CS264: Beyond Worst-Case Analysis
Lecture #6: Clustering in Approximation-Stable Instances*

Tim Roughgarden†

October 10, 2014

1 Preamble

In some optimization problems, the objective function can be taken quite literally. If one wants to maximize profit or accomplish some goal at minimum cost, then the goal translates directly into a numerical objective function.

In other applications, an objective function is only a means to an end. Consider, for example, the problem of *clustering*. Given a set of data points, the goal is to cluster them into "coherent groups," with points in the same group being "similar" and those in different groups being "dissimilar." There is not an obvious, unique way to translate this goal into a numerical objective function, and as a result many different objective functions have been studied (k-means, k-median, k-center, etc.) with the intent of making the fuzzy notion of a "good/meaningful clustering" into a concrete optimization problem. In this case, we do not care about the objective function value per se; rather, we want to discover interesting structure in the data. So we're perfectly happy to compute a "meaningful clustering" with suboptimal objective function value, and would be highly dissatisfied with an "optimal solution" that fails to indicate any patterns in the data (which suggests that we were asking the wrong question, or expecting structure where none exists).

The point is that *if we are trying to cluster a data set, then we are implicitly assuming that interesting structure exists in the data.*[1] This perspective suggests that an explicit model of data could sharpen the insights provided by a traditional worst-case analysis framework (cf., modeling locality of reference in online paging). This lecture begins our exploration of the conjecture that *clustering is hard only when it doesn't matter*. That is, clustering

†Department of Computer Science, Stanford University, 474 Gates Building, 353 Serra Mall, Stanford, CA 94305. Email: tim@cs.stanford.edu.

[1]There are exceptions of course. For example, if one is partitioning a graph as part of a divide and conquer algorithm, then the partition is only means to an end and is not interesting in its own right.

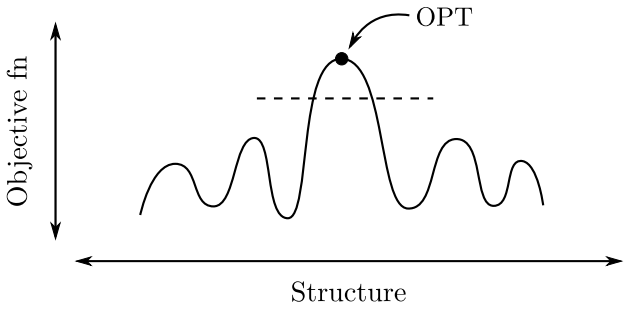

Figure 1: Numerical near-optimality implies structural near-optimality.

instances with "meaningful solutions" are computationally easier to solve than worst-case instances. There has been an explosion of work on this idea over the past five years, and we'll only have time to give an incomplete survey in these lectures.

2 Approximation Stability and the k-Median Problem

2.1 Definitions

The high-level idea of this lecture's model is to impose two assumptions on clustering instances that are meant to model the existence of a "meaningful solution." First, we assume that the data admits a "ground truth" target clustering — for example, the classification of images into those of cats and those of dogs, or the classification of proteins by function. The goal is to recover, perhaps approximately, this ground truth clustering using an algorithm. Of course, for there to be any hope in achieving this goal, the input must somehow reflect what the target clustering is.

The second assumption is that, with respect to some numerical objective function, every near-optimal clustering is nearly the same as the target clustering. That is, numerical near-optimality implies structural near-optimality. See Figure 1 for a cartoon depiction of this idea. This assumption is only well defined with respect to a choice of objective function value. We'll choose a standard one for this lecture; analogous results hold also for other popular objective functions (like k-means).

