IDIAP RESEARCH REPORT

A SUB-QUADRATIC EXACT MEDOID ALGORITHM

James Newling[a] Francois Fleuret

Idiap-RR-19-2017

JULY 2017

[a]Idiap

Centre du Parc, Rue Marconi 19, P.O. Box 592, CH - 1920 Martigny
T +41 27 721 77 11 F +41 27 721 77 12 info@idiap.ch www.idiap.ch

A Sub-Quadratic Exact Medoid Algorithm

James Newling
Idiap Research Institute & EPFL

François Fleuret
Idiap Research Institute & EPFL

Abstract

We present a new algorithm `trimed` for obtaining the *medoid* of a set, that is the element of the set which minimises the mean distance to all other elements. The algorithm is shown to have, under certain assumptions, expected run time $O(N^{\frac{3}{2}})$ in $\mathbb{R}^d$ where N is the set size, making it the first sub-quadratic exact medoid algorithm for $d > 1$. Experiments show that it performs very well on spatial network data, frequently requiring two orders of magnitude fewer distance calculations than state-of-the-art approximate algorithms. As an application, we show how `trimed` can be used as a component in an accelerated K-medoids algorithm, and then how it can be relaxed to obtain further computational gains with only a minor loss in cluster quality.

1 Introduction

A popular measure of the centrality of an element of a set is its mean distance to all other elements. In network analysis, this measure is referred to as *closeness centrality*, we will refer to it as *energy*. Given a set $\mathcal{S} = \{x(1), \dots, x(N)\}$ the energy of element $i \in \{1, \dots, N\}$ is thus given by,

$$E(i) = \frac{1}{N} \sum_{j \in \{1,\dots,N\}} \text{dist}(x(i), x(j)).$$

An element in $\mathcal{S}$ with minimum energy is referred to as a *1-median* or a *medoid*. Without loss of generality, we will assume that $\mathcal{S}$ contains a unique medoid. The problem of determining the medoid of a set arises in the contexts of clustering, operations research, and network analysis. In clustering, the Voronoi iteration K-medoids algorithm (Hastie et al., 2001; Park and Jun, 2009) requires determining the medoid of each of K clusters at each iteration. In operations research, the facility location problem requires placing one or several facilities so as to minimise the cost of connecting to clients. In network analysis, the medoid may represent an influential person in a social network, or the most central station in a rail network.

1.1 Medoid algorithms and our contribution

A simple algorithm for obtaining the medoid of a set of N elements computes the energy of all elements and selects the one with minimum energy, requiring $\Theta(N^2)$ time. In certain settings $\Theta(N)$ algorithms exist, such as in 1-D where the problem is solved by Quickselect (Hoare, 1961), and more generally on trees. However, no general purpose $o(N^2)$ algorithm exists. An example illustrating the impossibility of such an algorithm is presented in Supplementary Material B (SM-A). Related to finding the medoid of a set is finding the *geometric median*, which in vector spaces is defined as the point in the vector space with minimum energy. The relationship between the two problems is discussed in §2.1.

Much work has been done to develop approximate algorithms in the context of network analysis. The `RAND` algorithm of Eppstein and Wang (2004) can be used to estimate the energy of all nodes in a graph. The accuracy of `RAND` depends on the diameter of the network, which motivated Cohen et al. (2014) to use pivoting to make `RAND` more effective for large diameter networks. The work most closely related to ours is that of Okamoto et al. (2008), where `RAND` is adapted to the task of finding the k lowest energy nodes, $k = 1$ corresponding to the medoid problem. The resulting `TOPRANK` algorithm of Okamoto et al. (2008) has run time $\tilde{O}(N^{5/3})$ under certain assumptions, and returns the medoid with probability $1 - O(1/N)$, that is *with high probability* (w.h.p.). Note that only their run time result requires any assumption, obtaining the medoid w.h.p. is guaranteed. `TOPRANK` is discussed in §2.2.

In this paper we present an algorithm which has ex-

Proceedings of the 20^{th} International Conference on Artificial Intelligence and Statistics (AISTATS) 2017, Fort Lauderdale, Florida, USA. JMLR: W&CP volume 54.

pected run time $O(N^{3/2})$ under certain assumptions and always returns the medoid. In other words, we present an exact medoid algorithm with improved complexity over the state-of-the-art approximate algorithm, `TOPRANK`. We show through experiments that the new algorithm works well for low-dimensional data in $\mathbb{R}^d$ and for spatial network data. Our new medoid algorithm, which we call `trimed`, uses the triangle inequality to quickly eliminate elements which cannot be the medoid. The $O(N^{3/2})$ run time follows from the surprising result that all but $O(N^{1/2})$ elements can be eliminated in this way.

The complexity bound on expected run time which we derive contains a term which grows exponentially in dimension d, and experiments show that in very high dimensions `trimed` often ends up computing $O(N^2)$ distances.

1.2 K-medoids algorithms and our contribution

The K-medoids problem is to partition a set into K clusters, so as to minimise the sum over elements of dissimilarites with their nearest medoids. That is, to choose $\mathcal{M} = \{m(1), \dots, m(K)\} \subset \{1, \dots, N\}$ to minimise,

$$\mathcal{L}(\mathcal{M}) = \sum_{i=1}^{N} \min_{k \in \{1,\dots,K\}} \text{diss}(x(i), x(m(k))).$$

We focus on the special case where the dissimilarity is a distance (diss = dist), which is still more general than K-means which only applies to vector spaces. K-medoids is used in bioinformatics where elements are genetic sequences or gene expression levels (Chipman et al., 2003) and has been applied to clustering on graphs (Rattigan et al., 2007). In machine vision, K-medoids is often preferred, as a medoid is more easily interpretable than a mean (Frahm et al., 2010).

The K-medoids problem is NP-hard, but there exist approximation algorithms. The Voronoi iteration algorithm, appearing in Hastie et al. (2001) and later in Park and Jun (2009), consists of alternating between updating medoids and assignments, much in the same way as Lloyd's algorithm works for the K-means problem. We will refer to it as `KMEDS`, and to Lloyd's K-means algorithm as `lloyd`.

One significant difference between `KMEDS` and `lloyd` is that the computation of a medoid is quadratic in the number of elements per cluster whereas the computation of a mean is linear. By incorporating our new medoid algorithm into `KMEDS`, we break the quadratic dependency of `KMEDS`, bringing it closer in performance to `lloyd`. We also show how ideas for accelerating `lloyd` presented in Elkan (2003) can be used in `KMEDS`.

It should be noted that algorithms other than `KMEDS` have been proposed for finding approximate solutions to the K-medoids problem, and have been shown to be very effective in Newling and Fleuret (2016b). These include `PAM` and `CLARA` of Kaufman and Rousseeuw (1990), and `CLARANS` of Ng et al. (2005). In this paper we do not compare cluster qualities of previous algorithms, but focus on accelerating the `lloyd` equivalent for K-medoids as a test setting for our medoid algorithm `trimed`.

2 Previous works

2.1 A related problem: the geometric median

A problem closely related to the medoid problem is the geometric median problem. In the vector space $\mathcal{V}$ the geometric median, assuming it is unique, is defined as,

$$g(\mathcal{S}) = \arg\min_{v \in \mathcal{V}} \left(\sum_{x \in \mathcal{S}} \|v - x\| \right). \tag{1}$$

While the medoid of a set is defined in any space with a distance measure, the geometric median is specific to vector spaces, where addition and scalar multiplication are defined. The convexity of the objective function being minimised in (1) has enabled the development of fast algorithms. In particular, Cohen et al. (2016) present an algorithm which obtains an estimate for the geometric median with relative error $1 + O(\epsilon)$ with complexity $O(Nd \log^3(\frac{N}{\epsilon}))$ in $\mathbb{R}^d$. In $\mathbb{R}^d$, one may hope that such an algorithm can be converted into an exact medoid algorithm, but it is not clear how to do this.

Thus, while it may be possible that fast geometric median algorithms can provide inspiration in the development of medoid algorithms, they do not work out of the box. Moreover, geometric median algorithms cannot be used for network data as they only work in vector spaces, thus they are useless for the spatial network datasets which we consider in §5.

2.2 Medoid Algorithms : `TOPRANK` and `TOPRANK2`

In Eppstein and Wang (2004), the `RAND` algorithm for estimating the energy of all elements of a set $\mathcal{S} = \{x(1), \dots, x(N)\}$ is presented. While `RAND` is presented in the context of graphs, where the N elements are nodes of an undirected graph and the metric is shortest path length, it can equally well be applied to any set endowed with a distance. The simple idea of `RAND` is to estimate the energy of each element from a sample of *anchor* nodes I, so that for $j \in \{1, \dots, N\}$,

$$\hat{E}(j) = \frac{1}{|I|} \sum_{i \in I} \text{dist}(x(j), x(i)).$$

An elegant feature of RAND in the context of sparse graphs is that Dijkstra's algorithm needs only be run from anchor nodes $i \in I$, and not from every node. The key result of Eppstein and Wang (2004) is the following. Suppose that $\mathcal{S}$ has diameter Δ, that is

$$\Delta = \max_{(i,j)\in\{1,\dots,N\}^2} \text{dist}(x(i), x(j)),$$

and let $\epsilon > 0$ be some error tolerance. If I is of size $\Omega(\log(N)/\epsilon)$, then $\mathbb{P}(|E(j) - \hat{E}(j)| > \epsilon\Delta)$ is $O\left(\frac{1}{N^2}\right)$ for all $j \in \{1,\dots,N\}$. Using the union bound, this means there is a $O\left(\frac{1}{N}\right)$ probability that at least one energy estimate is off by more than $\epsilon\Delta$, and so we say that *with high probability* (w.h.p.) all errors are less than $\epsilon\Delta$.

RAND forms the basis of the TOPRANK algorithm of Okamoto et al. (2008). Whereas RAND w.h.p. returns an element which has energy within ϵ of the minimum, TOPRANK is designed to w.h.p. return the true medoid. In motivating TOPRANK, Okamoto et al. (2008) observe that the expected difference between consecutively ranked energies is $O(\Delta/N)$, and so if one wishes to correctly rank all nodes, one needs to distinguish between energies at a scale $\epsilon = \Delta/N$, for which the result of Eppstein and Wang (2004) dictates that $\Theta(N \log N)$ anchor elements are required with RAND, which is more elements than $\mathcal{S}$ contains. However, to obtain just the highest ranked node should require less information than obtaining a full ranking of nodes, and it is to this task that TOPRANK is adapted.

The idea behind TOPRANK is to accurately estimate only the energies of promising elements. The algorithm proceeds in two passes, where in the first pass promising elements are earmarked. Specifically, the first pass runs RAND with $N^{2/3}\log^{1/3}(N)$ anchor elements to obtain $\hat{E}(i)$ for $i \in \{1,\dots,N\}$, and then discards elements whose $\hat{E}(i)$ lies below threshold τ given by,

$$\tau = \min_{j\in\{1,\dots,N\}} \hat{E}(j) + 2\hat{\Delta}\alpha' \left(\frac{\log n}{n}\right)^{\frac{1}{3}}, \qquad (2)$$

where $\hat{\Delta}$ is an upper bound on Δ obtained from the anchor nodes, and α' is some constant satisfying $\alpha' > 1$. The second pass computes the true energy of the undiscarded elements, returning the one with lowest true energy. Note that a smaller α' value results in a lower (better) threshold, we discuss this point further in SM-C.

To obtain run time guarantees, TOPRANK requires that the distribution of node energies is non-decreasing near to the minimum, denoted by E^*. More precisely, letting f_E be the probability distribution of energies, the algorithms require the existence of $\epsilon > 0$ such that,

$$E^* \leq \tilde{e} < e < E^* + \epsilon \implies f_E(\tilde{e}) \leq f_E(e). \qquad (3)$$

If assumption 3 holds, then the run time is $\tilde{O}(N^{\frac{5}{3}})$. A second algorithm presented in Okamoto et al. (2008) is TOPRANK2, where the anchor set I is grown incrementally until some heuristic criterion is met. There is no runtime guarantee for TOPRANK2, although it has the potential to run much faster than TOPRANK under favourable conditions. Pseudocode for RAND, TOPRANK and TOPRANK2 is presented in SM-C.

2.3 K-medoids algorithm : KMEDS

The Voronoi iteration algorithm, which we refer to as KMEDS, is similar to lloyd, the main difference being that cluster medoids are computed instead of cluster means. It has been desribed in the literature at least twice, once in Hastie et al. (2001) and then in Park and Jun (2009), where a novel initialisation scheme is developed. Pseudocode is presented in SM-B.

