On Efficient Low Distortion Ultrametric Embedding

Abstract

A classic problem in unsupervised learning and data analysis is to find *simpler* and *easy-to-visualize* representations of the data that preserve its essential properties. A widely-used method to preserve the underlying hierarchical structure of the data while reducing its complexity is to find an embedding of the data into a tree or an ultrametric. The most popular algorithms for this task are the classic *linkage* algorithms (single, average, or complete). However, these methods on a data set of n points in $\Omega(\log n)$ dimensions exhibit a quite prohibitive running time of $\Theta(n^2)$.

In this paper, we provide a new algorithm which takes as input a set of points P in R^d, and for every $c \geq 1$, runs in time $n^{1+O(1/c^2)}$ to output an ultrametric Δ such that for any two points u, v in P, we have $\Delta(u, v)$ is within a multiplicative factor of $5c$ to the distance between u and v in the "best" ultrametric representation of P. Here, the best ultrametric is the ultrametric $\widetilde{\Delta}$ that minimizes the maximum distance distortion with respect to the ℓ_2 distance, namely that minimizes $\max_{u,v \in P} {}^{\widetilde{\Delta}(u,v)}/_{\|u-v\|_2}$.

We complement the above result by showing that under popular complexity theoretic assumptions, for every constant $\epsilon > 0$ there exists a constant $\delta > 0$ such that no algorithm with running time $n^{2-\epsilon}$ can distinguish between inputs that admit isometric embedding and inputs that can incur a distortion of $3/2$ in ℓ_∞-metric.

Finally, we present empirical evaluation on classic machine learning datasets and show that the output of our algorithm is comparable to the output of the linkage algorithms while achieving a much faster running time.

1. Introduction

The *curse of dimensionality* has ruthlessly been haunting machine learning and data mining researchers. On the one hand, high dimensional representation of data elements allows fined-grained description of each datum and so can lead to more accurate models, prediction and understanding. On the other hand, obtaining a significant signal in each dimension often requires a huge amount of data and high-dimensional data requires algorithms that can efficiently handle it. Hence, computing a *simple* representation of a high-dimensional dataset while preserving its most important properties has been a central problem in a large number of communities since the 50s.

Of course, computing a simple representation of an arbitrary high-dimensional set of data elements necessarily incurs some information loss. Thus, the main question has been to find dimensionality reduction techniques that would preserve – or better, *reveal* – some structure of the data. An example of such a successful approach has been the *principal component analysis* which can be used to denoise a dataset and obtain a low-dimensional representation where 'similar' data elements are mapped to close-by locations. This approach has thus become a widely-used, powerful tool to identify cluster structures in high-dimensional datasets.

Yet, in many cases more complex structures underlie the datasets and it is crucial to identify this structure. For example, given similarity relations between species, computing a phylogenetic tree requires more than identifying a 'flat' clustering structure, it is critical to identify the whole hierarchy of species. Thus, computing a simple representation of an input containing a hierarchical structure has drawn a lot of attention over the years, in particular from the computational biology community. The most popular approaches are arguably the *linkage* algorithms, average-linkage, single-linkage, Ward's method and complete-linkage, which produce an embedding of the original metric into an ultrametric [1], see for example the seminal work of Carlsson and Mémoli (2010). Unfortunately, these approaches come with a major drawback: all these methods, have quadratic running time – even in the best case – when the input consists of points in $\Omega(\log n)$ dimensions (where n is the number of points) making them impractical for most applications nowadays. Obtaining efficient algorithm for computing "good" hierarchical representation has thus been a major problem (see Section 1.2 for more details).

[1] An ultrametric (X, d) is a metric space where for each $x, y, z \in X$, $\Delta(x, y) \leq \max(\Delta(x, z)(, z, y))$

In this paper we are interested in defining the quality of an embedding so that it measures how much the hierarchical structure underlying the input is preserved. For example, given three points a, b, c, we would like that if a is more similar to b than to c (and so a is originally closer to b than to c in the high-dimensional representation), then its distance to b in the ultrametric is lower than its distance to c. More formally, given a set of points X in a Euclidean space, a *good ultrametric* representation Δ is such that for each pair of points a, b, $||a-b||_2 \leq \Delta(a,b) \leq \alpha||a-b||_2$ for the smallest possible α (see formal definition in Section 1.3). Interestingly, and perhaps surprisingly, this problem can be solved in $O(n^2 d + n^2 \log n)$ using an algorithm by Farach et al. (1995). Unfortunately, this algorithm also suffers from a quite prohibitive quadratic running time. We thus ask: Is there an easy-to-implement, efficient algorithm for finding good ultrametric representation of high-dimensional inputs?

1.1. Our Results

We focus on the problem mentioned above, which we refer to as the BEST ULTRAMETRIC FIT under the ℓ_∞ metric ($\texttt{BUF}_\infty$) and which is formally defined in Section 1.3. We provide a simple algorithm, with running time $O(nd + n^{1+1/\gamma^2} \log n)$ that returns a 5γ-approximation for the $\texttt{BUF}_\infty$ problem, or a near-linear time algorithm that returns an $O(\sqrt{\log n})$-approximation.

