Modern Aspects of Unsupervised Learning

Yingyu Liang

August, 2013

School of Computer Science
Georgia Institute of Technology
Atlanta, GA 30332

Thesis Committee:
Maria-Florina Balcan, Chair
Charles L. Isbell, Jr.
Dana Randall
Le Song

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy.

Abstract

Unsupervised learning has become more and more important due to the recent explosion of data. Clustering, a key topic in unsupervised learning, is a well-studied task arising in many applications ranging from computer vision to computational biology to the social sciences. This thesis is a collection of work exploring three modern aspects of clustering.

In the first part, we study clustering under perturbation resilience. As an alternative approach to worst case analysis, this novel theoretical framework aims at understanding the complexity of clustering instances that satisfy natural stability assumptions. In particular, we show how to correctly cluster an instance whose optimal solution is resilient to small multiplicative perturbations on the distances between data points. We propose to explore the combination of perturbation resilience with other stability assumptions.

In the second part, we study the problem of distributed clustering where the data is distributed across nodes which communicate over the edges of a connected graph. We provide algorithms with small communication cost and provable guarantees on the clustering quality. We also propose to explore the communication complexity of some other machine learning tasks in this distributed setting, such as the minimum enclosing ball problem.

In the third part, we consider algorithms that output a collection of clusters that are more general than a partition of the data. This allows us to model rich structures in the data that cannot be represented by a partition output by traditional methods. For example, for community detection tasks, we propose a theoretical model that formalizes the communities in the network and design an algorithm that provably detects all communities in our model. We also study the structure of weak clusters introduced in computational biology, and design a new, faster algorithm for computing the collection of all weak clusters.

Keywords: Unsupervised Learning, Clustering, Perturbation Resilience, Center-based Clustering, k-Median, k-Means, Min-Sum, Distributed Clustering, Coresets, Community Detection, Community Hierarchies, Weak Clusters, Weak Hierarchies.

Chapter 1

Overview

This thesis develops new frameworks and designs algorithms for new emerging aspects of clustering, a key topic in unsupervised learning. These modern aspects are areas of significant practical importance and rising concerns. Traditional clustering analysis tends to not capture these new aspects, hence new frameworks and algorithms are desirable.

Generally, the goal of clustering is to partition n data points into k meaningful subsets (called clusters) given access to a distance function on the data points. From the machine learning perspective, we assume that there is some unknown target clustering, and our algorithm should produce results close to the target. In general, we need to assume some connection between the target and the distance function. A common paradigm in clustering is objective-based, imposing a quantitative objective such as k-median or k-means and assuming that the target clustering is equal or close to the partition that optimizes the objective. Another paradigm is target-based, assuming that the target clustering satisfies certain property based on the distance function (for example, the strict separation property that any two points in any target cluster are closer to each other than to any points outside). Both paradigms are discussed in the thesis.

1.1 Clustering under Perturbation Resilience

For most natural clustering objectives, finding the optimal solution to the objective function is NP-hard. As a consequence, there has been substantial work on approximation algorithms [44, 23, 18, 31, 3] with both upper and lower bounds on the approximability of these objective functions on worst case instances. Bilu and Linial [21] suggested an interesting alternative approach aimed at understanding the complexity of clustering instances which arise in practice. They argued that interesting instances should be resilient to small perturbations in the distances, and specifically defined an instance to be α-perturbation resilient for an objective Φ if perturbing pairwise distances by multiplicative factors in the range $[1, \alpha]$ does not change the optimum clustering under Φ. Two important questions raised are: (1) how much resilience is required so that one can develop algorithms for important clustering objectives? (2) the resilience definition requires the optimum solution to remain *exactly* the same after perturbation: can one succeed under weaker conditions?

In [14], we address both these questions. First, for the center-based objectives we design a polynomial time algorithm for finding the optimum solution for instances resilient to perturbations of value $\alpha = 1+\sqrt{2}$, thus beating the previously best known factor of 3 of Awasthi et al [5]. Second, for k-median (which is a specific center-based objective), we consider a weaker, relaxed, and more realistic notion of perturbation-resilience where we allow the optimal clustering of the perturbed instance to differ from the optimal of the original in a small ϵ fraction of the points. Compared to the original perturbation resilience assumption, this

is arguably a more natural though also more difficult condition to deal with. We give positive results for this case as well, showing for somewhat larger values of α that we can still achieve a near-optimal clustering on the given instance. We give positive results for min-sum clustering which is a generally much harder objective than center-based objectives. We additionally provide sublinear-time algorithms both for the k-median and min-sum objectives, showing algorithms that can return an implicit clustering from only access to a small random sample. These results are presented in Chapter 2.

1.2 Distributed Clustering

Most classic clustering algorithms are designed for the centralized setting, but in recent years data has become distributed over different locations. As a consequence it has become crucial to develop clustering algorithms which are effective in the distributed setting. Several algorithms for distributed clustering have been proposed and empirically tested. Some of these algorithms [35, 67, 30] are direct adaptations of centralized algorithms which rely on statistics that are easy to compute in a distributed manner. Other algorithms [45, 46] generate summaries of local data and transmit them to a central coordinator which then performs the clustering algorithm. No theoretical guarantees are provided for the clustering quality in these algorithms, and they do not try to minimize the communication cost. Additionally, most of these algorithms assume that the distributed nodes can communicate with all other sites or that there is a central coordinator that communicates with all other sites.

We study the problem of distributed clustering where the data is distributed across nodes which communicate over the edges of an arbitrary connected graph. We provide algorithms with small communication cost and provable guarantees on the clustering quality. Our technique for reducing communication in general graphs is based on the construction of a small set of points called coreset [42], which act as a proxy for the entire data set. A coreset is a weighted set of points whose cost on any set of centers is approximately the cost of the original data on those same centers, thus an approximate solution for the coreset is also an approximate solution for the original data. We propose a distributed algorithm for k-means and k-median, by which each node constructs a local portion of a global coreset. Communicating the total cost of local approximate solutions to each node is enough for the construction, leading to low communication cost overall. The nodes then share the local portions of the coreset, which can be done efficiently in general graphs using a message passing approach. This result is discussed in Chapter 3.

1.3 Cluster Hierarchies

In many modern applications, the goal is to identify a collection of clusters, which generally do not form a partition as in traditional clustering. For example, in analyzing social networks, it is meaningful to identify interesting subsets called communities based on the affinity between members of the network. In [15], we take a target-based paradigm for community detection. We propose a theoretical model that formalizes the collection of target communities. In our model, each member of a community falls into a sub-community and the sub-communities within this community have active interactions with each other, while entities outside this community have fewer interactions with members inside. Given this formalization, we then propose an efficient algorithm that detects all the communities in this model, and prove that all the communities form a tree hierarchy. This result is discussed in Section 4.1.

We also study the notion of weak clusters introduced by Bandelt and Dress [17], and propose to use the collection of all weak clusters for cluster analysis. The collection of weak clusters forms a weak hierarchy,

a generalization of the tree hierarchy, and has nice structure properties that we utilize to design a new faster algorithm for constructing this hierarchy. This result is discussed in Section 4.2.

Chapter 2

Clustering under Perturbation Resilience

Problems of clustering data from pairwise distance information are ubiquitous in science. A common approach for solving such problems is to view the data points as nodes in a weighted graph (with the weights based on the given pairwise information), and then to design algorithms to optimize various objective functions such as k-median or min-sum. For example, in the k-median clustering problem the goal is to partition the data into k clusters C_i, giving each a center c_i, in order to minimize the sum of the distances of all data points to the centers of their cluster. In the min-sum clustering approach the goal is to find k clusters C_i that minimize the sum of all intra-cluster pairwise distances. Yet unfortunately, for most natural clustering objectives, finding the optimal solution to the objective function is NP-hard. As a consequence, there has been substantial work on approximation algorithms [44, 23, 18, 31, 3] with both upper and lower bounds on the approximability of these objective functions on worst case instances.

Recently, Bilu and Linial [21] suggested an interesting alternative approach aimed at understanding the complexity of clustering instances which arise in practice. Motivated by the fact that distances between data points in clustering instances are often based on a heuristic measure, they argue that interesting instances should be resilient to small perturbations in these distances. In particular, if small perturbations can cause the optimum clustering for a given objective to change drastically, then that probably is not a meaningful objective. Bilu and Linial [21] specifically define an instance to be α-perturbation resilient[1] for an objective Φ if perturbing pairwise distances by multiplicative factors in the range $[1, \alpha]$ does not change the optimum clustering under Φ. They consider in detail the case of max-cut clustering and give an efficient algorithm to recover the optimum when the instance is resilient to perturbations on the order of $\alpha = O(\sqrt{n})$.

Two important questions raised by the work of Bilu and Linial [21] are: (1) the degree of resilience needed for their algorithm to succeed is quite high: can one develop algorithms for important clustering objectives that require much less resilience? (2) the resilience definition requires the optimum solution to remain *exactly* the same after perturbation: can one succeed under weaker conditions? In the context of *center-based* clustering objectives such as k-median and k-center, Awasthi et al. [5] partially address the first of these questions and show that an algorithm based on the single-linkage heuristic can be used find the optimal clustering for α-perturbation-resilient instances for $\alpha = 3$. They also conjecture it to be NP-hard to beat 3 and prove beating 3 is NP-hard for a related notion.

In this work, we address both questions raised by [21] and additionally improve over [5]. First, for the center-based objectives we design a polynomial time algorithm for finding the optimum solution for instances resilient to perturbations of value $\alpha = 1 + \sqrt{2}$, thus beating the previously best known factor of 3 of Awasthi et al [5]. Second, for k-median (which is a specific center-based objective), we consider a weaker,

[1] Bilu and Linial [21] refer to such instances as perturbation stable instances.

relaxed, and more realistic notion of perturbation-resilience where we allow the optimal clustering of the perturbed instance to differ from the optimal of the original in a small ϵ fraction of the points. Compared to the original perturbation resilience assumption, this is arguably a more natural though also more difficult condition to deal with. We give positive results for this case as well, showing for somewhat larger values of α that we can still achieve a near-optimal clustering on the given instance (see Section 1.1 below for precise results). We additionally give positive results for min-sum clustering which is a generally much harder objective than center-based objectives. For example, the best known guarantee for min-sum clustering on worst-case instances is an $O(\delta^{-1}\log^{1+\delta} n)$-approximation algorithm that runs in time $n^{O(1/\delta)}$ due to Bartal et al. [18]; by contrast, the best guarantee known for k-median is factor $1+\sqrt{3}+\epsilon$ [51].

Our results are achieved by carefully deriving structural properties of perturbation-resilience. At a high level, all the algorithms we introduce work by first running appropriate linkage procedures to produce a hierarchical clustering, and then running dynamic programming to retrieve the best k-clustering present in the tree. To ensure that (under perturbation resilient instances) the hierarchy output in the first step has a pruning of low cost, we derive new linkage procedures (closure linkage and approximate closure linkage) which are of independent interest. While the overall analysis is quite involved, the clustering algorithms we devise are simple and robust. This simplicity and robustness allow us to show how our algorithms can be made sublinear-time by returning an implicit clustering from only a small random sample of the input.

Our Results: In this paper, we greatly advance the line of work of [21] by solving a number of important problems of clustering perturbation-resilient instances under metric center-based and min-sum objectives.

In Section 2.2 we improve on the bounds of [5] for α-perturbation resilient instances for center-based objectives, giving an algorithm that efficiently[2] finds the optimum clustering for $\alpha = 1+\sqrt{2}$. Most of the frequently used center-based objectives, such as k-median, are NP-hard to even approximate, yet we can recover the exact solution for perturbation resilient instances. Our algorithm is based on a new linkage procedure using a new notion of distance (closure distance) between sets that may be of independent interest.

In Section 2.3 we consider the more challenging and more general notion of (α,ϵ)-perturbation resilience for k-median, where we allow the optimal solution after perturbation to be ϵ-close to the original. We provide an efficient algorithm which for $\alpha > 2+\sqrt{7}$ produces $(1+O(\epsilon/\rho))$-approximation to the optimum, where ρ is the fraction of the points in the smallest cluster. The key structural property we derive and exploit is that, except for ϵn bad points, most points are α closer to their own center than to any other center. Using this fact, we then design an approximate version of the closure linkage criterion that allows us to carefully eliminate the noise introduced by the bad points and construct a tree that has a low-cost pruning that is a good approximation to the optimum.

In Section 2.4 we provide the first efficient algorithm for optimally clustering α-min-sum perturbation resilient instances. Our algorithm is based on an appropriate modification of average linkage that exploits the structure of min-sum perturbation resilient instances.

We also provide sublinear-time algorithms both for the k-median and min-sum objectives (Sections 2.3.3 and 2.4), which can return an implicit clustering from only access to a small random sample.

Related Work: A subsequent work of [21] by Bilu, Daniely, Linial and Saks [20] studied the max cut problem under Bilu and Linial stability, and showed how to solve in polynomial time $(1+\epsilon)$-stable instances of metric and dense max cut, and $\Omega(\sqrt{n})$-stable instances of general max cut. The later bound is further improved by Makarychev, Makarychev and Vijayaraghavan [55], that proposed a polynomial time exact algorithm for $\Omega(\sqrt{\log n}\log\log n)$-stable Max-Cut instances based on semidefinite programming.

In the context of objective based clustering, several recent papers have showed how to exploit other notions of stability for overcoming the existing hardness results on worst case instances. The ORSS stability

[2]For clarity, in this chapter efficient means polynomial in both n (the number of points) and k (the number of clusters).

notion of Ostrovsky, Rabani, Schulman and Swamy [61, 5] assumes that the cost of the optimal k-means solution is small compared to the cost of the optimal $(k-1)$-means solution. The BBG approximation stability condition of Balcan, Blum and Gupta [10] assumes that every nearly optimal solution is close to the target clustering. Awasthi, Sheffet and Blum [4] proposed a stability condition called weak-deletion stability, and showed that it is implied by both the ORSS stability and the BBG stability. Kumar and Kannan [48] proposed a proximity condition which assumes that in the target clustering, most data points satisfy that they are closer to their center than to any other center by an additive factor in the order of the maximal standard variance of their clusters in any direction. Their results are improved by Awasthi and Sheffet [6], which proposed a weaker version of the proximity condition called center separation, and designed algorithms achieving stronger guarantees under this weaker condition.