All N^2 distances are computed and stored upfront with KMEDS. Then, at each iteration, KN comparisons are made during assignment and $\Omega(N^2/K)$ additions are made during medoid update. The initialisation scheme of KMEDS requires all N^2 distances. Each iteration of KMEDS requires retrieving at least $\max\left(KN, N^2/K\right)$ distinct distances, as can be shown by assuming balanced clusters.

As an alternative to computing all distances upfront, one could store per-cluster distance matrices which get updated on-the fly when assignments change. Using such an approach, the best one could hope for would be $\max\left(KN, N^2/K\right)$ distance calculations and $\Theta(N^2/K)$ memory. If one were to completely forego storing distances in memory and calculate distances only when needed, the number of distance calculations would be at least $r(KN + N^2/K)$, where r is the number of iterations.

The initialisation scheme of Park and Jun (2009) selects K well centered elements as initial medoids. This goes against the general wisdom for K-means initialisation, where centroids are initialised to be well separated (Arthur and Vassilvitskii, 2007). While the new scheme of Park and Jun (2009) performs well on a limited number of small 2-D datasets, we show in § 3 that in general uniform initialisation performs as well or better.

3 Our new medoid algorithm : trimed

We present our new algorithm, trimed, for determining the medoid of set $\mathcal{S} = \{x(1),\dots,x(N)\}$. Whereas the approach with TOPRANK is to empirically *estimate* $E(i)$ for $i \in \{1,\dots,N\}$, the approach with trimed, presented as Alg. 1, is to *bound* $E(i)$. When trimed terminates, an index $m^* \in \{1,\dots,N\}$ has been de-

termined, along with lower bounds $l(i)$ for all $i \in \{1,\dots,N\}$, such that $E(m^*) \le l(i) \le E(i)$, and thus $x(m^*)$ is the medoid. The bounding approach uses the triangle inequality, as depicted in Figure 1.

Algorithm 1 The `trimed` algorithm for computing the medoid of $\{x(1),\dots,x(N)\}$.

1: $l \leftarrow \underline{0}_N$ // lower bounds on energies, maintained such that $l(i) \le E(i)$ and initialised as $l(i) = 0$.
2: $m^{cl}, E^{cl} \leftarrow -1, \infty$ // index of best medoid candidate found so far, and its energy.
3: **for** $i \in$ `shuffle`$(\{1,\dots,N\})$ **do**
4: **if** $l(i) < E^{cl}$ **then**
5: **for** $j \in \{1,\dots,N\}$ **do**
6: $d(j) \leftarrow \mathrm{dist}(x(i), x(j))$
7: **end for**
8: $l(i) \leftarrow \frac{1}{N}\sum_{j=1}^{N} d(j)$ // set $l(i)$ to be tight, that is $l(i) = E(i)$.
9: **if** $l(i) < E^{cl}$ **then**
10: $m^{cl}, E^{cl} \leftarrow i, l(i)$
11: **end if**
12: **for** $j \in \{1,\dots,N\}$ **do**
13: $l(j) \leftarrow \max(l(j), |l(i) - d(j)|)$ // using $E(i)$ and $\mathrm{dist}(x(i), x(j))$ to possibly improve bound on $E(j)$.
14: **end for**
15: **end if**
16: **end for**
17: $m^*, E^* \leftarrow m^{cl}, E^{cl}$
18: **return** $x(m^*)$

The algorithm `trimed` iterates through the N elements of $\mathcal{S}$. Each time a new element with energy lower than the current lowest energy (E^{cl}) is found, the index of the current best medoid (m^{cl}) is updated (line 10). Lower bounds on energies are used to quickly eliminate poor medoid candidates (line 4). Specifically, if lower bound $l(i)$ on the energy of element i is greater than or equal to E^{cl}, then i is eliminated. If the bound test fails to eliminate element i, then it is *computed*, that is, all distances to element i are computed (line 6). The computed distances are used to potentially improve lower bounds for all elements (line 13). Theorem 3.1 states that `trimed` finds the medoid. The proof relies on showing that lower bounds remain consistent when updated (line 13).

The algorithm is very straightforward to implement, and requires only two additional floating point values per datapoint: for sample i, one for $l(i)$ and one for $d(i)$. Computing either all or no distances from a sample makes particularly good sense for network data, where computing all distances to a single node is efficiently performed using Dijkstra's algorithm.

Theorem 3.1. *`trimed` returns the medoid of set $\mathcal{S}$.*

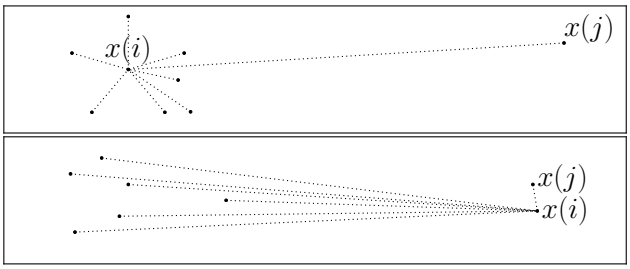

Figure 1: Using the inequality $E(j) \ge |E(i) - \mathrm{dist}(x(i), x(j))|$ to eliminate $x(j)$ as a medoid candidate. Computed element $x(i)$ with energy $E(i) \ge E^{cl}$ is used as a pivot to lower bound $E(j)$. The two cases where the inequality is effective are when (case 1, above) $\mathrm{dist}(x(i), x(j)) - E(i) \ge E^{cl}$ and (case 2, below) $E(i) - \mathrm{dist}(x(i), x(j)) \ge E^{cl}$, as both lead to $E(j) \ge E^{cl}$ which eliminates $x(j)$ as a medoid candidate.

Proof. We need to prove that $l(j) \le E(j)$ for all $j \in \{1,\dots,N\}$ at all iterations of the algorithm. Clearly, as $l(j) = 0$ at initialisation, we have $l(j) \le E(j)$ at initialisation. $E(j)$ does not change, and the only time that $l(j)$ may change is on line 13, where we need to check that $|l(i) - d(j)| \le E(j)$. At line 13, $l(i) = E(i)$ from line 8, and $d(j) = \mathrm{dist}(x(i), x(j))$, so at line 13 we are effectively checking that $|E(i) - \mathrm{dist}(x(i), x(j))| \le E(j)$. But this is a simple consequence of the triangle inequality, as we now show. Using the definition, $E(j) = \frac{1}{N}\sum_{l=1}^{N} \mathrm{dist}(x(l), x(j))$, we have on the one hand,

$$\begin{aligned} E(j) &\ge \frac{1}{N}\sum_{l=1}^{N} \mathrm{dist}(x(l), x(i)) - \mathrm{dist}(x(i), x(j)) \\ &\ge E(i) - \mathrm{dist}(x(i), x(j)), \end{aligned} \tag{4}$$

and on the other hand,

$$\begin{aligned} E(j) &\ge \frac{1}{N}\sum_{l=1}^{N} \mathrm{dist}(x(i), x(j)) - \mathrm{dist}(x(l), x(i)) \\ &\ge \mathrm{dist}(x(i), x(j)) - E(i). \end{aligned} \tag{5}$$

Combining (4) and (5) we obtain the required inequality $|E(i) - \mathrm{dist}(x(i), x(j))| \le E(j)$. □

The bound test (line 4) becomes more effective at later iterations, for two reasons. Firstly, whenever an element is computed, the lower bounds of other samples may increase. Secondly, E^{cl} will decrease whenever a better medoid candidate is found. The main result of this paper, presented as Theorem 3.2, is that in $\mathbb{R}^d$ the expected number of computed elements is $O(N^{\frac{1}{2}})$ under some weak assumptions. We show in §5 that

the $O(N^{\frac{1}{2}})$ result holds even in settings where the assumptions are not valid or relevent, such as for network data.

The shuffle on line 3 is performed to avoid w.h.p. pathological orderings, such as when elements are ordered in descending order of energy which would result in all N elements being computed.

Theorem 3.2. *Let $\mathcal{S} = \{x(1), \dots, x(N)\}$ be a set of N elements in $\mathbb{R}^d$, drawn independently from probability distribution function f_X. Let the medoid of $\mathcal{S}$ be $x(m^*)$, and let $E(m^*) = E^*$. Suppose that there exist strictly positive constants ρ, δ_0 and δ_1 such that for any set size N with probability $1 - O(1/N)$*

$$x \in \mathcal{B}_d(x(m^*), \rho) \implies \delta_0 \leq f_X(x) \leq \delta_1, \qquad (6)$$

where $\mathcal{B}_d(x, r) = \{x' \in \mathbb{R}^d \ : \ \|x' - x\| \leq r\}$. Let $\alpha > 0$ be a constant (independent of N) such that with probability $1 - O(1/N)$ all $i \in \{1, \dots, N\}$ satisfy,

$$x(i) \in \mathcal{B}_d(x(m^*), \rho) \implies E(i) - E^* \geq \alpha \|x(i) - x(m^*)\|^2. \qquad (7)$$

Then, the expected number of elements computed by `trimed` *is $O\left(\left(V_d[1]\delta_1 + d\left(\frac{4}{\alpha}\right)^d\right) N^{\frac{1}{2}}\right)$, where $V_d[1] = \pi^{\frac{d}{2}}/(\Gamma(\frac{d}{2}+1))$ is the volume of $\mathcal{B}_d(0,1)$.*

3.1 On the assumptions in Theorem 3.2

The assumption of constants ρ, δ_0 and δ_1 made in Theorem 3.2 is weak, and only pathological distributions might fail it, as we now discuss. For the assumptions to fail requires that f_X vanishes or diverges at the distribution medoid. Any reasonably behaved distribution does not have this behaviour, as illustrated in Figure 2. The constant α is a strong convexity constant. The existence of $\alpha > 0$ is guaranteed by the existence of ρ, δ_0 and δ_1, as the mean of a sum of uniformly spaced cones converges to a quadratic function. This is illustrated in 1-D in Figure 5 in SM-G, but holds true in any dimension.

Note that the assumptions made are on the distribution f_X, and not on the data itself. This must be so in order to prove complexity results in N.

3.2 Sketch of proof of Theorem 3.2

We now sketch the proof of Theorem 3.2, showing how (6) and (7) are used. A full proof is presented in SM-G. Firstly, let the index of the first element after the shuffle on line 3 be i'. Then, no elements beyond radius $2E(i')$ of $x(i')$ will subsequently be computed, due to type 1 eliminations (see Figure 1). Therefore, all computed elements are contained within $\mathcal{B}_d(x(i'), 2E(i'))$.

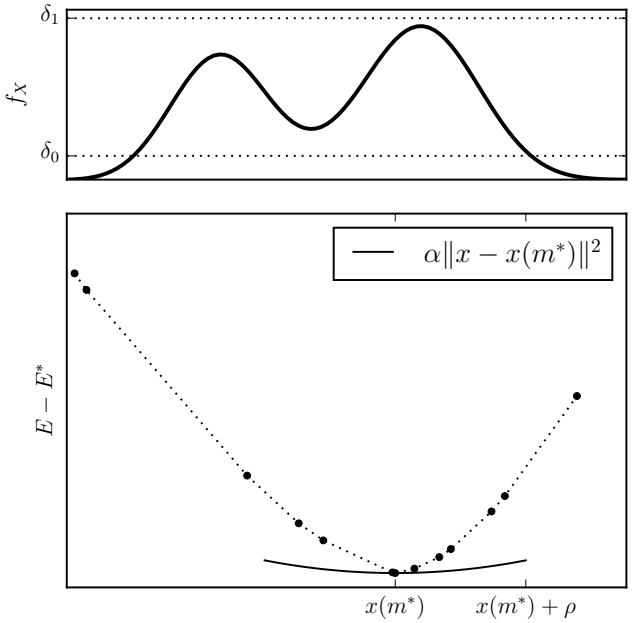

Figure 2: Illustration in 1-D of the constants used in Theorem 3.2. Above, δ_0 and δ_1 bound the probability density function in a region containing the distribution medoid. Below, the energy of samples grows quadratically around the medoid $x(m^*)$. The energy E is a sum of cones centered on samples, which is approximately quadratic unless f_X vanishes or explodes, guaranteeing the existence of $\alpha > 0$ required in Theorem 3.2.