Theorem 1.1 (Upper Bound). *For any $\gamma > 1$, there is an algorithm that produces a 5γ-approximation in time $nd + n^{1+O(1/\gamma^2)}$ for Euclidean instances of BUF_∞ of dimension d.*
Moreover, there is an algorithm that produces an $O(\sqrt{\log n})$-approximation in time $O(nd + n \log^2 n)$ for Euclidean instances of BUF_∞ of dimension d.

Importantly, and perhaps surprisingly, we show that finding a faster than $n^{2-\varepsilon}$ algorithm for this problem is beyond current techniques.

Theorem 1.2 (Lower Bound; Informal version of Theorem 4.1). *Assuming* SETH, *for every $\varepsilon > 0$, no algorithm running in time $n^{2-\varepsilon}$ can determine if an instance of $\texttt{BUF}_\infty$ of points in ℓ_∞-metric admits an isometric embedding or every embedding has distortion at least $3/2$.*

Empirical results We implemented our algorithm and performed experiments on three classic datasets (DIABETES, MICE, PENDIGITS). We compared the results with classic linkage algorithms (average, complete, single) and Ward's method from the Scikit-learn library (Pedregosa et al., 2011). For a parameter γ fixed to ≈ 2.5, our results are as follows. First, as complexity analysis predicts, the execution of our algorithm is much faster whenever the dataset becomes large enough: up to ≈ 36 (resp. 32, 7 and 35) times faster than average linkage (resp. complete linkage, single linkage and Ward's method) for moderate size dataset containing roughly 10000 points, and has comparable running time for smaller inputs. Second, while achieving a much faster running time, the quality of the ultrametric stays competitive to the distortion produced by the other linkage algorithms. Indeed, the maximum distortion is, on these three datasets, always better than Ward's method, while staying not so far from the others: in the worst case up to a factor ≈ 5.2 (resp. 4.3, 10.5) against average linkage (resp. complete and single linkages). This shows that our new algorithm is a reliable and efficient alternative to the linkage algorithms when dealing with massive datasets.

1.2. Related Work

Strengthening the foundations for hierarchical representation of complex data has received a lot of attention over the years. The thorough study of Carlsson and Mémoli (2010) has deepen our understanding of the linkage algorithms and the inputs for which they produce good representations, we refer the reader to this work for a more complete introduction to the linkage algorithms. Hierarchical representation of data and hierarchical clusterings are similar problems. A recent seminal paper by Dasgupta (2015) phrasing the problem of computing a good hierarchical clustering as an optimization problem has sparked a significant amount of work mixing theoretical and practical results. Cohen-Addad et al. (2018); Moseley and Wang (2017) showed that average-linkage achieves a constant factor approximation to (the dual of) Dasgupta's function and introduced new algorithms with worst-case and beyond-worst-case guarantees, see also (Roy and Pokutta, 2016; Charikar and Chatziafratis, 2017; Cohen-Addad et al., 2017; Charikar et al., 2019; 2018). Single-linkage is also known to be helfpul to identify 'flat' clusterings in some specific settings (Balcan et al., 2008). We would like to point out that this previous work did not considered the question of producing an ultrametric that is representative of the underlying (dis)similarities of the data and in fact most of the algorithms designed by previous work do not output ultrametrics at all. This paper takes a different perspective on the problem of computing a hierarchical clustering: we are interested in how well the underlying metric is preserved by the hierarchical clustering.

Finally, a related but orthogonal approach to ours was taken in recent papers by Cochez and Mou (2015) and Abboud et al. (2019). There, the authors design implementation of average-linkage and Ward's method that have subquadratic running time by approximating the greedy steps done by the algorithms. However, their results do not provide any approximation guarantees in terms of any objective function but rather on the quality of the approximation of the

greedy step and is not guaranteed to produce an ultrametric.

1.3. Preliminaries

We consider the BEST ULTRAMETRIC FIT problem under the ℓ_∞ metric ($\texttt{BUF}_\infty$), namely:

- Input: a set V of n elements $v_1, \ldots, v_n$ and a weight function $w : V \times V \mapsto \mathsf{R}$.
- output: an ultrametric dist^U such that $\forall v_i, v_j \in V$, $w(v_i, v_j) \leq \text{dist}^U(v_i, v_j) \leq \alpha \cdot w(v_i, v_j)$, for the minimal value α.

Note that we will abuse notation slightly and, for an edge $e = (v_i, v_j)$, write $w(e)$ to denote $w(v_i, v_j)$. We write dist^{OPT} to denote an optimal ultrametric, and let α_{OPT} denote the minimum α for which $\forall v_i, v_j \in V$, $w(v_i, v_j) \leq \text{dist}^{\text{OPT}}(v_i, v_j) \leq \alpha \cdot w(v_i, v_j)$.

We say that an ultrametric $\widehat{\text{dist}}$ is a γ-*approximation* to $\texttt{BUF}_\infty$ if $\forall v_i, v_j \in V$, $w(v_i, v_j) \leq \widehat{\text{dist}}(v_i, v_j) \leq \gamma \cdot \alpha_{\text{OPT}} \cdot w(v_i, v_j)$.

