Hierarchical Clustering for Euclidean Data

Moses Charikar
Stanford University

Vaggos Chatziafratis
Stanford University

Rad Niazadeh
Stanford University

Grigory Yaroslavtsev
Indiana University, Bloomington

Abstract

Recent works on Hierarchical Clustering (HC), a well-studied problem in exploratory data analysis, have focused on optimizing various objective functions for this problem under *arbitrary* similarity measures. In this paper we take the first step and give novel scalable algorithms for this problem tailored to Euclidean data in $\mathbb{R}^d$ and under vector-based similarity measures, a prevalent model in several typical machine learning applications. We focus primarily on the popular Gaussian kernel and other related measures, presenting our results through the lens of the objective introduced recently by [MW17]. We show that the approximation factor in [MW17] can be improved for Euclidean data. We further demonstrate both theoretically and experimentally that our algorithms scale to very high dimension d, while outperforming average-linkage and showing competitive results against other less scalable approaches.

1 Introduction

Hierarchical clustering is a popular data analysis method, with various applications in data mining [Ber06], phylogeny [ESBB98], and even finance [TLM10]. In practice, simple agglomerative procedures like average-linkage, single-linkage and complete-linkage are often used for this task. (See the book by Manning, Raghavan, Schütze [MRS08] for a comprehensive discussion). While the study of hierarchical clustering has focused on algorithms, there was a lack of objective functions to measure the quality of the output and compare the performance of algorithms. To remedy this, Dasgupta [Das16] recently introduced and studied an interesting objective function for hierarchical clustering for similarity measures.

The input to the above problem is a set of points V with a similarity measure w_{ij} between i and j. Given a hierarchical clustering represented by a tree $\mathcal{T}$ whose leaves correspond to points in V, Dasgupta's objective is defined as $\mathcal{F}^-(\mathcal{T}) = \sum_{(i,j)\in E} w_{ij}|\{x \in T(i,j)\}|$, where $T(i,j)$ is the subtree of $\mathcal{T}$ rooted in the least common ancestor of i and j in $\mathcal{T}$. [Das16] showed that solutions obtained from minimizing this objective has several desirable properties, which prompted a line of work on objective driven algorithms for hierarchical clustering, resulting in new algorithms as well as shedding light on the performance of classical methods [RP16, CC17, CAKMTM18].

Recently, this viewpoint has been applied to understand the performance of average linkage agglomerative clustering, one of the most popular algorithms used in practice. Moseley and Wang [MW17] introduced a new objective, in some sense dual to the objective introduced by Dasgupta:

$$\mathcal{F}^+(\mathcal{T}) = \sum_{(i,j)\in E} w_{ij}(|V| - |\{x \in T(i,j)\}|),$$

They showed that Average-Linkage obtains a $\frac{1}{3}$-approximation for maximizing this objective function.

It turns out that a random solution also achieves a $\frac{1}{3}$-approximation ratio for this problem. Recently [CCN19] showed that in the worst case, the approximation ratio achieved by Average-Linkage is no better than $\frac{1}{3}$ for maximizing $\mathcal{F}^+(\mathcal{T})$. They also gave an SDP based algorithm that achieves a $(\frac{1}{3}+\epsilon)$-approximation for the problem, for small constant ϵ.

One drawback of the prior work on these hierarchical clustering objectives is the fact that they all consider arbitrary similarity scores (specified as an $n \times n$ matrix); however, there is much more structure to such similarity scores in practice. In this paper, we initiate the study of the commonly encountered case of *Euclidean data*, where the similarity score w_{ij} is computed by applying a monotone decreasing function to the Euclidean distance between i and j. Roughly speaking, we show

Proceedings of the 22[nd] International Conference on Artificial Intelligence and Statistics (AISTATS) 2019, Naha, Okinawa, Japan. PMLR: Volume 89.

how to exploit this structure to design better/faster algorithms for hierarchical clustering and how to re-analyze algorithms commonly used in practice. In this paper, we focus on the problem of maximizing $\mathcal{F}^+(\mathcal{T})$, introduced by Moseley and Wang [MW17].

Arguably the most common distance-based similarity measure used for Euclidean data is the Gaussian kernel. Here we use the spherical version $w_{ij} \sim \exp(-\|v_i - v_j\|/2\sigma^2)$. The parameter σ^2, referred to as the *bandwidth*, plays an important role for applications and a large body of literature exists on selection of this parameter [ZMP05].

As discussed above, devising simple practical algorithms that improve on the $\frac{1}{3}$-approximation for general similarity measures appears to be a major challenge as observed in [MW17, CCN19]. In this paper, *we show that this approximation can be improved for Euclidean data, through fast algorithms that can be scaled to very large datasets.* While it might seem that the case of Euclidean data with the Gaussian kernel is a very restricted class of inputs to the HC problem, we show that for suitably high dimensions and suitable small choice of bandwidth σ^2 it can simulate arbitrary similarity scores (scaled appropriately). Thus any improvements to approximation guarantees for the Euclidean case (that do not apply to general similarity scores) must necessarily involve assumptions about the dimension of the data (not very high) or on σ^2 (not pathologically small). Such assumptions on σ^2 are consistent with common methods for computing the bandwidth parameter based on data.

Contributions: We start with the simplest case of 1-dimensional Euclidean data. Even in this seemingly simple setting, obtaining an efficient algorithm that produces an exactly optimum solution seems non-trivial, motivating the study of approximation algorithms. In Section 3 we prove that two algorithms – Random Cut (RC) and Average-Linkage (AL) – get a $\frac{1}{2}$-approximation (AL achieves this deterministically, while RC only in expectation). This beats the best known approximation for the general case, which is 0.336 [CCN19]. Here RC is substantially faster than AL, with a running time of $O(n \log n)$ vs. $O(n^2)$.

We next consider the high-dimensional case with the Gaussian Kernel and show that Average-Linkage cannot beat the factor $\frac{1}{3}$ even in poly-logarithmic dimensions. We propose the Projected Random Cut (PRC) algorithm that gets a constant improvement over $\frac{1}{3}$, irrespective of the dimension d (the improvement is a function of the ratio of the diameter to σ^2, and drops as this ratio gets large).

Furthermore, PRC runs in $O(n(d + \log n))$ time while Average-Linkage runs in $O(dn^2)$ time. Even single-linkage runs in almost-linear time only for constant d and has exponential dependence on the dimension (see e.g. [YV18]) and it is open whether it can be scaled to large d when n is large.