Next, notice that once an element $x(i)$ has been computed in `trimed`, no elements in the ball $\mathcal{B}_d(x(i), E(i) - E^{cl})$ will subsequently be computed, due to type 2 eliminations (see Figure 1). We refer to such a ball as an *exclusion ball*. By upper bounding the number of exclusion balls contained in $\mathcal{B}_d(x(i'), 2E(i'))$ using a volumetric argument, we can obtain a bound on the number of computed elements, but obtaining such an upper bound requires that the radii of exclusion ball $E(i) - E^{cl}$ be bounded below by a strictly positive value. However, by using a volumetric argument only beyond a certain positive radius of the medoid (a radius $N^{-1/2d}$), we have $\alpha > 0$ in (15) which provides a lower bound on exclusion ball radii, assuming $E^{cl} \approx E^*$. Using δ_0 we can show that E^{cl} approaches E^* sufficiently fastsufficiently fast to validate the approximation $E^{cl} \approx E^*$.

It then remains to count the number of computed elements within radius $N^{-1/2d}$ of the medoid. One cannot find a strict upper bound here, but using the boundedness of f_X provided by δ_1, we have w.h.p. that the number of elements computed within $N^{-1/2d}$ is $O(\delta_1 N^{1/2})$, as the volume of a sphere scales as the d'th power of its radius.

4 Our accelerated K-medoids algorithm : `trikmeds`

We adapt our new medoid algorithm `trimed` and borrow ideas from Elkan (2003) to show how `KMEDS` can be accelerated. We abandon the initial N^2 distance calculations, and only compute distances when necessary. The accelerated version of `lloyd` of Elkan (2003) maintains KN bounds on distances between points and centroids, allowing a large proportion of distance calculations to be eliminated. We use this approach to accelerate assignment in `trikmeds`, incurring a memory cost $O(KN)$. By adopting the algorithm of Newling and Fleuret (2016a) or that of Hamerly (2010), the memory overhead can be reduced to $O(N)$. We accelerate the medoid update step by adapting `trimed`, reusing lower bounds between iterations, so that `trimed` is only run from scratch once at the start. Details and pseudocode are presented in SM-H.

One can relax the bound test in `trimed` so that for $\epsilon > 0$ element i is computed if $l(i)(1+\epsilon) < E^{cl}$, guaranteeing that an element with energy within a factor $1+\epsilon$ of E^* is found. It is also possible to relax the bound tests in the assignment step of `trikmeds`, such that the distance to an assigned cluster's medoid is always within a factor $1+\epsilon$ of the distance to the nearest medoid. We denote by `trikmeds-`ϵ the `trikmeds` algorithm where the update and assignment steps are relaxed as just discussed, with `trikmeds-0` being exactly `trikmeds`. The motivation behind such a relaxation is that, at all but the final few iterations, it is probably a waste of computation obtaining medoids and assignments at high resolution, as in subsequent iterations they may change.

5 Results

We first compare the performance of the medoid algorithms `TOPRANK`, `TOPRANK2` and `trimed`. We then compare the K-medoids algorithms, `KMEDS` and `trikmeds`.

5.1 Medoid algorithm results

We compare our new exact medoid algorithm `trimed` with state-of-the-art approximate algorithms `TOPRANK` and `TOPRANK2`. Recall, Okamoto et al. (2008) prove that the approximate algorithms return w.h.p. the true medoid. We confirm that this is the case in all our experiments, where the approximate algorithms return the same element as `trimed`, which we know to be correct by Theorem 3.1. We now focus on comparing computational costs, which are proportional to the number of computed points.

Results on artificial datasets are presented in Figure 3, where our two main observations relate to scaling in N and dimension d. The artificial data are (left) uniformly drawn from $[0,1]^d$ and (right) drawn from $\mathcal{B}_d(0,1)$ with probability of lying within radius $1/2^{1/d}$ of $1/200$, as opposed to $1/2$ as would be the case under uniform density. Details about sampling from this distribution can be found in SM-F. Results on a mix of publicly available real and artificial datasets are presented in Table 1 and discussed in §5.1.2.

5.1.1 Scaling with N and d on artificial datasets

In Figure 3 we observe that the number of points computed by `trimed` is $O(N^{1/2})$, as predicted by Theorem 3.2. This is illustrated (right) by the close fit of the number of computed points to exact square root curves at sufficiently large N for $d \in \{2, 6\}$.

Recall that `TOPRANK` consists of two passes, a first where $N^{2/3}\log^{1/3} N$ anchor points are computed, and a second where all sub-threshold points are computed. We observe that for small N `TOPRANK` computes all N points, which corresponds to all points lying below threshold. At sufficiently large N the threshold becomes low enough for all points to be eliminated after the first pass. The effect is particularly dramatic in high dimensions ($d = 6$ on right), where a phase transition is observed between all and no points being computed in the second pass.

Dimension d appears in Theorem 3.2 through a factor $d(4/\alpha)^d$, where α is the strong convexity of the energy at the medoid. In Figure 3, we observe that the number of computed points increases with d for fixed N, corresponding to a relatively small α. The effect of α on the number of computed elements is considered in greater detail in SM-F.

In contrast to the above observation that the number of computed points increases as dimension increases for `trimed`, `TOPRANK` appears to scale favourably with dimension. This observation can be explained in terms of the distribution of energies, with energies close to E^* being less common in higher dimensions, as discussed in SM-J.

5.1.2 Results on publicly available real and simulated datasets

We present the datasets used here in detail in SM-I. For all datasets, algorithms `TOPRANK`, `TOPRANK2` and `trimed` were run 10 times with a distinct seed, and the mean number of iterations ($\hat{n}$) over the 10 runs was computed. We observe that our algorithm `trimed` is the best performing algorithm on all datasets, although in high-dimensions (MNIST-0) and on social

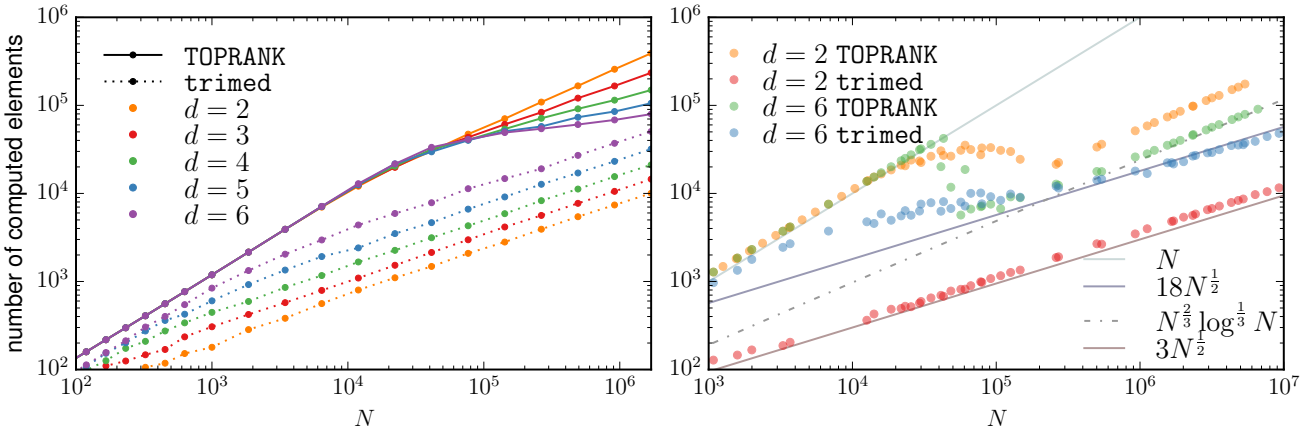

Figure 3: Comparison of `TOPRANK` and our algorithm `trimed` on simulated data. On the left, points are drawn uniformly from $[0,1]^d$ for $d \in \{2,\dots,6\}$, and on the right they are drawn from $\mathcal{B}_d(0,1)$ for $d \in \{2,6\}$, with an increased density near the edge of the ball. Fewer points (elements) are computed by `trimed` than by `TOPRANK` in all scenarios. For small N, `TOPRANK` computes $O(N)$ points, before transitioning to $\tilde{O}(N^{2/3})$ computed points for large N. `trimed` computes $O(N^{1/2})$ points. Note that `trimed` performs better in low-d than in high-d, with the reverse trend being true for `TOPRANK`. These observations are discussed in further detail in the text.

network data (Gnutella) no algorithm computes significantly fewer than N elements. The failure in high-dimensions (MNIST-0) of `trimed` is in agreement with Theorem 3.2, where dimension appears as the exponent of a constant term. The small world network data, Gnutella, can be embedded in a high-dimensional Euclidean space, and thus the failure on this dataset can also be considered as being due to high-dimensions. For low-dimensional real and spatial network data, `trimed` consistently computes $O(N^{1/2})$ elements.

5.1.3 But who needs the exact medoid anyway?

A valid criticism that could be raised at this stage would be that for large datasets, finding the exact medoid is probably overkill, as any point with energy reasonably close to E^* suffices for most applications. But consider, the `RAND` algorithm requires computing $\log N/\epsilon^2$ elements to confidently return an element with energy within ϵE^* of E^*. For $N = 10^5$ and $\epsilon = 0.05$, this is 4600, already more than `trimed` requires to obtain the exact medoid on low-d datasets of comparable size.

5.2 K-medoids algorithm results

With N elements to cluster, `KMEDS` is $\Theta(N^2)$ in memory, rendering it unusable on even moderately large datasets. To compare the initialisation scheme proposed in Park and Jun (2009) to random initialisation, we have performed experiments on 14 small datasets, with $K \in \{10, \lceil N^{1/2} \rceil, \lceil N/10 \rceil\}$. For each of these 42 experimental set-ups, we run the deterministic `KMEDS` initialisation once, and then uniform random initialisation, 10 times. Comparing the mean final energy of the two initialisation schemes, in only 9 of 42 cases does `KMEDS` initialisation result in a lower mean final energy. A Table containing all results from these experiments in presented in SM-E.

Having demonstrated that random uniform initialisation performs at least as well as the initialisation scheme of `KMEDS`, and noting that `trikmeds-0` returns exactly the same clustering as would `KMEDS` with uniform random initialisation, we turn our attention to the computational performance of `trikmeds`. Table 2 presents results on 4 datasets, each described in SM-I. The first numerical column is the relative number of distance calculations using `trikmeds-0` and `KMEDS`, where large savings in distance calculations, especially in low-dimensions, are observed. Columns ϕ_c and ϕ_E are the number of distance calculations and energies respectively, using $\epsilon \in \{0.01, 0.1\}$, relative to $\epsilon = 0$. We observe large reductions in the number of distance computations with only minor increases in energy.

6 Conclusion and future work

We have presented our new `trimed` algorithm for computing the medoid of a set, and provided strong theoretical guarantees about its performance in $\mathbb{R}^d$. In low-dimensions, it outperforms the state-of-the-art approximate algorithm on a large selection of datasets. The algorithm is very simple to implement, and can easily be extended to the general ranking problem. In the future, we propose to explore the idea of us-

			TOPRANK	TOPRANK2	trimed
dataset	type	N	$\hat{n}$	$\hat{n}$	$\hat{n}$
Birch 1	2-d	1.0×10^5	57944	100180	**2180**
Birch 2	2-d	1.0×10^5	66062	100180	**2208**
Europe	2-d	1.6×10^5	176095	169535	**2862**
U-Sensor Net	u-graph	3.6×10^5	113838	327216	**1593**
D-Sensor Net	d-graph	3.6×10^5	99896	176967	**1372**
Pennsylvania road	u-graph	1.1×10^6	216390	time-out	**2633**
Europe rail	u-graph	4.6×10^4	35913	47041	**518**
Gnutella	d-graph	6.3×10^3	7043	6407	**6328**
MNIST	784-d	6.7×10^3	7472	6799	**6514**

Table 1: Comparison of TOPRANK, TOPRANK2 and our algorithm trimed on publicly available real and simulated datasets. Column 2 provides the type of the dataset, where 'x-d' denotes x-dimensional vector data, while 'd-graph' and 'u-graph' denote directed and undirected graphs respectively. Column $\hat{n}$ gives the mean number of elements computed over 10 runs. Our proposed trimed algorithm obtains the true medoid with far fewer computed points in low dimensions and on spatial network data. On the social network dataset (Gnutella) and the very high-d dataset (MNIST), all algorithms fail to provide speed-up, computing approximately N elements.