1.4. Algorithm from (Farach et al., 1995)

Farach et al. (1995) provide an $O(n^2)$ algorithm to solve a "more general" problem (i.e., that is such that an optimal algorithm for this problem can be used to solve $\texttt{BUF}_\infty$), the so-called "sandwich problem". In the sandwich problem, the input consists of n elements $v_1, \ldots, v_n$ and two weight functions w_ℓ and w_h, and the goal is to output an ultrametric dist^U such that $\forall v_i, v_j \in V$, $w_\ell(v_i, v_j) \leq \text{dist}^U(v_i, v_j) \leq \alpha \cdot w_h(v_i, v_j)$ for the minimal α. Observe that an algorithm that solves the sandwich problem can be used to solve BEST ULTRAMETRIC FIT by setting $w_\ell = w_h = w$.

We now review the algorithm of (Farach et al., 1995). Given a tree T over the elements of V and an edge $e \in T$, removing e from the T creates two connected components, we call $L(e)$ and $R(e)$ the set of elements in these connected components respectively. Given $L(e)$ and $R(e)$, we define $P(e)$ to be the set of pairs of elements $v_i \in L(e)$ and $v_j \in R(e)$ such that the maximum weight of an edge of the path from v_i to v_j in T is $w_\ell(e)$.

A **cartesian tree** of a weighted tree T is a rooted tree T_C defined as follows: the root of T_C corresponds to the edge of maximal weight and the two children of T_C are defined recursively as the cartesian trees of $L(e)$ and $R(e)$, respectively. The leaves of T_C correspond to the nodes of T. Each node has an associated height. The height of any leaf is set to 0. For a non-leaf node $u \in T_C$, we know that u corresponds, by construction, to an edge e_u in T, which is the first edge (taken in decreasing order w.r.t their weights) that separates v_i from v_j in T. Set the height of u to be equal to the weight of e_u in T. A cartesian tree T_C naturally induces an ultrametric dist^{T_C}: the distance between two points v_i and v_j (i.e., two leaves of T_C) is defined as the height of their least common ancestor in T_C.

Finally, we define the **cut weight** of edge e to be

$$CW(e) = \max_{(v_i, v_j) \in P(e)} w_\ell(v_i, v_j).$$

The algorithm of (Farach et al., 1995) is as follow:

1. Compute a minimum spanning tree (MST) T over the complete graph G_h defined on V and with edge weights w_h;
2. Compute the cut weights with respect to the tree T;
3. Construct the cartesian tree T_C of the tree T' whose the structure is identical to T and the distance from an internal node of T_C to the leaves of its subtree is given by the cut weight of the corresponding edge in T.
4. Output the ultrametric induced by the tree metric of T_C.

The following theorem is proved in (Farach et al., 1995):

Theorem 1.3. *Given two weight functions w_ℓ and w_h, the above algorithm outputs an ultrametric dist^U such that for all $v_i, v_j \in V$*

$$w_\ell(v_i, v_j) \leq \text{dist}^U(v_i, v_j) \leq \alpha_{OPT} \cdot w_h(v_i, v_j)$$

for the minimal α_{OPT}.

2. APPROXBUF: An Approximation Algorithm for BUF_∞

In this section, we describe a new approximation algorithm for BUF_∞ and prove its correctness. We then show in the next section how it can be implemented efficiently for Euclidean inputs.

Given a spanning tree T over a graph G, any edge $e = (v_i, v_j) \in G \setminus T$ induces a unique cycle C_e^T which consists of the union of e and the unique path from x to y in T. We say that a tree T is a γ-*approximate Kruskal tree* (or shortly a γ-KT) if

$$\forall e \in G \setminus T, w(e) \geq \frac{1}{\gamma} \max_{e' \in C_e^T} w(e').$$

Moreover, given a tree T and and an edge e of T, we say that $\beta \in \mathbb{R}$ is a γ-estimate of $CW(e)$ if $CW(e) \leq \beta \leq \gamma \cdot CW(e)$. By extension, we say that a function

$$ACW : V \times V \mapsto \mathsf{R}$$

is a γ-estimate of the cut weights CW if, for any edge e, $ACW(e)$ is a γ-estimate of $CW(e)$.

The rest of this section is dedicated to proving that the following algorithm achieves a $\gamma\delta$-approximation to $\texttt{BUF}_\infty$, for some parameters $\gamma \geq 1, \delta \geq 1$ of the algorithm.

1. Compute a γ-KT T over the complete graph G_h defined on V and with edge weights w_h;
2. Compute a δ-estimate ACW of the cut weights of all the edge of the tree T;
3. Construct the cartesian tree T_C of the tree T' whose the structure is identical to T and the distance from an internal node of T_C to the leaves of its subtree is given by the ACW of the corresponding edge in T.
4. Output the ultrametric dist^{T_C}.

We want to prove the following:

Theorem 2.1. *For any $\gamma \geq 1, \delta \geq 1$, the above algorithm outputs an ultrametric dist^{T_C} which is a $\gamma\delta$-approximation to $\texttt{BUF}_\infty$, meaning that for all $v_i, v_j \in V$*

$$w_\ell(v_i, v_j) \leq \textit{dist}^{T_C}(v_i, v_j) \leq \gamma \cdot \delta \cdot \alpha_{\textit{OPT}} \cdot w_h(v_i, v_j)$$

Proof.