Experiments: Many existing algorithms have time efficiency shortcomings, and none of them can be used for really large datasets. On the contrary, our Projected Random Cut (see Section 4) is a fast HC algorithm that scales to the largest ML datasets. The running time scales almost linearly and the algorithm can be implemented in one pass without needing to store similarities, so the memory is $O(n)$. We also evaluate its quality on a small dataset (Zoo).

Open Problems: We list a number of open problems: (1) Can one get an improvement over $\frac{1}{3}$ for the problem of maximizing $\mathcal{F}^+(\mathcal{T})$ as a function of d for small d? (2) Can projection on low-dimensional subspaces be used to improve the approximation ratio for the high-dimensional case further? (3) Does Average-Linkage achieve a 3/4-approximation for 1-dimensional data?

2 Preliminaries

Euclidean data. We consider data sets represented as sets of d-dimensional *feature* vectors $v_1, \ldots, v_n \in \mathbb{R}^d$. We focus on similarity measures between pairs of data points, denoted by $[w_{ij}]_{i,j\in[n]}$, where the similarities only depend on the underlying vectors, i.e., $w_{ij} = f(v_i, v_j)$ for some function $f : \mathbb{R}^d \times \mathbb{R}^d \to [0, 1]$ and furthermore are fully determined by monotone functions of distances between them.

Definition 2.1 (Distance-based similarity measure)**.** *A similarity measure $w_{ij} = f(v_i, v_j)$ is "distance-based" if $f(v_i, v_j) = g(\|v_i - v_j\|_2)$ for some function $g\colon \mathbb{R}_{\geq 0} \to [0, 1]$, and is "monotone distance-based" if furthermore $g\colon \mathbb{R}_{\geq 0} \to [0, 1]$ is a monotone non-increasing function.*

As an example of the monotone similarity measure it is natural to consider the *Gaussian kernel similarity*, i.e.,

$$w_{ij} = (\sqrt{2\pi}\sigma)^{-n} e^{-\frac{\|v_i - v_j\|_2^2}{2\sigma^2}}, \qquad (\textit{Gaussian Kernel})$$

where σ is a normalization factor determining the *bandwidth* of the Gaussian kernel [Gär03]. For simplicity, we ignore the multiplicative factor $(\sqrt{2\pi}\sigma)^{-n}$ (unless noted otherwise), as our focus is on multiplicative approximations and scaling has no effects.

Linkage-based hierarchical clustering. Among various algorithms popular in practice for HC, we focus on two very popular ones: *single-linkage* and *average-linkage*, which are two simple agglomerative clustering algorithms that recursively merge pairs of nodes (or super nodes) to create the final binary tree. Single-linkage picks the pair of super nodes with the

maximum similarity weight between their data points, i.e., merges A and B maximizing $\max_{i\in A, j\in B} w_{ij}$. On the contrary, average-linkage picks the pair of super-nodes with maximum average similarity weight at each step, i.e., merges A and B maximizing $\frac{w_{AB}}{|A|\cdot|B|}$, where $w_{AB} := \sum_{i\in A, j\in B} w_{ij}$. Average-linkage has an approximation ratio of $1/3$ for maximizing the $\mathcal{F}^+$ objective function [MW17], and this factor is tight [CCN19].

General upper bounds. In order to analyze the linkage-based clustering algorithms and our proposed algorithms, we propose a natural upper bound on the value of the $\mathcal{F}^+$ objective function. The idea is to decompose the objective function $\mathcal{F}^+$ into contributions of *triple* of vertices:

$$\mathcal{F}^+(\mathcal{T}) = \sum_{i<j<k} \left(w_{ij}\mathbb{1}\left[\mathcal{E}^k_{ij}\right] + w_{jk}\mathbb{1}\left[\mathcal{E}^i_{jk}\right] + w_{ik}\mathbb{1}\left[\mathcal{E}^j_{ik}\right] \right)$$

where $\mathcal{E}^x_{yz}$ denotes the event that x is separated first among the vertices of triple $\{x, y, z\}$ in tree $\mathcal{T}$. Note that the final tree scores only one of the similarity weights between the triple $\{i, j, k\}$. Given this observation, we define the following benchmark:

$$\texttt{MAX-upper} \triangleq \sum_{i<j<k} \max(w_{ij}, w_{jk}, w_{ik}),$$

Clearly, for all trees $\mathcal{T}$, $\mathcal{F}^+(\mathcal{T}) \leq \texttt{MAX-upper}$.

One-dimensional benchmarks. Consider the special case of 1D data points (with any monotone distance-based similarity measure as in Definition 2.1), and suppose $v_1 \leq v_2 \leq \ldots \leq v_n \in \mathbb{R}$. Now, for any triple $i < j < k$, as a simple observation we have $w_{ik} \leq \min(w_{ij}, w_{jk})$. Hence we can modify the above benchmark to obtain two refined new benchmarks:

$$\texttt{1D-MAX-upper} \triangleq \sum_{i<j<k} \max(w_{ij}, w_{jk}),$$

$$\texttt{1D-SUM-upper} \triangleq \sum_{i<j<k} (w_{ij} + w_{jk}).$$

Again, clearly for all trees $\mathcal{T}$ we have:

$$\mathcal{F}^+(\mathcal{T}) \leq \texttt{1D-MAX-upper} \leq \texttt{1D-SUM-upper}$$

3 Hierarchical Clustering in 1D

In this section we look at the extreme case where the feature vectors have $d = 1$, and we try to analyze the popular/natural algorithms existing in this domain by evaluating how well they approximate the objective function $\mathcal{F}^+$. We focus on average-linkage and Random Cut (will be formally defined later). In particular, Random Cut is a building block of our algorithm for high-dimensional data given in Section 4.

We show (i) average-linkage gives a 1/2-approximation, and can obtain no better than 3/4 fraction of the optimal objective value in the worst-case. We then show (ii) Random Cut also is a 1/2-approximation (in expectation) and this factor is tight. In the supplement §C, we discuss other simple algorithms: single-linkage and greedy cutting (will be defined formally later). We start by the observation that greedy cutting and single-linkage output the same tree (and so are equivalent). We finish by showing (iii) there is an instance where single-linkage attains only $\frac{1}{2}$ of the optimum objective value. We further show on this instance both average-linkage and random cutting are almost optimal.

For the rest of this section, suppose we have 1D points $x_1 \leq \ldots, \leq x_n \in \mathbb{R}$, where $w_{ij} = g(\|x_i - x_j\|)$ for some monotone non-increasing function $g : \mathbb{R}_{\geq 0} \to [0, 1]$.

Average-linkage. In order to simplify the notation, we define w_{AB} to be $\sum_{i\in A, j\in B} w_{ij}$ for any two sets A and B. We first prove a simple structural property of average-linkage in 1D (proof in the supplement §B).