We study the *k-median* problem. The input is an n-point metric space (X, d), meaning that d is a nonnegative function on $X \times X$ satisfying $d(x, x) = 0$ for all $x \in X$, $d(x, y) = d(y, x)$ for all $x, y \in X$ (symmetry), and $d(x, y) \leq d(x, z) + d(z, y)$ for all $x, y, z \in X$ (triangle inequality). One generally interprets d as a distance function or (dis)similarity measure. A *k-clustering* of a finite metric space is a partition of its points into k non-empty sets.[2] The goal in the k-median problem is to identify a subset $S \subseteq X$ of k *centers* to minimize the sum

[2]We'll think of k as known a priori. In some applications one has domain knowledge about what k should be; in others one runs a k-clustering algorithm for varying values of k and goes with the solution that is "best" in some sense.

of the points' distances to their respective nearest centers:

$$\sum_{x \in X} \left(\min_{c \in S} d(c, x) \right).$$

Note that a choice of centers induces a k-clustering: the cluster of a center c is the set of points $x \in X$ for which c is the closest center.

The key definition is this.

Definition 2.1 (Approximation Stability [1]) The target clustering of a k-median instance is (c, ϵ)-*approximation stable* if every c-approximate k-clustering is ϵ-*accurate*, meaning that it agrees with the target clustering on at least a $1 - \epsilon$ fraction of the points.

When we speak of two (unordered) k-clusterings "agreeing" on at least a $1 - \epsilon$ fraction of the points, we mean that there is a way to label each of the two collections of k clusters with the names $1, 2 \ldots, k$ so that at least $(1 - \epsilon)n$ points of X get the same label in both k-clusterings. That is, we take an optimal correspondence of the clusters in the two k-clusterings.

The condition in Definition 2.1 grows more restrictive as one increases c or decreases ϵ. Throughout this lecture, you might want to think of c as a small constant (like 1.1, translating to 10% numerical error), the number of points n tending to infinity, and ϵ tending to 0 with n, say as $1/\sqrt{n}$. For example, with $n = 10^6$ points, this would give $\epsilon n = \sqrt{n} = 1000$, and Definition 2.1 would insist that all k-clusterings with at most 10% numerical error incorrectly classify at most 1000 out of the million points (.1% classification error).

2.2 Discussion of Approximation Stability

Definition 2.1 requires that *the only way to achieve a near-optimal k-median solution is to agree with the optimal solution on almost all points* — numerical similarity (in objective function value) implies structural similarity (in the classification of points). Is this a reasonable definition? Taken literally, the requirement is pretty strong, and it's not clear that it holds (with reasonable parameter values) in "real-world" instances. Since the definition references both an unknown target clustering and all near-optimal k-median solutions, it's also not clear how to check whether or not the condition holds in inputs of interest.

These criticisms are similar to those we lobbed at the definition of the recoverable value last lecture, and it's worth remembering the counterarguments. First, there is a plausible narrative about why "real" clustering instances should tend toward satisfying the definition of approximation stability, at least approximately. For images of cats and dogs, for example, it seems unlikely that a highly incorrect classification of the images would yield a numerically near-optimal 2-clustering (for any reasonable encoding of the problem). Second, the set of instances where one can formally prove guaranteed good performance of an algorithm is typically a small subset of the instances where the algorithm performs well empirically. Third, a condition like Definition 2.1 should ultimately be judged by the extent to which it leads to a better understanding of an existing algorithm or (in this case) the development of new algorithmic ideas that might prove useful for solving the problem.

2.3 Connection to Approximation Algorithms

By definition, running a c-approximation algorithm[3] on a (c, ϵ)-approximation stable instance yields an ϵ-accurate clustering. In this case, there is no need to design a new algorithm — we can use an existing one "off the shelf" to obtain a clustering that is close to the ground truth. Unfortunately, this "off the shelf" approach can only take us so far. Unless $P = NP$, there is no polynomial-time c-approximation algorithm for the k-median problem with $c < 1 + \frac{2}{e} \approx 1.736$ [2]. The best upper bound known is $1 + \sqrt{3} \approx 2.73$ [3].