First step: we prove that the γ-KT T computed at the first step of the algorithm can be seen an exact MST for a complete weighted graph G' defined on V and with a weight function w' satisfying

$$\forall v_i, v_j \in V, w'(v_i, v_j) \leq \gamma \cdot w_h(v_i, v_j).$$

We construct w' in the following way. For each pair of points (v_i, v_j):

- If $(v_i, v_j) \in T$, then set $w'(v_i, v_j) = w_h(v_i, v_j)$
- If $(v_i, v_j) \notin T$, then set $w'(v_i, v_j) = \gamma w_h(v_i, v_j)$.

By construction, it is clear that $w' \leq \gamma \cdot w_h$. To see that T is an (exact) MST of G', consider any MST F of G'. If $e = (v_i, v_j) \in F \backslash T$, then consider the first edge e' in the unique path from v_i to v_j in T that reconnects $F \setminus e$. By definition of w', we have $w'(e') = w_h(e')$ and $w'(e) = \gamma w_h(e)$. Since T is a γ-KT, we also have that $w_h(e) \geq \frac{1}{\gamma} w_h(e')$. Therefore $w'(e') \leq w'(e)$ and $\{F \cup e'\} \setminus e$ is a spanning tree of G' of weight smaller than or equal to the weight of F. This proves that $\{F \cup e'\} \setminus e$ is also a MST. Doing this process for all edges not in T gives eventually T and proves that T is also a MST of G', as desired.

Second step. Observe that the weight function w_h is not involved in steps 2,3 and 4 of the algorithm. Therefore, if steps 2,3,4 of the algorithm are made without approximation (meaning that we compute the exact cut weights CW associated to the γ-KT tree T and we output the ultrametric to the corresponding cartesian tree), then the output would be an ultrametric dist such that for all $v_i, v_j \in V$

$$w_\ell(v_i, v_j) \leq \text{dist}(v_i, v_j) \leq \alpha'_{\text{OPT}} \cdot w'(v_i, v_j) \qquad (1)$$

for the minimal such α'_{OPT}. This follows directly from Theorem 1.3 and the fact that T is an exact MST for the graph G' defined above. Note that $\alpha'_{\text{OPT}} \leq \alpha_{\text{OPT}}$ where α_{OPT} denotes the minimal constant such that there exists an ultrametric between w_l and $\alpha_{\text{OPT}} \cdot w_h$.

Now, consider the ultrametric dist^{T_C} associated to T and a δ-estimate ACW of the cut weights. We claim that for all $v_i, v_j \in V$

$$\text{dist}(v_i, v_j) \leq \text{dist}^{T_C}(v_i, v_j) \leq \delta \cdot \text{dist}(v_i, v_j). \qquad (2)$$

To see this, take any $v_i, v_j \in V$. By definition, $\text{dist}^{T_C}(v_i, v_j) = ACW(e)$ for the first edge e (taken in decreasing order w.r.t to ACW) that separates v_i from v_j in T. Let e_{v_i,v_j} be the first edge that separates v_i from v_j w.r.t to the actual cut weights CW. Again, we have by definition that $\text{dist}(v_i, v_j) = CW(e)$. We have that $ACW(e) \geq ACW(e_{v_i,v_j})$ since e is the first edge w.r.t ACW. Moreover $ACW(e_{v_i,v_j}) \geq CW(e_{v_i,v_j})$ because ACW is a δ-estimate of the cut weights: this gives us the first desired inequality

$$\text{dist}^{T_C}(v_i, v_j) \geq \text{dist}(v_i, v_j).$$

The upper bound is similar. We know that $ACW(e) \leq \delta \cdot CW(e)$ since ACW is a δ-estimate. We also have that $CW(e) \leq CW(e_{v_i,v_j})$ since e_{v_i,v_j} is the first separating edge w.r.t CW. This gives:

$$\text{dist}^{T_C}(v_i, v_j) \leq \delta \cdot \text{dist}(v_i, v_j).$$

All together, Equations 1 and 2 imply

$$\begin{aligned} w_\ell(v_i, v_j) \leq \text{dist}^{T_C}(v_i, v_j) &\leq \gamma \cdot \alpha'_{\text{OPT}} \cdot w'(v_i, v_j) \\ &\leq \gamma \cdot \delta \cdot \alpha'_{\text{OPT}} \cdot w_h(v_i, v_j) \\ &\leq \gamma \cdot \delta \cdot \alpha_{\text{OPT}} \cdot w_h(v_i, v_j) \end{aligned}$$

as desired. □

3. A Fast Implementation of ApproxBUF in Euclidean Space – Proof of Theorem 1.1

In this section, we consider inputs of BUF_∞ that consists of a set of points V in $\mathbb{R}^d$, and so for which $w(v_1, v_2) = ||v_1 - v_2||_2$. We now explain how to implement ApproxBUF efficiently for $\gamma \geq 1$ and $\delta = 5$.

Fast Euclidean γ-KT. For computing efficiently a γ-KT of a set of points in a Euclidean space of dimension d, we appeal to the result of Har-Peled et al. (2013) (if interested in doubling metrics, one can instead use the bound of Filtser and Neiman (2018)). The approach relies on *spanners*; A c-spanner of a set S of n points in $\mathbb{R}^d$ is a graph $G = (S, E)$ and a weight function $w : E \mapsto \mathbb{R}_+$ such that for any $u, v \in S$, the shortest path distance in G under the edge weights induced by w, $\text{dist}^G(u, v)$ satisfies $||u - v||_2 \leq \text{dist}^G(u, v) \leq c \cdot ||u - v||_2$.