Several recent papers have showed how to exploit the structure of perturbation resilient instances in order to obtain better approximation guarantees (than those possible on worst case instances) for other difficult optimization problems. These include the game theoretic problem of finding Nash equilibria [12, 52] and the classic traveling salesman problem [56]. Salil Vadhan [70] also pointed out that various forms of stability assumptions can be potentially useful for differential privacy analysis.

2.1 Preliminaries

In a clustering instance, we are given a set S of n points in a finite metric space, and we denote $d : S \times S \to \mathbb{R}_{\geq 0}$ as the distance function. Φ denotes the objective function over a partition of S into $k < n$ clusters which we want to optimize over the metric, ie. Φ assigns a score to every clustering. The optimal clustering w.r.t. Φ is denoted as $\mathcal{C} = \{C_1, C_2, \ldots, C_k\}$, and its cost is denoted as $\mathcal{OPT}$. We focus on the center-based and min-sum objectives.

For the *center-based objectives*, we consider separable center-based objectives defined by [5].

Definition 1. *A clustering objective is center-based if the optimal solution can be defined by k points $c_1, \cdots, c_k$ in the metric space called centers such that every data point is assigned to its nearest center. Such a clustering objective is separable if it furthermore satisfies the following two conditions:*

- *The objective function value of a given clustering is either a (weighted) sum or the maximum of the individual cluster scores.*
- *Given a proposed single cluster, its score can be computed in polynomial time.*

One particular center-based objective is the k-median objective. We partition S into k disjoint subsets $\mathcal{P} = \{P_1, P_2, \ldots, P_k\}$ and assign a set of centers $\mathbf{p} = \{p_1, p_2, \ldots, p_k\} \subseteq S$ for the subsets. The goal is to minimize $\Phi(\mathcal{P}, \mathbf{p}) = \sum_{i=1}^{k} \sum_{p \in P_i} d(p, p_i)$. The centers in the optimal clustering are denoted as $\mathbf{c} = \{c_1, \ldots, c_k\}$. Clearly, in an optimal solution, each point is assigned to its nearest center. In such cases, the objective is denoted as $\Phi(\mathbf{c})$.

For the *min-sum objective*, we partition S into k disjoint subsets $\mathcal{P} = \{P_1, P_2, \ldots, P_k\}$, and the goal is to minimize $\Phi(\mathcal{P}) = \sum_{i=1}^{k} \sum_{p,q \in P_i} d(p, q)$. Note that we sometimes denote Φ as Φ_S in the case where the distinction is necessary, such as in Section 2.3.3.

The core concept we study is the perturbation resilience notion introduced by [21]. Formally:

Definition 2. *A clustering instance (S, d) is α-**perturbation resilient** to a given objective Φ if for any function $d' : S \times S \to \mathbb{R}_{\geq 0}$ s.t. $\forall p, q \in S, d(p, q) \leq d'(p, q) \leq \alpha d(p, q)$, there is a unique optimal clustering $\mathcal{C}'$ for Φ under d' and this clustering is equal to the optimal clustering $\mathcal{C}$ for Φ under d.*

In Section 2.3 we consider a generalization of Definition 2 where we allow a small difference between the original optimum and the new optimum after perturbation. Formally:

Definition 3. *Let $\mathcal{C}$ be the optimal k-clustering and $\mathcal{C}'$ be another k-clustering of a set of n points. We say $\mathcal{C}'$ is ϵ-**close** to $\mathcal{C}$ if $\min_{\sigma \in \mathcal{S}_k} \sum_{i=1}^{k} |C_i \setminus C'_{\sigma(i)}| \leq \epsilon n$, where σ is a matching between indices of clusters of $\mathcal{C}'$ and those of $\mathcal{C}$.*

Definition 4. *A clustering instance (S, d) is (α, ϵ)-**perturbation resilient** to a given objective Φ if for any function $d' : S \times S \to \mathbb{R}_{\geq 0}$ s.t. $\forall p, q \in S, d(p, q) \leq d'(p, q) \leq \alpha d(p, q)$, the optimal clustering $\mathcal{C}'$ for Φ under d' is ϵ-close to the optimal clustering $\mathcal{C}$ for Φ under d.*

2.2 α-Perturbation Resilience for Center-based Objectives

In this section we show that, for $\alpha \geq 1 + \sqrt{2}$, if the clustering instance is α-perturbation resilient for center-based objectives, then we can in polynomial time find the optimal clustering. This improves on the $\alpha \geq 3$ bound of [5] and stands in sharp contrast to the NP-Hardness results on worst-case instances. Our algorithm succeeds for an even weaker property, the α-center proximity, introduced in [5].

Definition 5. *A clustering instance (S, d) satisfies the α-**center proximity** property if for any optimal cluster $C_i \in \mathcal{C}$ with center c_i, $C_j \in \mathcal{C} (j \neq i)$ with center c_j, any point $p \in C_i$ satisfies $\alpha d(p, c_i) < d(p, c_j)$.*

Lemma 1. *Any clustering instance that is α-perturbation resilient to center-based objectives also satisfies the α-center proximity.*

The proof follows easily by constructing a specific perturbation that blows up all the pairwise distances within cluster C_i by a factor of α. By α-perturbation resilience, the optimal clustering remains the same after this perturbation, which then implies the desired result. Then it is sufficient to prove our result for α-center proximity. The following properties implied by α-center proximity and triangle inequality are the key for our algorithm.

Lemma 2. *For any points $p \in C_i$ and $q \in C_j (j \neq i)$ in the optimal clustering of an α-center proximity instance, when $\alpha \geq 1 + \sqrt{2}$, we have: (1) $d(c_i, q) > d(c_i, p)$, (2) $d(p, c_i) < d(p, q)$.*

Lemma 2 implies that for any optimal cluster C_i, the ball of radius $\max_{p \in C_i} d(c_i, p)$ around the center c_i contains *only* points from C_i, and moreover, points inside the ball are each closer to the center than to any point outside the ball. Inspired by this structural property, we define the notion of closure distance between two sets as the radius of the minimum ball that covers the sets and has some margin from points outside the ball. We show that any (strict) subset of an optimal cluster has smaller closure distance to another subset in the same cluster than to any subset of other clusters or to unions of other clusters. Using this, we will be able to define an appropriate linkage procedure that, when applied to the data, produces a tree on subsets that will all be laminar with respect to the clusters in the optimal solution. This will then allow us to extract the optimal solution using dynamic programming applied to the tree.

We now define the notion of closure distance and then present our algorithm (Algorithm 1).

Definition 6. *Let $\mathbb{B}(p, r) = \{q : d(q, p) \leq r\}$. The **closure distance** $d_S(A, A')$ between two disjoint non-empty subsets A and A' of point set S is the minimum $d \geq 0$ such that there is a point $c \in A \cup A'$ satisfying the following requirements:*
(1) coverage: the ball $\mathbb{B}(c, d)$ covers A and A', i.e. $A \cup A' \subseteq \mathbb{B}(c, d)$;
(2) margin: points inside $\mathbb{B}(c, d)$ are closer to the center c than to points outside,
i.e. $\forall p \in \mathbb{B}(c, d), q \notin \mathbb{B}(c, d)$, we have $d(c, p) < d(p, q)$.

Algorithm 1 Center-based objectives, α perturbation resilience

Input: Data set S, distance function $d(\cdot, \cdot)$ on S.

▷ `Phase 1: tree construction`

Step 1 Begin with n singleton clusters.

Step 2 Repeat till only one cluster remains: merge clusters C, C' which minimize $d_S(C, C')$.

Step 3 Let T be the tree with single points as leaves and internal nodes corresponding to the merges.

▷ `Phase 2: pruning`

Step 4 Apply dynamic programming on T to get the minimum cost pruning $\tilde{\mathcal{C}}$.

Output: Clustering $\tilde{\mathcal{C}}$.

It is sufficient to show that the linkage builds a tree with the optimal clustering as a pruning. Intuitively, the closure distance is defined such that the distance between any two subsets from the same optimal cluster is at most the radius of the cluster. On the other hand, by Lemma 2, we can show that the distance between any (strict) subset of an optimal cluster and any subset outside the cluster is larger than the radius of the cluster. Therefore, the algorithm will merge points from the same optimal cluster before merging them with points outside, which ensures that the optimal clustering is a pruning of the tree built.

Theorem 1. *For $(1 + \sqrt{2})$-center proximity instances, Phase 1 of Algorithm 1 (the closure linkage phase) constructs a binary tree such that the optimal clustering is a pruning of this tree.*

Proof. We prove correctness by induction. In particular, assume that our current clustering is *laminar* with respect to the optimal clustering – that is, for each cluster A in our current clustering and each C in the optimal clustering, we have either $A \subseteq C$, or $C \subseteq A$ or $A \cap C = \varnothing$. This is clearly true at the start. To prove that the merge steps keep the laminarity, we need to show the following: if A is a strict subset of an optimal cluster C_i, A' is a subset of another optimal cluster or the union of one or more other clusters, then there exists B from $C_i \setminus A$, such that $d_S(A, B) < d_S(A, A') = d_S(A', A)$.

We first prove that there is a cluster $B \subseteq C_i \setminus A$ in the current cluster list such that $d_S(A, B) \leq d = \max_{p \in C_i} d(c_i, p)$. There are two cases. First, if $c_i \notin A$, then define B to be the cluster in the current cluster list that contains c_i. By induction, $B \subseteq C_i$ and thus $B \subseteq C_i \setminus A$. Then we have $d_S(B, A) \leq d$ since there is $c_i \in B$, and (1) for any $p \in A \cup B$, $d(c_i, p) \leq d$, (2) for any $p \in S$ satisfying $d(c_i, p) \leq d$, and any $q \in S$ satisfying $d(c_i, q) > d$, by Lemma 2 we know $p \in C_i$ and $q \notin C_i$, and thus $d(c_i, p) < d(p, q)$. In the second case when $c_i \in A$, we pick any $B \subseteq C_i \setminus A$ and a similar argument gives $d_S(A, B) \leq d$.

As a second step, we need to show that $d < \hat{d} = d_S(A, A')$. There are two cases: the center for $d_S(A, A')$ is in A or in A'. In the first case, there is a point $c \in A$ such that c and $\hat{d}$ satisfy the requirements of the closure distance. Pick a point $q \in A'$, and define C_j to be the cluster in the optimal clustering that contains q. As $d(c, q) \leq \hat{d}$, and by Lemma 2 $d(c_j, q) < d(c, q)$, we must have $d(c_j, c) \leq \hat{d}$ (otherwise it violates the second requirement of closure distance). Suppose $p = \arg\max_{p' \in C_i} d(c_i, p')$. Then we have $d = d(p, c_i) < d(p, c_j)/\alpha \leq (d + d(c_i, c) + d(c, c_j))/\alpha$ where the first inequality comes from Lemma 1 and the second from the triangle inequality. Since $d(c_i, c) < d(c, c_j)/\alpha$, we can combine the above inequalities and compare d and $d(c, c_j)$, and when $\alpha \geq 1 + \sqrt{2}$ we have $d < d(c, c_j) \leq \hat{d}$.

Now consider the second case, when there is a point $c \in A'$ such that c and $\hat{d}$ satisfy the requirements in the definition of the closure distance. Select an arbitrary point $q \in A$. We have $\hat{d} \geq d(c, q)$ from the first requirement, and $d(c, q) > d(c_i, q)$ by Lemma 2. Then from the second requirement of closure distance

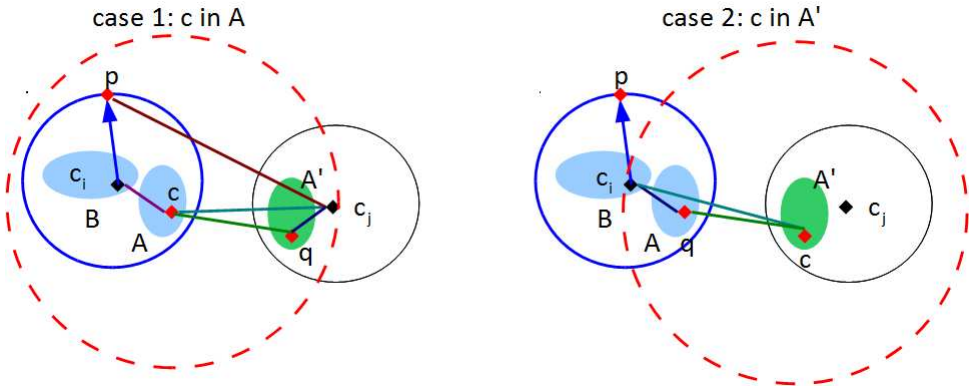


Figure 2.1: Illustration for comparing d and $d_S(A, A')$ in Theorem 1

$d(c_i, c) \leq \hat{d}$. And by Lemma 2, $d = d(c_i, p) < d(c_i, c)$, we have $d < d(c_i, c) \leq \hat{d}$. □

Note: Our factor of $\alpha = 1 + \sqrt{2}$ beats the NP-hardness *lower bound* of $\alpha = 3$ of [5] for center-proximity instances. The reason is that the lower bound of [5] requires the addition of Steiner points that can act as centers but are not part of the data to be clustered (though the upper bound of [5] does not allow such Steiner points). One can also show a lower bound for center-proximity instances without Steiner points. In particular one can show that for any $\epsilon > 0$, the problem of solving $(2 - \epsilon)$-center proximity k-median instances is NP-hard [64].

2.3 (α, ϵ)-Perturbation Resilience for the k-Median Objective

In this section we consider a natural relaxation of the α-perturbation resilience, the (α, ϵ)-perturbation resilience property, that requires the optimum after perturbation of up to a multiplicative factor α to be ϵ-close to the original (one should think of ϵ as sub-constant). We show that if the instance is (α, ϵ)-perturbation resilient, with $\alpha > 2 + \sqrt{7}$ and $\epsilon = O(\epsilon'\rho)$ where ρ is the fraction of the points in the smallest cluster, then we can in polynomial time output a clustering that provides a $(1 + \epsilon')$-approximation to the optimum. Thus this improves over the best worst-case approximation guarantees known when $\epsilon' \leq \sqrt{3}$ and also beats the lower bound of $(1 + 1/e)$ on the best approximation achievable on worst case instances for the metric k-median objective [41, 44] when $\epsilon' \leq 1/e$.