What about the converse? If all we care about is correctly classifying most points, do we necessarily have to approximate the k-median objective? The main take-away point of this lecture is a negative answer: in approximation-stable k-median instances, *accurate classification is strictly easier than numerical approximation.*

Theorem 2.2 ([1]) *For every pair $\alpha, \epsilon > 0$ of constants, there is a polynomial-time algorithm that, for every $(1 + \alpha, \epsilon)$-approximation stable k-median instance, recovers a k-clustering that agrees with the target clustering on all but an $O(\epsilon/\alpha)$ fraction of the points.*

Note that when $\alpha < 2/e$, Theorem 2.2 guarantees nearly correct classification even when numerical approximation is NP-hard.[4] This is possible because approximation stability — which cannot be directly checked by an efficient algorithm — implies the presence of detectable structure that is sufficient for near-optimal classification.

2.4 Discussion: Input Assumptions in the Analysis vs. the Design of Algorithms

When previous lectures introduced an assumption about the inputs to a problem, it was to better understand the performance of a general-purpose algorithm. The LRU paging algorithm, for example, is well defined on every request sequence, and its performance improves with the locality of the sequence. Similarly, all of the independent set heuristics discussed last lecture can be run on arbitrary problem instances — it's just our analysis of these algorithms that exploits assumptions about vertex degrees.

A stronger way to exploit an assumption about problem instances is in the *design* of an algorithm, rather than merely in the analysis. Such an algorithm only solves a "promise problem" — it is guaranteed to perform well on inputs that satisfy some assumptions but might seg fault on inputs that don't (Figure 2). This lecture is the first to present such an algorithm.[5]

[3] Recall that a c-approximation algorithm for a minimization problem returns a feasible solution with objective function value at most c times the minimum possible.

[4] The inapproximability factor of $1 + \frac{2}{e}$ for general k-median instances carries over to (c, ϵ)-approximation stable instances; see Homework #3.

Also, it turns out that ϵ, α need to be strictly positive to enable the polynomial-time recovery of a good clustering, though this is not obvious [1].

[5] Indeed, while the algorithm includes some ideas with potential for broad use, its details are a bit tailored to the underlying data model. The next two lectures propose alternative conditions on clustering instances that permit the recovery of optimal clusterings by more straightforward algorithms.

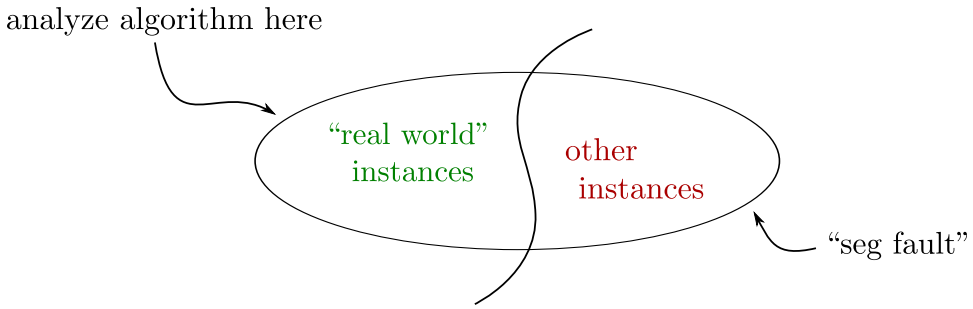

Figure 2: Promise problem: the algorithm is promised that the input satisfies some property, and is free to give arbitrary answers otherwise. We implicitly believe that the promised property holds (or nearly holds) in the instances that we care about.

All else being equal, one should prefer algorithms that solve a problem in general rather than only a "promise" version. Ideally, a general-purpose algorithm automatically takes advantage of problem structure when it exists (LRU being a good example). For many problems, however, structure needs to be explicitly looked for and exploited. For example, the effective exploitation of graph structure (such as planarity, bounded treewidth, etc.) typically requires algorithms that are specifically tailored to such structure.