The result of Har-Peled et al. (2013) states that there is an algorithm that for any set S of n points in $\mathbb{R}^d$ produces an $O(\gamma)$-spanner for S with $O(n^{1+1/c^2} \log^2 n)$ edges in time $O(nd + n^{1+1/c^2} \log^2 n)$. The algorithm uses the locality sensitive hash family of Andoni and Indyk (2006), or alternatively for $\gamma = \sqrt{\log n}$ the Lipschitz partitions of Charikar et al. (1998).

An immediate application of Kruskal classic algorithm for computing a minimum spanning tree on the spanner yields an algorithm with running time $O(nd + n^{1+1/c^2} \log^3 n)$. Moreover, we claim that a minimum spanning tree on a c-spanner G is indeed a c-KT for the original point set. Assume towards contradiction that this is not the case. Then there exists an edge $e = (u, v) \notin T$ such that $||u - v||_2 < \max_{(x,y)\in C_e^T} ||x - y||_2/c$. By correctness of the c-spanner we have that $\text{dist}^G(u, v) \leq c||u - v||_2 < \max_{(x,y)\in C_e^T} ||x - y||_2 \leq \max_{(x,y)\in C_e^T} \text{dist}^G(x, y)$. A contradiction to the fact that T is an MST of the c-spanner.

Fast Estimation of the Cut Weights. We explain how to compute in time $O(nd + n \log n)$ a 5-estimate of the cut weights. To do this, we maintain a disjoint-set data structure on X with the additional property that each equivalence class C (we call such an equivalence class *cluster*) has a special vertex r_C and we store m_C the maximal distance between r_C and a point in the cluster. We now consider the edges of the MST T in increasing order (w.r.t their weights). When at edge $e = (x, y)$, we look at the two clusters C and D coming from the equivalence classes that respectively contain x and y. We claim that

$$E = 5. \max(d(r_C, r_D), m_C - d(r_C, r_D), m_D - d(r_C, r_D))$$

is a 5-approximation of the cut weight for e. To see this, observe that if x, y are the farthest points respectively in C, D, then:

$$\begin{aligned} d(x, y) &\leq d(x, r_C) + d(r_C, r_D) + d(r_D, y) \\ &\leq d(x, r_C) - d(r_C, r_D) + 3d(r_C, r_D) \\ &\qquad + d(r_D, y) - d(r_C, r_D) \\ &\leq 5. \max(d(r_C, r_D), m_C - d(r_C, r_D), \\ &\qquad m_D - d(r_C, r_D)) \\ &\leq E \end{aligned}$$

On the other hand

$$\begin{aligned} d(r_C, r_D) &\leq d(x, y) \\ m_C - d(r_C, r_D) &\leq d(x, r_D) \leq d(x, y) \\ m_D - d(r_C, r_D) &\leq d(y, r_C) \leq d(x, y) \end{aligned}$$

and therefore $E \leq 5.d(x, y)$, as wanted.

Merging C and D can simply be done via a classic disjoint-set data structure. Thus, the challenge is to update $m_{C\cup D}$. To do so, we consider the smallest cluster, say D, query $d(x, r_C)$ for each point $x \in D$ and update accordingly $r_{C\cup D}$ if a bigger value is found. Therefore the running time to update $m_{C\cup D}$ is $O(|D| \times d)$ (we compute $|D|$ distances in a space of dimension d). The overall running time to compute the approximate cut weights is $O(nd+n \log n)$: sorting the edges requires $O(n \log n)$ and constructing bottom-up the cut-weights with the disjoint-set data structure takes $O(nd + n\alpha(n))$, where $\alpha(n)$ denotes the inverse of the Ackermann function (this part comes from the disjoint-set structure). To conclude, note that $n\alpha(n)$ is much smaller than $n \log n$.

4. Hardness of BUF_∞ for High-Dimensional Inputs

We complement Theorem 1.1 with a hardness of approximation result in this section. Our lower bound is based on the well-studeid Strong Exponential Time Hypothesis (SETH) (Impagliazzo and Paturi, 2001; Impagliazzo et al., 2001; Calabro et al., 2006) which roughly states that SAT on n variables cannot be solved in time less than $2^{n(1-o(1))}$. SETH is a popular assumption to prove lower bounds for problems in P (see the following surveys (Williams, 2015; 2016; 2018) for a discussion).

Theorem 4.1. *Assuming* SETH*, for every $\varepsilon > 0$, no algorithm running in time $n^{2-\varepsilon}$ can, given as input an instance of* BUF_∞ *consisting n points of dimension $d := O_\varepsilon(\log n)$ in ℓ_∞-metric, distinguish between the following two cases.*

Completeness: *There is an isometric ultrametric embedding.*

Soundness: *The distortion of the best ultrametric embedding is at least* $3/2$.

Note that the above theorem morally[2] rules out approximation algorithms running in subquadratic time which can approximate the best ultrametric to $3/2 - o(1)$ factor.

Finally, we remark that all the results in this section can be based on a weaker assumption called the Orthogonal Vectors Hypothesis (Williams, 2005) instead of SETH. Before we proceed to the proof of the above theorem, we prove below a key technical lemma.

