Clustering Perturbation Resilient Data

Maria Florina Balcan
School of Computer Science
Georgia Institute of Technology
Atlanta, GA 30332
ninamf@cc.gatech.edu

Yingyu Liang
School of Computer Science
Georgia Institute of Technology
Atlanta, GA 30332
yliang39@gatech.edu

Abstract

Recently, Bilu and Linial [6] formalized an implicit assumption often made when choosing a clustering objective: that the optimum clustering to the objective should be preserved under small multiplicative perturbations to distances between points. Balcan and Liang [4] generalized this to a relaxed notion where the optimal clustering after perturbation is allowed to change slightly. In this paper, we propose an efficient algorithm for k-median instances under the generalized notion, achieving theoretical guarantees that significantly improve over previous known results. Additionally, we give a sublinear-time algorithm which can return an implicit clustering from only access to a small random sample.

1 Introduction

Problems of clustering data from pairwise distance information are a classical topic in machine learning. A common approach is to optimize various objective functions such as k-median, k-means or min-sum. However, for most natural clustering objectives, finding the optimal solution is NP-hard. There has been substantial work on approximation algorithms [10, 7, 5, 8, 1] with both upper and lower bounds on the approximability of these objectives on worst case instances.

Recently, Bilu and Linial [6] suggested an exciting, alternative approach aimed at understanding the complexity of practical clustering instances. Motivated by the fact that distances are often based on a heuristic measure, they argued that interesting instances should be resilient to small perturbations in these distances. Specifically, they defined an instance to be α-perturbation resilient if perturbing pairwise distances by multiplicative factors in the range $[1, \alpha]$ does not change the optimum clustering. Balcan and Liang [4] generalized this to a weaker, relaxed, and more realistic notion of (α, ϵ)-perturbation resilience where the optimal clustering of the perturbed instance is allowed 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.

In this paper, we propose an efficient algorithm for (α, ϵ)-perturbation resilient k-median instances, which for $\alpha > 4$ produces $(1 + O(\epsilon/\rho))$-approximation to the optimum, where ρ is the fraction of the points in the smallest cluster. This significantly improves over the bound $\alpha > 2 + \sqrt{7}$ in [4]. The algorithm is based on the key structural property that, except for ϵn bad points, most points are α times closer to their own center than to any other center. To eliminate the noise introduced by the bad points, we carefully partition the points into a list of sufficiently large blobs, each of which contains only good points from one optimal cluster. This then allows us to construct a tree on the blobs with a low-cost pruning that is a good approximation to the optimum. Additionally, the robustness to the bad points allows us to make the algorithm sublinear-time by returning an implicit clustering from only a small random sample of the input. The construction of the implicit clustering takes time poly-logarithmic in the size of the data, which makes the sublinear-time version preferable for large scale data sets.

2 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. In k-median clustering, 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 the objective $\Phi(\mathcal{P}, \mathbf{p}) = \sum_{i=1}^{k} \sum_{p \in P_i} d(p, p_i)$. When the partition $\mathcal{P}$ is obtained by assigning each point to its nearest center in $\mathbf{p}$, the objective is shorten as $\Phi(\mathbf{c})$. The optimal centers are denoted as $\mathbf{c} = \{c_1, \ldots, c_k\}$, the optimal clustering is denoted as $\mathcal{C} = \{C_1, C_2, \ldots, C_k\}$, and its cost is denoted as $\mathcal{OPT}$.

The core concept we study in this paper is the following (α, ϵ)-perturbation resilience notion.

Definition 1. *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 2. *A clustering instance (S, d) is (α, ϵ)-**perturbation resilient** to a given objective Φ if for any function $d' : S \times S \to \mathbb{R}_{\geq 0}$ such that $\forall p, q \in S, d(p, q) \leq d'(p, q) \leq \alpha d(p, q)$, the optimal clustering $\mathcal{C}'$ for Φ under d' is ϵ-close to the optimal clustering $\mathcal{C}$ for Φ under d.*

3 Clustering (α, ϵ)-Perturbation Resilient k-Median Instances

In this section we show that if the instance is (α, ϵ)-perturbation resilient, with $\alpha > 4$ 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. Formally,

Theorem 1. *If the clustering instance is (α, ϵ)-perturbation resilient for $\alpha > 4$ and $\epsilon \leq \rho/30$ where $\rho = \frac{\min_i |C_i|}{n}$, then Algorithm 1 produces a clustering which is $(1 + \frac{5\epsilon}{\rho})$-approximation to the optimal clustering with respect to the k-median objective in time $O(n^{\omega+1})$, where $O(n^{\omega})$ is the state of the art for matrix multiplication.*

This significantly improves over the bound $\alpha > 2 + \sqrt{7}$ in [4]. It also improves over the best worst-case approximation guarantees known [11] 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 [9, 10] when $\epsilon' \leq 1/e$.