3 Approximate Recovery in Approximation-Stable k-Median Instances

We show a variant of Theorem 2.2 that is easier to prove. The result itself will be incomparable to Theorem 2.2: it requires an additional non-trivial assumption about the input (see below) but correctly classifies all but an ϵ fraction of the points (rather than only an $O(\epsilon/\alpha)$ fraction).

Consider a $(1+\alpha, \epsilon)$-approximation sable k-median instance, where $\alpha, \epsilon > 0$ are constants. For simplicity, we assume that the target clustering is the same as the one that minimizes the k-median objective. This is almost without loss of generality; see Exercise #22 in the homework. Also for simplicity, we assume that we know the objective function value OPT of the optimal k-clustering $C_1^*, \ldots, C_k^*$ (but not the C_i^*'s themselves). Exercise #25 asks you to verify that the following algorithm and analysis can be adapted to solve the real problem (where OPT is unknown), by running the algorithm multiple times while varying its parameters.

3.1 Structural Consequences of Approximation Stability

Before describing an algorithm, we begin by deducing structural consequences of the approximation-stable condition that can be detected and exploited by efficient algorithms. The next definition talks about points being "close to" or "far from" various centers in the optimal solution

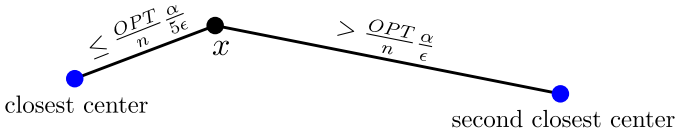

Figure 3: The point x is good if it is both close to its center in the optimal solution, and far from any other center in the optimal solution.

(a.k.a. target clustering), relative to the average distance between a point and its closest center (namely, OPT/n).

Definition 3.1 For $x \in X$, let $w(x)$ denote x's contribution ($\min_{c \in S^*} d(c, x)$, for the optimal centers S^*) to the optimal solution.

1. The point x is *close* if
$$w(x) \leq \frac{OPT}{n} \cdot \frac{\alpha}{5\epsilon}. \tag{1}$$

2. The point x is *well separated* if
$$w_2(x) - w(x) > \frac{OPT}{n} \cdot \frac{\alpha}{\epsilon}, \tag{2}$$
where $w_2(x)$ denotes the distance between x and its second-closest center in the optimal solution S^*.

3. The point x is *good* if it is close and well separated, and *bad* otherwise.

See also Figure 3. Definition 3.1 is just for the sake of analysis; an algorithm (which is ignorant of the optimal solution) has no idea which points are good and which points are bad.

The next two lemmas argue that the optimal clustering is nicely structured, comprising tightly knit and well-separated clusters (modulo a few outliers).

First, a constant fraction of all points are close (whether the instance is approximation stable or not).

Lemma 3.2 *For every k-median instance, all but $\frac{5\epsilon}{\alpha}n$ points are close.*

After all, if more than $\frac{5\epsilon}{\alpha}n$ points were not close, then the sum of the distances between points and their closest centers would be more than OPT.

Using approximation stability, we can also limit the number of points that are not well separated.

Lemma 3.3 *Fewer than ϵn points are not well separated.*

Proof: If not, start with the optimal clustering and consider reassigning ϵn points that are not well separated to their second-closest centers. This yields a clustering that misclassifies at most an ϵ fraction of the points, and that has objective function value at most

$$OPT + \epsilon n \frac{OPT}{n} \cdot \frac{\alpha}{\epsilon} = (1+\alpha) \cdot OPT.$$

This contradicts the assumption that the instance is $(1+\alpha, \epsilon)$-approximation stable.[6] ■

Combining Lemmas 3.2 and 3.3, we find that there are at most

$$b := \epsilon n \left(1 + \frac{5}{\alpha}\right) \tag{3}$$

bad points.

The "large clusters" assumption: For the rest of this lecture, we assume that every cluster C_i^* in the optimal clustering contains at least $2b+2$ points, and hence at least $b+2$ good points. For example, if $n = 10^6$, $\epsilon = 1/\sqrt{n} = .001$, and $\alpha = .1$, then this assumption requires that each optimal cluster contains at least 10% of the points (and hence there are at most 10 clusters).