Definition 4.2 (Point-set S^*)**.** *For every* $\gamma, \gamma' \geq 0$ *and every* $p \in \mathbb{R}_{\geq 1} \cup \{\infty\}$*, we define the discrete point-set* $S^*(\gamma, \gamma', p) := \{a, a', b\}$ *in the* ℓ_p*-metric as follows:*

$$\|a-b\|_p \leq 1,\ \|a-a'\|_p \leq 1+\gamma',\ \text{and}\ \|a'-b\|_p \geq 1+\gamma.$$

Lemma 4.3 (Distortion in Ultrametric Embedding)**.** *Fix* $\gamma, \gamma' \geq 0$ *and* $p \in \mathbb{R}_{\geq 1} \cup \{\infty\}$*. Then we have that any embedding of* $S^*(\gamma, \gamma', p) := \{a, a', b\}$ *into ultrametric incurs a distortion of at least* $\frac{1+\gamma}{1+\gamma'}$.

Proof. Let the distortion of S^* to the ultrametric be at most ρ. Let T^* be a tree with exactly 3 leaves $v_a, v_{a'}$, and v_b. For any two nodes v, v' of T^* let $\Delta(v, v')$ be the length of the shortest path from v to v'. Let $\alpha \in \mathbb{R}^+$ be the scaling factor of the embedding from the ℓ_p-metric to the ultrametric.

$$\begin{aligned}(1+\gamma)\cdot\alpha &\leq \Delta(v_{a'}, v_b)\\ &\leq \max\{\Delta(v_a, v_b), \Delta(v_a, v_{a'})\}\\ &\leq \rho\cdot(1+\gamma')\cdot\alpha\end{aligned}$$

Thus we have that $\rho \geq \frac{1+\gamma}{1+\gamma'}$. □

We combine the above lemma with David et al.'s conditional lower bound (stated below) on approximating the Bichromatic Closest Pair problem in the ℓ_∞-metric to obtain Theorem 4.1.

Theorem 4.4 ((David et al., 2019))**.** *Assuming* SETH*, for any* $\varepsilon > 0$*, no algorithm running in time* $n^{2-\varepsilon}$*, given* $A, B \subseteq \mathbb{R}^d$ *as input, where* $|A| = |B| = n$ *and* $d = O_\varepsilon(\log n)$*, distinguish between the following two cases:*

Completeness: *There exists* $(a, b) \in A \times B$ *such that* $\|a - b\|_\infty = 1$.

Soundness: *For every* $(a, b) \in A \times B$ *we have* $\|a-b\|_\infty = 3$.

Moreover this hardness holds even with the following additional properties:

- *Every distinct pair of points in* A *(resp.* B*) are at distance 2 from each other in the* ℓ_∞*-metric.*
- *All pairs of points in* $A \times B$ *are at distance either 1 or 3 from each other in the* ℓ_∞*-metric.*

Proof of Theorem 4.1. Let (A, B) be the input to the hard instances of the Bichromatic Closest Pair problem as given in the statement of Theorem 4.4 (where $A, B \subseteq \mathbb{R}^d$ and $|A| = |B| = n$). We show that if for every $(a, b) \in A \times B$ we have $\|a-b\|_\infty = 3$ then there is an isometric embedding of $A \cup B$ into an ultrametric and if there exists $(a, b) \in A \times B$ such that $\|a - b\|_\infty = 1$ then any embedding of $A \cup B$ to an ultrametric incurs a distortion of $3/2$. Once we show this, the proof of the theorem statement immediately follows.

Suppose that for every $(a, b) \in A \times B$ we have $\|a-b\|_\infty = 3$. We construct the following ultrametric embedding. Let T be a tree with root r. Let r have two children c_A and c_B. Both c_A and c_B each have n leaves which we identify with the points in A and points in B respectively. Then we subdivide the edge between c_A and its leaves and c_B and its leaves. Notice that any pair of leaves corresponding to two distinct points in A (resp. in B) are at distance four away in T. Also notice that any pair of leaves corresponding to a pair of points in $A \times B$ are at distance six. Therefore the aforementioned embedding is isometric.

Next, suppose that there exists $(a, b) \in A \times B$ such that $\|a - b\|_\infty = 1$. We also suppose that there exists $(a', b) \in A \times B$ such that $\|a' - b\|_\infty = 3$. We call Lemma 4.3 with the point-set $\{a, a', b\}$ and parameters $\gamma = 2$ and $\gamma' = 1$. Thus we have that even just embedding $\{a, a', b\}$ into an ultrametric incurs distortion of $3/2$. □

One may wonder if one can extend Theorem 4.1 to other ℓ_p-metrics to rule out approximation algorithms running in subquadratic time which can approximate the best ultrametric to arbitrary factors close to 1. More concretely, one may look at the hardness of approximation results of (Rubinstein, 2018; Karthik C. S. and Manurangsi, 2019) on Closest Pair problem, and try to use them as the starting point of the reduction. An immediate obstacle to do so is that in the soundness case of the closest pair problem (i.e., the completeness case of the computing ultrametric distortion problem), there is no good bound on the range of all pairwise distances, and thus the distortion cannot be estimated to yield a meaningful reduction.