Lemma 3.1. *For $d = 1$, under any monotone distance-based similarity measure $w_{ij} = g(\|x_i - x_j\|)$, average-linkage can always merge neighbouring clusters.*

Theorem 3.2. *For $d = 1$, under any monotone distance-based similarity measure $w_{ij} = g(\|x_i - x_j\|)$, average-linkage obtains at least $\frac{1}{2}$ of the 1D-SUM-upper, and hence is a $\frac{1}{2}$-approximation for the objective $\mathcal{F}^+$.*

Proof. The proof uses a potential function argument. Given a partitioning of points $x_1 \leq x_2 \leq \ldots \leq x_n$ into sets $S_1, \ldots, S_m$, a triple of points $i < j < k$ is called *separated* if no pair of these three points belong to the same set. Now, the potential function Φ gets $\{S_1, \ldots, S_m\}$ as input, and maps it to a summation over all separated triples by $\{S_i\}_{i\in[m]}$ as below:

$$\Phi(S_1, \ldots, S_m) = \sum_{i<j<k:\ (i,j,k) \text{ is separated}} (w_{ij} + w_{jk})$$

Note that $\Phi(\{x_1\}, \ldots, \{x_n\}) = \texttt{1D-SUM-upper}$, and $\Phi(\{x_1, \ldots, x_n\}) = 0$.

We now run average-linkage. Based on the definition of $\mathcal{F}^+$, every time that average-linkage merges two super nodes A and B it scores $w_{AB} \cdot (n - |A| - |B|)$, and sum over of all these per-step scores is equal to its final objective value. Let Score-AL denote the variable that stores the score of average-linkage over time. At every step we keep track of (1) the change in the potential function, denoted by $\Delta\Phi$ and (2) how much progress average-linkage had towards the final objective value, denoted by ΔScore-AL. In order to prove 1/2-approximation, it suffices to show that at every step of average-linkage we have:

$$\Delta\texttt{Score-AL} + \frac{1}{2}\Delta\Phi \geq 0. \qquad (\star)$$

To see this note that average-linkage starts with all points separated, and ends with one cluster/super node with all the points ($\mathcal{T}_{\text{AL}}$ is the generated tree). Therefore, by summing eq. ($\star$) over all merging steps of average-linkage and canceling the terms in the telescopic sum, we have:

$$(\mathcal{F}^+(\mathcal{T}_{\text{AL}}) - 0) + \frac{1}{2}\left(\Phi(\{x_1,\dots,x_n\}) - \Phi(\{x_1\},\dots,\{x_n\})\right) \geq 0.$$

Plugging values of Φ at the start and the end, we get:

$$\mathcal{F}^+(\mathcal{T}_{\text{AL}}) \geq \frac{1}{2}(\texttt{1D-SUM-upper}),$$

which implies the 1/2-approximation factor.

To prove eq. ($\star$), we focus on a single step of average-linkage where by Lemma 3.1 some two neighboring clusters denoted as A and B get merged. Let C denote the nodes on the left of A and let D denote the nodes on the right of B (Figure 1).[1] By merging two clusters A, B, the change in the score of average-linkage is:

$$\Delta\texttt{Score-AL} = w_{AB}(|C| + |D|)$$

Moreover, any separated triple $i < j < k$ such that

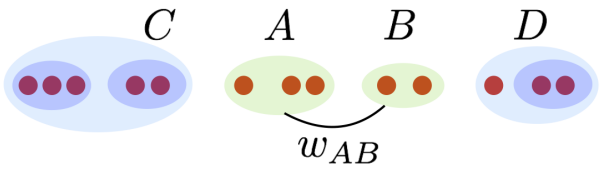

Figure 1: Illustration of the merging process in 1D.

either $i \in C, j \in A, k \in B$ or $i \in A, j \in B, k \in D$ will not be separated anymore after this merge. For each such triple, the potential function drops by $w_{ij} + w_{jk}$. Therefore:

$$-\Delta\Phi = (w_{AB}|C| + w_{AC}|B|) + (w_{AB}|D| + w_{BD}|A|)$$

To compare the two, we show that $w_{AB}(|C| + |D|) \geq w_{AC}|B| + w_{BD}|A|$, and hence:

$$\begin{aligned}\Delta\texttt{Score-AL} &= w_{AB}(|C| + |D|) \geq w_{AC}|B| + w_{BD}|A| \\ &= -\Delta\Phi - \Delta\texttt{Score-AL}\ ,\end{aligned}$$

which implies eq. ($\star$) as desired. To prove the last claim, note that average-linkage picks the pair (A, B) over both (A, C) and (B, D). Therefore, by definition of average-linkage:

$$\frac{w_{AB}}{|A||B|} \geq \frac{w_{AC}}{|A||C|} \implies w_{AB}|C| \geq w_{AC}|B|$$

$$\frac{w_{AB}}{|A||B|} \geq \frac{w_{BD}}{|B||D|} \implies w_{AB}|D| \geq w_{BD}|A|$$

By summing above inequalities, we get $w_{AB}(|C| + |D|) \geq w_{AC}|B| + w_{BD}|A|$, which finishes the proof. □

As a final note, in Section 5 we discuss hard instances for average-linkage under Gaussian kernels when $d = 1$, where we essentially show the result of Theorem 3.2 is tight when comparing against `1D-SUM-upper`, and there is no hope to get an approximation factor better than $\frac{3}{4}$ for average-linkage in general for $d = 1$.

Random Cut. The following algorithm (termed as Random Cut) picks a uniformly random point r in the range $[x_1, x_n]$ and divides the set of points into left and right using r as the splitter. The same process is applied recursively until the leaves are reached.