The key idea is to understand and leverage the structure implied by (α, ϵ)-perturbation resilience. We show that perturbation resilience implies that there exists only a small fraction of points that are bad in the sense that their distance to their own center is not α times smaller than their distance to any other centers in the optimal solution. We then propose a robust version of the closure linkage algorithm which carefully deals with the bad points and produces a tree with a pruning that approximates the optimal clustering. The robustness also allows us to make the algorithm sublinear-time by returning an implicit clustering from only a small random sample of the input.

2.3.1 Structure of (α, ϵ)-Perturbation Resilience

In the following we call a point *good* if it is α times closer to its own center than to any other center in the optimal clustering; otherwise we call it *bad*. Let B_i be the set of bad points in C_i, and let $G_i = C_i \setminus B_i$ be the good points. We show that under perturbation resilience we do not have too many bad points. Formally:

Theorem 2. *Suppose the clustering instance is (α, ϵ)-perturbation resilient and $\min_i |C_i| > (2+\frac{2\alpha}{\alpha-1})\epsilon n + \frac{2\alpha(\alpha+1)}{\alpha-1}$. Then $|\bigcup_i B_i| \leq \epsilon n$.*

Proof Sketch: The main idea is to construct a specific perturbation that forces certain selected bad points to move from their original optimal clusters. Then the (α, ϵ)-perturbation resilience leads to a bound on the number of selected bad points, which can also be proved to be a bound on all the bad points. For technical reasons, the selected bad points $\hat{B}_i$ in cluster C_i are defined by arbitrarily selecting $\min(\epsilon n + 1, |B_i|)$ points from B_i. For $p \in \hat{B}_i$, let $c(p) = \arg\min_{c_j, j \neq i} d(p, c_j)$ denote its second nearest center; for $p \in C_i \setminus \hat{B}_i$, $c(p) = c_i$. The perturbation we consider blows up all distances by a factor of α except for those distances between p and $c(p)$. Formally, we define d' as $d'(p, q) = d(p, q)$ if $p = c(q)$ or $q = c(p)$, and $d'(p, q) = \alpha d(p, q)$ otherwise.

The key challenge in proving a bound on the selected bad points is to show that the optimal centers do not change after the perturbation we constructed. Then in the optimum under d' each point p is assigned to the center $c(p)$, and therefore the selected bad points ($\bigcup_i \hat{B}_i$) will move from their original optimal clusters. By (α, ϵ)-perturbation resilience property we get an upper bound ϵn on the number of selected bad points, which is also an upper bound for all bad points by construction.

At a high level, we prove that the optimal centers do not change as follows. Without loss of generality, assume the optimal clustering $\mathcal{C}'$ under d' is indexed so that C'_i corresponds to C_i, and the distance between $\mathcal{C}$ and $\mathcal{C}'$ is $\sum_{i=1}^k |C_i \setminus C'_i|$. Suppose C'_i is obtained by adding point set A_i and removing point set M_i from C_i, i.e. $A_i = C'_i \setminus C_i, M_i = C_i \setminus C'_i$. Denote by c'_i the center of C'_i. To prove $c_i = c'_i$ for all i, we first show that for each cluster, its new center is close to its old center, roughly speaking since the new and old clusters have a lot in common (Claim 1). We then show if $c_i \neq c'_i$ for some i, then the weighted sum of the distances $\sum_{1 \leq i \leq k}(|A_i| + \alpha + 1 + |M_i|)d(c_i, c'_i)$ should be large (Claim 2). However, this contradicts Claim 1, so the centers do not move after the perturbation.

For $A, B \subseteq S$, define $d_{sum}(A, B) \doteq \sum_{p \in A} \sum_{q \in B} d(p, q)$ and $d_{sum}(p, B) \doteq d_{sum}(\{p\}, B)$. We have:

Claim 1. *For each i, $d_{sum}(c_i, (C_i \cap C'_i) \setminus \hat{B}_i) \geq \frac{\alpha}{\alpha+1}(|(C_i \cap C'_i) \setminus \hat{B}_i| - |\hat{B}_i \setminus M_i| - |A_i| - (\alpha+1))d(c_i, c'_i)$, i.e., $d(c_i, c'_i)$ can be bounded approximately by the average distance between c_i and a large portion of C_i.*

Proof Sketch: The intuition is that under d', c'_i is the optimal center for C'_i, so it has no more cost than c_i on C'_i. Since $\hat{B}_i \setminus M_i$ and A_i are small compared to $(C_i \cap C'_i) \setminus \hat{B}_i$, c'_i cannot save much cost on $\hat{B}_i \setminus M_i$ and A_i, thus it cannot have much more cost on $(C_i \cap C'_i) \setminus \hat{B}_i$ than c_i. Then c'_i is close to $(C_i \cap C'_i) \setminus \hat{B}_i$, and so is c_i, then c'_i is close to c_i.

Formally, we have $d'_{sum}(c'_i, C'_i) \leq d'_{sum}(c_i, C'_i)$. We divide C'_i into three parts $(C_i \cap C'_i) \setminus \hat{B}_i, \hat{B}_i \setminus M_i$ and A_i, and move terms on $(C_i \cap C'_i) \setminus \hat{B}_i$ to one side (the cost more than c_i on $(C_i \cap C'_i) \setminus \hat{B}_i$), the rest terms to another side (the cost saved on $\hat{B}_i \setminus M_i$ and A_i). After translating the terms from d' to d, we apply the triangle inequality and obtain the desired result. □

Claim 2. *Let $I_i = 1$ if $c_i \neq c'_i$ and $I_i = 0$ otherwise. Then $(\alpha - 1)\sum_{1 \leq i \leq k} I_i d_{sum}(c_i, (C_i \cap C'_i) \setminus \hat{B}_i) \leq \alpha \sum_{1 \leq i \leq k}(|A_i| + \alpha + 1 + |M_i|)d(c_i, c'_i)$.*

Proof Sketch: The intuition is that if $c'_i \neq c_i$, then under d, they have similar costs on $(C_i \cap C'_i) \setminus \hat{B}_i$ (See the intuition for Claim 1). However, this means that under d', the new centers c'_i have significantly larger cost than the old centers c_i on $(C_i \cap C'_i) \setminus \hat{B}_i$, since $d'(c'_i, (C_i \cap C'_i) \setminus \hat{B}_i) = \alpha d(c'_i, (C_i \cap C'_i) \setminus \hat{B}_i)$ while $d'(c_i, (C_i \cap C'_i) \setminus \hat{B}_i) = d(c_i, (C_i \cap C'_i) \setminus \hat{B}_i)$. This difference needs be compensated by those on other points in C'_i, since c'_i is optimal center for C'_i. This means that c_i should have much more cost than c'_i on other points in C'_i, so the distance between c_i and c'_i should be large.

Formally, we have $\sum_i d_{sum}(c_i, C_i) \leq \sum_i d_{sum}(c'_i, C_i)$ by using the fact that c_i are the optimal centers for C_i under d; and $\sum_i d'_{sum}(c'_i, C'_i) \leq \sum_i [d'_{sum}(c_i, C'_i - \hat{B}_i) + \sum_{p \in \hat{B}_i \setminus M_i} d'(c(p), p)]$ by using the fact that c'_i are the optimal centers for C'_i under d'. We multiply the first inequality by α and add it to the second inequality. Then we divide C_i into three parts $M_i, \hat{B}_i \setminus M_i$ and $(C_i \cap C'_i) \setminus \hat{B}_i$, C'_i into $A_i, \hat{B}_i \setminus M_i$ and $(C_i \cap C'_i) \setminus \hat{B}_i$, and $C'_i \setminus \hat{B}_i$ into A_i and $(C_i \cap C'_i) \setminus \hat{B}_i$. After translating the terms from d' to d, we notice that the clustering that under d' assigns points in $C'_i \setminus \hat{B}_i$ to c_i and points p in $\hat{B}_i \setminus M_i$ to $c(p)$ (corresponding to the right hand side of the second inequality) saves as much cost as $(\alpha - 1) \sum_i d_{sum}(c_i, (C_i \cap C'_i) \setminus \hat{B}_i)$ on $(C_i \cap C'_i) \setminus \hat{B}_i$ compared to the optimum clustering under d'. So, the optimum clustering under d' must save this cost on other parts, i.e. A_i and $\hat{B}_i \setminus M_i$. By triangle inequality, we obtain the desired result. □

Claim 2 implies that if $c'_i \neq c_i$, then c_i should be far from c'_i, which then contradicts Claim 1. So $c'_i = c_i$ for all i, which leads to a bound on the selected bad points and consequently a bound on all bad points. Formally, by combining Claims 1 and 2, we get

$$\sum_{1 \leq i \leq k} \alpha d(c_i, c'_i) \big[(|A_i| + \alpha + 1 + |M_i|) - \frac{\alpha - 1}{\alpha + 1} (|(C_i \cap C'_i) \setminus \hat{B}_i| - |\hat{B}_i \setminus M_i| - |A_i| - (\alpha + 1)) I_i \big] \geq 0.$$

If $I_i = 0$, we have $d(c_i, c'_i) = 0$; if $I_i = 1$, since $|C_i| > (2 + \frac{2\alpha}{\alpha - 1})\epsilon n + \frac{2\alpha(\alpha+1)}{\alpha-1}$, the coefficient of $d(c_i, c'_i)$ is negative. So the left hand side is no greater than 0. Therefore, all terms are equal to 0, i.e. for all $1 \leq i \leq k$, $d(c_i, c'_i) = 0$. Then points in $\hat{B}_i$ will move to other clusters after perturbation, which means that $\hat{B}_i \subseteq M_i$. Then $|\bigcup_i \hat{B}_i| \leq |\bigcup_i M_i| \leq \epsilon n$. Specially, $|\hat{B}_i| \leq \epsilon n$ for any i. Then $|B_i| \leq \epsilon n$, otherwise $|\hat{B}_i|$ would be $\epsilon n + 1$. So $\hat{B}_i = B_i$, and $|\bigcup_i B_i| = |\bigcup_i \hat{B}_i| \leq \epsilon n$. □

2.3.2 Approximating the Optimum Clustering

Since (α, ϵ)-perturbation resilient instances have at most ϵn bad points, we can show that for $\alpha > 4$ such instances satisfy the ϵ-strict separation property (the property that after eliminating an ϵ fraction of the points, the remaining points are closer to points in their own cluster than to other points in different clusters). Therefore, we could use the clustering algorithms in [11, 13] to output a hierarchy such that the optimal clustering is ϵ-close pruning of this tree. However, this pruning might not have a small cost and it is not clear how to retrieve a small cost clustering from the tree constructed by these generic algorithms.

We design a new algorithm for obtaining a good approximation to the optimum for (α, ϵ)-perturbation resilient instances. This algorithm uses a novel linkage procedure to first construct a tree that we further process to output a desired clustering. This linkage based procedure uses an approximate version of the closure condition discussed in Section 2.2. More precisely, we say that the ball $\mathbb{B}_{p,q} = \{x \in S : d(p, x) \leq d(p, q)\}$ satisfies the approximate closure condition with respect to a clustering, if the clusters that have significantly amount of points within the ball satisfy the coverage and margin conditions after removing a few bad points: the points in the union of these clusters fall within the ball; points inside have some margin from points outside. The approximate closure condition is carefully designed to be robust to the bad points. So, if we begin with a clustering with each point in its own cluster, and repeatedly check whether $\mathbb{B}_{p,q}$ satisfies the condition for increasing pairwise distances $d(p, q)$ and merge clusters in the ball if it does, then we will merge good points from the same optimal cluster before merging them with good points outside. This linkage procedure thus produces a tree with a pruning that correctly clusters all good points. Some post-processing steps and dynamic programming on the tree then lead to an approximation solution. The formal guarantee is presented in Theorem 3, and the complete proof can be found in [14].

Theorem 3. *Suppose the clustering instance is (α, ϵ)-perturbation resilient to k-median. If $\alpha > 2+\sqrt{7}$ and $\epsilon \leq \rho/5$ where $\rho = (\min_i |C_i| - 15)/n$, then there exists a polynomial time algorithm that outputs a tree $\tilde{T}$ that contains a pruning that is ϵ-close to the optimum clustering. Moreover, if $\epsilon \leq \rho\epsilon'/5$ where $\epsilon' \leq 1$, the clustering produced is a $(1+\epsilon')$-approximation to the optimum.*

2.3.3 Sublinear Time Algorithm for the k-Median Objective

Consider a clustering instance (X, d) that is (α, ϵ)-perturbation resilient to k-median. We show that our algorithm can be made sublinear-time by running it on a small sample S of X. More specifically, let $N = |X|$, $\rho = \min_i |C_i|/N$ denote the fraction of the points in the smallest cluster, $D = \max_{p,q \in X} d(p, q)$ denote the diameter of X, $\zeta = \Phi_X(\mathbf{c})/N$ denote the average cost of the points in the optimum clustering. We have:

Theorem 4. *Suppose (X, d) is (α, ϵ)-perturbation resilient for $\alpha > 8$, $\epsilon < \rho/20$. Let $0 < \lambda < 1$. Then w.p. $\geq 1-\delta$, we can get an implicit clustering that is $2\frac{1+\lambda}{1-\lambda}(1+\frac{8\epsilon}{\rho-12\epsilon})$-approximation in time $O((\frac{kD^2}{\lambda^2\epsilon^2\zeta^2}\ln\frac{N}{\delta})^5)$.*

Proof Sketch: We sample a set S of size $n = \Theta(\frac{kD^2}{\lambda^2\epsilon^2\zeta^2}\ln\frac{N}{\delta})$ and run the algorithm for (α, ϵ)-perturbation resilient k-median on S to obtain the minimum cost pruning $\tilde{\mathcal{C}}$ and its centers $\tilde{\mathbf{c}}$. We then output the implicit clustering of the whole space X that assigns each point in X to its nearest neighbor in $\tilde{\mathbf{c}}$. Here we describe a proof sketch that shows $\Phi_X(\tilde{\mathbf{c}}) \approx \Phi_X(\mathbf{c})$.

Since when n is sufficiently large, w.h.p. $\Phi_X(\tilde{\mathbf{c}})/N \approx \Phi_S(\tilde{\mathbf{c}})/n$ and $\Phi_X(\mathbf{c})/N \approx \Phi_S(\mathbf{c})/n$, so it is sufficient to show $\Phi_S(\tilde{\mathbf{c}})$ is not much larger than $\Phi_S(\mathbf{c})$. However, $\tilde{\mathcal{C}}$ may be different from $\mathcal{C} \cap S$, so we need a bridge for the two.

