every arm i.

[4]When the number of points *under consideration* is more than 2000, we draw the distribution using the true distances of 200 random samples.

References

10xGenomics. *1.3 Million Brain Cells from E18 Mice*. 10x Genomics, 2017. available at `https://support.10xgenomics.com/single-cell-gene-expression/datasets/1M_neurons`.

T. Batu, L. Fortnow, R. Rubinfeld, W. D. Smith, and P. White. Testing that distributions are close. In *Foundations of Computer Science, 2000. Proceedings. 41st Annual Symposium on*, pages 259–269. IEEE, 2000.

J. Bennett, S. Lanning, et al. The netflix prize. In *Proceedings of KDD cup and workshop*, volume 2007, page 35. New York, NY, USA, 2007.

S. Bubeck, N. Cesa-Bianchi, and G. Lugosi. Bandits with heavy tail. *IEEE Transactions on Information Theory*, 59(11):7711–7717, 2013.

O. Catoni et al. Challenging the empirical mean and empirical variance: a deviation study. In *Annales de l'Institut Henri Poincaré, Probabilités et Statistiques*, volume 48, pages 1148–1185. Institut Henri Poincaré, 2012.

S. Daruru, N. M. Marin, M. Walker, and J. Ghosh. Pervasive parallelism in data mining: dataflow solution to co-clustering large and sparse netflix data. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1115–1124. ACM, 2009.

D. Eppstein and J. Wang. Fast approximation of centrality. *Graph Algorithms and Applications 5*, 5:39, 2006.

E. Even-Dar, S. Mannor, and Y. Mansour. Pac bounds for multi-armed bandit and markov decision processes. In *International Conference on Computational Learning Theory*, pages 255–270. Springer, 2002.

E. Even-Dar, S. Mannor, and Y. Mansour. Action elimination and stopping conditions for the multi-armed bandit and reinforcement learning problems. *Journal of machine learning research*, 7(Jun):1079–1105, 2006.

P. Fränti, M. Rezaei, and Q. Zhao. Centroid index: cluster level similarity measure. *Pattern Recognition*, 47(9):3034–3045, 2014.

C. A. Hoare. Algorithm 65: find. *Communications of the ACM*, 4(7):321–322, 1961.

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

Z. Karnin, T. Koren, and O. Somekh. Almost optimal exploration in multi-armed bandits. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pages 1238–1246, 2013.

L. Kaufman and P. Rousseeuw. *Clustering by means of medoids*. North-Holland, 1987.

A. M. Klein, L. Mazutis, I. Akartuna, N. Tallapragada, A. Veres, V. Li, L. Peshkin, D. A. Weitz, and M. W. Kirschner. Droplet barcoding for single-cell transcriptomics applied to embryonic stem cells. *Cell*, 161(5):1187–1201, 2015.

L. Kocsis and C. Szepesvári. Bandit based monte-carlo planning. In *European conference on machine learning*, pages 282–293. Springer, 2006.

T. L. Lai and H. Robbins. Asymptotically efficient adaptive allocation rules. *Advances in applied mathematics*, 6(1):4–22, 1985.

Y. LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

J. Leskovec and A. Krevl. SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/data`, June 2014.

J. Leskovec, A. Rajaraman, and J. D. Ullman. *Mining of massive datasets*. Cambridge university press, 2014.

G. C. Linderman and S. Steinerberger. Clustering with t-sne, provably. *arXiv preprint arXiv:1706.02582*, 2017.

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

E. Z. Macosko, A. Basu, R. Satija, J. Nemesh, K. Shekhar, M. Goldman, I. Tirosh, A. R. Bialas, N. Kamitaki, E. M. Martersteck, et al. Highly parallel genome-wide expression profiling of individual cells using nanoliter droplets. *Cell*, 161(5):1202–1214, 2015.

J. 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.

M. Meuser. Europe railways. `http://www.mapcruzin.com/free-europe-arcgis-maps-shapefiles.htm`, 2016. [Online; accessed 24-December-2017].

J. Newling. Implementation of trimed, toprand, and rand. `https://github.com/idiap/trimed`, 2017. [Online; accessed 24-December-2017].

J. Newling and F. Fleuret. A sub-quadratic exact medoid algorithm. *Proceedings of the 20th Inter-*

national Conference on Artificial Intelligence and Statistics, 2017.

V. Ntranos, G. M. Kamath, J. M. Zhang, L. Pachter, and D. N. Tse. Fast and accurate single-cell rna-seq analysis by clustering of transcript-compatibility counts. *Genome biology*, 17(1):112, 2016.

K. Okamoto, W. Chen, and X.-Y. Li. Ranking of closeness centrality for large-scale social networks. In *International Workshop on Frontiers in Algorithmics*, pages 186–195. Springer, 2008.

M. Ripeanu, I. Foster, and A. Iamnitchi. Mapping the gnutella network: Properties of large-scale peer-to-peer systems and implications for system design. *arXiv preprint cs/0209028*, 2002.

H. Steinhaus. Sur la division des corp materiels en parties. *Bull. Acad. Polon. Sci*, 1(804):801, 1956.

C. Sudlow, J. Gallacher, N. Allen, V. Beral, P. Burton, J. Danesh, P. Downey, P. Elliott, J. Green, M. Landray, et al. Uk biobank: an open access resource for identifying the causes of a wide range of complex diseases of middle and old age. *PLoS medicine*, 12 (3):e1001779, 2015.

A. W. Van der Vaart. *Asymptotic statistics*, volume 3. Cambridge university press, 1998.

T. Zhang, R. Ramakrishnan, and M. Livny. Birch: A new data clustering algorithm and its applications. *Data Mining and Knowledge Discovery*, 1(2):141–182, 1997.

G. X. Zheng, J. M. Terry, P. Belgrader, P. Ryvkin, Z. W. Bent, R. Wilson, S. B. Ziraldo, T. D. Wheeler, G. P. McDermott, J. Zhu, et al. Massively parallel digital transcriptional profiling of single cells. *Nature communications*, 8:14049, 2017.