This "large optimal clusters" assumption is strong enough that the k-median problem becomes polynomial-time solvable — it implies that $k = O(1/\epsilon)$ and hence an optimal solution can be computed in polynomial time via brute-force search (see Exercise #21). In the next section we give a much more efficient algorithm for this case. Balcan et al. [1] use a few extra ideas to also handle the case where some optimal clusters are small, which proves Theorem 2.2 without the large clusters assumption.

3.2 The BBG Algorithm and Analysis

We now give an intertwined presentation of the algorithm and analysis of [1] for the "large optimal clusters" version of Theorem 2.2. The high-level idea is to form a graph on the points X based on pairwise distances, and to run some efficient algorithms on this graph to illuminate easily detectable signatures of the hidden optimal clusters $C_1^*, \ldots, C_k^*$. This algorithmic approach is a promising one for attacking "well-structured" clustering problems quite generally — and recall that the primary point of the present modeling and algorithm design exercise is to generate such ideas.

[6] A subtle point: we are implicitly assuming here that if $\hat{C}_1, \ldots, \hat{C}_k$ is a k-clustering obtained from $C_1^*, \ldots, C_k^*$ by re-assigning ϵn points, then the two k-clusterings disagree on an ϵ fraction of the points. This is not true in general (Exercise #23). It is true, however, under the "large clusters assumption" that we're about to make (Exercise #24).

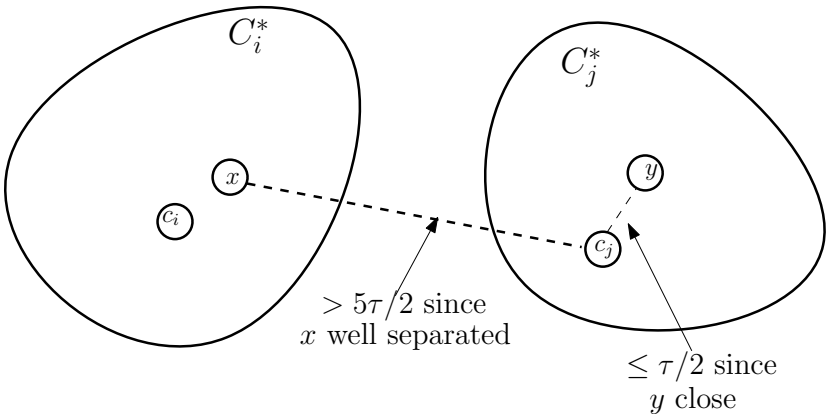

Figure 4: Observation #2 in the analysis of the BBG algorithm.

Step 1 of the BBG Algorithm: Set

$$\tau := \frac{OPT}{n} \cdot \frac{2\alpha}{5\epsilon},$$

which is twice the right-hand side of (1) and 2/5 times the right-hand side of (2). Form the graph $G = (X, E)$, where the edge (x, y) is in E if and only if $d(x, y) \leq \tau$.[7]

The hope is that the optimal clusters $C_1^*, \ldots, C_k^*$ show up in G in a thinly veiled, detectable way.

Observation #1: *All good points within a single C_i^* form a clique in G.* The reason is that, by (1), all such points are within distance $\tau/2$ of the center of C_i^*, and so by the triangle inequality they are all within distance τ of each other.

This observation is a good start, but of course in general cliques are NP-hard to detect in graphs. We need additional structure.

Observation #2: *If x, y are both good and are in different optimal clusters C_i^* and C_j^*, then there are no 1-hop or 2-hop x-y paths in G.* To see this, refer to Figure 4. Since x is well separated, its distance from the center c_j of C_j^* is more than $\frac{5}{2}\tau$. Since y is close, its distance to c_j is at most $\tau/2$. The triangle inequality implies that $d(x, y) > 2\tau$, and hence at least 3 hops in G are needed to get between x and y.