Notice w.h.p. S has only $2\epsilon n$ bad points, and each cluster $C_i \cap S$ is large. Moreover, when $\alpha > 8$, any good point is 3 times closer to those in the same cluster than to those in other clusters. Then even if $\mathbf{c}$ are not sampled, we can choose an arbitrary good point from each cluster $C_i \cap S$ to be its center, so that we can prove the algorithm for (α, ϵ)-perturbation resilience for k-median still forms nodes N_i approximating the clusters, and $\tilde{T}$ has a pruning $\mathcal{P}'$, which is different from $\mathcal{C} \cap S$ only on the bad points. Suppose in S, $\mathbf{c}'$ are the optimal centers for $\mathcal{P}'$. Then we can use $\Phi_S(\mathcal{P}', \mathbf{c}')$ as a bridge for comparing $\Phi_S(\tilde{\mathbf{c}})$ and $\Phi_S(\mathbf{c})$.

On one hand, $\Phi_S(\tilde{\mathbf{c}}) \leq \Phi_S(\mathcal{P}', \mathbf{c}')$. This is because (1) since $\tilde{\mathcal{C}}$ is the minimum cost pruning, $\Phi_S(\tilde{\mathcal{C}}, \tilde{\mathbf{c}}) \leq \Phi_S(\mathcal{P}', \mathbf{c}')$; (2) since in $\Phi_S(\tilde{\mathbf{c}})$ each point is assigned to its nearest center but in $\Phi_S(\tilde{\mathcal{C}}, \tilde{\mathbf{c}})$ this may not be true, $\Phi_S(\tilde{\mathbf{c}}) \leq \Phi_S(\tilde{\mathcal{C}}, \tilde{\mathbf{c}})$. On the other hand, $\Phi_S(\mathcal{P}', \mathbf{c}')$ is not much larger than $\Phi_S(\mathbf{c})$. This is because (1) $\Phi_S(\mathcal{P}', \mathbf{c})$ is different from $\Phi_S(\mathbf{c})$ only on the bad points, and thus we can show the increase of cost is limited; (2) by the triangle inequality we have $\Phi_S(\mathcal{P}', \mathbf{c}') \leq 2\Phi_S(\mathcal{P}', \mathbf{c})$. □

2.4 α-Perturbation Resilience for the Min-Sum Objective

In this section we provide an efficient algorithm for clustering α-perturbation resilient instances for the min-sum k-clustering problem (Algorithm 2). We use the following notations: $d_{avg}(A, B) \doteq d_{sum}(A, B)/(|A||B|)$ and $d_{avg}(p, B) \doteq d_{avg}(\{p\}, B)$. For simplicity, we will assume that $\min_i |C_i|$ is known. (Otherwise, we can simply search over the n possible different values.)

Theorem 5. *For $(3\frac{\max_i |C_i|}{\min_i |C_i|-1})$-perturbation resilient instances, Algorithm 2 outputs the optimal min-sum k-clustering in polynomial time.*

Proof Sketch: First we show that the α-perturbation resilience property implies that for any two different optimal clusters C_i and C_j and any $A \subseteq C_i$, we have $\alpha d_{sum}(A, C_i \setminus A) < d_{sum}(A, C_j)$. This follows by

Algorithm 2 min-sum, α perturbation resilience

Input: Data set S, distance function $d(\cdot,\cdot)$ on S, $\min_i |C_i|$.

▷ `Phase 1: tree construction`

Step 1: Connect each point with its $\frac{1}{2}\min_i |C_i|$ nearest neighbors.

Step 2: Initialize the clustering $\mathcal{C}'$ with each connected component being a cluster.

Step 3: Repeat till one cluster remains in $\mathcal{C}'$: merge clusters C, C' in $\mathcal{C}'$ which minimize $d_{avg}(C, C')$.

Step 4: Let T be the tree with components as leaves and internal nodes corresponding to the merges.

▷ `Phase 2: pruning`

Step 5: Apply dynamic programming on T to get the minimum min-sum cost pruning $\tilde{\mathcal{C}}$.

Output: Output $\tilde{\mathcal{C}}$.

considering the perturbation where $d'(p,q) = \alpha d(p,q)$ if $p \in A, q \in C_i \backslash A$ and $d'(p,q) = d(p,q)$ otherwise, and using the fact that the optimum does not change after the perturbation. This can be used to show that when $\alpha > 3\frac{\max_i |C_i|}{\min_i |C_i|-1}$ we have: (1) for any optimal clusters C_i and C_j and any $A \subseteq C_i$, $A' \subseteq C_j$ s.t. $\min(|C_i \setminus A|, |C_j \setminus A'|) > \min_i |C_i|/2$ we have $d_{avg}(A, A') > \min\{d_{avg}(A, C_i \setminus A), d_{avg}(A', C_j \setminus A')\}$; (2) for any point p in the optimal cluster C_i, twice its average distance to points in $C_i \setminus \{p\}$ is smaller than the distance to any point in other optimal cluster C_j. Fact (2) implies that for any point $p \in C_i$ its $|C_i|/2$ nearest neighbors are in the same optimal cluster, so the leaves of the tree T are laminar to the optimum clustering. Fact (1) can be used to show that the merge steps preserve the laminarity with the optimal clustering, so the minimum cost pruning of T will be the optimal clustering, as desired. □

The algorithm can also be made sublinear-time. More specifically, consider a clustering instance (X, d) that is α-perturbation resilient to min-sum, and let $N = |X|$, $\rho = \min_i |C_i|/N$ denote the fraction of the points in the smallest cluster, $D = \max_{p,q \in X} d(p,q)$ denote the diameter of X, $\eta = \min_{p \in X, 1 \le i \le k} d_{avg}(p, C_i)$ denote the minimum average distance between points and optimal clusters. We have:

Theorem 6. *Suppose the clustering instance (X, d) is $(6\frac{\max_i |C_i|}{\min_i |C_i|-1})$-perturbation resilient to the min-sum objective. Then w.p. $\geq 1 - \delta$, we can get an implicit optimum clustering in time $O((\frac{D^2}{\rho^2\eta^2} \ln \frac{Nk}{\delta})^3)$.*

Proof Sketch: We sample a set S of size $n = \Theta(\frac{D^2}{\rho^2\eta^2} \ln \frac{Nk}{\delta})$ and run Algorithm 2 on S. We then output the implicit clustering of the whole space X that assigns each point $p \in X$ to $\tilde{C}_i \in \tilde{\mathcal{C}}$ s. t. $d_{sum}(p, \tilde{C}_i)$ is minimized. We have that for any $p \in C_i$ and $C_j (j \neq i)$, $3\frac{\max_i |C_i \cap S|}{\min_i |C_i \cap S|-1} d_{sum}(p, C_i \cap S) < d_{sum}(p, C_j \cap S)$, since when n is sufficiently large, $d_{sum}(p, C_i \cap S) \approx d_{sum}(p, C_i)|S|/|X|$, $d_{sum}(p, C_j \cap S) \approx d_{sum}(p, C_j)|S|/|X|$ and $\frac{\max_i |C_i \cap S|}{\min_i |C_i \cap S|-1} \approx \frac{\max_i |C_i|}{\min_i |C_i|-1}$. So the tree T is laminar to $\mathcal{C} \cap S$. Since clusters in $\mathcal{C} \cap S$ are far apart, the cost increased by joining different clusters in it is larger than that saved by splitting clusters, so $\mathcal{C} \cap S$ is the minimum cost pruning, so Algorithm 2 on S outputs $\mathcal{C} \cap S$, and the theorem follows. □

2.5 Future Directions

A natural direction for future investigation is to explore whether one can take advantage of smaller perturbation factors for perturbation resilient instances. More precisely, can we still get the optimal clustering for α-perturbation resilient k-median when $\alpha < 1 + \sqrt{2}$? Can we deal with unbalanced clusters for α-perturbation resilient min-sum, i.e. get rid of the $\max_i |C_i| / \min_i |C_i|$ factor in the bound for α?

Another natural direction is to design an algorithm for (α, ϵ)-perturbation resilient min-sum instances. While we give an (α, ϵ)-perturbation resilience analysis for the k-median objective, such an analysis for the more difficult min-sum objective remains open. The (α, ϵ)-perturbation resilient k-median instances have nice structure property that there are only a small fraction of bad points, which is utilized to design our algorithm. What would be an similar structure property for min-sum? How can we utilize the property to design approximation algorithms?

It is also interesting to study perturbation resilient instances in Euclidian spaces. That is, while d is a Euclidean metric, the perturbed distance d' need not be so as in Definitions 2 and 4. Alternatively, one could also consider a natural version of Definitions 2 and 4 in which d' must be Euclidean as well, and in fact implemented via a perturbation of coordinate values.

Finally, another interesting direction is to study the combination of stability properties. Besides perturbation resilience, there also exist other notions of stability properties for clustering [61, 5, 10, 4, 48, 6]. If the clustering instance satisfies simultaneously two or more of such properties, can we design algorithms that succeed with lower degrees of stability and/or have better approximation factor?

Chapter 3

Distributed Clustering

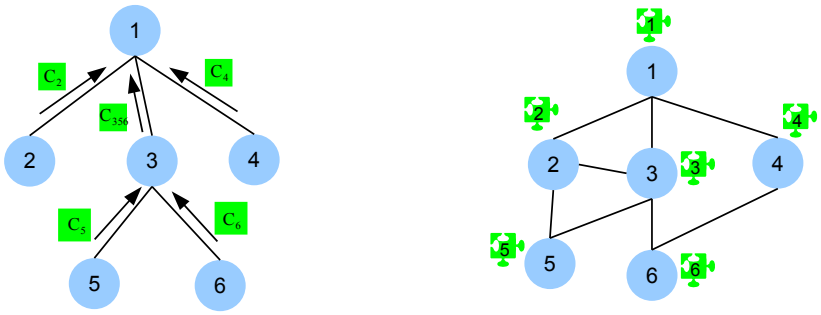
Most classic clustering algorithms are designed for the centralized setting, but in recent years data has become distributed over different locations, such as distributed databases [60, 29], images and videos over networks [57], surveillance [39] and sensor networks [28, 40]. In many of these applications the data is inherently distributed because, as in sensor networks, it is collected at different sites. As a consequence it has become crucial to develop clustering algorithms which are effective in the distributed setting.

Several algorithms for distributed clustering have been proposed and empirically tested. Some of these algorithms [35, 67, 30] are direct adaptations of centralized algorithms which rely on statistics that are easy to compute in a distributed manner. Other algorithms [45, 46] generate summaries of local data and transmit them to a central coordinator which then performs the clustering algorithm. No theoretical guarantees are provided for the clustering quality in these algorithms, and they do not try to minimize the communication cost. Additionally, most of these algorithms assume that the distributed nodes can communicate with all other sites or that there is a central coordinator that communicates with all other sites.

Here we study the problem of distributed clustering where the data is distributed across nodes whose communication is restricted to the edges of an arbitrary graph. We provide algorithms with small communication cost and provable guarantees on the clustering quality. Our technique for reducing communication is based on the construction of a small set of points called *coreset* that act as a proxy for the entire data set.

An ϵ-*coreset* is a weighted set of points whose cost on any set of centers is approximately the cost of the original data on those same centers up to accuracy ϵ. Thus an approximate solution for the coreset is also an approximate solution for the original data. Coresets have previously been studied in the centralized setting ([42, 33]) but have also recently been used for distributed clustering as in [72] and as implied by [34]. In this work, we propose a distributed algorithm for k-means and k-median, by which each node constructs a local portion of a global coreset. Communicating the total cost of local approximate solutions to each node is enough for the construction, leading to low communication cost overall. The nodes then share the local portions of the coreset, which can be done efficiently in general graphs using a message passing approach.

Comparison to Other Coreset Algorithms for Distributed Clustering: Since coresets summarize local information they are a natural tool to use when trying to reduce communication complexity. If each node constructs an ϵ-coreset on its local data, then the union of these coresets is clearly an ϵ-coreset for the entire data set. Unfortunately the size of the coreset using this natural approach increases greatly with the number of nodes. More sophisticated approaches, such as [72], reduce the size of the global coreset by approximating the union of local coresets with another coreset. They assume nodes communicate over a rooted tree, with each node passing its coreset to its parent. Because the approximation factor of the constructed coreset depends on the quality of its component coresets, the accuracy a coreset needs (and thus



(a) Zhang et al.[72] (b) Our Construction

Figure 3.1: **(a)** Each node computes a coreset on the weighted pointset for its own data and its subtrees' coresets. **(b)** Local constant approximation solutions are computed, and the costs of these solutions are used to coordinate the construction of a local portion on each node.

the overall communication complexity) scales with the height of this tree. Although it is possible to find a spanning tree in any communication network, when the graph has large diameter every tree has large height. In particular many natural networks such as grid networks have a large diameter ($\Omega(\sqrt{n})$ for grids) which greatly increases the size of coresets which must be communicated across the lower levels of the tree.

We show that it is possible to construct a global coreset with low communication overhead. This is done by distributing the coreset construction procedure rather than combining local coresets. The communication needed to construct this coreset is negligible – just a single value from each data set representing the approximate cost of their local optimal clustering. Since the sampled global ϵ-coreset is the same size as any local ϵ-coreset, this leads to an improvement of the communication cost over the other approaches. See Figure 3.1 for an illustration. The constructed coreset is smaller by a factor of n in general graphs, and is independent of the underlying communication topology. This method excels in sparse networks with large diameters, where the previous approaches require coresets that are quadratic in the size of the diameter for k-median and quartic for k-means.

3.1 Preliminaries

Let $d(p, q)$ denote the Euclidean distance between any two points $p, q \in \mathbf{R}^d$. The goal of k-means clustering is to find a set of k centers $\mathbf{x} = \{x_1, x_2, \ldots, x_k\}$ which minimize the k-means cost of data set $P \subseteq \mathbf{R}^d$. Here the k-means cost is defined as $\text{cost}(P, \mathbf{x}) = \sum_{p \in P} d(p, \mathbf{x})^2$ where $d(p, \mathbf{x}) = \min_{x \in \mathbf{x}} d(p, x)$. If P is a weighted data set with a weighting function w, then the k-means cost is defined as $\sum_{p \in P} w(p) d(p, \mathbf{x})^2$. Similarly, the k-median cost is defined as $\sum_{p \in P} d(p, \mathbf{x})$. Both k-means and k-median cost functions are known to be **NP**-hard to minimize, so we generally aim at approximation solutions.