In the following, we first review the structural properties utilized, and then describe our algorithm and provide a sketch of the analysis.

3.1 Structural Property

The key structural property we exploit is that, except for ϵn bad points, most points are α times closer to their own center than to any other center. Specifically, 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. That is, $B_i = \{p \in C_i : \exists j \neq i, \alpha d(c_i, p) > d(c_j, p)\}$. Let $G_i = C_i \setminus B_i$ be the good points in cluster C_i and let $B = \bigcup_i B_i$.

Theorem 2. *(Theorem 1 in [4]) Suppose the clustering instance is (α, ϵ)-perturbation resilient and $\min_i |C_i| > (3 + \frac{2\alpha}{\alpha - 1})\epsilon n + 9\alpha$. Then $|B| \leq \epsilon n$.*

We can see that by definition, the good points are far apart from each other. This then implies that for any good point, most of its nearest neighbors are from its own optimal cluster. Formally,

Lemma 1. *When $\alpha > 4$, for any good points $p_1, p_2 \in G_i, q \in G_j (j \neq i)$, we have $d(p_1, p_2) < d(p_1, q)$. Consequently, for any good point $p \in G_i$, all its $|G_i|$ nearest neighbors belong to $C_i \cup B$.*

3.2 Approximation Algorithm

In the following, we utilize the bound on the bad points and the property of the good points to design an efficient approximation algorithm. In order to get rid of the influence of the bad points, we

Algorithm 1 k-median, (α, ϵ) perturbation resilience

1: **Input:** Distance function $d(\cdot, \cdot)$ on S, the size of the smallest optimal cluster $\min_i |C_i|$, $\epsilon > 0$
2: Run Algorithm 2 to generate a list $\mathcal{L}$ of blobs.
3: Run the robust linkage procedure in [3] to get a cluster tree T.
4: Run dynamic programming on T to get the lowest cost pruning $\tilde{\mathcal{C}}$ and its centers $\tilde{\mathbf{c}}$.
5: **Output:** Clustering $\tilde{\mathcal{C}}$ and its centers $\tilde{\mathbf{c}}$.

Algorithm 2 Generating interesting blobs

1: **Input:** Distance function $d(\cdot, \cdot)$ on S, $\min_i |C_i|$, $\epsilon > 0$.
2: Let $N_r(p)$ denote the r nearest neighbors of p in S.
3: Let $\mathcal{L} = \emptyset$, $A_S = S$. Let the initial threshold $t = \min_i |C_i|$.
4: Construct a graph F_t by connecting $p, q \in A_S$ if $|N_t(p) \cap N_t(q)| > t - 2\epsilon n$.
5: Construct a graph H_t by connecting points $p, q \in A_S$ that have more than ϵn neighbors in F_t.
6: Add to $\mathcal{L}$ all the components C of H_t with $|C| \geq \frac{1}{2} \min_i |C_i|$ and remove them from A_S.
7: For each point $p \in A_S$, check if most of $N_t(p)$ are in $\mathcal{L}$ and if there exists $C \in \mathcal{L}$ containing a significant number of points in $N_t(p)$. More precisely, check if
(1) $|N_t(p) \setminus \mathcal{L}| \leq \frac{1}{2} \min_i |C_i| + 2\epsilon n$;
(2) $\mathcal{L}_p \neq \emptyset$ where $\mathcal{L}_p = \{C \in \mathcal{L} : |C \cap N_t(p)| \geq \frac{2}{5}|C|\}$.
If so, assign p to the blob in $\mathcal{L}_p$ of smallest median distance, and remove p from A_S.
8: While $|A_S| > 0$, increase t by 1 and go to Step 4.
9: **Output:** The list $\mathcal{L}$.

generate a list of blobs, which form a partition of the data points, and each of which contains only good points from one optimal cluster. Then we construct a tree on the list of blobs with a pruning that assigns all good points correctly. We will show that this pruning has low cost, so the lowest cost pruning of the tree is a good approximation. The details are described in Algorithm 1. We now provide a sketch of the analysis of its two key steps and the final approximation guarantee.