Algorithm 1 Random Cut

Input: Integer n, points $x_1 \leq \cdots \leq x_n$.
Output: Binary tree with leaves $(x_1, \dots, x_n)$

if $n == 1$ **then**
  return New leaf containing x_1.
end if
Pick $r \sim U([x_1, x_n])$
Let m be the largest integer such that $x_m \leq r$.
Create new internal tree node x.
$x.left =$ Random Cut$(m, x_1, \dots, x_m)$
$x.right =$ Random Cut$(n - m, x_{m+1}, \dots, x_n)$
return x

Lemma 3.3. *For $d = 1$ under any monotone distance-based similarity measure $w_{ij} = g(x_i, x_j)$ the algorithm* Random Cut *obtains at least 1/2 fraction of the `1D-MAX-upper`, and hence gives a $\frac{1}{2}$-approximation for the objective $\mathcal{F}^+$ in expectation.*

Proof. For every triple $i < j < k$ conditioned on partitioning the interval $[i, k]$ for the first time the longer edge amongst (x_i, x_j) and (x_j, x_k) gets cut with probability $p_1 \geq 1/2$ and the shorter with probability $p_2 \leq 1/2$ so that $p_1 + p_2 = 1$. W.l.o.g let's assume that (x_i, x_j) is the longer edge. Then the algorithm Random Cut scores $w_{jk}p_1 + w_{ij}(1 - p_1)$ in expectation for the (i, j, k) triple. Note that:

$$\begin{aligned}&w_{jk}p_1 + w_{ij}(1 - p_1) \\ &= (w_{jk} - w_{ij})\left(p_1 - \frac{1}{2}\right) + \frac{1}{2}(w_{ij} + w_{jk}) \geq \frac{1}{2}(w_{ij} + w_{jk})\end{aligned}$$

By the linearity of expectation taking the sum over all triples $i < j < k$ Random Cut scores at least 1/2 of `1D-MAX-upper` in expectation and hence gives $\frac{1}{2}$-approximation for the objective $\mathcal{F}^+$. □

[1] Note that C, D may be empty sets.

4 Hierarchical Clustering in High Dimensions

We now describe an algorithm Projected Random Cut which we use for high-dimensional data. This algorithm is given as Algorithm 2. It first projects on a random spherical Gaussian vector and then clusters the resulting projections using Random Cut.

Algorithm 2 Projected Random Cut

Input: Integer n, vectors $v_1, \ldots, v_n \in \mathbb{R}^d$.
Output: Binary tree with leaves $(v_1, \ldots, v_n)$

Pick a random Gaussian vector $\mathbf{g} \sim N_d(0, 1)$
Compute dot products $x_i = \langle v_i, \mathbf{g} \rangle$
$x_{i_1}, \ldots x_{i_n} = \text{Sort}(x_1, \ldots, x_n)$
return Random Cut$(n, x_{i_1}, \ldots, x_{i_n})$

Theorem 4.1. *For any input set of vectors $v_1, \ldots, v_n \in \mathbb{R}^d$ the algorithm* Projected Random Cut *gives an α-approximation (in expectation) for the objective $\mathcal{F}^+$ under the Gaussian kernel similarity measure $w_{ij} \sim e^{-\|v_i - v_j\|_2^2/2\sigma^2}$ where $\alpha = (1+\delta)/3$ for $\delta = \min_{i,j} \exp(-\frac{\|v_i - v_j\|_2^2}{2\sigma^2})$.*

Proof. [2] Recall an upper bound on the optimum:

$$OPT \leq \texttt{MAX-upper} = \sum_{i<j<k} \max(w_{ij}, w_{ik}, w_{jk}).$$

Fix any triple (i, j, k) where $i < j < k$. Note that the objective value achieved by the algorithm Projected Random Cut can also be expressed as $ALG = \sum_{i<j<k} ALG_{i,j,k}$ where $ALG_{i,j,k}$ is the contribution to the objective from the triple (i, j, k) defined as follows. Consider the tree constructed by the algorithm. If v_k is the first vector in the triple (v_i, v_j, v_k) to be separated from the other two in the hierarchical partition (starting from the root) then $A_{i,j,k}$ is defined to be w_{ij} (in the other two cases when i or j are separated first the definition is analogous). Note that since Projected Random Cut is a randomized algorithm $ALG_{i,j,k}$ is a random variable. By the linearity of expectation we have $\mathbb{E}[ALG] = \sum_{i<j<k} \mathbb{E}[ALG_{i,j,k}]$. Thus in order to complete the proof it suffices to show that for every $i < j < k$ it holds that:

$$\mathbb{E}[ALG_{i,j,k}] \geq \alpha \max(w_{ij}, w_{ik}, w_{jk}).$$

Fix any triple (v_i, v_j, v_k) which forms a triangle in $\mathbb{R}^d$. Conditioned on cutting this triangle for the first time let (p_{ij}, p_{ik}, p_{jk}) be the vector of probabilities corresponding to the events that the corresponding edge is not cut, i.e., this is the probability that we score the contribution of this edge in the objective. Note that $p_{ij} + p_{ik} + p_{jk} = 1$.

[2] In the supplement, you can find this result (using exactly the same proof) restated in terms of the multiplicative Lipschitz constant of any given distance-based similarity measure.

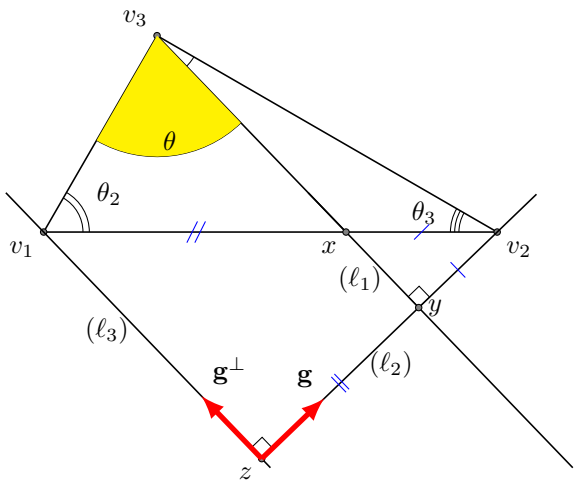

Figure 2: Projecting the triangle (v_1, v_2, v_3) on $\mathbf{g}$.

Consider any triangle whose vertices are v_i, v_j, v_k. To simplify presentation we set $i = 1, j = 2, k = 3$. We can assume that $\|v_2 - v_1\| \geq \|v_2 - v_3\| \geq \|v_1 - v_3\|$. Let $\theta_1 = \angle(v_1 - v_3, v_2 - v_3)$, $\theta_2 = \angle(v_2 - v_1, v_3 - v_1)$ and $\theta_3 = \angle(v_1 - v_2, v_3 - v_2)$ so that $\theta_1 \geq \theta_2 \geq \theta_3$. See Figure 2. Note that the probability that the i-th longest side of the triangle has the longest projection is then θ_i/π.

Lemma 4.2. *If (v_1, v_3) is the shortest edge in the triangle (v_1, v_2, v_3) then it holds that $p_{13} \geq \frac{1}{3}$*

Proof. Suppose that $\mathbf{g}$ forms an angle $\pi/2 - \theta$ with $(v_3 - v_1)$, i.e. the vector $\mathbf{g}^\perp$ orthogonal to $\mathbf{g}$ forms angle θ with $v_3 - v_1$. We define three auxiliary points x, y, z as follows (see also Figure 2.). Let ℓ_1 be a line parallel to $\mathbf{g}^\perp$ going through v_3, let ℓ_2 to be a line parallel to $\mathbf{g}$ going through v_2 and let ℓ_3 be the line parallel to ℓ_1 going through v_1. We then let x be the intersection of ℓ_1 and (v_1, v_2), y be the intersection of ℓ_1 and ℓ_2 and z be the intersection of ℓ_2 and ℓ_3 (see Fig 2).