In the distributed clustering task, we consider a set of n nodes $V = \{v_i, 1 \leq i \leq n\}$ which communicate on an undirected connected graph $G = (V, E)$ with $m = |E|$ edges. More precisely, an edge $(v_i, v_j) \in E$ indicates that v_i and v_j can communicate with each other. Here we measure the communication cost in number of points, and assume for simplicity that there is no latency in the communication. On each node v_i, there is a local set of data points P_i, and the global data set is $P = \bigcup_{i=1}^{n} P_i$. The goal is to find a set of k centers $\mathbf{x}$ which optimize $\text{cost}(P, \mathbf{x})$ while keeping the computation efficient and the communication

cost as low as possible. Our focus is to reduce the total communication cost while preserving theoretical guarantees for approximating clustering cost.

3.1.1 Coresets

For the distributed clustering task, a natural approach to avoid broadcasting raw data is to generate a local summary of the relevant information. In the centralized setting, the idea of summarization with respect to the clustering task is captured by the concept of coresets [42, 33]. A coreset is a set of points, together with a weight for each point, such that the cost of this weighted set approximates the cost of the original data for any set of k centers. Formally:

Definition 7 (coreset). *An ϵ-coreset for a set of points P with respect to a center-based cost function is a set of points S and a set of weights $w : S \to \mathbf{R}$ such that for any set of centers $\mathbf{x}$,*

$$(1-\epsilon)\text{cost}(P,\mathbf{x}) \leq \sum_{p\in S} w(p)\text{cost}(p,\mathbf{x}) \leq (1+\epsilon)\text{cost}(P,\mathbf{x}).$$

In the centralized setting, many coreset construction algorithms have been proposed for k-median, k-means and some other cost functions. For example, for points in $\mathbf{R}^d$, algorithms in [33] construct coresets of size $t = \tilde{O}(kd/\epsilon^4)$ for k-means and coresets of size $t = \tilde{O}(kd/\epsilon^2)$ for k-median. In the distributed setting, it is natural to ask whether there exists an algorithm that constructs a small coreset for the entire point set but still has low communication cost. We present a distributed algorithm which constructs a global coreset roughly the same size as the centralized construction and only needs a single value[1] communicated to each node. This serves as the basis for our distributed clustering algorithm.

3.2 Distributed Coreset Construction

In this section, we design a distributed coreset construction algorithm for k-means and k-median. Note that the underlying technique can be extended to other additive clustering objectives such as k-line median.

To gain some intuition on the distributed coreset construction algorithm, we briefly review the coreset construction algorithm in [33] in the centralized setting. The coreset is constructed by computing a constant approximation solution for the entire data set, and then sampling points proportional to their contributions to the cost of this solution. Intuitively, the points close to the nearest centers can be approximately represented by the nearest centers while points far away cannot be well represented. Thus, points should be sampled with probability proportional to their contributions to the cost.

Directly adapting the algorithm to the distributed setting would require computing a constant approximation solution for the entire data set. We show that a global coreset can be constructed in a distributed fashion by estimating the cost of the entire data set with the sum of local approximations. With this approach, it suffices for nodes to communicate the total costs of their local solutions. The distributed coreset construction algorithm is described in Algorithm 3, and the formal guarantee is presented in Theorem 7.

Theorem 7. *For distributed k-means and k-median clustering on a graph, there exists an algorithm such that with probability at least $1-\delta$, the union of its output on all nodes is an ϵ-coreset for $P = \bigcup_{i=1}^{n} P_i$. The size of the coreset is $O(\frac{1}{\epsilon^4}(kd+\log\frac{1}{\delta})+nk\log\frac{nk}{\delta})$ for k-means, and $O(\frac{1}{\epsilon^2}(kd+\log\frac{1}{\delta})+nk)$ for k-median. The total communication cost is $O(mn)$.*

[1] The value that is communicated is the sum of the costs of approximations to the local optimal clustering. This is guaranteed to be no more than a constant factor times larger than the optimal cost.

Algorithm 3 Distributed coreset construction

Input: $t, \{P_i, 1 \leq i \leq n\}$.

Round 1: on each node $v_i \in V$

Step 1 Compute a constant approximation B_i for P_i.
Communicate $\text{cost}(P_i, B_i)$ to all other nodes.

Round 2: on each node $v_i \in V$

Step 2 Set $t_i = \frac{t \text{ cost}(P_i, B_i)}{\sum_{j=1}^{n} \text{cost}(P_j, B_j)}$ and $m_p = \text{cost}(p, B_i), \forall p \in P_i$.

Step 3 Pick a non-uniform random sample S_i of t_i points from P_i,
where for every $q \in S_i$ and $p \in P_i$, we have $q = p$ with probability $m_p / \sum_{q \in P_i} m_q$.
Let $w_p = \frac{\sum_i \sum_{q \in P_i} m_q}{t m_p}$ for each $p \in S_i$.

Step 4 For $\forall b \in B_i$, let $P_b = \{p \in P_i : d(p, b) = d(p, B_i)\}$, $w_b = |P_b| - \sum_{p \in P_b \cap S_i} w_p$.

Output: $S_i \cup B_i, \{w_p : p \in S_i \cup B_i\}, 1 \leq i \leq n$.

Proof Sketch for k-means: The key is a sampling lemma using the dimension of a function space.

Definition 8 ([33])**.** *Let H be a finite set of functions from a set $\mathbf{X}$ to $\mathbf{R}_{\geq 0}$. Let $\text{range}(H, \mathbf{x}, r) = \{h \in H : h(\mathbf{x}) \leq r\}$. The dimension of the function space $\dim(H, \mathbf{X})$ is the smallest integer d such that for any $G \subseteq H$,*

$$\left|\{G \cap \text{range}(H, \mathbf{x}, r) : \mathbf{x} \in \mathbf{X}, r \geq 0\}\right| \leq |G|^d.$$

Note that let $I_{\mathbf{x},r}(h) = I[h(\mathbf{x}) \leq r]$, and $I = \{I_{\mathbf{x},r} : \mathbf{x} \in \mathbf{X}, r \geq 0\}$, then the dimension $\dim(H, \mathbf{X})$ is closely related to the VC dimension of I on H, i.e. by definition we have $\dim(H, \mathbf{X}) = \Theta(\text{VC-dim}(I))$. A sampling argument using the dimension then leads to the following lemma.

Lemma 3. *Fix a set H of functions $h_p : \mathbf{X} \to \mathbf{R}_{\geq 0}$, a set of weights $m_p \in \mathbf{R}_{>0}$ for $p \in P$. Let S be a sample drawn i.i.d. from P where each p is sampled with probability $\frac{m_p}{\sum_{q \in P} m_q}$, and let $w_p = \frac{\sum_{q \in P} m_q}{m_p |S|}$ for $p \in S$. If for a sufficiently large c,*

$$|S| \geq \frac{c}{\epsilon^2}\left(\dim(H, \mathbf{X}) + \log\frac{1}{\delta}\right)$$

then with probability at least $1 - \delta, \forall \mathbf{x} \in \mathbf{X}$:

$$\left|\sum_{p \in P} h_p(\mathbf{x}) - \sum_{p \in S} w_p h_p(\mathbf{x})\right| \leq \epsilon \max_{p \in P} \frac{h_p(\mathbf{x})}{m_p} \sum_{q \in P} m_q.$$

To apply the lemma, we note that the bound provided depends on two terms: $\max_{p \in P} \frac{h_p(\mathbf{x})}{m_p}$ and $\sum_{q \in P} m_q$. Ideally, we want $\max_{p \in P} \frac{h_p(\mathbf{x})}{m_p}$ to be bounded and $\sum_{q \in P} m_q$ to be small. If we could choose $m_p = h_p(\mathbf{x})$, then the bound would be $\epsilon \sum_p h_p(\mathbf{x})$, which means the weighted cost of the sample approximates the original cost up to ϵ factor. However, m_p should be independent of $\mathbf{x}$. A better strategy is to choose m_p to be the upper bound for $h_p(\mathbf{x})$, then the bound is $\epsilon \sum_{q \in P} m_q$. The key now becomes to design $h_p(\mathbf{x})$ with suitable upper bounds for our problems, i.e. we should choose $h_p(\mathbf{x})$ such that its value does not vary much with $\mathbf{x}$.

We now consider applying the lemma to the k-median problem by choosing suitable $\{h_p\}$ and $\{m_p\}$. First, note that we cannot apply this lemma directly to the cost, since a suitable upper bound is not available by choosing $h_p(\mathbf{x}) := \text{cost}(p, \mathbf{x})$. We therefore consider the local approximation solutions. Let b_p denote the closest center to p, then $\text{cost}(b_p, \mathbf{x})$ will be close to $\text{cost}(p, \mathbf{x})$. We now aim to approximate the error $\sum_p [\text{cost}(p, \mathbf{x}) - \text{cost}(b_p, \mathbf{x})]$, rather than to approximate $\sum_p \text{cost}(p, \mathbf{x})$ directly. More precisely, let $h_p(\mathbf{x}) := \text{cost}(p, \mathbf{x}) - \text{cost}(b_p, \mathbf{x}) + \text{cost}(p, b_p)$, where $\text{cost}(p, b_p)$ is added so that $h_p \geq 0$. We apply the lemma to $h_p(\mathbf{x})$ and $m_p = \text{cost}(p, b_p)$. Since $0 \leq h_p \leq 2\text{cost}(p, b_p)$, the lemma bounds the difference between the cost of h_p and its sampled cost by $2\epsilon \sum_{p \in P} \text{cost}(p, b_p)$, so we have an $O(\epsilon)$-approximation.

Second, note that the difference between the true cost and the cost of the sampled points is not the same as the difference between the cost of $\{h_p\}$ and its sampled cost. We show that the gap depends only on the costs of the approximation points $\{b_p\}$, i.e.

$$\sum_{p \in P} h_p(x) - \sum_{p \in S} w_p h_p(x) \quad = \quad \sum_{p \in P} \text{cost}(p, x) - \sum_{p \in S} w_p \text{cost}(p, x) - f(\{\text{cost}(b_p, x)\}).$$

for some function f. The approximation points $\{b_p\}$ are then weighted and added to the coreset, so that the difference between the true cost and the cost of the coreset equals the difference between the cost of $\{h_p\}$ and its sampled cost, leading to an approximation good for every set of centers.

Formally, we have the following lemma for k-median with

$$H = \{h_p : h_p(\mathbf{x}) = d(p, \mathbf{x}) - d(b_p, \mathbf{x}) + d(p, b_p), p \in P\}.$$

Lemma 4. *For k-median, the output of Algorithm 3 is an ϵ-coreset with probability at least $1 - \delta$, if*

$$t \geq \frac{c}{\epsilon^2} \left(\dim(H, (\mathbf{R}^d)^k) + \log \frac{1}{\delta} \right)$$

for a sufficiently large constant c.

The proof is completed by noting $\dim(H, (\mathbf{R}^d)^k) = O(kd)$ (see [33]), and bounding the total communication cost by $O(mn)$ since we can broadcast the local costs with $O(mn)$ communication (see Algorithm 5).

Proof Sketch for k-means: We have for k-means a similar lemma that when $t = O(\frac{1}{\epsilon^4}(kd + \log \frac{1}{\delta}) + nk \log \frac{nk}{\delta}))$, the algorithm constructs an ϵ-coreset with probability at least $1 - \delta$. The key idea is the same as that for k-median: we use centers $\{b_p\}$ from the local approximation solutions as an approximation to the original data $\{p\}$, and show that the error between the total cost and the weighted sample cost is approximately the error between the cost of h_p and its sampled cost (compensated by the weighted centers), which is shown to be small by Lemma 3.

The key difference between k-means and k-median is that triangle inequality applies directly to the k-median cost. In particular, for the k-median problem note that $\text{cost}(b_p, p) = d(b_p, p)$ is an upper bound for the error of b_p on any set of centers, i.e. $\forall \mathbf{x} \in (\mathbf{R}^d)^k$,

$$d(b_p, p) \geq |d(p, \mathbf{x}) - d(b_p, \mathbf{x})| = |\text{cost}(p, \mathbf{x}) - \text{cost}(b_p, \mathbf{x})|$$

by triangle inequality. Then we can construct $h_p(\mathbf{x}) := \text{cost}(p, \mathbf{x}) - \text{cost}(b_p, \mathbf{x}) + d(b_p, p)$ such that $h_p(\mathbf{x})$ is bounded. In contrast, for k-means, the error $|\text{cost}(p, \mathbf{x}) - \text{cost}(b_p, \mathbf{x})| = |d(p, \mathbf{x})^2 - d(b_p, \mathbf{x})^2|$ does not have such an upper bound. The main change to the analysis is that we divide the points into two categories: good points whose costs approximately satisfy the triangle inequality (up to a factor of $1/\epsilon$) and bad points. The good points for a fixed set of centers $\mathbf{x}$ are defined as

$$G(\mathbf{x}) = \{p \in P : |\text{cost}(p, \mathbf{x}) - \text{cost}(b_p, \mathbf{x})| \leq \Delta_p\}$$

where the upper bound is $\Delta_p = \frac{\text{cost}(p,b_p)}{\epsilon}$. Good points we can bound as before. For bad points we can show that while the difference in cost is larger than Δ_p, it must still be small, namely $O(\epsilon \min\{\text{cost}(p,\mathbf{x}), \text{cost}(b_p,\mathbf{x})\})$.

Formally, the functions h_p are restricted to be defined only over good points:

$$h_p = \begin{cases} \text{cost}(p,\mathbf{x}) - \text{cost}(b_p,\mathbf{x}) + \Delta_p & \text{if } p \in G(\mathbf{x}), \\ 0 & \text{otherwise.} \end{cases}$$

This leads to

$$\begin{aligned} & \sum_{p\in P} \text{cost}(p,\mathbf{x}) - \sum_{p\in S\cup B} w_p \text{cost}(p,\mathbf{x}) \\ = & \sum_{p\in P} h_p(\mathbf{x}) - \sum_{p\in S} w_p h_p(\mathbf{x}) && (3.1) \\ & + \sum_{p\in P\setminus G(\mathbf{x})} [\text{cost}(p,\mathbf{x}) - \text{cost}(b_p,\mathbf{x}) + \Delta_p] && (3.2) \\ & - \sum_{p\in S\setminus G(\mathbf{x})} w_p[\text{cost}(p,\mathbf{x}) - \text{cost}(b_p,\mathbf{x}) + \Delta_p] && (3.3) \end{aligned}$$

Lemma 3 bounds (3.1) by $O(\epsilon)\text{cost}(P,\mathbf{x})$, but we need an accuracy of ϵ^2 to compensate for the $1/\epsilon$ factor in the upper bound, resulting in a $O(1/\epsilon^4)$ factor in the sample complexity.