The next observation is an easy consequence of the previous one.

[7] The algorithm is in a position to construct G, provided it knows OPT (as discussed earlier) and $\frac{\alpha}{\epsilon}$. This is the only point where the algorithm references these parameters. More generally, one can keep rerunning the following algorithm with increasing thresholds τ – there are only $\binom{n}{2}$ thresholds that matter — and return the best solution found (cf., Exercise #25).

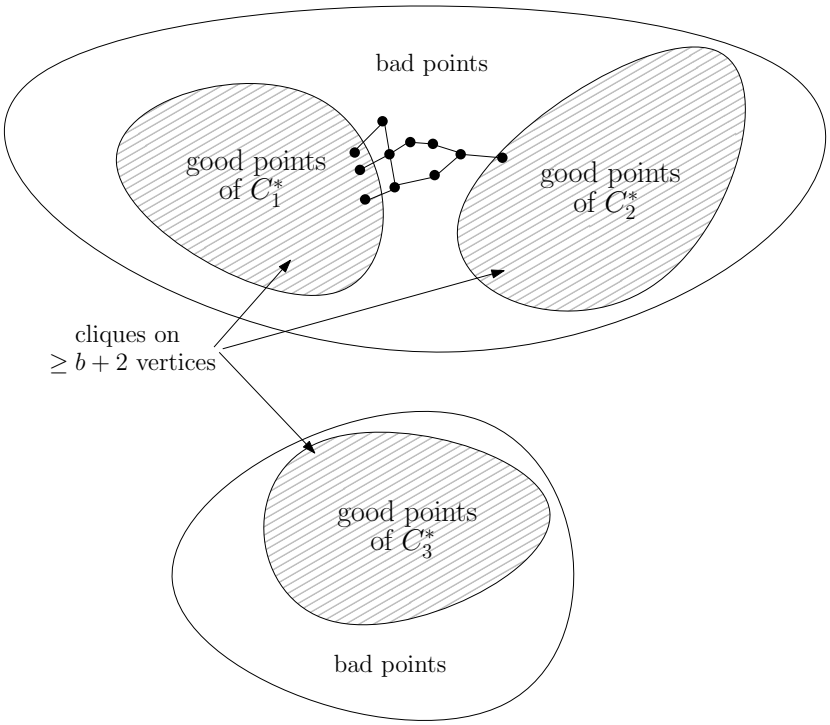

Figure 5: The graph G after the first step of the BBG algorithm.

Observation #3: *The neighbors of a bad point in G consist of other bad points (at most $b-1$ of them – recall (3)) and possibly some good points from a single cluster C_i^*.*

Figure 5 summarizes what the graph G looks like in light of these observations — a clique for each cluster C_i^*, with different cliques connected only by relatively long paths. It's still not clear that we have enough structure to efficiently detect the cliques that correspond to the optimal clusters.

Cliques in G can only be connected to each other by relatively long chains. The next step of the algorithm filters the edges of G to sever these long chains (without destroying the cliques) to better highlight the cliques.

Step 2 of the BBG Algorithm: Form the graph $H = (X, F)$, where $(x, y) \in F$ if and only if $(x, y) \in E$ and x, y have at least b common neighbors in G.[8]

Our fourth observation is immediate from the definitions and the large clusters assumption.

Observation #4: *The cliques of good points in G (each on at least $b+2$ nodes) survive in H.*

[8]Both distance thresholding and edge filtering using common neighbors seem like good algorithmic ideas to try across a range of clustering problems.

The fifth observation is a consequence of the third one.

Observation #5: *No paths between good points in different optimal clusters survive in H.*

By Observation #3 and the definition of Step 2, two bad points are adjacent in H only if each is adjacent in G to good points from the same cluster C_i^*, and thus necessarily to no good points from the other clusters. The second observation then implies that every (at least 3-hop) path between good points of different optimal clusters is severed in H — there is some edge (u, v) on every such a path where u and v are bad points that do not have neighbors in a common cluster C_i^*, and this edge does not survive in H.