			$K = 10$					$K = \lceil\sqrt{N}\rceil$				
			$\epsilon = 0$	$\epsilon = 0.01$		$\epsilon = 0.1$		$\epsilon = 0$	$\epsilon = 0.01$		$\epsilon = 0.1$	
Dataset	N	d	N_c/N^2	ϕ_c	ϕ_E	ϕ_c	ϕ_E	N_c/N^2	ϕ_c	ϕ_E	ϕ_c	ϕ_E
Europe	1.6×10^5	2	0.067	0.33	1.004	0.01	1.054	0.008	0.68	1.031	0.39	1.090
Conflong	1.6×10^5	3	0.042	0.67	1.001	0.08	1.014	0.006	0.92	1.003	0.61	1.026
Colormo	6.8×10^4	9	0.163	0.92	1.000	0.35	1.015	0.011	0.98	1.000	0.82	1.005
MNIST50	6.0×10^4	50	0.280	0.99	1.000	0.95	1.001	0.019	0.99	1.001	0.97	1.001

Table 2: Relative numbers of distance calculations and final energies using trikmeds-ϵ for $\epsilon \in \{0, 0.01, 0.1\}$. The number of distance calculations with trikmeds-0 is N_c, presented here relative to the number computed using KMEDS (N^2) in column N_c/N^2. The number of distance calculations with $\epsilon \in \{0.01, 0.1\}$ relative to trikmeds-0 are given in columns ϕ_c, so $\phi_c = 0.33$ means $3\times$ fewer calculations than with $\epsilon = 0$. The final energies with $\epsilon \in \{0.01, 0.1\}$ relative to trikmeds-0 are given in columns ϕ_E. We see that trikmeds-0 uses significantly fewer distance calculations than would KMEDS, especially in low-dimensions where a greater than $K\times$ reduction is observed ($N_C/N^2 < 1/K$). For low-d, additional relaxation further increases the saving in distance calculations with little cost to final energy.

ing more complex triangle inequality bounds involving several points, with as goal to improve on the $O(N^{1/2})$ number of computed points.

We have demonstrated how trimed, when combined with the approach of Elkan (2003), can greatly reduce the number of distance calculations required by the Voronoi iteration K-medoids algorithm of Park and Jun (2009). In the future we would like to replace the strategy of Elkan (2003) with that of Hamerly (2010), which will be better adapted to graph clustering as either all or no distances are computed with it, making it more amenable to Dijkstra's algorithm.

Acknowledgements

The authors are grateful to Wei Chen for helpful discussions of the TOPRANK algorithm. James Newling was funded by the Hasler Foundation under the grant 13018 MASH2.

References

Arthur, D. and Vassilvitskii, S. (2007). K-means++: The advantages of careful seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '07, pages 1027–1035, Philadelphia, PA, USA. Society for Industrial and Applied Mathematics.

Chipman, H., Hastie, T., and Tibshirani, R. (2003).

Statistical Analysis of Gene Expression Microarray Data. Chapman & Hall. Chapter 4.

Cohen, E., Delling, D., Pajor, T., and Werneck, R. F. (2014). Computing classic closeness centrality, at scale. In *Proceedings of the Second ACM Conference on Online Social Networks*, COSN '14, pages 37–50, New York, NY, USA. ACM.

Cohen, M. B., Lee, Y. T., Miller, G. L., Pachocki, J. W., and Sidford, A. (2016). Geometric median in nearly linear time. In *STOC16*. submitted.

Elkan, C. (2003). Using the triangle inequality to accelerate k-means. In *Machine Learning, Proceedings of the Twentieth International Conference (ICML 2003), August 21-24, 2003, Washington, DC, USA*, pages 147–153.

Eppstein, D. and Wang, J. (2004). Fast approximation of centrality. *J. Graph Algorithms Appl.*, 8(1):39–45.

Frahm, J.-M., Fite-Georgel, P., Gallup, D., Johnson, T., Raguram, R., Wu, C., Jen, Y.-H., Dunn, E., Clipp, B., Lazebnik, S., and Pollefeys, M. (2010). Building rome on a cloudless day. In *Proceedings of the 11th European Conference on Computer Vision: Part IV*, ECCV'10, pages 368–381, Berlin, Heidelberg. Springer-Verlag.

Hamerly, G. (2010). Making k-means even faster. In *SDM*, pages 130–140.

Hastie, T. J., Tibshirani, R. J., and Friedman, J. H. (2001). *The elements of statistical learning : data mining, inference, and prediction*. Springer series in statistics. Springer, New York.

Hoare, C. A. R. (1961). Algorithm 65: Find. *Commun. ACM*, 4(7):321–322.

Kaufman, L. and Rousseeuw, P. J. (1990). *Finding groups in data : an introduction to cluster analysis*. Wiley series in probability and mathematical statistics. Wiley, New York. A Wiley-Interscience publication.

Newling, J. and Fleuret, F. (2016a). Fast k-means with accurate bounds. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 936–944.

Newling, J. and Fleuret, F. (2016b). K-medoids for k-means seeding. arXiv:1609.04723. Under review.

Ng, R. T., Han, J., and Society, I. C. (2005). Clarans: A method for clustering objects for spatial data mining. *IEEE Transactions on Knowledge and Data Engineering*, pages 1003–1017.

Okamoto, K., Chen, W., and Li, X.-Y. (2008). Ranking of closeness centrality for large-scale social networks. In *Proceedings of the 2Nd Annual International Workshop on Frontiers in Algorithmics*, FAW '08, pages 186–195, Berlin, Heidelberg. Springer-Verlag.

Park, H.-S. and Jun, C.-H. (2009). A simple and fast algorithm for k-medoids clustering. *Expert Syst. Appl.*, 36(2):3336–3341.

Rattigan, M. J., Maier, M., and Jensen, D. (2007). Graph clustering with network structure indices. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, pages 783–790, New York, NY, USA. ACM.

SM-A On the difficulty of the medoid problem

We construct an example showing that no general purpose algorithm exists to solve the medoid problem in $o(N^2)$. Consider an almost fully connected graph containing $N = 2m + 1$ nodes, where the graph is exactly m edges short of being fully connected: one node has $2m$ edges and the others have $2m - 1$ edges. The graph has $2m^2$ edges. With the shortest path metric, it is easy to see that the node with $2m$ edges is the medoid, hence the medoid problem is as difficult as finding the node with $2m$ edges. But, supposing that the edges are provided as an unsorted adjacency list, it is clearly an $O(m^2)$ task to determine which node has $2m$ edges as one must look at all edges until a node with $2m$ edges is found. Thus determining the medoid is $O(m^2)$ which is $O(N^2)$.

SM-B `KMEDS` pseudocode

Alg. 2 presents the `KMEDS` algorithm of Park and Jun (2009), with the novel initialisation of `KMEDS` on line 1. `KMEDS` is essentially `lloyd`, with medoids instead of means.

Algorithm 2 `KMEDS` for clustering data $\{x(1), \ldots, x(N)\}$ around K medoids

1: Set all distances $D(i,j) \leftarrow \|x(i) - x(j)\|$ and sums $S(i) \leftarrow \sum_{j \in \{1,\ldots,N\}} D(i,j)$
2: Initialise medoid indices as K indices minimising $f(i) = \sum_{j \in \{1,\ldots,N\}} D(i,j)/S(j)$
3: **while** Some convergence criterion has not been met **do**
4: Assign each element to the cluster whose medoid is nearest to the element
5: Update cluster medoids according to assignments made above
6: **end while**

SM-C `RAND`, `TOPRANK` and `TOPRANK2` pseudocode

We present pseudocode for the `RAND`, `TOPRANK` and `TOPRANK2` algorithms of Okamoto et al. (2008), and discuss the explicit and implicit constants.

SM-C.1 On the number of anchor elements in `TOPRANK` : the constant in $\Theta(N^{\frac{2}{3}} (\log N)^{\frac{1}{3}})$

Note that the number of anchor points used in `TOPRANK` does not affect the result that the medoid is w.h.p. returned. However, Okamoto et al. (2008) show that by choosing the size of the anchor set to be $q (\log N)^{\frac{1}{3}}$ for any q, the run time is guaranteed to be $\tilde{O}(N^{5/3})$. They do not suggest a specific q, the optimal q being dataset dependant. We choose $q = 1$.

Consider Figure 3 in Section 5.1 for example, where $q = 1$. Had q be chosen to be less than 1, the line $\texttt{ncomputed} = N^{2/3} \log^{1/3} N$ to which `TOPRANK` runs parallel for large N would be shifted up or down by $\log q$, however the N at which the transition from $\texttt{ncomputed} = N^{2/3} \log^{1/3} N$ to $\texttt{ncomputed} = N^{2/3} \log^{1/3} N$ takes place would also change.

SM-C.2 On the parameter α' in `TOPRANK` and `TOPRANK2`

The threshold τ in (2) is proportional to the parameter α'. In Okamoto et al. (2008), it is stated that α' should be some value greater than 1. Note that the smaller α' is, the lower the threshold is, and hence fewer the number of computed points is, thus $\alpha' = 1.00001$ would be a fair choice. We use $\alpha' = 1$ in our experiments, and observe that the correct medoid is returned in all experiments.

Personal correspondence with the authors of Okamoto et al. (2008) has brought into doubt the proof of the result that the medoid is w.h.p. returned for any α' where $\alpha' > 1$. In our most recent correspondence, the authors suggest that the w.h.p. result can be proven with the more conservative bound of $\alpha' > \sqrt{1.5}$. Moreover, we show in SM-D that $\alpha' > 1$ is good enough to return the medoid with probability $N^{-(\alpha'-1)}$, a probability which still tends to 0 as N grows large, but not a w.h.p. result. Please refer to SM-D for further details on our correspondence with the authors.

SM-C.3 On the parameters specific to TOPRANK2

In addition to α', TOPRANK2 requires two parameters to be set. The first is l_0, the starting anchor set size, and the second is q, the amount by which l should be incremented at each iteration. Okamoto et al. (2008) suggest taking l_0 to be the number of top ranked nodes required, which in our case would be $l_0 = k = 1$. However, in our experience this is too small as all nodes lie well within the threshold and thus when l increases there is no change to number below threshold, which makes the algorithm break out of the search for the optimal l too early. Indeed, l_0 needs to be chosen so that at least some points have energies greater than the threshold, which in our experiments is already quite large. We choose $l_0 = \sqrt{N}$, as any value larger than $N^{2/3}$ would make TOPRANK2 redundant to TOPRANK. The parameter q we take to be $\log N$ as suggested by Okamoto et al. (2008).

Algorithm 3 RAND for estimating energies of elements of set S (Eppstein and Wang, 2004).

$I \leftarrow$ random uniform sample from $\{1, \dots, N\}$
// Compute all distances from anchor elements (I), using Dijkstra's algorithm on graphs
for $i \in I$ **do**
  for $j \in \{1, \dots, N\}$ **do**
    $d(i, j) \leftarrow \|x(i) - x(j)\|$,
  end for
end for
// Estimate energies as mean distances to anchor elements
for $j \in \{1, \dots, N\}$ **do**
  $\hat{E}(j) \leftarrow \frac{1}{|I|} \sum_{i \in I} d(i, j)$
end for
return $\hat{E}$

Algorithm 4 TOPRANK for obtaining top k ranked elements of S (Okamoto et al., 2008).

$l \leftarrow N^{\frac{2}{3}} (\log N)^{\frac{1}{3}}$ // Okamoto et al. (2008) state that l should be $\Theta((\log N)^{\frac{1}{3}})$, the choice of 1 as the constant is arbitrary (see comments in the text of Section SM-C.1).
Run RAND with uniform random I of size l to get $\hat{E}(i)$ for $i \in \{1, \dots, N\}$.
Sort $\hat{E}$ so that $\hat{E}[1] \leq \hat{E}[2] \leq \dots \leq \hat{E}[N]$
$\hat{\Delta} \leftarrow 2 \min_{i \in I} \max_{j \in \{1, \dots, N\}} \|x(i) - x(j)\|$ // where $\|x(i) - x(j)\|$ computed in RAND
$Q \leftarrow \left\{ i \in \{1, \dots, N\} \mid \hat{E}(i) \leq \hat{E}[k] + 2\alpha' \hat{\Delta} \sqrt{\frac{\log(n)}{l}} \right\}$.
Compute exact energies of all elements in Q and return the element with the lowest energy.