Note that for (3.2) and ((3.3), $|\text{cost}(p,\mathbf{x}) - \text{cost}(b_p,\mathbf{x})| > \Delta_p$ since $p \notin G(\mathbf{x})$. Furthermore, $p \notin G(\mathbf{x})$ only when p and b_p are close to each other and far away from $\mathbf{x}$. This then leads to

$$|\text{cost}(p,\mathbf{x}) - \text{cost}(b_p,\mathbf{x})| \le O(\epsilon) \min\{\text{cost}(p,\mathbf{x}), \text{cost}(b_p,\mathbf{x})\}$$

which can then bound (3.2) and (3.3) by $O(\epsilon)\text{cost}(P,\mathbf{x})$. The proof is then completed by choosing a suitable ϵ, and bounding $\dim(H, (\mathbf{R}^d)^k) = O(kd)$ as in the k-median case.

3.3 Effect of Network Topology on Communication Cost

Combined the distributed coreset construction algorithm with a message passing approach for globally sharing information, and use it for collecting information for coreset construction and sharing the local portions of the coreset, we have Algorithm 4 for distributed clustering.

This leads immediately to the following main result for distributed clustering on general graphs.

Theorem 8. *Given an α-approximation algorithm for weighted k-means (k-median respectively) as a subroutine, there exists an algorithm that with probability at least $1-\delta$ outputs a $(1+\epsilon)\alpha$-approximation solution for distributed k-means (k-median respectively) clustering. The total communication cost is $O(m(\frac{1}{\epsilon^4}(kd + \log\frac{1}{\delta}) + nk\log\frac{nk}{\delta}))$ for k-means, and $O(m(\frac{1}{\epsilon^2}(kd+\log\frac{1}{\delta})+nk))$ for k-median.*

Our algorithm can also be applied on a rooted tree when restricting message passing to operating along this tree, leading to the following theorem for this special case.

Theorem 9. *Given an α-approximation algorithm for weighted k-means (k-median respectively) as a subroutine, there exists an algorithm that with probability at least $1-\delta$ outputs a $(1+\epsilon)\alpha$-approximation solution for distributed k-means (k-median respectively) clustering on a rooted tree of height h. The total communication cost is $O(h(\frac{1}{\epsilon^4}(kd+\log\frac{1}{\delta}) + nk\log\frac{nk}{\delta}))$ for k-means, and $O(h(\frac{1}{\epsilon^2}(kd+\log\frac{1}{\delta})+nk))$ for k-median.*

Algorithm 4 Distributed clustering on a graph

Input: $\{P_i\}, \{N_i\}, \mathcal{A}_\alpha$. Here N_i is the neighbors of v_i, and $\mathcal{A}_\alpha$ is a α-approximation algorithm for weighted clustering instances.

Round 1: on each node v_i

Step 1 Construct its local portion D_i of an $\epsilon/2$-coreset by Algorithm 3, using Message-Passing for communicating the local costs.

Round 2: on each node v_i

Step 2 Call Message-Passing(D_i, N_i).

Step 3 $\mathbf{x} = \mathcal{A}_\alpha(\bigcup_j D_j)$.

Output: $\mathbf{x}$

Algorithm 5 Message-Passing(I_i, N_i)

Input: I_i is the message, N_i are the neighbors.

Step 1 Let R_i denote the information received.
Initialize $R_i = \{i\}$. Send I_i to all the neighbors.

Step 2 While $R_i \neq [n]$:
If receive message I_j,
$R_i = R_i \cup \{j\}$ and send I_j to all the neighbors.

Our approach improves the cost of $\tilde{O}(\frac{nh^2kd}{\epsilon^2})$ for k-median in [72] [2] . In a general graph, any rooted tree will have its height h at least as large as half the diameter. For sensors in a grid network, this implies $h = \Omega(\sqrt{n})$. In this case, our algorithm gains a significant improvement over existing algorithms.

Acknowledgements This work in this chapter is joint work with Steven Ehrlich.

3.4 Future Directions

The current work leads to open questions and future directions, some of which are presented below.

Communication Complexity of Distributed Clustering It is of great interest to further reduce the communication cost in distributed clustering, or get lower bounds that show the necessary dependence of the communication cost on some parameters (such as the number of clusters, the number of nodes).

It is especially interesting to reduce the dependence on $1/\epsilon$ where ϵ is the accuracy parameter. The following are two possible approaches to achieve this goal.

- It would be interesting to design an algorithm that constructs coresets of smaller size. Note that there are other coreset construction algorithms, such as [24], that have better dependence on $1/\epsilon$ for k-means, but they have worse dependence on other parameters like number and dimension of

[2] Their result depended on older coreset constructions, with larger size. When we compare to [72] we assume they use the coreset construction technique of [33] to reduce their coreset size and communication cost.

data points. We would like to improve the dependence on $1/\epsilon$, and also maintain or improve the dependence on other parameters.

- Our current results are based on coresets, and the approach consists of two rounds of communication: broadcasting the costs of local approximation solutions for coreset construction and broadcasting the local portions of the coreset constructed. Is it possible to use multiple rounds of communication to reduce the cost? For example, in each round the nodes communicate with each other adaptively based on the "coarse" coresets obtained so far, resulting in "finer" coresets. The problem lies in the following questions. In what sense are the coresets coarse or fine? Is this with respect to the worse accuracy of approximating the clustering cost of the original data? Or is it with respect to the range of sets of centers on which the coreset can approximate the cost? Once we have coarse coresets, how can we refine them with low communication cost?

It is also interesting to get lower bounds, especially those showing dependence on $1/\epsilon$. To emphasize the communication complexity, we can assume that the nodes have infinite computation power, as is typical in the communication complexity literature.

Distributed SVM Support Vector Machine (SVM) is a well-known method for supervised learning with successful results in many applications. A key observation made by [69, 68] is that the quadratic programming problem underlying SVM is equivalent to that defining a minimal enclosing ball (MEB) problem, that is, the problem of computing the ball of smallest radius containing a set of points. Recent work in computational geometry [7, 71, 25] introduced the notion of ϵ-coreset for MEB, defined as a subset of points whose MEB approximates the MEB of the original data with accuracy ϵ. They also proposed algorithms capable to construct an ϵ-coreset of size $O(1/\epsilon)$. Therefore, a small coreset can first be selected and then used for training SVM [69, 68].

For distributed SVM, [54] proposed to combine coresets for local data sets and use the union for training SVM. They only showed that the union of exact local coresets is an exact global coreset (i.e. $\epsilon = 0$). It is an interesting open question whether this is also true for ϵ-coreset when $\epsilon > 0$. Another more interesting question is whether the idea of distributed construction of a global coreset works for SVM. If this is true, then we can improve the communication cost of the simple approach of combining local coresets.

Chapter 4

Cluster Hierarchies

Classical clustering tasks aim at finding a partition of the data points. However, in many modern applications, the goal is to identify meaningful clusters, which generally do not form a partition. For example, in analyzing social networks, it is meaningful to identify interesting subsets called communities based on the affinity between members of the network. For such tasks, we aim to output a collection of clusters which include the target clusters. To avoid outputting too many clusters (for example, all subsets of the data), the collection should have some structure such as a tree hierarchy or other more general hierarchies. In the following, we consider two such tasks: detecting community hierarchies and computing all weak clusters.

4.1 Modeling and Detecting Community Hierarchies

The structure of networks has been extensively studied over the past several years in many disciplines, ranging from mathematics and computer science to sociology and biology. A significant amount of recent work in this area has focused on the development of community detection algorithms. The community structure reflects how entities in a network form meaningful groups such that interactions within the groups are more active compared to those between the groups and the outside world. The discovery of these communities is useful for understanding the structure of the underlying network, or making decisions in the network [36, 37, 58, 59].

Generally, a community should be thought of as a subset whose members have more interactions with each other than with the remainder of the network. This intuition is captured by some recently proposed models [2, 8, 1, 43, 47]. Additionally, recent studies show that networks often exhibit hierarchical organization, in which communities can contain groups of sub-communities, and so forth over multiple scales. For example, this can be observed in ecological niches in food webs, modules in biochemical networks or groups of common interest in social websites [63, 49, 27]. It is also shown empirically and theoretically that hierarchical structures can simultaneously explain and quantitatively reproduce many commonly observed topological properties of networks [26, 66, 38]. This suggests that the hierarchical structure should also be reflected when modeling real world communities.

Although some heuristic approaches [38, 50] have been proposed to detect community hierarchies, few works have formalized this hierarchical property, and there are no theoretical performance guarantees for the algorithms. Inspired by the related work in clustering [9], we define a notion of communities that both reflects the tight connections within communities and explicitly models the hierarchy of communities. In our model, each member of a community falls into a sub-community, and the sub-communities within this community have active interactions with each other while entities outside this community have fewer

interactions with members inside. Given this formalization, we then propose an efficient algorithm that detects all the communities in this model, and prove that all the communities form a hierarchy.

4.1.1 Hierarchical Community Model

A network is typically represented as a graph $G = (V, E)$ on a set of $n = |V|$ points [1], where the edges could be undirected or directed, unweighted or weighted. The graph implicitly specifies a neighborhood structure on the points, i.e. for each point there is a ranking of all other points according to the level of possible interaction. More precisely, we assume that we have a neighborhood function N which given a point p and a threshold t outputs a list $N_t(p)$ containing the t nearest neighbors of p in V.

The neighborhood function can be used to formalize a model of hierarchical communities. Using this neighborhood function, the tight connections within communities can be naturally rephrased as follows: for suitable t, most points p in the community have most of the nearest neighbors $N_t(p)$ from the community while points outside have just a few nearest neighbors from the community. Besides this, we also want to formalize the hierarchical structure that sub-communities in a lower, more local level actively interacting with each other form a community in a higher, more global level. The connections between the sub-communities can also be rephrased using the language of neighborhood: a majority of points in each sub-community have most of the nearest neighbors from the sub-communities in the same community.

In the remainder of the section, we specify our model based on the neighborhood function. We begin with the following notion of compact blobs, which will serve as a building block for our model.

Definition 9. *A subset A of points is called an α-compact blob, if out of the $|A|$ nearest neighbors:*

- *any point $p \in A$ has at most αn neighbors outside A, i.e. $|N_{|A|}(p) \setminus A| \leq \alpha n$;*
- *any point $q \notin A$ has at most αn neighbors inside A, i.e. $|N_{|A|}(q) \cap A| \leq \alpha n$.*

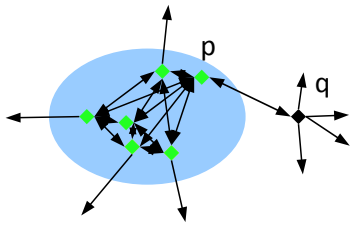


Figure 4.1: Illustration of an α-compact blob. An edge (x, y) means that y is one of x's nearest neighbors.

Note that the notion of compact blobs is the same as the clusters that satisfy the α-good neighborhood property defined in [9]. The notion captures the desired property of communities to be detected: members in the community have many more interactions with other members inside the community and have fewer interactions with those outside. However, in practice, the notion may seem somewhat restricted. First, it requires all the members in the community have most interactions with other members inside the community, which may not be the case in real life. For example, some members in the boundary may have more interactions with the outside world, i.e. they have more than αn neighbors from outside. Based on this consideration, we define the (α, β)-stable property as follows.

[1] We distinguish the nodes in the hierarchy our algorithm builds from the points in the graph.

Definition 10. *A community C is (α, β)-stable if*

- *any point $p \in C$ falls into a α-compact blob $A_p \subseteq C$ of size greater than $6\alpha n$,*
- *for any point $p \in C$, at least β fraction of points in A_p have all but at most αn nearest neighbors from C out of their $|C|$ nearest neighbors,*
- *any point q outside C has at most αn nearest neighbors from C out of their $|C|$ nearest neighbors.*

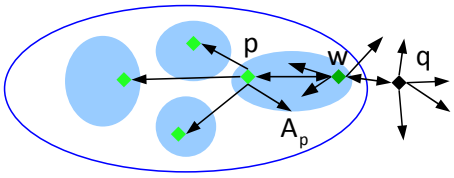


Figure 4.2: Illustration of an (α, β)-stable community. An edge (x, y) means that y is one of x's nearest neighbors. Note that point w lies on the "boundary" of the community. It falls into the compact blob A_p, but does not have most of its nearest neighbors from the community.

Informally, the first condition means that every point falls into a sufficiently large compact blob in its community. This condition formalizes the local neighborhood structure that each member interacts actively with sufficiently many members in the community. Note that the compact blob should be large enough so that the membership of the point is clearly established, i.e. it should have size comparable to αn, the number of connections to points outside. Here we choose a minimum size of $6\alpha n$ mainly because it guarantees that our algorithm can still identify the blob in the worst case. The second condition means that at least β fraction of points in these compact blobs have most of their nearest neighbors from the community. This condition formalizes more global neighborhood structure about how the compact blobs interact with each other to form a community. The third condition formalizes how the community is separated from the outside.

Note that we no longer require all the members in the community have most interactions inside; we only require each member interacts with sufficiently many members and a majority of members in these local groups interact actively. Also note that the definition is hierarchical in nature: sufficiently large compact blobs clearly satisfy the definition of (α, β)-stable property and thus can be viewed as communities in lower levels. Furthermore, in the next section we will show that all the (α, β)-stable communities form a hierarchy. We show this by presenting an algorithm and proving that each (α, β)-stable community is a node in the hierarchy output by the algorithm. So our formulation explicitly models the hierarchical structure of communities observed in networks.

Next we propose a further generalization that considers possible noise in real world data. There may be some abnormal points that do not exhibit clear membership to any community, in the presence of which our definition above does not model the communities well. For example, suppose there is a point that has connections to all other points in the network, then no non-trivial subsets satisfy our definition above. We call such points bad since they do not fit into our community model above. To deal with the noise, we can naturally relax the (α, β)-stable property to the (α, β, ν)-stable property defined as follows. Informally, it requires that the target community satisfies the (α, β)-stable property after removing a few bad points B. For convenience, we call the other points in $S \setminus B$ good points.