Nonetheless, we introduce a new complexity theoretic hypothesis in the supplementary material and show how that extends Theorem 4.1 to other ℓ_p-metrics.

[2] We say "morally" because our hardness results are for the decision version, but doesn't immediately rule out algorithms that find approximately optimal embedding, as computing the distortion of an embedding (naively) requires n^2 time. So the search variant cannot be naively reduced to the decision variant.

5. Experiments

We present some experiments performed on three standard datasets: DIABETES (768 samples, 8 features), MICE (1080 samples, 77 features), PENDIGITS (10992 samples, 16 features) and compare our C++ implementation of the algorithm described above to the classic linkage algorithms (average, complete, single or ward) as implemented in the Scikit-learn library (note that the implementation is also in C++). The measure we are interested in is the maximum distortion $\max_{(u,v)\in P} \frac{\Delta(u,v)}{\|u-v\|_2}$, where P is the dataset and Δ the ultrametric output by the algorithm. Note that average linkage, single and ward linkage can underestimate distances, i.e., $\frac{\Delta(u,v)}{\|u-v\|_2} < 1$ for some points u and v. In practice, the smallest ratio given by average linkage lies often between 0.4 and 0.5 and between 0.8 and 0.9 for ward linkage. For single linkage, the maximum distortion is always 1 and hence the minimum distortion can be very small. For a fair comparison, we normalize the ultrametrics by multiplying every distances by the smallest value for which $\frac{\Delta(u,v)}{\|u-v\|_2}$ becomes greater than or equal to 1 for all pairs. Note that what matters most in hierarchical clustering is the structure of the tree induced by the ultrametric and performing this normalization (a uniform scaling) does not change this structure.

ApproxBUF stands for the C++ implementation of our algorithm. To compute the γ-approximate Kruskal tree, we implemented the idea from Har-Peled et al. (2013), that uses the locality-sensitive hash family of Andoni and Indyk (2006) and runs in time $O(nd + n^{1+1/\gamma^2}\log^2 n)$. The parameter γ is related to choices in the design of the locality-sensitive hash family. It is hard to give the precise γ that we choose during our experiments since this one relies on theoretical and asymptotic analysis. However, we choose parameters to have, in theory, a γ around 2.5. Observe that our algorithm is roughly cut into two distinct parts: computing a γ-KT tree T, and using T to compute the approximate cut weights and the corresponding cartesian tree. Each of these parts play a crucial role in the approximation guarantees. To understand better how important it is to have a tree T close to an exact MST, we also implemented a slight variant of **ApproxBUF**, namely **ApproxBUF2**, in which T is replaced by an exact MST. The best known algorithm for computing an exact MST of a set of high-dimensional set of points is $\Theta(n^2)$ and so **ApproxBUF** did not exhibit a competitive running time and was not included in Figure 1.

Table 1 shows the maximum distortions of the different algorithms. For the linkage algorithms, the results are deterministic hence exact (up to rounding) while the output of our algorithm is probabilistic (this probabilistic behavior comes from the locality-sensitive hash families). We performed 100 runs for each datasets. . We observe that **ApproxBUF** performs better than Ward's method while being not too far from the others . **ApproxBUF2** performs almost better than all algorithms except single linkage, this emphasizes the fact that finding efficiently accurate γ-KT is important.

	DIABETES	MICE	PENDIGITS
Average	11.1	9.7	27.5
Complete	18.5	11.8	33.8
Single	6.0	4.9	14.0
Ward	61.0	59.3	433.8
ApproxBUF	41.0	51.2	109.8
ApproxBUF2	9.6	9.4	37.2

Table 1. Max distortions

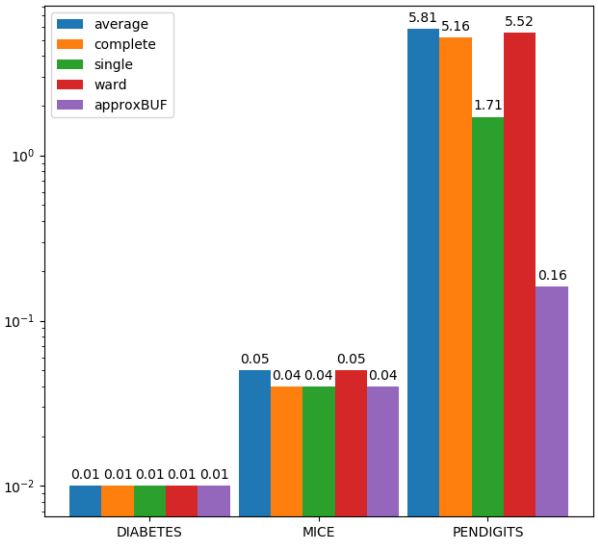

Figure 1. Average running time, in seconds. Logarithmic scale.

Figure 1 shows the average running time, rounded to 10^{-2} seconds. We see that for small datasets, **ApproxBUF** is comparable to linkage algorithms, while **ApproxBUF** is much faster on a large dataset, as the complexity analysis predicts (roughly 36 times faster than the slowest linkage algorithm and 10 times faster than the fastest one).

References

Abboud, A., Cohen-Addad, V., and Houdrouge, H. (2019). Subquadratic high-dimensional hierarchical clustering. In Wallach, H. M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E. B., and Garnett, R., editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada*, pages 11576–11586.