SM-D On the proof that TOPRANK returns the medoid with high probability

Through correspondence with the authors of Okamoto et al. (2008), we have located a small problem in the proof that the medoid is returned w.h.p. for $\alpha' > 1$, the problem lying in the second inequality of Lemma 1. To arrive at this inequality, the authors have used the fact that for all i,

$$\mathbb{P}(E(i) \geq \hat{E}(i) + f(l) \cdot \Delta) \geq 1 - \frac{1}{2N^2}, \tag{8}$$

which is a simple consequence of the Hoeffding inequality as shown in Eppstein and Wang (2004). Essentially (8) says that, for a fixed node i, from which the mean distance to other nodes is $E(i)$, if one uniformly samples l distances to i and computes the mean $\hat{E}(i)$, the probability that $\hat{E}(i)$ is less than $E(i) + f(l)$ is greater than $1 - \frac{1}{2N^2}$.

The inequality (8) is true for a fixed node i. However, it no longer holds if i is selected to be the node with the lowest $\hat{E}(i)$. To illustrate this, suppose that $E(i) = 1$ for all i, and compute $\hat{E}(i)$ for all i. Let $\hat{E}^* = \arg\min_i \hat{E}(i)$. Now, we have a strong prior on $\hat{E}^*$ being significantly less than 1, and (8) no longer holds as a statement made about $\hat{E}^*$.

In personal correspondence, the authors show that the problem can be fixed by the use of an additional layer of union bounding, with a correction to be published (if not already done so at time of writing). However, the

Algorithm 5 TOPRANK2 for obtaining top k ranked elements of S (Okamoto et al., 2008).

// In Okamoto et al. (2008), it is suggested that l_0 be taken as k, which in the case of the medoid problem is 1. We have experimented with several choices for l_0, as discussed in the text.
$l \leftarrow l_0$
Run RAND with uniform random I of size l to get $\hat{E}(i)$ for $i \in \{1, \ldots, N\}$.
$\hat{\Delta} \leftarrow 2 \min_{i \in I} \max_{j \in \{1,\ldots,N\}} \|x(i) - x(j)\|$ // where $\|x(i) - x(j)\|$ computed in RAND
Sort $\hat{E}$ so that $\hat{E}[1] \leq \hat{E}[2] \leq \ldots \leq \hat{E}[N]$
$Q \leftarrow \left\{ i \in \{1, \ldots, N\} \mid \hat{E}(i) \leq \hat{E}[k] + 2\alpha' \Delta \sqrt{\frac{\log(n)}{l}} \right\}$.
$g \leftarrow 1$
while g is 1 **do**
  $p \leftarrow |Q|$
  // The recommendation for q in Okamoto et al. (2008) is $\log(n)$, we follow the suggestion
  Increment I with q new anchor points
  Update $\hat{E}$ for all data according to new anchor points
  $l \leftarrow |I|$
  $\hat{\Delta} \leftarrow 2 \min_{i \in I} \max_{j \in \{1,\ldots,N\}} \|x(i) - x(j)\|$
  Sort $\hat{E}$ so that $\hat{E}[1] \leq \hat{E}[2] \leq \ldots \leq \hat{E}[N]$
  $Q \leftarrow \left\{ i \in \{1, \ldots, N\} \mid \hat{E}(i) \leq \hat{E}[k] + 2\alpha' \Delta \sqrt{\frac{\log(n)}{l}} \right\}$
  $p' \leftarrow |Q|$
  if $p - p' < \log(n)$ **then**
    $g \leftarrow 0$
  end if
end while
Compute exact energies of all elements in Q and return the element with the lowest energy

additional layer of union bound requires a more conservative constraint on α', which is $\alpha' > 2$, although the authors propose that the w.h.p. result can be proven with $\alpha' > \sqrt{1.5}$ for N sufficiently large. We now present a small proof proving the w.h.p. result for $\alpha' > \sqrt{2}$ for N sufficiently large, with at the same time $\alpha' > 1$ guaranteeing that the medoid is returned with probability $O(N^{\alpha'-1})$.

SM-D.1 That the medoid is returned *with high probability* holds for $\alpha' > \sqrt{2}$ and that *with vanishing probability* it is returned for $\alpha' > 1$

Recall that we have N nodes with *energies* $E(1), \ldots, E(n)$. We wish to find the k lowest energy nodes (the original setting of Okamoto et al. (2008)). From Hoeffding's inequality we have,

$$\mathbb{P}(|E(i) - \hat{E}(i)| \geq \epsilon\Delta) \leq 2\exp\left(-l\epsilon^2\right). \tag{9}$$

Set the probability on the right hand side of 9 to be $2/N^{1+\beta}$, that is,

$$2\exp\left(-l\epsilon^2\right) = 2/N^{1+\beta},$$

which corresponds to

$$\epsilon = \sqrt{\left(\frac{1+\beta}{l}\right)\log(N)} \quad := \quad \tilde{f}(l).$$

Clearly $\sqrt{1+\beta}$ corresponds to α'. With this notation we have,

$$\mathbb{P}(|E(i) - \hat{E}(i)| \geq \tilde{f}(l)\Delta) \leq \frac{2}{N^{1+\beta}}. \tag{10}$$

Applying the union bound to (10) we have,

$$\mathbb{P}\left(\neg\left(\wedge_{i \in \{1,\ldots,N\}} |E(i) - \hat{E}(i)| \leq \tilde{f}(l)\Delta\right)\right) \leq \frac{2}{N^{\beta}}. \tag{11}$$

			$K = 10$		$K = \lceil\sqrt{N}\rceil$		$K = \lceil \frac{N}{10} \rceil$	
Dataset	N	d	μ_u/μ_{park}	σ_u/μ_{park}	μ_u/μ_{park}	σ_u/μ_{park}	μ_u/μ_{park}	σ_u/μ_{park}
gassensor	256	128	1.09	0.08	**0.90**	0.03	**0.83**	0.01
house16H	1927	17	1.01	0.02	**0.97**	0.01	**0.93**	0.01
S1	5000	2	1.05	0.05	**0.75**	0.01	**0.32**	0.01
S2	5000	2	1.04	0.07	**0.68**	0.01	**0.34**	0.00
S3	5000	2	1.03	0.05	**0.76**	0.01	**0.35**	0.00
S4	5000	2	1.02	0.03	**0.75**	0.01	**0.41**	0.01
A1	3000	2	**0.82**	0.03	**0.43**	0.01	**0.19**	0.00
A2	5250	2	**0.98**	0.03	**0.47**	0.01	**0.25**	0.00
A3	7500	2	**0.96**	0.02	**0.42**	0.02	**0.22**	0.00
thyroid	215	5	**0.95**	0.08	**0.97**	0.04	**0.93**	0.04
yeast	1484	8	1.00	0.02	**0.96**	0.02	**0.91**	0.02
wine	178	14	1.01	0.02	1.02	0.01	**0.98**	0.02
breast	699	9	**0.79**	0.03	**0.77**	0.02	**0.68**	0.02
spiral	312	3	1.03	0.03	**0.99**	0.02	**0.82**	0.03

Table 3: Comparing the initialisation scheme proposed in Park and Jun (2009) with random uniform initialisation for the `KMEDS` algorithm. The final energy using the deterministic scheme proposed in Park and Jun (2009) is μ_{park}. The mean over 10 random uniform initialisations is μ_u, and the corresponding standard deviation is σ_u. For small K ($K = 10$), the performances using the two schemes are comparable, while for larger K, it is clear that uniform initialisation performs much better on the majority of datasets.

Recall that we wish to obtain the k nodes with lowest energy. Denote by $r(j)$ the index of the node with the j'th lowest energy, so that

$$E(r(1)) \leq \ldots \leq E(r(j)) \leq \ldots \leq E(r(N)).$$

Denote by $\hat{r}(j)$ the index of the node with the j'th lowest estimated energy, so that

$$\hat{E}(\hat{r}(1)) \leq \ldots \leq \hat{E}(\hat{r}(j)) \leq \ldots \leq \hat{E}(\hat{r}(N)).$$

Now assume that for all i, it is true that $|E(i) - \hat{E}(i)| \leq \tilde{f}(l)$. Then consider, for $j \leq k$,

$$\begin{aligned} \hat{E}(\hat{r}(k)) - \hat{E}(r(j)) &= \underbrace{\Big(\hat{E}(\hat{r}(k)) - E(r(k))\Big)}_{\geq -\tilde{f}(l)\Delta} + \underbrace{\Big(E(r(k)) - E(r(j))\Big)}_{\geq 0} + \underbrace{\Big(E(r(j)) - \hat{E}(r(j))\Big)}_{\geq -\tilde{f}(l)\Delta}, \\ &\geq -2\tilde{f}(l)\Delta. \end{aligned} \tag{12}$$

The first bound in (12) is obtained by considering the most extreme case possible under the assumption, which is $\hat{E}(i) = a(E) - \tilde{f}(l)$ for all i. The second bound follows from $j \leq k$, and the third bound follows directly from the assumption. We thus have that, under the assumption,

$$\hat{E}(r(j)) \leq \hat{E}(\hat{r}(k)) + 2\tilde{f}(l)\Delta,$$

which says that all nodes of rank less than or equal to k have approximate energy less than $\hat{E}(\hat{r}(k)) + 2\tilde{f}(l)\Delta$. As the assumption holds with probability greater than $1 - 2/N^{\beta}$ by (11), we are done. Take $\beta = 1$ if you want the statement *with high probability*, that is

$$\epsilon = \sqrt{\frac{2\log(n)}{l}},$$

but for any $\beta > 0$, which corresponds to $\alpha' > 1$, the probability of failing to return the k lowest energy nodes tends to 0 as N grows.

SM-E On the initialisation of Park and Jun (2009)

In Table 3 we present the full results of the 48 experiments comparing the initialisation proposed in Park and Jun (2009) with simple uniform initialisation. The 14 datasets are all available from `https://cs.joensuu.fi/sipu/datasets/`.

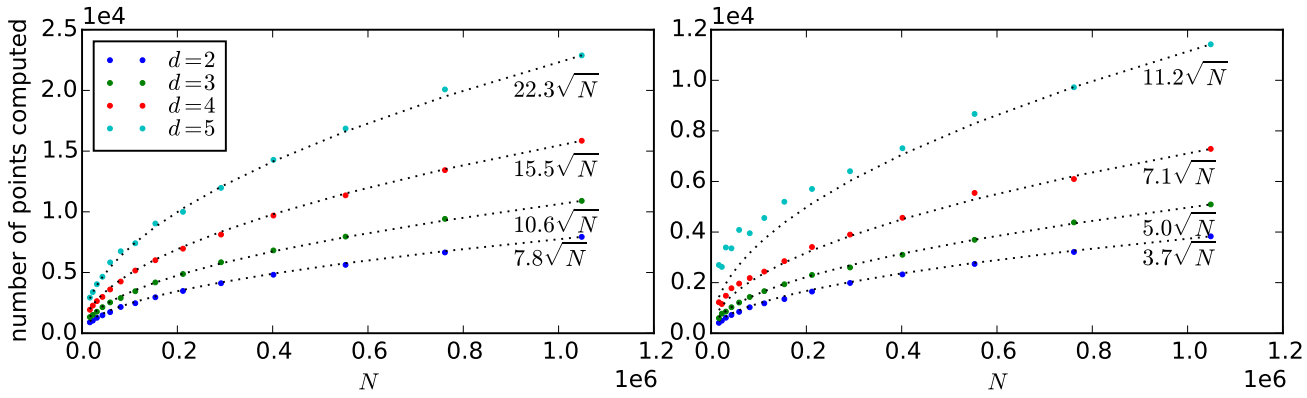

Figure 4: Number of points computed on simulated data. Points are drawn from $\mathcal{B}_d(0,1)$, for $d \in \{2,3,4,5\}$. On the left, points are drawn uniformly, while on the right, the density in $\mathcal{B}_d(0,(1/2)^{1/d})$ is 19× lower that in $\mathcal{A}_d(0,(1/2)^{1/2},1)$, where recall that $\mathcal{A}_d(x,r_1,r_2)$ denotes an annulus centred at x of inner radius r_1 and outer radius r_2. We observe a near perfect fit of the number of computed points to $\xi\sqrt{N}$ where the constant ξ depends on the dimension and the distribution (left and right). The number of computed points increases with dimension. The strong convexity constant of the distribution on the right is larger, corresponding to fewer distance calculations as predicted by Theorem 3.2.