Generating Blobs The first key step is to generate the list of almost "pure" blobs, which is described in Algorithm 2. Informally, the algorithm maintains a threshold t. At each threshold, for each point p that has not been added to the list, the algorithm checks its t nearest neighbors $N_t(p)$. It constructs a graph F_t by connecting any two points that have most neighbors in common. It then builds another graph H_t by connecting any two points that have sufficiently many neighbors in F_t, and adds sufficiently large components in H_t to the list. Finally, for each remaining point p, it checks if most of p's neighbors are in the list and if there are blobs containing a significant amount of p's neighbors. If so, it inserts p into such a blob with the smallest median distance. Then the threshold is increased and the above steps are repeated.

The intuition behind Algorithm 2 is as follows. The algorithm works when for any i and any good point $p \in G_i$, the $|G_i|$ nearest neighbors of p contain no good points outside C_i. To see this, assume without loss of generality that $|C_1| \leq |C_2| \leq \cdots \leq |C_k|$. When $t \leq |C_1|$, good points in different clusters do not have most neighbors in common and thus are not connected in F_t. However, they may be connected by a path of bad points. So we further build the graph H_t to disconnect such paths, which ensures that the blobs added into the list contain only good points from one optimal cluster. The final insert step (Step 7) makes sure that when $t = |C_1|$, all remaining good points in C_1 will be added to the list and will not affect the construction of blobs from other optimal clusters. We can show by induction that, at the end of the iteration $t = |C_i|$, all good points in $C_j (j \leq i)$ are added to the list. When t is large enough, any remaining bad points are inserted into the list, so the points are partitioned into a list of almost pure blobs.

Linking Blobs Another key step is to construct a tree on these blobs. Since good points are closer to good points in the same optimal cluster than to those in other clusters (Lemma 1), there exist algorithms that can build a tree with a pruning that assigns all good points correctly. In particular, we can use the robust linkage procedure in [3], which repeatedly merges the two blobs C, C' with the maximum $\text{score}(C, C')$ defined as follows. For each $p \in C$, sort the other blobs in decreasing order of the median distance between p and points in the blob, and let $\text{rank}(p, C')$ denote the rank of C'. Then define $\text{rank}(C, C') = \text{median}_{x \in C}[\text{rank}(x, C')]$ and

$\text{score}(C, C') = \min[\text{rank}(C, C'), \text{rank}(C', C)]$. Intuitively, for any blobs A, A' from the same optimal cluster and D from a different cluster, good points in A always rank A' later than D in the sorted list, so $\text{rank}(A, A') > \text{rank}(A, D)$. Similarly, $\text{rank}(A', A) > \text{rank}(A', D)$, and thus $\text{score}(A', A) > \text{score}(A, D)$. Then the algorithm always merges blobs from the same cluster before merging them with blobs outside, and thus there is a pruning that assigns all good points correctly.

Approximation Guarantee As described above, Algorithm 2 partitions the points into a list of blobs, each of which has size at least $\frac{1}{2}\min_i |C_i|$ and contains only good points from one optimal cluster. Let B_i' denote the bad points that are assigned to blobs containing good points in C_i. Then the robust linkage procedure on $\mathcal{L}$ guarantees that $\{G_i \cup B_i'\}_{i=1}^k$ is a pruning of the tree output (see Theorem 9 in [3]). It suffices to show that this pruning, using the optimal centers $\{c_i\}$, is a $(1 + \frac{5\epsilon}{\rho})$-approximation to $\mathcal{OPT}$. Since all good points are correctly assigned, we only need to bound the cost increased by assigning a bad point $q \in C_i$ to a blob containing good points from a different optimal cluster C_j. Intuitively, the bad point must have many nearest neighbors in that blob, then it is closer to a significant number of good points in that optimal cluster than to a significant number of good points in its own optimal cluster. Formally, we have

Lemma 2. *If a bad point $q \in B_i$ is assigned to a blob C containing good points from a different optimal clustering C_j, then there exist $m = \frac{1}{5}\min_i |C_i|$ points Z_i from C_i, and m points Z_j from C_j, such that $d(q, Z_i) \geq d(q, Z_j)$. Consequently, $d(q, c_j) - d(q, c_i) \leq \frac{\mathcal{OPT}}{m}$.*

As there are at most ϵn bad points and $m = \frac{\min_i |C_i|}{5}$, the increase of cost is at most $\frac{5\epsilon}{\rho}\mathcal{OPT}$.