Definition 11. *A community C is (α, β, ν)-stable if there exist a subset of bad points B of size at most νn, such that*

- *any good point $p \in G = C \setminus B$ falls into a compact blob $A_p \subseteq C$ of size greater than $6(\alpha + \nu)n$,*
- *for any point $p \in G$, at least β fraction of points in A_p have all but at most αn nearest neighbors from G out of their $|G|$ nearest neighbors in $S \setminus B$,*
- *any good point q outside $C \cup B$ has at most αn nearest neighbors from G out of their $|G|$ nearest neighbors in $S \setminus B$.*

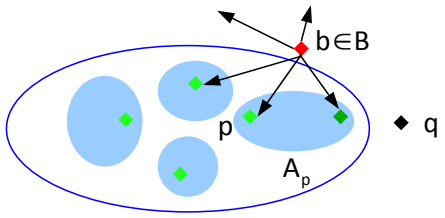


Figure 4.3: Illustration of an (α, β, ν)-stable community. An edge (x, y) means that y is one of x's nearest neighbors. Point b is a bad point and does not exhibit clear membership to any community.

We comment that the input of the community detection task is usually a graph representing the network, and there are different ways to lift the graph to a neighborhood function. The simplest one is to directly sort for each point p all the other points q according to the weights of the edges (p, q) and break ties randomly (we assume without loss of generality that the weights are in $[0, 1]$ and the weight of an edge not in E is regarded as 0). However, as pointed out in [8], we also have alternative approaches to convert the observed graph into a neighborhood function. For example, based on the belief that random walks on the graph can reflect the similarities between entities, we can define the affinity to be the diffusion kernel $\exp\{\lambda A\}$ where A is the adjacent matrix and λ is a parameter.

4.1.2 Hierarchical Community Detection Algorithm

To illustrate our technique, we first consider a compact blob whose size n_C is known. In this case it is quite easy to recover the blob as follows. We first construct a graph F whose vertices are points in V; we connect two points in F if they share at least $n_C - 2(\nu + \alpha)n$ points in common among their n_C nearest neighbors. Then by the stable property, when the blob is not too small (say $n_C > 6(\nu + \alpha)n$), no good point inside the blob can be connected with good points outside, and all good points inside the blob are connected. So, if there are no bad points, one component in F represents the blob. If there are bad points, we can construct a new graph H whose vertices are points in V; we connect in H points that share more than $(\alpha + \nu)n$ neighbors in the graph F. The key point is that in F a bad point cannot be connected to both good points inside the blob and good points outside, which then ensures that good points in the blob and good points outside are not connected in H. Consequently, a component of H represents the target blob.

In the case when we do not know the size n_C of the blob, we can start with a low threshold $t \leq n_C$ that is not too small (say $6(\nu + \alpha)n < t \leq n_C$), build the graph F_t on V by connecting points if they share at least $t - 2(\nu + \alpha)n$ points in common out of their t nearest neighbors, and build H_t on V by connecting points if they share more than $(\alpha + \nu)n$ neighbors in the graph F_t. It is safe to do so, since when $t \leq n_C$, good points inside and outside the blob share less than $t - 2(\nu + \alpha)n$ neighbors, and thus are not connected in F_t and H_t. We can increase the threshold and repeatedly build F_t and H_t. By the time we reach n_C, all good points in the blob will get connected in H_t, and so a component of H_t represents the target blob.

For detecting the general stable communities, things are more subtle. Similar ideas can be applied but with several modifications. Specifically, we maintain a list $\mathcal{C}'$ of subsets of points. At each threshold, we try to identify subsets from the same target clusters using H_t and then update the list by merging the subsets identified. For this reason, we build F_t on the points in V as before, but build H_t on the subsets in $\mathcal{C}'$ (instead of on the points in V) by connecting subsets if their points share enough neighbors in F_t. When building H_t we distinguish between singleton subsets and non-singleton subsets, since we can use median test for non-singleton subsets to outvote the noise. This algorithm makes sure that all the compact blobs are detected. Furthermore, for a general community C of size n_C, we can show the following key points: when $t \leq n_C$, in H_t no subsets containing good points in C will be connected with subsets containing good points outside C; when $t = n_C$, good points in C form a clique in F_t and all compact blobs in C have been formed, then all subsets containing good points in C are connected in H_t, and thus are merged. Then we have a subset that contains all good points in C_i and no good points outside.

The details are described in Algorithm 6, and the guarantee is that each stable community is close to one node in the tree output by the algorithm. This is formalized in Theorem 10, where we say that a community C is ν-close to another community C' if $|C \setminus C'| + |C' \setminus C| \leq \nu n$.

Algorithm 6 Hierarchical Community Detection Algorithm

Input: neighborhood function N on a set of points V, $n = |V|$, $\alpha > 0, \nu > 0$.

Step 1 Initialize $\mathcal{C}'$ to be a set of singleton points, and $t = 6(\alpha + \nu)n + 1$.
while $|\mathcal{C}'| > 1$ **do**
Step 2 Build F_t on V as follows.
for any $x, y \in V$ that satisfy $|N_t(x) \cap N_t(y)| \geq t - 2(\alpha + \nu)n$ **do**
Connect x, y in F_t.
Step 3 Build H_t on $\mathcal{C}'$ as follows. Let $N_F(x)$ denote the neighbors of x in F_t.
for any $U, W \in \mathcal{C}'$ **do**
if U, W are singleton subsets, i.e. $U = \{x\}, W = \{y\}$ **then**
Connect U, W in H_t, if $|N_F(x) \cap N_F(y)| > (\alpha + \nu)n$.
else
Set $S_t(x, y) = |N_F(x) \cap N_F(y) \cap (U \cup W)|, \forall x \in U, y \in W$.
Connect U, W in H_t, if $\text{median}_{x \in U, y \in W} S_t(x, y) > \frac{|U|+|W|}{4}$.
Step 4 **for** any component R in H_t that satisfies $|\bigcup_{C \in R} C| \geq 4(\alpha + \nu)n$ **do**
Update $\mathcal{C}'$ by merging subsets in R into one subset.
Step 5 $t = t + 1$.
end while

Output: Hierarchy T with single points as leaves and internal nodes corresponding to the merges.

Theorem 10. *Algorithm 6 outputs a hierarchy such that any community satisfying the (α, β, ν)-stable property with $\beta \geq 5/6$ is ν-close to a node in the hierarchy. The algorithm runs in time $O(n^{\omega+1})$, where $O(n^{\omega})$ is the state of the art for matrix multiplication.*

4.2 Weak Clusters

Cluster analysis for complex data sets in many applications today requires overlapping clusters, which are not available in classic clustering models finding partitions or tree hierarchies of the data. For example, a gene presented in multiple pathways due to trans-regulation mechanism is forced to a specific function in a gene hierarchy, resulting in inconsistency with existing gene function classifications [53]. To overcome these restrictions, several extensions have been proposed, allowing overlapping clusters in the output collection of clusters, such as weakly indexed pseudo-hierarchies (pyramids) [19, 32], PoClustering [53] and weakly indexed valued set systems [62]. Even if these models are interesting, they have their own drawbacks. In the pyramid model, each cluster can only overlap with two other clusters. PoClustering and weakly indexed valued set systems may contain an exponential number of clusters, which makes it difficult to find interesting ones. Here we focus on an extension called weak clusters introduced in computational biology literature by Bandelt and Dress [17]. We show that the collection of all weak clusters has nice hierarchical structure properties, and utilize these properties to design a new, faster algorithm for computing the collection.

4.2.1 Preliminaries on Weak Clusters

Let X denote the set of all objects in the data set, and $n = |X|$. Let $d(\cdot, \cdot)$ be the dissimilarity function. We assume the dissimilarity function to be symmetric, but not necessary a metric.

Definition 12. *A set $C \subseteq X$ is called a weak cluster, if for any $x_1, x_2 \in C$, $y \notin C$,*

$$d(x_1, x_2) < \max\{d(x_1, y), d(x_2, y)\}.$$

The requirement in the definition of weak clusters is rather weak. It is a natural assumption about possibly interesting clusters in the data, regardless of the domain where the data are obtained. Also, it allows for overlapping clusters. For example, suppose we have three groups of literatures: computer science, bioinformatics and biology. Computer science literatures are closer to bioinformatics than to biology, so the two form a weak cluster. Similarly, biology and bioinformatics literatures can form another weak cluster. These clusters are overlapping but both meaningful. So weak cluster is an interesting notion for cluster analysis. With the collection of weak clusters, the experts with specific domain knowledge can navigate the collection and select useful clusters. Alternatively, the collection can serve as candidates for further process, such as generating multiple clusterings.

[17] showed that there are at most $\binom{n+1}{2}$ non-empty weak clusters and proposed a $O(n^5)$ algorithm to compute all the weak clusters. Their result is based on the following notion of weak hierarchies.

Definition 13. *A non-empty collection $\mathcal{H}$ consisting of subsets of X is called a weak hierarchy, if the intersection of any three members C_1, C_2, C_3 of $\mathcal{H}$ is equal to one of the binary intersections $C_1 \cap C_2$, $C_2 \cap C_3$, $C_3 \cap C_1$.*

Lemma 5. ***([17])** If $\mathcal{H}$ is a collection of weak clusters, then $\mathcal{H}$ is a weak hierarchy.*

Proof. Assume that $\mathcal{H}$ is not a weak hierarchy. By definition, there exist $C_1, C_2, C_3 \in \mathcal{H}$ such that $C_1 \cap C_2 \cap C_3 \notin \{C_1 \cap C_2, C_1 \cap C_3, C_2 \cap C_3\}$. In other words, there exist $x_i \in (C_j \cap C_k) \setminus C_i$ for $\{i, j, k\} = \{1, 2, 3\}$. Without loss of generality we have

$$d(x_1, x_2) \geq \max\{d(x_1, x_3), d(x_2, x_3)\}.$$

This, however, is impossible because C_3 is a weak cluster, and $x_1, x_2 \in C_3, x_3 \notin C_3$. □

By Lemma 5, to investigate the properties of a collection of weak clusters, it is sufficient to do so for weak hierarchies. [17] showed the following nice structure property of weak hierarchies, which bounds the number of weak clusters by $\binom{n+1}{2}$.

Definition 14. *The closure $\langle A\rangle_{\mathcal{H}}$ (or briefly $\langle A\rangle$) of a set $A \subseteq X$ is the intersection of all members of $\mathcal{H}$ containing A, i.e. $\langle A\rangle = \cap_{A\subseteq C\in\mathcal{H}} C$. Let $\langle x, y\rangle$ be a shorthand for $\langle\{x, y\}\rangle$.*

Lemma 6. ***([17])** If $\mathcal{H}$ is a weak hierarchy, then for every non-empty subset A of X, there exist $x, y \in A$ such that $A = \langle x, y\rangle$. Call (x, y) the trace of A.*

Proof. Choose $x, y \in A$ such that $|\langle x, y\rangle|$ is maximal. Suppose $\langle x, y\rangle \neq A$, so that one can find some $z \in A - \langle x, y\rangle$. Then, because of maximality, we have $x \notin \langle y, z\rangle$ and $y \notin \langle x, z\rangle$, whence there exists $C_1, C_2, C_3 \in H$, such that $x, y \in C_1$, $y, z \in C_2$, $x, z \in C_3$, but $z \notin C_1, x \notin C_2, y \notin C_3$. Hence $C_1 \cap C_2 \cap C_3$ is properly contained in each of $C_1 \cap C_2$, $C_1 \cap C_3$, $C_2 \cap C_3$, thus giving a contradiction. □

4.2.2 Algorithm for Computing All Weak Clusters

Here we assume for any $\{p, q\} \neq \{x, y\}$, $d(p, q) \neq d(x, y)$. For a dissimilarity function d which does not satisfy the assumption, we can easily make small perturbation so that the perturbed dissimilarity d' satisfies the assumption and preserves all weak clusters. In [16], we reveal more properties of weak clusters, and use them to design our algorithm. The high level idea is that we begin with the trace of a weak cluster C, then repeatedly expand it by adding points that must belong to C until no points can be added.

Definition 15. *Let the expansion of A be $f(A) = A \cup B$ where $B = \{y \notin A | \exists x_1, x_2 \in A, d(x_1, x_2) > \max\{d(x_1, y), d(x_2, y)\}\}$. Note that for any $A \subseteq X$ there exists $t \leq n = |X|$, such that $f^{t+1}(A) = f^t(A)$. Let $F(A) = f^t(A)$ be the maximal expansion of A. For simplicity, use $F(x, y)$ as a shorthand of $F(\{x, y\})$.*

Lemma 7. *For any $A \subseteq X$, $F(A)$ is a weak cluster.*

Lemma 8. *If (x, y) is the trace of a weak cluster C, then $F(x, y) = C$.*

From Lemma 8 and the fact that every weak cluster has a trace, we know that $\{F(x, y)|x, y \in X\}$ is the collection of all weak clusters in X. A naive algorithm would be to enumerate all pairs of points and compute their maximal expansions. However, notice that if $d(x, y) > \max\{d(x, z), d(y, z)\}$, then $F(x, z) \subset F(x, y)$ and $F(y, z) \subset F(x, y)$. This means we can first compute $F(x, z)$ and $F(y, z)$, and utilize these results when computing $F(x, y)$. More specifically, we can build a graph on pairs of points, by adding a directed edge from (x, y) to (x, z) and (y, z) for all z such that $d(x, y) > \max\{d(x, z), d(y, z)\}$, then $F(x, y)$ is the union of all the maximal expansions on the children of (x, y). The details are described in Algorithm 7, and the theoretical guarantee is presented in Theorem 11.

Theorem 11. *Algorithm 7 outputs all non-empty weak clusters of X in time $O(n^4)$.*

4.3 Future Directions

For community detection, we plan to perform systematic empirical study of our model and algorithm using more neighborhood functions and on more real-world data sets. Another direction would be to speed up the neighborhood function computation and the algorithm, or design algorithms for large-scale scenarios (e.g. local algorithms whose running time depends on the size of the detected community, rather than the size of the entire network).

Algorithm 7 Computing All Weak Clusters

Input: dissimilarity $d(\cdot,\cdot)$ on X.