SM-F Scaling with α, N, and dimension d

We perform more experiments to provide further validation of Theorem 3.2. In particular, we check how the number of computed elements scales with N, d and α. We generate data from a unit ball in various dimensions, according to two density functions with different strong convexity constants α. The first density function is uniform, so that the density everywhere in the ball is uniform. To sample from this distribution, we generate two random variables, $X_1 \sim \mathcal{N}_d(0,1)$ and $X_2 \sim U(0,1)$ and use

$$X_3 = X_1/\|X_1\| \cdot X_2^{\frac{1}{d}}, \tag{13}$$

as a sample from the unit ball $\mathcal{B}_d(0,1)$ with uniform distribution. The second distribution we consider has a higher density beyond radius $(1/2)^{1/d}$. Specifically, within this radius the density is 19× lower than beyond this radius. To sample from this distribution, we sample X_3 according to (13), and then points lying within radius $(1/2)^{1/d}$ are with probability 1/10 re-sampled uniformly beyond this radius.

The second distribution has a larger strong convexity constant α. To see this, note that the strong convexity constant at the center of the ball depends only on the density of the ball on its surface, that is at radius 1, as can be shown using an argument based on cancelling energies of internal points. As the density at the surface under distribution 2 is approximately twice that of under distribution 1, the change in energy caused by a small shift in the medoid is twice as large under distribution 2. Thus, according to Theorem 3.2, we expect the number of computed points to be larger under distribution 1 than under distribution 2. This is what we observe, as shown in Figure 4, where distribution 1 is on the left and distribution 2 is on the right.

In Figure 4 we observe a near perfect $N^{1/2}$ scaling of number of computed points. Dashed curves are exact $N^{1/2}$ relationships, while the coloured points are the observed number of computed points.

SM-G Proof of Theorem 3.2 (See page 5)

Theorem 3.2. *Let $\mathcal{S} = \{x(1),\ldots,x(N)\}$ be a set of N elements in $\mathbb{R}^d$, drawn independently from probability distribution function f_X. Let the medoid of $\mathcal{S}$ be $x(m^*)$, and let $E(m^*) = E^*$. Suppose that there exist strictly positive constants ρ, δ_0 and δ_1 such that for any set size N with probability $1 - O(1/N)$*

$$x \in \mathcal{B}_d(x(m^*),\rho) \implies \delta_0 \leq f_X(x) \leq \delta_1, \tag{6}$$

where $\mathcal{B}_d(x,r) = \{x' \in \mathbb{R}^d \ : \ \|x'-x\| \leq r\}$. *Let* $\alpha > 0$ *be a constant (independent of* N*) such that with probability* $1 - O(1/N)$ *all* $i \in \{1,\dots,N\}$ *satisfy,*

$$x(i) \in \mathcal{B}_d(x(m^*),\rho) \implies E(i) - E^* \geq \alpha \|x(i) - x(m^*)\|^2. \tag{7}$$

Then, the expected number of elements computed by `trimed` *is* $O\left(\left(V_d[1]\delta_1 + d\left(\frac{4}{\alpha}\right)^d\right)N^{\frac{1}{2}}\right)$, *where* $V_d[1] = \pi^{\frac{d}{2}}/(\Gamma(\frac{d}{2}+1))$ *is the volume of* $\mathcal{B}_d(0,1)$.

Proof. We show that the assumptions made in Th. 3.2 validate the assumptions required in Thm SM-G.1. Firstly, if $e(i) > \rho$ then $e(i) \geq \alpha\rho^2 e(i) > \rho$, which follows from the convexity of the loss function and. Secondly, the existance of β follows from continuity of the gradient of the distance, combined with the existence of δ_1 (non-exploding). □

Theorem SM-G.1 (Main Theorem Expanded)**.** *Let* $\mathcal{S} = \{x(1),\dots,x(N)\} \subset \mathbb{R}^d$ *have medoid* $x(m^*)$ *with minimum energy* $E(m^*) = E^*$, *where elements in* $\mathcal{S}$ *are drawn independently from probability distribution function* f_X. *Let* $e(i) = \|x(i) - x(m^*)\|$. *Suppose that for* f_X *there exist strictly positive constants* $\alpha, \beta, \rho, \delta_0$ *and* δ_1 *satisfying,*

$$x \in \mathcal{B}_d(x(m^*),\rho) \implies \delta_0 \leq f_X(x) \leq \delta_1, \tag{14}$$

where $\mathcal{B}_d(x,r) = \{x' \in \mathbb{R}^d \ : \ \|x' - x\| \leq r\}$, *and that for any set size* N, *w.h.p. all* $i \in \{1,\dots,N\}$ *satisfy,*

$$E(i) - E^* \geq \begin{cases} \alpha e(i)^2 & \text{if} \quad e(i) \leq \rho, \\ \alpha\rho^2 & \text{if} \quad e(i) > \rho, \end{cases} \tag{15}$$

and,

$$E(i) - E^* \leq \beta e(i)^2 \quad \text{if} \quad e(i) \leq \rho. \tag{16}$$

Then the expected number of elements computed, which is to say not eliminated on line 4 of `trimed`, *is* $O\left(\left(V_d[1]\delta_1 + d\left(\frac{4}{\alpha}\right)^d\right)N^{\frac{1}{2}}\right)$, *where* $V_d[1] = \pi^{\frac{d}{2}}/(\Gamma(\frac{d}{2}+1))$ *is the volume of* $\mathcal{B}_d(0,1)$.

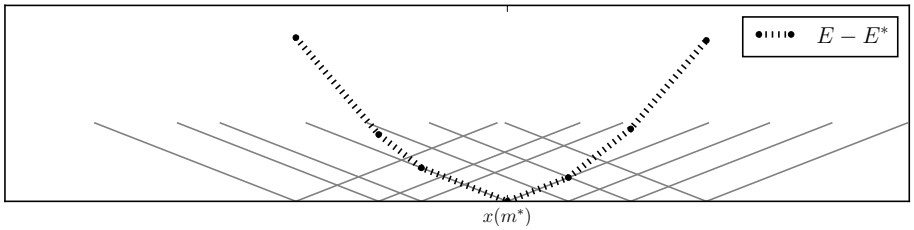

Figure 5: A sum of uniformly distributed cones is approximately quadratic.

Proof. We first show that the expected number of computed elements in $\mathcal{B}_d(x(m^*), N^{-\frac{1}{2d}})$ is $O(V_d[1]\delta_1 N^{\frac{1}{2}})$. When N is sufficiently large, $f_X(x) \leq \delta_1$ within $\mathcal{B}_d(x(m^*), N^{-\frac{1}{2d}})$. The expected number of samples in $\mathcal{B}_d(x(m^*), N^{-\frac{1}{2d}})$ is thus upper bounded by δ_1 multiplied by the volume of the ball. But the volume of a ball of radius $N^{-\frac{1}{2d}}$ in $\mathbb{R}^d$ is $V_d[1]N^{-\frac{1}{2}}$.

In Lemma SM-G.2 we use a packing argument to show that the number of computed elements in the annulus $\mathcal{A}_d(x(m^*), N^{-\frac{1}{2d}}, \infty)$ is $O\left(d\left(\frac{4}{\alpha}\right)^d N^{\frac{1}{2}}\right)$, but we there assume that the medoid index m^* is the first element in `shuffle`$(\{1,\dots,N\})$ on line 3 of `trimed` and thus that the medoid energy is known from the first iteration ($E^{cl} = E^*$). We now extend Lemma SM-G.2 to the case where the medoid is not the first element processed. We do this by showing that w.h.p. an element with energy very close to E^* has been computed after $N^{-\frac{1}{2}}$ iterations of `trimed`, and thus that the bounds on numbers of computed elements obtained using the packing arguments underlying Lemma SM-G.2 are all correct to within some small factor after $N^{-\frac{1}{2}}$ iterations.

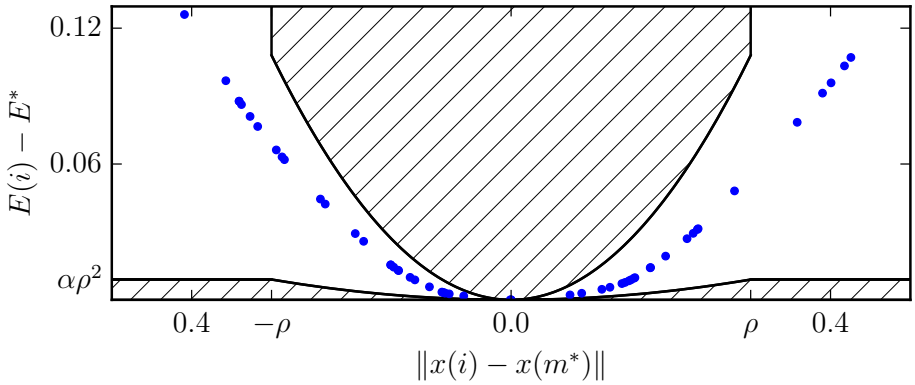

Figure 6: Illustrating the parameters α, β and ρ of Theorem 3.2. Here we draw $N = 101$ samples uniformly from $[-1, 1]$ and compute their energies, plotted here as the series of points. Theorem 3.2 states that their exists α, β and ρ such that irrespective of N, all energies (points) will lie in the envelope (non-hatched region).

The probability of a sample lying within radius $N^{-\frac{2}{3d}}$ of $x(m^*)$ is $\Omega(\delta_0 N^{-\frac{2}{3}})$, and so the probability that none of the first $N^{\frac{1}{2}}$ samples lies within radius $N^{-\frac{2}{3d}}$ is $O((1-\delta_0 N^{-\frac{2}{3d}})^{N^{\frac{1}{2}}})$ which is $O(\frac{1}{N})$. Thus w.h.p. after $N^{\frac{1}{2}}$ iterations of `trimed`, E^{cl} is within $\beta N^{-\frac{4}{3d}}$ of E^*, which means that the radii of the balls used in the packing argument are overestimated by at most a factor $N^{-\frac{1}{3d}}$. Thus w.h.p. the upper bounds obtained with the packing argument are correct to within a factor $1 + N^{-\frac{1}{3}}$. The remaining $O(\frac{1}{N})$ cases do not affect the expectation, as we know that no more than N elements can be computed. □

Lemma SM-G.2 (Packing beyond the vanishing radius)**.** *If we assume* (15) *from Theorem 3.2 and that the medoid index m^* is the first element processed by* `trimed`*, then the number of elements computed in $\mathcal{A}_d(x(m^*), N^{-\frac{1}{2d}}, \infty)$ is $O\left(d\left(\frac{4}{\alpha}\right)^d N^{\frac{1}{2}}\right)$.*

Proof. Follows from Lemmas SM-G.3 and SM-G.4. □

Lemma SM-G.3 (Packing from the vanishing radius $N^{-\frac{1}{d}}$ to ρ)**.** *If we assume* (15) *from Theorem 3.2 and that the medoid index m^* is the first element processed in* `trimed`*, then the number of computed elements in $\mathcal{A}(x(m^*), N^{-\frac{1}{2d}}, \rho)$ is $O(d\left(\frac{4}{\alpha}\right)^d N^{\frac{1}{2}})$.*

Proof. According to Assumption 15, an element at radius $r < \rho$ has surplus energy at least αr^2. This means that, assuming that the medoid has already been computed, an element computed at radius r will be surrounded by an exclusion zone of radius αr^2 in which no element will subsequently be computed. We will use this fact to upper bound the number of computed elements in $\mathcal{A}(x(m^*), N^{-\frac{1}{2d}}, \rho)$, firstly by bounding the number in an annulus of inner radius r and width αr^2, that is the annulus $\mathcal{A}_d(x(m^*), r, r + \alpha r^2)$, and then summing over concentric rings of this form which cover $\mathcal{A}(x(m^*), N^{-\frac{1}{2d}}, \rho)$. Recall that the number of computed elements in $\mathcal{A}_d(x(m^*), r, r + \alpha r^2)$ is denoted by $N_c(x(m^*), r, r + \alpha r^2)$.