Andoni, A. and Indyk, P. (2006). Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In *2006 47th annual IEEE symposium on foundations of computer science (FOCS'06)*, pages 459–468. IEEE.

Balcan, M.-F., Blum, A., and Vempala, S. (2008). A discriminative framework for clustering via similarity functions. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 671–680. ACM.

Calabro, C., Impagliazzo, R., and Paturi, R. (2006). A duality between clause width and clause density for SAT. In *21st Annual IEEE Conference on Computational Complexity (CCC 2006), 16-20 July 2006, Prague, Czech Republic*, pages 252–260.

Carlsson, G. and Mémoli, F. (2010). Characterization, stability and convergence of hierarchical clustering methods. *Journal of machine learning research*, 11(Apr):1425–1470.

Charikar, M. and Chatziafratis, V. (2017). Approximate hierarchical clustering via sparsest cut and spreading metrics. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 841–854. Society for Industrial and Applied Mathematics.

Charikar, M., Chatziafratis, V., and Niazadeh, R. (2019). Hierarchical clustering better than average-linkage. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2291–2304. SIAM.

Charikar, M., Chatziafratis, V., Niazadeh, R., and Yaroslavtsev, G. (2018). Hierarchical clustering for euclidean data. *arXiv preprint arXiv:1812.10582*.

Charikar, M., Chekuri, C., Goel, A., Guha, S., and Plotkin, S. (1998). Approximating a finite metric by a small number of tree metrics. In *Proceedings 39th Annual Symposium on Foundations of Computer Science (Cat. No. 98CB36280)*, pages 379–388. IEEE.

Cochez, M. and Mou, H. (2015). Twister tries: Approximate hierarchical agglomerative clustering for average distance in linear time. In *Proceedings of the 2015 ACM SIGMOD international conference on Management of data*, pages 505–517. ACM.

Cohen-Addad, V., Kanade, V., and Mallmann-Trenn, F. (2017). Hierarchical clustering beyond the worst-case. In *Advances in Neural Information Processing Systems*, pages 6201–6209.

Cohen-Addad, V., Kanade, V., Mallmann-Trenn, F., and Mathieu, C. (2018). Hierarchical clustering: Objective functions and algorithms. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 378–397. SIAM.

Dasgupta, S. (2015). A cost function for similarity-based hierarchical clustering. *arXiv preprint arXiv:1510.05043*.

David, R., Karthik C. S., and Laekhanukit, B. (2019). On the complexity of closest pair via polar–pair of point–sets. *SIAM J. Discrete Math.*, 33(1):509–527.

Farach, M., Kannan, S., and Warnow, T. J. (1995). A robust model for finding optimal evolutionary trees. *Algorithmica*, 13(1/2):155–179.

Filtser, A. and Neiman, O. (2018). Light spanners for high dimensional norms via stochastic decompositions. In *26th Annual European Symposium on Algorithms, ESA 2018, August 20-22, 2018, Helsinki, Finland*, pages 29:1–29:15.

Har-Peled, S., Indyk, P., and Sidiropoulos, A. (2013). Euclidean spanners in high dimensions. In *Proceedings of the twenty-fourth annual ACM-SIAM symposium on Discrete algorithms*, pages 804–809. SIAM.

Impagliazzo, R. and Paturi, R. (2001). On the complexity of k-sat. *J. Comput. Syst. Sci.*, 62(2):367–375. Preliminary version in CCC'99.

Impagliazzo, R., Paturi, R., and Zane, F. (2001). Which problems have strongly exponential complexity? *J. Comput. Syst. Sci.*, 63(4):512–530. Preliminary version in FOCS'98.

Karthik C. S. and Manurangsi, P. (2019). On closest pair in euclidean metric: Monochromatic is as hard as bichromatic. In *10th Innovations in Theoretical Computer Science Conference, ITCS 2019, January 10-12, 2019, San Diego, California, USA*, pages 17:1–17:16.

Moseley, B. and Wang, J. (2017). Approximation bounds for hierarchical clustering: Average linkage, bisecting k-means, and local search. In *Advances in Neural Information Processing Systems*, pages 3094–3103.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Roy, A. and Pokutta, S. (2016). Hierarchical clustering via spreading metrics. In *Advances in Neural Information Processing Systems*, pages 2316–2324.

Rubinstein, A. (2018). Hardness of approximate nearest neighbor search. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 1260–1268.

Williams, R. (2005). A new algorithm for optimal 2-constraint satisfaction and its implications. *Theor. Comput. Sci.*, 348(2-3):357–365.

Williams, V. V. (2015). Hardness of easy problems: Basing hardness on popular conjectures such as the strong exponential time hypothesis (invited talk). In *10th International Symposium on Parameterized and Exact Computation, IPEC 2015, September 16-18, 2015, Patras, Greece*, pages 17–29.

Williams, V. V. (2016). Fine-grained algorithms and complexity (invited talk). In *33rd Symposium on Theoretical Aspects of Computer Science, STACS 2016, February 17-20, 2016, Orléans, France*, pages 3:1–3:1.

Williams, V. V. (2018). On some fine-grained questions in algorithms and complexity. In *Proc. Int. Cong. of Math.*, volume 3, pages 3431–3472.