▷ Construct a directed graph $G = (V, E)$ as follows

Step 1 Let $V = \{(x, y)|x, y \in X\}$.

Step 2 For any two points $x, y \in X$,
add two directed edges $((x, y), (x, x))$ and $((x, y), (y, y))$ into E.

Step 3 For any three points $x, y, z \in X$,
If $d(x, y) > \max\{d(x, z), d(y, z)\}$,
add two directed edges $((x, y), (x, z))$ and $((x, y), (y, z))$ into E.

▷ Construct the collection of weak clusters

Step 4 Initialize $\mathcal{H} = \emptyset$.

Step 5 In topological order from the sink to the source, traverse the node $r = (x, y)$ of G:
If $x = y$, then let $C_r = \{x\}$; otherwise, $C_r = \cup_{s\in V,(r,s)\in E} C_s$.
$\mathcal{H} = \mathcal{H} \cup \{C_r\}$.

Output: $\mathcal{H}$.

For weak clusters, we improve the running time of the algorithm for computing all weak clusters from $O(n^5)$ to $O(n^4)$. It is an interesting research direction to improve it further. Another interesting direction would be to generalize the notion of weak clusters. Noticing that real world clusters may satisfy the weak cluster definition only approximately, we can define a more general notion called noisy weak clusters to formalize this observation. More specifically, a cluster C is called a (α, ν)-weak cluster if after removing at most νn noisy points, for each point p in C, at least $(1 - \alpha)$ fraction of points p' in C satisfy the weak cluster requirement. Experiments on real data show that real world clusters satisfy the definition for small α and ν, so it would be meaningful to develop algorithms for finding noisy weak clusters.

There are also other notions for cluster hierarchies rather than tree and weak hierarchies. For instances, the neighbor-net algorithm [22] builds a circular split hierarchy given that the input distance function is circular. It generalizes the neighbor-joining algorithm [65], which given a tree metric recovers the underlying tree hierarchy. Possible research directions are:

- Are the circular set hierarchy and the neighbor-net algorithm (and its variants) suitable for community detection? What are the criteria for measuring whether a hierarchy model (or a hierarchical construction algorithm) is suitable for a particular task? In our work on community detection, we measured the average recover accuracy of the communities, but did not consider the possible redundance in the hierarchy output, i.e. the size of the hierarchy. Further quality criteria are important for designing and evaluating hierarchical construction algorithms.

- If the target cluster hierarchy belongs to a larger model than the one we consider, we would like to design an algorithm whose output is guaranteed to be (close to) the optimum in the hierarchy model used. For example, if the distance function is defined by a circular set hierarchy but our algorithm only recovers a tree, then we would like to guarantee that the output tree is (close to) the "optimal" tree. What would be a formal metric that measures the closeness of two hierarchy models? Such questions also arise in other areas such as network topology recovery and latent variable model construction.

Bibliography

[1] E. M. Airoldi, D. M. Blei, S. E. Fienberg, and E. P. Xing. Mixed membership stochastic blockmodels. *The Journal of Machine Learning Research*, 2008.

[2] S. Arora, R. Ge, S. Sachdeva, and G. Schoenebeck. Finding overlapping communities in social networks: toward a rigorous approach. In *Proceedings of the ACM Conference on Electronic Commerce*, 2012.

[3] V. Arya, N. Garg, R. Khandekar, A. Meyerson, K. Munagala, and V. Pandit. Local search heuristics for k-median and facility location problems. *SIAM Journal of Computing*, 2004.

[4] P. Awasthi, A. Blum, and O. Sheffet. Stability yields a ptas for k-median and k-means clustering. In *Proceedings of the IEEE Annual Symposium on Foundations of Computer Science*, 2010.

[5] P. Awasthi, A. Blum, and O. Sheffet. Center-based clustering under perturbation stability. *Information Processing Letters*, 2012.

[6] P. Awasthi and O. Sheffet. Improved spectral-norm bounds for clustering. In *Approximation, Randomization, and Combinatorial Optimization: Algorithms and Techniques*. 2012.

[7] M. Bădoiu and K. L. Clarkson. Optimal core-sets for balls. *Computational Geometry*, 2008.

[8] M. Balcan, C. Borgs, M. Braverman, J. Chayes, and S. Teng. Finding endogenously formed communities. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms*, 2013.

[9] M. Balcan and P. Gupta. Robust hierarchical clustering. In *Proceedings of the Conference on Learning Theory*, 2010.

[10] M. F. Balcan, A. Blum, and A. Gupta. Approximate clustering without the approximation. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms*, 2009.

[11] M. F. Balcan, A. Blum, and S. Vempala. A discriminative framework for clustering via similarity functions. In *Proceedings of the ACM Symposium on the Theory of Computing*, 2008.

[12] M. F. Balcan and M. Braverman. Approximate nash equilibria under stability conditions. *CoRR*, abs/1008.1827, 2010.

[13] M. F. Balcan and P. Gupta. Robust hierarchical clustering. In *Proceedings of the Annual Conference on Learning Theory*, 2010.

[14] M. F. Balcan and Y. Liang. Clustering under perturbation resilience. In *Proceedings of the International Colloquium on Automata, Languages, and Programming*, 2012.

[15] M. F. Balcan and Y. Liang. Modeling and detecting community hierarchies. In *Proceedings of the International Workshop on Similarity-Based Pattern Analysis and Recognition*, 2013.

[16] M.-F. Balcan and Y. Liang. Weak clusters: Towards flexible and concise clustering, 2013.

[17] H. Bandelt and A. Dress. Weak hierarchies associated with similarity measures – an additive clustering technique. *Bulletin of Mathematical Biology*, 1989.

[18] Y. Bartal, M. Charikar, and D. Raz. Approximating min-sum k-clustering in metric spaces. In *Proceedings of the Annual ACM Symposium on Theory of Computing*, 2001.

[19] P. Bertrand and M. F. Janowitz. Pyramids and weak hierarchies in the ordinal model for clustering. *Discrete Applied Mathematics*, 2002.

[20] Y. Bilu, A. Daniely, N. Linial, and M. Saks. On the practically interesting instances of maxcut. In *Proceedings of the International Symposium on Theoretical Aspects of Computer Science*, 2013.

[21] Y. Bilu and N. Linial. Are stable instances easy? In *Proceedings of the Symposium on Innovations in Computer Science*, 2010.

[22] D. Bryant and V. Moulton. Neighbor-net: an agglomerative method for the construction of phylogenetic networks. *Molecular biology and evolution*, 2004.

[23] M. Charikar, S. Guha, É. Tardos, and D. B. Shmoys. A constant-factor approximation algorithm for the k-median problem. *Journal of Computer and System Sciences*, 2002.

[24] K. Chen. On k-median clustering in high dimensions. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms*, 2006.

[25] K. L. Clarkson. Coresets, sparse greedy approximation, and the frank-wolfe algorithm. *ACM Transactions on Algorithms*, 2010.

[26] A. Clauset, C. Moore, and M. Newman. Hierarchical structure and the prediction of missing links in networks. *Nature*, 2008.

[27] A. Clauset, M. E. J. Newman, and C. Moore. Finding community structure in very large networks. *Physical Review E*, 2004.

[28] J. Considine, F. Li, G. Kollios, and J. Byers. Approximate aggregation techniques for sensor databases. In *Proceedings of the International Conference on Data Engineering*, 2004.

[29] J. C. Corbett, J. Dean, M. Epstein, A. Fikes, C. Frost, J. Furman, S. Ghemawat, A. Gubarev, C. Heiser, P. Hochschild, et al. Spanner: Googles globally-distributed database. In *Proceedings of the USENIX Symposium on Operating Systems Design and Implementation*, 2012.

[30] S. Datta, C. Giannella, H. Kargupta, et al. K-means clustering over peer-to-peer networks. In *Proceedings of the International Workshop on High Performance and Distributed Mining*, 2005.

[31] W. F. de la Vega, M. Karpinski, C. Kenyon, and Y. Rabani. Approximation schemes for clustering problems. In *Proceedings of the Annual ACM Symposium on Theory of Computing*, 2003.

[32] C. Durand and B. Fichet. One-to-one correspondences in pyramidal representation: A unified approach. *Classification and Related Methods of Data Analysis*, 1988.

[33] D. Feldman and M. Langberg. A unified framework for approximating and clustering data. In *Proceedings of the Annual ACM Symposium on Theory of Computing*, 2011.

[34] D. Feldman, A. Sugaya, and D. Rus. An effective coreset compression algorithm for large scale sensor networks. In *Proceedings of the International Conference on Information Processing in Sensor Networks*, 2012.

[35] G. Forman and B. Zhang. Distributed data clustering can be efficient and exact. *ACM SIGKDD Explorations Newsletter*, 2000.

[36] S. Fortunato. Community detection in graphs. *Physics Reports*, 2010.

[37] M. Girvan and M. E. J. Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, 2002.

[38] M. Girvan and M. E. J. Newman. Community structure in social and biological networks. 2002.

[39] S. Greenhill and S. Venkatesh. Distributed query processing for mobile surveillance. In *Proceedings of the International Conference on Multimedia*, 2007.

[40] M. Greenwald and S. Khanna. Power-conserving computation of order-statistics over sensor networks. In *Proceedings of the ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, 2004.

[41] S. Guha and S. Khuller. Greedy strikes back: Improved facility location algorithms. *Journal of Algorithms*, 1999.

[42] S. Har-Peled and S. Mazumdar. On coresets for k-means and k-median clustering. In *Proceedings of the Annual ACM Symposium on Theory of Computing*, 2004.

[43] J. He, J. Hopcroft, H. Liang, S. Suwajanakorn, and L. Wang. Detecting the structure of social networks using (α, β)-communities. In *Proceedings of the International Conference on Algorithms and Models for the Web Graph*, 2011.

[44] K. Jain, M. Mahdian, and A. Saberi. A new greedy approach for facility location problems. In *Proceedings of the Annual ACM Symposium on Theory of Computing*, 2002.

[45] E. Januzaj, H. Kriegel, and M. Pfeifle. Towards effective and efficient distributed clustering. In *Workshop on Clustering Large Data Sets in the IEEE International Conference on Data Mining*, 2003.

[46] H. Kargupta, W. Huang, K. Sivakumar, and E. Johnson. Distributed clustering using collective principal component analysis. *Knowledge and Information Systems*, 2001.

[47] M. Kim and J. Leskovec. Latent multi-group membership graph model. In *Proceedings of the International Conference on Machine Learning*, 2012.

[48] A. Kumar and R. Kannan. Clustering with spectral norm and the k-means algorithm. In *Proceedings of the IEEE Annual Symposium on Foundations of Computer Science*, 2010.

[49] M. C. Lagomarsino, P. Jona, B. Bassetti, and H. Isambert. Hierarchy and feedback in the evolution of the Escherichia coli transcription network. *Proceedings of the National Academy of Sciences*, 2007.

[50] A. Lancichinetti, S. Fortunato, and J. Kertész. Detecting the overlapping and hierarchical community structure in complex networks. *New Journal of Physics*, 2009.

[51] S. Li and O. Svensson. Approximating k-median via pseudo-approximation. In *Proceedings of the ACM Symposium on the Theory of Computing*, 2013.

[52] R. J. Lipton, E. Markakis, and A. Mehta. On stability properties of economic solution concepts. Manuscript, 2006.

[53] J. Liu, Q. Zhang, W. Wang, L. McMillan, and J. Prins. Poclustering: Lossless clustering of dissimilarity data. In *Proceedings of the SIAM International Conference on Data Mining*, 2007.

[54] S. Lodi, R. Nanculef, and C. Sartori. Svm learning with distributed data.

[55] K. Makarychev, Y. Makarychev, and A. Vijayaraghavan. Bilu-linial stable instances of max cut. *CoRR*, abs/1305.1681, 2013.

[56] M. Mihalák, M. Schöngens, R. Šrámek, and P. Widmayer. On the complexity of the metric tsp under stability considerations. In *Proceedings of the International Conference on Current Trends in Theory and Practice of Computer Science*, 2011.

[57] S. Mitra, M. Agrawal, A. Yadav, N. Carlsson, D. Eager, and A. Mahanti. Characterizing web-based video sharing workloads. *ACM Transactions on the Web*, 2011.

[58] M. E. J. Newman. Detecting community structure in networks. *The European Physical Journal B - Condensed Matter and Complex Systems*, 2004.

[59] M. E. J. Newman. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences*, 2006.

[60] C. Olston, J. Jiang, and J. Widom. Adaptive filters for continuous queries over distributed data streams. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 2003.

[61] R. Ostrovsky, Y. Rabani, L. J. Schulman, and C. Swamy. The effectiveness of lloyd-type methods for the k-means problem. In *Proceedings of the Annual IEEE Symposium on Foundations of Computer Science*, 2006.

[62] Patrice and Bertrand. Set systems and dissimilarities. *European Journal of Combinatorics*, 2000.

[63] E. Ravasz, A. L. Somera, D. A. Mongru, Z. N. Oltvai, and A. L. Barabási. Hierarchical Organization of Modularity in Metabolic Networks. *Science*, 2002.

[64] L. Reyzin. Data stability in clustering: A closer look. In *Proceedings of the International Conference on Algorithmic Learning Theory*, 2012.

[65] N. Saitou and M. Nei. The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Molecular biology and evolution*, 1987.

[66] M. Schweinberger and T. A. B. Snijders. Settings in social networks: A measurement model. *Sociological Methodology*, 2003.

[67] D. Tasoulis and M. Vrahatis. Unsupervised distributed clustering. In *Proceedings of the International Conference on Parallel and Distributed Computing and Networks*, 2004.

[68] I. W. Tsang, A. Kocsor, and J. T. Kwok. Simpler core vector machines with enclosing balls. In *Proceedings of the International Conference on Machine Learning*, 2007.

[69] I. W. Tsang, J. T. Kwok, and P.-M. Cheung. Core vector machines: Fast svm training on very large data sets. In *Journal of Machine Learning Research*, 2005.

[70] S. Vadhan. Cs229r: Mathematical approaches to data privacy additional topics and project ideas, 2013.

[71] E. A. Yildirim. Two algorithms for the minimum enclosing ball problem. *SIAM Journal on Optimization*, 2008.

[72] Q. Zhang, J. Liu, and W. Wang. Approximate clustering on distributed data streams. In *Proceedings of the IEEE International Conference on Data Engineering*, 2008.