Summarizing, the graph H has precisely k connected components that have size at least $b + 2$, and each of these contains all of the good points from some C_i^*. These k components can be identified in linear time, and any small components (of bad points) can be merged in arbitrarily to get a k-clustering $(\hat{C}_1, \ldots, \hat{C}_k)$. This k-clustering correctly classifies all good points, which constitute at least a $1 - \epsilon(1 + \frac{5}{\alpha})$ fraction of all of the points. For example, with our running example parameters $n = 10^6$, $\epsilon = 10^{-3}$, and $\alpha = .1$, we obtain a clustering that misclassifies at most $\approx 5\%$ of the points.

Step 3 of the BBG Algorithm. After Step 2, we have correctly classified all but an $O(\epsilon/\alpha)$ fraction of the points (all but the bad points). This is the guarantee asserted in the original statement of Theorem 2.2, without the "big optimal clusters" assumption. We now show that under the latter assumption, we can correctly classify all but the ϵ fraction of the points that are not well separated.[9] The goal of this "clean-up" step is to correctly classify points that are well separated but that might not be close. The idea is to use a clever form of majority vote.

Precisely, we execute the following procedure in parallel (i.e., independently) for each $x \in X$:

1. For each cluster $\widehat{C}_i$ computed in Step 2, look at the *median* distance $d(x, y)$ among points $y \in \widehat{C}_i$ (not including x itself, in case it is already in $\widehat{C}_i$).

2. Re-assign x to the cluster to which its median distance is the smallest.

That is, each point x looks around to see which cluster $\hat{C}_i$ it is closest to (according to the median distance) and then joins it.

Every Well Separated Point Is Correctly Re-assigned: Suppose x rightfully belongs to the optimal cluster C_i^*. (It may or may not be there already.) Then for all of the $\geq (b+1)$ good points of C_i^* that are different from x (which are all also members of $\widehat{C}_i$), the triangle inequality implies that

$$d(x, y) \leq \underbrace{d(x, c_i)}_{\text{(where } c_i \text{ is the center of } C_i^*)} + \underbrace{d(c_i, y)}_{\leq \tau/2 \text{ since } y \text{ is close}} \leq d(x, c_i) + \frac{\tau}{2}. \tag{4}$$

[9] Previous uses of the "big optimal clusters" assumption can be essentially removed with some further work. However, it is not clear how to implement this third step without this assumption.

Since at most b points of $\widehat{C}_i$ are not good, the median distance $d(x, y)$ with $y \in \widehat{C}_i \setminus \{x\}$ also satisfies (4).

For some other cluster $\widehat{C}_j$, and for all of the good points y in C_j^* (and $\widehat{C}_j$) different than x, the triangle inequality implies that

$$d(x, y) \geq \underbrace{d(x, c_j)}_{\geq d(x,c_i)+5\tau/2 \text{ since } x \text{ well separated}} - \underbrace{d(c_i, y)}_{\leq \tau/2 \text{ since } y \text{ is close}} \geq d(x, c_i) + 2\tau. \tag{5}$$

Again, the median distance $d(x, y)$ with $y \in \widehat{C}_i \setminus \{x\}$ must also satisfy this inequality. Comparing (4) and (5), we see that every well separated point x is inevitably re-assigned to the correct cluster.

References

[1] M.-F. Balcan, A. Blum, and A. Gupta. Approximate clustering without the approximation. In *Proceedings of the 20th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1068–1077, 2009.

[2] S. Guha and S. Khuller. Greedy strikes back: Improved facility location algorithms. *Journal of Algorithms*, 31(1):228–248, 1999.

[3] S. Li and O. Svensson. Approximating k-median via pseudo-approximation. In *Proceedings of the 45th Annual ACM Symposium on Theory of Computing (STOC)*, pages 901–910, 2013.