Thus the projections of $v_3 - v_1$, $v_2 - v_1$ and $v_2 - v_3$ on $\mathbf{g}$ are $y - z$, $v_2 - z$ and $v_2 - y$. Hence conditioned on (v_1, v_2) having the longest projection the probability of scoring the contribution of (v_2, v_3) is given as $p_{23}^{12}(\theta) = \frac{\|y-z\|}{\|z-v_2\|}$ since we are applying the Random Cut algorithm after the projection. Note that by Thales's theorem we have $p_{23}^{12}(\theta) = \frac{\|y-z\|}{\|z-v_2\|} = \frac{\|x-v_1\|}{\|v_2-v_1\|}$. Applying the law

of sines to the triangles (v_1, v_2, v_3) and (x, v_1, v_3) we have:

$$\frac{\sin\theta_1}{\|v_1 - v_2\|} = \frac{\sin(\theta_1+\theta_2)}{\|v_1 - v_3\|}, \quad \frac{\sin\theta}{\|v_1 - x\|} = \frac{\sin(\theta+\theta_2)}{\|v_1 - v_3\|},$$

where we used the fact that $\sin\theta_3 = \sin(\pi - \theta_1 - \theta_2) = \sin(\theta_1 + \theta_2)$ and similarly $\sin(\angle(v_3 - x, v_1 - x)) = \sin(\theta + \theta_2)$. Using the above we can express $p_{23}^{12}(\theta)$ as:

$$p_{23}^{12}(\theta) = \frac{\sin\theta}{\sin(\theta+\theta_2)} \frac{\sin(\theta_1+\theta_2)}{\sin\theta_1}$$

Thus the overall probability of scoring the contribution of the edge (v_1, v_3) conditioned on (v_1, v_2) having the longest projection which we denote as p_{13}^{12} is given as:

$$p_{23}^{12} = \frac{1}{\theta_1}\int_0^{\theta_1} p_{23}^{12} d\theta$$

Similarly, consider the probability p_{13}^{12} of scoring the contribution of (v_1, v_3) conditioned on (v_1, v_2) having the longest projection. We can express it as:

$$p_{13}^{12} = \frac{1}{\theta_1}\int_0^{\theta_1} p_{13}^{12}(\theta) d\theta,$$

where

$$p_{13}^{12}(\theta) = \frac{\sin\theta}{\sin(\theta+\theta_3)} \frac{\sin(\theta_1+\theta_3)}{\sin\theta_1}.$$

Below we will show that $p_{13}^{12} \geq p_{23}^{12}$. In fact, we will show that for any fixed $\theta \in [0, \theta_1]$ it holds that $p_{13}^{12}(\theta) \geq p_{23}^{12}(\theta)$. Comparing the expressions for both it suffices to show that $\frac{\sin(\theta_1+\theta_3)}{\sin(\theta+\theta_3)} \geq \frac{\sin(\theta_1+\theta_2)}{\sin(\theta+\theta_2)}$ for all $\theta \in [0, \theta_1]$. Since $\theta_1 = \pi - \theta_2 - \theta_3$ this is equivalent to:

$$\frac{\sin\theta_3}{\sin(\theta+\theta_2)} \leq \frac{\sin\theta_2}{\sin(\theta+\theta_3)}$$

It suffices to show that:

$$\sin\theta_3 \sin(\theta+\theta_3) \leq \sin\theta_2 \sin(\theta+\theta_2)$$

Using the formula $\sin\alpha\sin\beta = \frac{1}{2}(\cos(\alpha-\beta) - \cos(\alpha+\beta))$ it suffices to show that:

$$\cos(\theta + 2\theta_3) \geq \cos(\theta + 2\theta_2).$$

The above inequality follows for all $\theta \in [0, \pi - \theta_2 - \theta_3]$ since $\theta_3 \leq \theta_2$. This shows that $p_{13}^{12} \geq p_{23}^{12}$. Since the probability that (v_1, v_2) has the longest projection is θ_1/π we have that the probability of scoring (v_1, v_3) and (v_1, v_2) having the longest projection is at least $\frac{\theta_1}{2\pi}$. An analogous argument shows that the probability of scoring (v_1, v_3) and (v_2, v_3) having the longest projection is at least $\frac{\theta_2}{2\pi}$.

Putting things together:

$$p_{13} \geq \frac{1}{2}\frac{\theta_1+\theta_2}{\pi} = \frac{1}{2}\frac{\pi-\theta_3}{\pi} \geq \frac{1}{3},$$

where we used that $\theta_3 \leq \pi/3$ since $\theta_3 \leq \theta_2 \leq \theta_1$. □

We are now ready to complete the proof of Theorem 4.1. Let $\gamma = \frac{\frac{2}{3}\delta}{(1+\delta)}$ and note that $\gamma \geq \delta/3$ since $\delta \leq 1$. If $p_{13} \geq 1/3 + \gamma$ then the desired guarantee follows immediately. Otherwise, if $p_{13} \leq 1/3 + \gamma$ then we have:

$$\begin{aligned}\frac{\mathbb{E}[ALG_{1,2,3}]}{OPT_{1,2,3}} &= \frac{p_{13}w_{13} + p_{12}w_{12} + p_{23}w_{23}}{w_{13}} \\ &\geq \frac{1}{3} + \left(\frac{2}{3} - \gamma\right)\frac{w_{12}}{w_{13}} \geq \frac{1}{3} + \left(\frac{2}{3} - \gamma\right)\delta \\ &= \frac{1}{3} + \frac{2}{3}\frac{\delta}{1+\delta} \geq \frac{1+\delta}{3},\end{aligned}$$

where we used the fact that

$$\frac{w_{12}}{w_{13}} = e^{\frac{\|v_1-v_3\|_2^2 - \|v_1-v_2\|_2^2}{2\sigma^2}} \geq e^{\frac{-\|v_1-v_2\|_2^2}{2\sigma^2}} \geq \delta. \quad \square$$

4.1 Gaussian Kernels with small δ

Theorem 4.1 only provides an improved approximation guarantee for Projected Random Cut compared to the factor 1/3 (i.e., the tight approximation guarantees of average-linkage in high dimensions; see Section 5) if δ is not too small, where $\delta = \min_{i,j}\exp(-\frac{\|v_i-v_j\|_2^2}{2\sigma^2})$. In particular, we get constant improvement if $\delta = \Omega(1)$. Is this a reasonable assumption? Interestingly, we answer this question in the affirmative by showing that if we have $\delta = \exp(-\tilde{\Omega}(n))$, then the Gaussian kernel can encode *arbitrary* similarity weights (up to scaling, which has no effects on multiplicative approximations). For simplicity, we only prove this result for $\{\epsilon, 1\}$ weights here, while it can be generalized to arbitrary weights.