4 Sublinear Time Algorithm for (α, ϵ)-Perturbation Resilient Instances

Many clustering applications have recently faced an explosion of data, and it is often expensive to run an algorithm over the entire data. Here we show that for perturbation resilient k-median instances, we can overcome this difficulty by running our algorithm on a small random sample.

More precisely, consider a clustering instance (X, d) that is (α, ϵ)-perturbation resilient to k-median. For simplicity, suppose the distances are normalized to $[0, 1]$. Let $N = |X|$ and let $\rho = \min_i |C_i|/N$ denote the fraction of the points in the smallest cluster. Let Φ_X denote the cost on X, and let $\zeta = \Phi_X(\mathbf{c})/N$ denote the average cost of the points in the optimum clustering.

Theorem 3. *Suppose (X, d) is (α, ϵ)-perturbation resilient for $\alpha > 4$, $\epsilon < \rho/100$. Then with probability $\geq 1 - \delta$, we can get an implicit clustering that is $2(1 + \frac{16\epsilon}{\rho})$-approximation in time $poly(\log \frac{N}{\delta}, k, \frac{1}{\epsilon}, \frac{1}{\zeta})$.*

The main idea is to run Algorithm 1 on a random sample S of size $n = \Theta(\frac{k}{\epsilon^2\zeta^2}\ln\frac{N}{\delta})$ to obtain the minimum cost pruning and the corresponding centers $\tilde{\mathbf{c}}$. Then the implicit clustering of the whole space X assigns each point in X to its nearest center in $\tilde{\mathbf{c}}$.

In the following, we describe the idea to show that $\tilde{\mathbf{c}}$ is a good approximation solution to the optimal centers $\mathbf{c}$ for X. First, when n is sufficiently large, with high probability, $\Phi_X(\tilde{\mathbf{c}})/N \approx \Phi_S(\tilde{\mathbf{c}})/n$ and $\Phi_X(\mathbf{c})/N \approx \Phi_S(\mathbf{c})/n$. Then it is sufficient to show $\Phi_S(\tilde{\mathbf{c}})$ is close to $\Phi_S(\mathbf{c})$. Next, we can show that Algorithm 1 builds a tree with a pruning $\mathcal{P}'$ that assigns all good points correctly. The key is to use the cost of this pruning as a bridge for comparing $\Phi_S(\tilde{\mathbf{c}})$ and $\Phi_S(\mathbf{c})$.

On one hand, $\Phi_S(\tilde{\mathbf{c}}) \leq \Phi_S(\tilde{\mathcal{C}}, \tilde{\mathbf{c}}) \leq \Phi_S(\mathcal{P}', \mathbf{c}')$. The first inequality comes from the fact that in $\Phi_S(\tilde{\mathbf{c}})$ each point is assigned to its nearest center and the second comes from the fact that $\tilde{\mathcal{C}}$ is the minimum cost pruning. On the other hand, $\Phi_S(\mathcal{P}', \mathbf{c}') \leq 2\Phi_S(\mathcal{P}', \mathbf{c}) \leq 2(1 + \frac{12\epsilon}{\rho})\Phi_S(\mathbf{c})$. The second inequality comes from an argument similar to that in Theorem 1 and the fact that $\Phi_S(\mathcal{P}', \mathbf{c})$ is different from $\Phi_S(\mathbf{c})$ only on the bad points. The first inequality comes from the triangle inequality. More precisely, for any cluster $N_i' \in \mathcal{P}'$,

$$2|N_i'| \sum_{p \in N_i'} d(p, c_i) = \sum_{p,q \in N_i'} [d(p, c_i) + d(q, c_i)] \geq \sum_{p,q \in N_i'} d(p, q) \geq \sum_{p,q \in N_i'} d(q, c_i') = |N_i'| \sum_{q \in N_i'} d(q, c_i').$$

Note: If we have an oracle that given a set of points C_i' finds the best center *in* X for that set, then we can save a factor of 2 in the bound.

References

[1] 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.

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

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

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

[5] 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.

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

[7] 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.

[8] 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.

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

[10] 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.

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