We use Lemma SM-G.5 to bound $N_c(x(m^*), r, r+\alpha r^2)$,

$$\begin{aligned}
N_c(x(m^*), r, r+\alpha r^2) &\leq (d+1)^2 \left(\frac{4}{\sqrt{3}}\right)^d \frac{\alpha r^2 (r+\alpha r^2)^{d-1}}{(\alpha r^2)^d} \\
&\leq (d+1)^2 \left(\frac{4}{\sqrt{3}}\right)^d \left(1+\frac{1}{\alpha r}\right)^{d-1} \\
&\leq (d+1)^2 \left(\frac{4}{\sqrt{3}}\right)^d \left(\max\left(2, \frac{2}{\alpha r}\right)\right)^{d-1} \\
&\leq (d+1)^2 \left(\frac{4}{\sqrt{3}}\right)^d \left(\max\left(2^{d-1}, \left(\frac{2}{\alpha r}\right)^{d-1}\right)\right) \\
&\leq (d+1)^2 \left(\frac{4}{\sqrt{3}}\right)^d \left(2^{d-1} + \left(\frac{2}{\alpha r}\right)^{d-1}\right) \\
&\leq (d+1)^2 \left(\frac{8}{\sqrt{3}}\right)^d + (d+1)^2 \left(\frac{8}{\sqrt{3}}\right)^d \left(\frac{1}{\alpha r}\right)^{d-1}
\end{aligned}$$

Let $r_0 = N^{-\frac{1}{2d}}$ and $r_{i+1} = r_i + \alpha r_i^2$, and let T be the smallest index i such that $r_i \leq \rho$. With this notation in hand, we have

$$N_c(x(m^*), N^{-\frac{1}{2d}}, \rho) \leq \sum_{i=0}^{T} N_c(x(m^*), r_i, \alpha r_i + r_i^2).$$

The summation on the right-hand side can be upper-bounded by an integral. Using that the difference between r_i and r_{i+1} is αr_i^2, we need to divide terms in the sum by αr_i^2 when converting to an integral. Doing this, we obtain,

$$\begin{aligned}
N_c(x(m^*), N^{-\frac{1}{2d}}, \rho) &\leq \int_{N^{-\frac{1}{2d}}}^{\rho+\alpha\rho^2} N_c(x(m^*), r, \alpha r^2) dr \\
&\leq \text{const } + (d+1)^2 \left(\frac{8}{\sqrt{3}}\right)^d \left(\frac{1}{\alpha}\right)^d \int_{N^{-\frac{1}{2d}}}^{\infty} r^{-(1+d)} dr \\
&\leq \text{const } + (d+1) \left(\frac{4}{\alpha}\right)^d N^{\frac{1}{2}}.
\end{aligned}$$

This completes the proof, and provides the hidden constant of complexity as $(d+1)\left(\frac{4}{\alpha}\right)^d$. Thus larger values for α should result in fewer computed elements in the annulus $\mathcal{A}_d(x(m^*), r, r+\alpha r^2)$, which makes sense given that large values of α imply larger surplus energies and thus larger elimination zones. □

Lemma SM-G.4 (Packing beyond ρ)**.** *If we assume* (15) *from Theorem 3.2 and that the medoid index m^* is the first element processed by* `trimed`*, then the number of computed elements in $\mathcal{A}_d(x(m^*), \rho, \infty)$ is less than $(1 + 4E^*/(\alpha\rho^2))^d$.*

Proof. Recall that we at assuming $m^* = 1$, that is that the medoid is the first element processed in `trimed`. All elements beyond radius $2E^*$ are eliminated by type 1 eliminations (Figure 1), which provides the first inequality below. Then, as the excess energy is at least $\epsilon = \alpha\rho^2$ for all elements beyond radius ρ of $x(m^*)$, we apply Lemma SM-G.8 with $\epsilon = \alpha\rho^2/2$ to obtain the second inequality below,

$$\begin{aligned}
N_c(m(x), \rho, \infty) &\leq N_c(m(x), \rho, 2E^*) \\
&\leq \frac{(2E^* + \frac{1}{2}\alpha\rho^2)^d}{(\frac{1}{2}\alpha\rho^2)^d} \\
&\leq \left(1 + \frac{4E^*}{\alpha\rho^2}\right)^d.
\end{aligned}$$

□

Lemma SM-G.5 (Annulus packing). *For $0 \leq r$ and $0 < \epsilon \leq w$. If*

$$\mathcal{X} \subset \mathcal{A}_d(0, r, r+w),$$

where

$$\forall x \in \mathcal{X}, \mathcal{B}_d(x, \epsilon) \cup \mathcal{X} = \{x\}, \tag{17}$$

then,

$$|\mathcal{X}| \leq (d+1)^2 \left(\frac{4}{\sqrt{3}}\right)^d \frac{w\,(r+w)^{d-1}}{\epsilon^d}.$$

Proof. The condition (17) implies,

$$\forall x, x' \in \mathcal{X} \times \mathcal{X}, \mathcal{B}\left(x, \frac{\epsilon}{2}\right) \cup \mathcal{B}\left(x', \frac{\epsilon}{2}\right) = \emptyset. \tag{18}$$

Using that $\epsilon \in (0, w]$ and Lemma SM-G.6, one can show that for all $x \in \mathcal{A}(0, r, r+w)$,

$$\text{volume}\left(\mathcal{B}\left(x, \frac{\epsilon}{2}\right) \cap \mathcal{A}(0, r, r+w)\right) > \frac{1}{d+1}\left(\frac{3}{4}\right)^{\frac{d}{2}} V_d\left[\frac{\epsilon}{2}\right] \tag{19}$$

Combining (18) with (19) we have,

$$\text{volume}\left(\bigcup_{x \in \mathcal{X}} \mathcal{B}\left(x, \frac{\epsilon}{2}\right) \cap \mathcal{A}(0, r, r+w)\right) > \frac{V_d\,[1]}{d+1}\left(\frac{\sqrt{3}}{4}\right)^d |\mathcal{X}|\epsilon^d. \tag{20}$$

Letting $S_d\,[\epsilon]$ denote the surface area of a $\mathcal{B}(0, \epsilon)$, it is easy to see that

$$\text{volume}\left(\mathcal{A}(0, r, r+w)\right) < S_d\,[1]\, w\,(r+w)^{d-1}. \tag{21}$$

Combining (20) with (21) we get,

$$\frac{V_d\,[1]}{d+1}\left(\frac{\sqrt{3}}{4}\right)^d |\mathcal{X}|\epsilon^d < S_d\,[1]\, w\,(r+w)^{d-1}.$$

which combined with the fact that

$$\begin{aligned}\frac{S_d\,[1]}{V_d\,[1]} &= \left(\frac{\frac{dV_d}{dr}}{V_d}\right)_{r=1} \\ &= d,\end{aligned}$$

provides us with,

$$|\mathcal{X}| \leq (d+1)^2 \left(\frac{4}{\sqrt{3}}\right)^d \frac{w\,(r+w)^{d-1}}{\epsilon^d}.$$

□

Lemma SM-G.6 (Volume of ball intersection). *For $x_0, x_1 \in \mathbb{R}^d$ with $\|x_0 - x_1\| = 1$,*

$$\frac{\text{volume}\left(\mathcal{B}_d\left(x_0, 1\right) \cap \mathcal{B}_d\left(x_1, 1\right)\right)}{\text{volume}\left(\mathcal{B}_d\left(x_0, 1\right)\right)} \geq \frac{1}{d+1}\left(\frac{3}{4}\right)^{\frac{d}{2}}.$$

Proof. Let $V_d[r]$ denote the volume of $\mathcal{B}_d(0,r)$. It is easy to see that,

$$\begin{aligned}
\text{volume}\left(\mathcal{B}_d(x_0,1)\cap\mathcal{B}_d(x_1,1)\right) &= 2\int_0^{\frac{1}{2}} V_{d-1}\left[\sqrt{x(2-x)}\right]dx \\
&\geq 2\int_0^{\frac{1}{2}} V_{d-1}\left[\sqrt{\frac{3}{2}x}\right]dx \\
&\geq 2V_{d-1}[1]\int_0^{\frac{1}{2}}\left(\frac{3}{2}x\right)^{\frac{d-1}{2}}dx \\
&\geq 2V_{d-1}[1]\left(\frac{3}{2}\right)^{\frac{d-1}{2}}\left(\frac{2}{d+1}\right)\left(\frac{1}{2}\right)^{\frac{d+1}{2}} \\
&\geq V_{d-1}[1]\left(\frac{3}{2}\right)^{\frac{d-1}{2}}\left(\frac{2}{d+1}\right)\left(\frac{1}{2}\right)^{\frac{d-1}{2}} \\
&\geq V_{d-1}[1]\left(\frac{3}{4}\right)^{\frac{d-1}{2}}\left(\frac{2}{d+1}\right).
\end{aligned}$$

Using that $\dfrac{V_{d-1}[1]}{V_d[1]} > \dfrac{1}{\sqrt{\pi}}$, we divide the intersection volume through by $V_d[1]$ to obtain,

$$\begin{aligned}
\frac{\text{volume}\left(\mathcal{B}_d(x_0,1)\cap\mathcal{B}_d(x_1,1)\right)}{\text{volume}\left(\mathcal{B}_d(x_0,1)\right)} &\geq \left(\frac{3}{4}\right)^{\frac{d-1}{2}}\left(\frac{2}{\sqrt{\pi}(d+1)}\right) \\
&\geq \frac{1}{d+1}\left(\frac{3}{4}\right)^{\frac{d}{2}}
\end{aligned}$$

□

Lemma SM-G.7 (Packing balls in a ball)**.** *The number of non-intersecting balls of radius ϵ which can be packed into a ball of radius r in $\mathbb{R}^d$ is less than $\left(\frac{r}{\epsilon}\right)^d$*

Proof. The technique used here is a loose version of that used in proving Lemma SM-G.5. The volume of $\mathcal{B}_d(0,\epsilon)$ is a factor $(r/\epsilon)^d$ smaller than that of $\mathcal{B}_d(0,r)$. As the balls of radius ϵ are non-overlapping, the volume of their union is simply the sum of their volumes. The result follow from the fact that the union of the balls of radius ϵ is contained within the ball of radius r. □

Lemma SM-G.8 (Packing points in a ball)**.** *Given $\mathcal{X}\subset\mathcal{B}_d(0,r)$ such that no two elements of $\mathcal{X}$ lie within a distance of ϵ of each other, $|\mathcal{X}| < \left(\frac{2r+\epsilon}{\epsilon}\right)^d$.*

Proof. As no two elements lie within distance ϵ of each other, balls of radius $\epsilon/2$ centred at elements are non-intersecting. As each of the balls of radius $\epsilon/2$ centred at elements of $\mathcal{X}$ lies entirely within $\mathcal{B}_d(0,r+\epsilon/2)$, we can apply Lemma (SM-G.7), arriving at the result. □

SM-H Pseudocode for `trikmeds`

In Alg. (6) we present `trikmeds`. It is decomposed into algorithms for initialisation (7), updating medoids (8), assigning data to clusters (9) and updating bounds on the `trimed` derived bounds (10). Table 4 summarised all of the variables used in `trikmeds`.

When there are no distance bounds, the location of the bottleneck in terms of distance calculations depends on N/K^2. If $N/K \gg K$, the bottleneck lies in updating medoids, which can be improved through the strategy used in `trimed`. If $N/K \ll K$, the bottleneck lies in assigning elements to clusters, which is effectively handled through the approach of Elkan (2003).