Theorem 4.3. *Given any undirected graph $G = (V, E)$ on n nodes and $\epsilon > 0$, there exist unit vectors $\{k_v\}_{v\in V}$ in $\mathbb{R}^d$ and bandwidth parameter $\sigma \in \mathbb{R}_+$, such that $d = O(n^2)$, $\frac{1}{\sigma^2} = \Omega(n\log(1/\epsilon))$, and for some $\alpha > 0$ we have:*

$$\forall(u,v) \in E : e^{-\frac{\|k_u-k_v\|_2^2}{2\sigma^2}} = \alpha,$$
$$\forall(u,v) \notin E : e^{-\frac{\|k_u-k_v\|_2^2}{2\sigma^2}} = \alpha\epsilon.$$

Proof. Our proof is constructive. Let $d = \binom{n}{2}$. Pick orthogonal vectors $\{x_e\}_{e\in E}$ in $\mathbb{R}^d$ such that $\|x_e\|_2 = \frac{1}{d_u d_v}$, where d_u is the degree of node u in graph G. For each $v \in V$, define $y_v \in \mathbb{R}^d$ as follows:

$$y_v \triangleq \sum_{(u,v)\in E} (d_u d_v) x_{uv}.$$

Note that $\|y_v\|_2^2 = \sum_{(u,v)\in E} d_u^2 d_v^2 \frac{1}{d_u^2 d_v^2} = d_v$ As the next step, pick a set of n orthonormal vectors $\{z_v\}$ in the null space of $\{y_v\}_{v\in V}$. Finally, for each $v \in V$, define the final vector $k_v \in \mathbb{R}^d$ as follows:

$$k_v \triangleq \sqrt{1 - \frac{d_v}{n}} z_v + \sqrt{\frac{1}{n}} y_v$$

First, note that these vectors have unit length:

$$\|k_v\|_2^2 = 1 - \frac{d_v}{n} + \frac{\|y_v\|_2^2}{n} = 1 - \frac{d_v}{n} + \frac{d_v}{n} = 1$$

Now, pick any two vertices u and v. If $(u,v) \notin E$, then:

$$\begin{aligned}\langle k_u, k_v\rangle &= \langle \sqrt{1-\frac{d_v}{n}}z_u + \sqrt{\frac{1}{n}}y_u, \sqrt{1-\frac{d_v}{n}}z_v + \sqrt{\frac{1}{n}}y_v\rangle \\ &= \frac{1}{n}\langle y_u, y_v\rangle = 0\ ,\end{aligned}$$

where we used the fact that $\langle z_u, z_v\rangle = 0$, $\langle z_u, y_v\rangle = \langle z_v, y_u\rangle = 0$ and the fact that $\langle x_e, x_{e'}\rangle = 0$ for every edge e incident to u and every edge e' incident to v, as $e \neq e'$ when $(u,v) \notin E$. Similarly, when $(u,v) \in E$:

$$\langle k_u, k_v\rangle = \frac{1}{n}\langle y_u, y_v\rangle = \frac{1}{n}(d_u^2 d_v^2 \|x_{uv}\|_2^2) = \frac{1}{n}$$

Now, consider a Gaussian kernel with bandwidth $\sigma = (n\log(1/\epsilon))^{-\frac{1}{2}}$ and vectors $\{k_v\}_{v\in V}$. Since $\|k_u - k_v\|_2^2 = 2(1-\langle k_u, k_v\rangle)$ from the above calculations of the inner products it follows that:

$$\begin{aligned}\forall (u,v) \in E : e^{-\frac{\|k_u-k_v\|_2^2}{2\sigma^2}} &= e^{-\frac{1-1/n}{\sigma^2}}, \\ \forall (u,v) \notin E : e^{-\frac{\|k_u-k_v\|_2^2}{2\sigma^2}} &= e^{-\frac{1}{\sigma^2}}.\end{aligned}$$

Thus the ratio is $e^{-\frac{1}{n\sigma^2}} = \epsilon$, as desired. □

5 Hard Instances for Average-Linkage under Gaussian Kernel Similarity

High-dimensional case. We embed the construction of [CCN19] shown in Figure 3 into vectors with similarities given by the Gaussian kernel.

Theorem 5.1. *There exists a set of vectors $v_1, \ldots, v_n \in \mathbb{R}^d$ for $d = poly(\log n)$ for which the average-linkage clustering algorithm achieves an approximation at most $\frac{1}{3} + o(1)$ for $\mathcal{F}^+$ under the Gaussian kernel similarity measure.*

Proof of Theorem 5. We start by this theorem:

Theorem 5.2 ([CCN19])**.** *For any constant $\epsilon \in (0,1)$ the instance $\mathcal{I}$ average-linkage clustering achieves the value of $\mathcal{F}^+$ at most $\frac{1}{6}n^{8/3} + O(n^{7/3})$ while the optimum is at least $\frac{1}{2}n^{8/3} - O(n^{7/3})$.*

Let $\Delta, \tau > 0$ be real-valued parameters to be chosen later. We use indices i and j to index our set of vectors. For $i \in \{1, 2, \ldots, n^{1/3}\}$, $j \in \{1, 2, \ldots, n^{2/3}\}$ let

$$v_{i,j} = \Delta(e_i + (1+\tau)e_{k+j}),$$

where $k = n^{1/3}$ and e_i is the t-th standard unit vector $e_t = (0 \ldots, 1, \ldots, 0)$ with the 1 in the t-th entry. Then

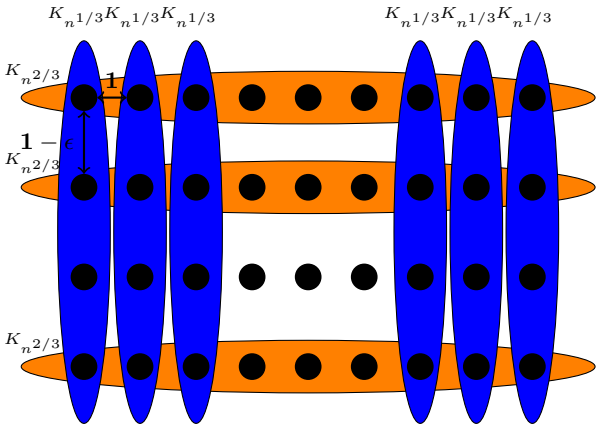

Figure 3: Hard instance $\mathcal{I}$ from [CCN19]. Vertices in orange blocks form cliques of size $n^{2/3}$ connected by edges of similarity $1-\epsilon$, vertices in blue blocks form cliques of size $n^{1/3}$ connected by edges of similarity 1, all other pairs have similarity 0.

it is easy to see that for any fixed $i \in [n^{1/3}]$ and $j_1 \neq j_2 \in [n^{2/3}]$ it holds that:

$$\|v_{i,j_1} - v_{i,j_2}\|_2^2 = 2(1+\tau)^2\Delta^2$$

Similarly, for any fixed $j \in [n^{2/3}]$ and $i_1 \neq i_2 \in [n^{1/3}]$ it holds that:

$$\|v_{i_1,j} - v_{i_2,j}\|_2^2 = 2\Delta^2.$$

Otherwise if $i_1 \neq i_2 \in [n^{1/3}]$ and $j_1 \neq j_2 \in [n^{2/3}]$ then:

$$\|v_{i_1,j_1} - v_{i_2,j_2}\|_2^2 = 2\Delta^2 + 2(1+\tau)^2\Delta^2 \geq 4\Delta^2.$$

By setting $\Delta^2 = 2\sigma^2 c\log n$ for a sufficiently large constant c the contribution of pairs of vectors with $i_1 \neq i_2$ and $j_1 \neq j_2$ can be made negligible. Let $2(1+\tau)^2\Delta^2 = \alpha$ and $2\Delta^2 = \beta$. The rest of the pairs correspond to a hard instance $\mathcal{I}$ for which average-linkage only achieves a $\frac{1}{3} + o(1)$-approximation compared to the optimum. By setting $\tau = 1/poly(\log(n))$ we have $e^{(\beta^2-\alpha^2)/2\sigma^2} = \Omega(1)$ and hence by Theorem 5.2 it follows that average-linkage clustering can't achieve better than $1/3 + o(1)$ approximation for this instance.

Finally, note that by applying the Johnson-Lindenstrauss transform we can reduce the dimension required for the above reduction to $d = poly(\log n)$. Indeed, projecting on a random subspace of dimension $O(\frac{\log n}{\xi^2})$ would preserve ℓ_2^2-distances between all pairs of vectors up to a multiplicative factor of $(1 \pm \xi)$ setting $\xi = \frac{\tau}{10} = 1/poly(\log n)$ it follows that our hard instance can be embedded in dimension $d = poly(\log n)$. □

Low-dimensional case. For $d = 1$ a hard instance for average-linkage clustering can also be constructed.

Lemma 5.3. *For points $x_1, \ldots, x_n \in \mathbb{R}$ average-linkage clustering achieves approximation at most 3/4 for $\mathcal{F}^+$ under the Gaussian kernel similarity measure.*

Proof. The instance consists of four equally spaced points on a line, i.e. $0, \Delta, 2\Delta, 3\Delta$. Slightly shifting the two middle points leads average-linkage to first connect the two middle points. An alternative solution is to merge $(0, \Delta)$ and $(2\Delta, 3\Delta)$ and then merge those together (calculations are in the supplement). □

Corollary 5.4. *In the instance of Lemma 5.3, the* `1D-SUM-upper` *evaluates to $(\sqrt{2\pi}\sigma)^{-n} 6e^{-\Delta^2/2\sigma^2}$, and hence average-linkage cannot obtain more than 1/2 of* `1D-SUM-upper`.

6 Experimental Results

In this section we demonstrate the quality of the solution returned by Projected Random Cut (PRC) on a small dataset, and highlight that running time only scales linearly on a large dataset. PRC does not compute the similarity weights (i.e., it only takes one pass over the feature vectors with dimension-free memory requirements) and then sorts the projected points in time $O(n \log n)$. Hence, it is fast even in use-cases with over millions of datapoints (in contrast to average-linkage, spectral clustering, or even single-linkage which all have superlinear running times in high dimensions). We run PRC algorithm on two real datasets from the UCI ML repository [Lic13]: (i) the Zoo dataset [Lic13, VD16] (the *small* dataset) that contains 100 animals given as 16D feature vectors (this dataset comes from applications in biology), and (ii) the SIFT10M dataset [FMMA14] (the *large* dataset) that contains around 10M datapoints, where each datapoint is a 128D Scale Invariant Feature Transform (SIFT) vector (this dataset comes from applications in computer vision). For more details, refer to the supplement §D.

Small dataset (Zoo). We compare PRC to (i) the recursive spectral clustering using the second eigenvector of the normalized Laplacian of the weight matrix (Spectral) [CNC18], (ii) average-linkage (AL), and (iii) `MAX-upper`, an upper-bound on the objective value of the optimum tree; see Section 2. In contrast to PRC, these three benchmarks need to compute the weights and are slow on large datasets. Table 3 summarizes the results of this experiment for various values of σ. The fifth column gives an empirically observed approximation guarantee for PRC by comparing it against the upper bound on optimum `MAX-upper` for $\mathcal{F}^+$. We observe that PRC attains better approximation factors ($\approx [0.75, \ldots, 0.92]$) compared to Spectral and AL, even though it runs faster (in almost linear-time).

σ	PRC	Spectral	AL	`MAX-upper`	$\frac{\text{PRC}}{\texttt{MAX-upper}}$
1.5	48	61	28	64	0.75
2	64	83	47	87	0.74
2.5	83	100	66	105	0.79
3	100	112	82	117	0.85
3.5	111	121	95	126	0.87
4	117	128	105	132	0.88
4.5	123	133	114	137	0.91
5	129	137	120	140	0.92

Table 1: Values of the objective (times 10^{-3}) on the Zoo dataset (averaged over 10 runs).

Size	PRC (seconds)	1 Data Pass (seconds)
10K	1.7	1.5
100K	13	9.6
500K	67	46.4
1M	135	99.7
10M	1592	1144

Table 2: Running times of PRC and one pass.