Algorithm 6 `trikmeds`

```
initialise()
while not converged do
  update-medoids()
  assign-to-clusters()
  update-sum-bounds()
end while
```

Algorithm 7 `initialise`

```
// Initialise medoid indices, uniform random sample without replacement (or otherwise)
{m(1), ..., m(K)} ← uniform-no-replacement({1, ..., N})
for k = 1 : K do
  // Initialise medoid and set cluster count to zero
  c(k) ← x(m(k))
  v(k) ← 0
  // Set sum of in-cluster distances to medoid to zero
  s(k) ← 0
end for
for i = 1 : N do
  for k = 1 : K do
    // Tightly initialise lower bounds on data-to-medoid distances
    l_c(i, k) ← ‖x(i) − c(k)‖
  end for
  // Set assignments and distances to nearest (assigned) medoid
  a(i) ← arg min_{k∈{1,...,K}} l_c(i, k)
  d(i) ← l_c(i, a(i))
  // Update cluster count
  v(a(i)) ← v(a(i)) + 1
  // Update sum of distances to medoid
  s(a(i)) ← s(a(i)) + d(i)
  // Initialise lower bound on sum of in-cluster distances to x(i) to zero
  l_s(i) ← 0
end for
V(0) ← 0
for k = 1 : K do
  // Set cumulative cluster count
  V(k) ← V(k − 1) + v(k)
  // Initialise lower bound on in-cluster sum of distances to be tight for medoids
  l_s(m(k)) ← s(k)
end for
// Make clusters contiguous
contiguate()
```

Algorithm 8 `update-medoids`

for $k = 1 : K$ **do**
  for $i = V(k-1) : V(k) - 1$ **do**
    // If the bound test cannot exclude i as $m(k)$
    if $l_s(i) < s(k)$ **then**
      // Make $l_s(i)$ tight by computing and cumulating all in-cluster distances to $x(i)$,
      $l_s(i) \leftarrow 0$
      for $i' = V(k-1) : V(k) - 1$ **do**
        $\tilde{d}(i') \leftarrow \|x(i) - x(i')\|$
        $l_s(i) \leftarrow l_s(i) + \tilde{d}(i')$
      end for
      // Re-perform the test for i as candidate for $m(k)$, now with exact sums. If i is the new best candidate, update some cluster information
      if $l_s(i) < s(k)$ **then**
        $s(k) \leftarrow l_s(i)$
        $m(k) \leftarrow i$
        for $i' = V(k-1) : V(k) - 1$ **do**
          $d(i') \leftarrow \|x(i) - x(i')\|$
        end for
      end if
      // Use computed distances to i to improve lower bounds on sums for all samples in cluster k (see Figure X)
      for $i' = V(k-1) : V(k) - 1$ **do**
        $l_s(i') \leftarrow \max\left(l_s(i'), |\tilde{d}(i')v(k) - l_s(i)|\right)$
      end for
    end if
  end for
  // If the medoid of cluster k has changed, update cluster information
  if $m(k) \neq V(k-1)$ **then**
    $p(k) \leftarrow \|c(k) - x(m(k))\|$
    $c(k) \leftarrow x(m(k))$
  end if
end for

Algorithm 9 `assign-to-clusters`

```
// Reset variables monitoring cluster fluxes,
for k = 1 : K do
    // the number of arrivals to cluster k,
    Δ_{n-in}(k) ← 0
    // the number of departures from cluster k,
    Δ_{n-out}(k) ← 0
    // the sum of distances to medoid k of samples which leave cluster k
    Δ_{s-out}(k) ← 0
    // the sum of distances to medoid k of samples which arrive in cluster k
    Δ_{s-in}(k) ← 0
end for
for i = 1 : N do
    // Update lower bounds on distances to medoids based on distances moved by medoids
    for k = 1 : K do
        l(i,k) = l(i,k) − p(k)
    end for
    // Use the exact distance of current assignment to keep bound tight (might save future calcs)
    l(i,a(i)) = d(i)
    // Record current assignment and distance
    a_old = a(i)
    d_old = d(i)
    // Determine nearest medoid, using bounds to eliminate distance calculations
    for k = 1 : K do
        if l(i,k) < d(i) then
            l(i,k) ← ‖x(i) − c(k)‖
            if l(i,k) < d(i) then
                a(i) = k
                d(i) = l(i,k)
            end if
        end if
    end for
    // If the assignment has changed, update statistics
    if a_old ≠ a(i) then
        v(a_old) = v(a_old) − 1
        v(a(i)) = v(a(i)) + 1
        l_s(i) = 0
        Δ_{n-in}(a(i)) = Δ_{n-in}(a(i)) + 1
        Δ_{n-out}(a_old) = Δ_{n-out}(a_old) + 1
        Δ_{s-in}(a(i)) = Δ_{s-in}(a(i)) + d(i)
        Δ_{s-out}(a_old) = Δ_{s-out}(a_old) + d_old
    end if
end for
// Update cumulative cluster counts
for k = 1 : K do
    V(k) ← V(k − 1) + v(k)
end for
contiguate()
```

Table 4: Table Of Notation For `trikmeds`

N	:	number of training samples
i	:	index of a sample, $i \in \{1, \dots, N\}$
$x(i)$	:	sample i
K	:	number of clusters
k	:	index of a cluster, $k \in \{1, \dots, K\}$
$m(k)$	:	index of current medoid of cluster k, $m(k) \in \{1, \dots, N\}$
$c(k)$	:	current medoid of cluster k, that is $c(k) = x(m(k))$
$n_1(i)$	:	cluster index of centroid nearest to $x(i)$
$a(i)$	:	cluster to which $x(i)$ is currently assigned
$d(i)$	:	distance from $x(i)$ to $c(a(i))$
$v(k)$	:	number of samples assigned to cluster k
$V(k)$	:	number of samples assigned to a cluster of index less than $k+1$
$l_c(i,k)$	:	lowerbound on distance from $x(i)$ to $m(k)$
$l_s(i)$	:	lowerbound on $\sum_{i':a(i')=a(i)} \|x(i') - x(i)\|$
$p(k)$	:	distance moved (teleported) by $m(k)$ in last update
$s(k)$	:	sum of distances of samples in cluster k to medoid k

Algorithm 10 `update-sum-bounds`

for $k = 1 : K$ **do**
  // Obtain absolute and net fluxes of energy and count, for cluster k
  $\mathcal{J}_s^{abs}(k) = \Delta_{s-in}(k) + \Delta_{s-out}(k)$
  $\mathcal{J}_s^{net}(k) = \Delta_{s-in}(k) - \Delta_{s-out}(k)$
  $\mathcal{J}_n^{abs}(k) = \Delta_{n-in}(k) + \Delta_{n-out}(k)$
  $\mathcal{J}_n^{net}(k) = \Delta_{n-in}(k) - \Delta_{n-out}(k)$
  for $i = V(k-1) : V(k) - 1$ **do**
    // Update the lower bound on the sum of distances
    $l_s(i) \leftarrow l_s(i) - \min(\mathcal{J}_s^{abs}(k) - \mathcal{J}_n^{net}(k)d(i), \mathcal{J}_n^{abs}(k)d(i) - \mathcal{J}_s^{net}(k))$
  end for
end for

SM-I Datasets

- *Birch1*, *Birch2* : Synthetic 2-D datasets available from `https://cs.joensuu.fi/sipu/datasets/`
- *Europe* : Border map of Europe available from `https://cs.joensuu.fi/sipu/datasets/`
- *U-Sensor Net* : Undirected 2-D graph data. Points drawn uniformly from unit square, with an undirected edge connecting points when the distance between them is less than $1.25\sqrt{N}$
- *D-Sensor Net* : Directed 2-D graph data. Points drawn uniformly from unit square, with directed edge connecting points when the distance between them is less than $1.45\sqrt{N}$, direction chosen at random.
- *Europe rail* : The European rail network, the shapefile is available at `http://www.mapcruzin.com/free-europe-arcgis-maps-shapefiles.htm`. We extracted edges from the shapefile using `networkx` available at `https://networkx.github.io/`.

Algorithm 11 `contiguate`

// This function performs an in place rearrangement over of variables a, d, l, x and m
// The permutation applied to a, d, l and x has as result a sorting by cluster,
// $a(i) = k$ if $i \in \{V(k-1), V(k)\}$ for $k \in \{1, \dots, K\}$
// and moreover that the first element of each cluster is the medoid,
// $m(k) = V(k-1)$ for $k \in \{1, \dots, K\}$

- *Pennsylvania road* The road network of Pennsylvania, the edge list is available directly from `https://snap.stanford.edu/data/`
- *Gnutella* Peer-to-peer network data, available from `https://snap.stanford.edu/data/`
- *MNIST (0)* The '0's in the MNIST training dataset.
- *Conflong* The conflongdemo data is available from `https://cs.joensuu.fi/sipu/datasets/`
- *Colormo* The colormoments data is available at `http://archive.ics.uci.edu/ml/datasets/Corel+Image+Features`
- *MNIST50* The MNIST dataset, projected into 50-dimensions using a random projection matrix where each of the 784×50 elements in the matrix is i.i.d. $\mathcal{N}(0, 1)$.
- *S1, S2, S3, S4, A1, A2, A3* All of these synthetic datasets are available from `https://cs.joensuu.fi/sipu/datasets/`.
- *thyroid, yeast, wine, breast, spiral* All of these real world datasets are available from `https://cs.joensuu.fi/sipu/datasets/`.

SM-J Scaling with dimension of TOPRANK and TOPRANK2

Recall the assumption (3) made for the TOPRANK and TOPRANK2 algorithms. The assumption states that as one approaches the minimum energy E^* from above, the density of elements decreases. In other words, the lowest energy elements stand out from the rest and are not bunched up with very similar energies.

Consider the case where elements are points in $\mathbb{R}^d$. Suppose that the density f_X of points around the medoid is bounded by $0 < \rho_0 \leq f_X \leq \rho_1$, and that the energy grows quadratically in radius about the medoid. Then, as the number of points at radius ϵ is $O(\epsilon^{d-1})$, the density (by energy) of points at radius ϵ is $O(\epsilon^{d-2})$. Thus for $d = 1$ the assumption for TOPRANK and TOPRANK does not hold, which results in poor performance for $d = 1$. For $d = 2$, the assumption holds, as the density (by energy) of points is constant. For $d \geq 2$, as d increases the energy distribution becomes more and more favourable for TOPRANK and TOPRANK2, as the low ranking elements become more and more distinct with low energies becoming less probable. This explains the observation that TOPRANK scales well with dimension in Figure 3.

SM-K Example where geometric median is a poor approximation of medoid

There is no guarantee that the geometric median is close to the set medoid. Moreover, the element in $\mathcal{S}$ which is nearest to $g(\mathcal{S})$ is not necessarily the medoid, as illustrated in the following example. Suppose $S = \{x(1), \ldots, x(20)\} \subset \mathbb{R}^2$, with $x(i) = (0, 1)$ for $i \in \{1, \ldots, 9\}$, $x(i) = (0, -1)$ for $i \in \{10, \ldots, 18\}$, $x(19) = (1/2, 0)$ and $x(20) = (-1/2, 0)$. The geometric median is $(0, 0)$ and the nearest points to the geometric median, $x(19)$ and $x(20)$ have energy $1 + 18\sqrt{3}/2 \approx 16.6$. However, points $\{x(1), \ldots, x(18)\}$ have energy $2\sqrt{3}/2 + 9 = 10.7$. Thus by choosing a point in $\mathcal{S}$ which is nearest to the geometric median, one is choosing the element with the highest energy, the opposite of the medoid.

Note the above example appears to violate the assumptions required for $O(N^{3/2})$ convergence of `trimed`, as it requires that the probability density function vanishes at the distribution median. Indeed, in $\mathbb{R}^d$ it is the case that if the $O(N^{3/2})$ assumptions are satisfied, the set medoid converges to the geometric median, and so the geometric median is a good approximation. We stress however that the geometric median is only relevant in vector spaces.

SM-L Miscellaneous

Figure 7 illustrates the idea behind algorithm `trimed`, comments in the caption.

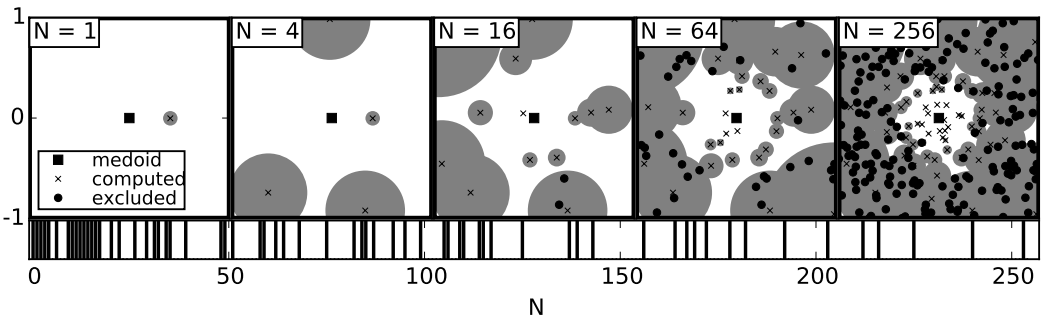

Figure 7: Eliminating samples as potential medoids using only type 1 elimination, where we assume that the medoid and its energy E^* are known, and so the radius of the exclusion ball of an element x is $E(x) - E^*$. Uniformly sampling from $[-1, 1] \times [-1, 1]$, energies are computed only if the sample drawn does not lie in the exclusion zone (union of balls). If the energy at x is computed, the exclusion zone is augmented by adding $\mathcal{B}_d(x, E(x) - E^*)$. Top left to right: the distribution of samples which are computed and excluded. Bottom: the times at which samples are computed. We prove that probability of computation at time n is $O(n^{-\frac{1}{2}})$.