On the choice of bandwidth parameter σ, as a rule of thumb, we find an interval $[\alpha, \beta]$ such that the similarity scores are meaningful: notice that if σ is too small ($\sigma < \alpha$) or too large ($\sigma > \beta$), then all similarities become close to 0 or 1 respectively, due to the exponentiation of the Gaussian similarity; this would result to artificially inflated approximation guarantees ($> 98\%$), since the graph is now almost a clique (with the same edge weights) and it is known [Das16] that all HC of such cliques have the same objective value. We end up with $\alpha = 1.5, \beta = 5$. For completeness, in the supplement, we also extend our experiments by evaluating 4 more similarity scores: cosine similarity, appropriately scaled ℓ_1 and ℓ_2 norms and Jaccard similarity, achieving comparable results to the Gaussian kernel.

Large dataset (SIFT10M) The focus of this experiment is measuring the running time of PRC, and showing that it only scales linearly with the dataset size. Note that evaluating the performance of any other algorithm or upper bound (or even one pass over the similarity matrix) would be prohibitive. We run PRC on truncated versions of SIFT10M of sizes 10K, 100K, 500K, 1M and 10M. σ is set to 450 [3]. We emphasize that our PRC algorithm runs extremely fast on a 2014 MacBook[4]. The running times are summarized in Table 2. Observe that PRC scales almost linearly with the data and has almost the same running time as just a single pass over the datapoints.

[3] Note that the σ value does not affect the running time.

[4] System specs: 8 GB 1600 MHz DDR3 RAM, 2,5 GHz Intel Core i5 CPU.

References

[BBD+17] Mohammadhossein Bateni, Soheil Behnezhad, Mahsa Derakhshan, MohammadTaghi Hajiaghayi, Raimondas Kiveris, Silvio Lattanzi, and Vahab Mirrokni. Affinity clustering: Hierarchical clustering at scale. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 6864–6874. Curran Associates, Inc., 2017.

[Ber06] Pavel Berkhin. A survey of clustering data mining techniques. In *Grouping multidimensional data*, pages 25–71. Springer, 2006.

[CAKMT17] Vincent Cohen-Addad, Varun Kanade, and Frederik Mallmann-Trenn. Hierarchical clustering beyond the worst-case. In *Advances in Neural Information Processing Systems*, pages 6202–6210, 2017.

[CAKMTM18] Vincent Cohen-Addad, Varun Kanade, Frederik Mallmann-Trenn, and Claire Mathieu. Hierarchical clustering: Objective functions and algorithms. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 378–397. SIAM, 2018.

[CC17] Moses Charikar and Vaggos Chatziafratis. 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, 2017.

[CCN19] Moses Charikar, Vaggos Chatziafratis, and Rad Niazadeh. Hierarchical clustering better than average-linkage. In *Proceeding of the ACM-SIAM Symposium on Discrete Algorithms*, 2019.

[CNC18] Vaggos Chatziafratis, Rad Niazadeh, and Moses Charikar. Hierarchical clustering with structural constraints. In *International Conference on Machine Learning*, pages 773–782, 2018.

[Das16] Sanjoy Dasgupta. A cost function for similarity-based hierarchical clustering. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pages 118–127. ACM, 2016.

[DBE+15] Ibai Diez, Paolo Bonifazi, Iñaki Escudero, Beatriz Mateos, Miguel A Muñoz, Sebastiano Stramaglia, and Jesus M Cortes. A novel brain partition highlights the modular skeleton shared by structure and function. *Scientific reports*, 5:10532, 2015.

[ESBB98] Michael B Eisen, Paul T Spellman, Patrick O Brown, and David Botstein. Cluster analysis and display of genome-wide expression patterns. *Proceedings of the National Academy of Sciences*, 95(25):14863–14868, 1998.

[FMMA14] Xiping Fu, Brendan McCane, Steven Mills, and Michael Albert. Nokmeans: Non-orthogonal k-means hashing. In *Asian Conference on Computer Vision*, pages 162–177. Springer, 2014.

[Gär03] Thomas Gärtner. A survey of kernels for structured data. *ACM SIGKDD Explorations Newsletter*, 5(1):49–58, 2003.

[GHP07] Gregory Griffin, Alex Holub, and Pietro Perona. Caltech-256 object category dataset. 2007.

[JS68] N Jardine and R Sibson. A model for taxonomy. *Mathematical Biosciences*, 2(3-4):465–482, 1968.

[Lic13] Moshe Lichman. Uci machine learning repository, zoo dataset, 2013.

[LRU14] Jure Leskovec, Anand Rajaraman, and Jeffrey David Ullman. *Mining of massive datasets*. Cambridge university press, 2014.

[McS01] Frank McSherry. Spectral partitioning of random graphs. In *focs*, page 529. IEEE, 2001.

[MMO08] Charles F Mann, David W Matula, and Eli V Olinick. The use of sparsest cuts to reveal the hierarchical community structure of social networks. *Social Networks*, 30(3):223–234, 2008.

[MRS08] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to information retrieval*. Cambridge University Press, 2008.

[MW17] Benjamin Moseley and Joshua Wang. Approximation bounds for hierarchical clustering: Average linkage, bisecting k-means, and local search. In *Advances in Neural Information Processing Systems*, pages 3097–3106, 2017.

[RP16] Aurko Roy and Sebastian Pokutta. Hierarchical clustering via spreading metrics. In *Advances in Neural Information Processing Systems*, pages 2316–2324, 2016.

[SKK+00a] Michael Steinbach, George Karypis, Vipin Kumar, et al. A comparison of document clustering techniques. In *KDD workshop on text mining*, volume 400, pages 525–526. Boston, 2000.

[SKK+00b] Michael Steinbach, George Karypis, Vipin Kumar, et al. A comparison of document clustering techniques. In *KDD workshop on text mining*, volume 400, pages 525–526. Boston, 2000.

[SS62] Peter HA Sneath and Robert R Sokal. Numerical taxonomy. *Nature*, 193(4818):855–860, 1962.

[TLM10] Michele Tumminello, Fabrizio Lillo, and Rosario N Mantegna. Correlation, hierarchies, and networks in financial markets. *Journal of Economic Behavior & Organization*, 75(1):40–58, 2010.

[VD16] Sharad Vikram and Sanjoy Dasgupta. Interactive bayesian hierarchical clustering. In *International Conference on Machine Learning*, pages 2081–2090, 2016.

[VF10] Andrea Vedaldi and Brian Fulkerson. Vlfeat: An open and portable library of computer vision algorithms. In *Proceedings of the 18th ACM international conference on Multimedia*, pages 1469–1472. ACM, 2010.

[YV18] Grigory Yaroslavtsev and Adithya Vadapalli. Massively parallel algorithms and hardness for single-linkage clustering under ℓ_p distances. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, pages 5596–5605, 2018.

[ZMP05] Lihi Zelnik-Manor and Pietro Perona. Self-tuning spectral clustering. In *Advances in neural information processing systems*, pages 1601–1608, 2005.