

---

# Robust Hierarchical Clustering

---

**Maria Florina Balcan**  
Georgia Institute of Technology  
School of Computer Science  
ninamf@cc.gatech.edu

**Pramod Gupta**  
Georgia Institute of Technology  
School of Computer Science  
pramodgupta@gatech.edu

## Abstract

One of the most widely used techniques for data clustering is agglomerative clustering. Such algorithms have been long used across many different fields ranging from computational biology to social sciences to computer vision in part because their output is easy to interpret. Unfortunately, it is well known, however, that many of the classic agglomerative clustering algorithms are *not* robust to noise [14]. In this paper we propose and analyze a new robust algorithm for bottom-up agglomerative clustering. We show that our algorithm can be used to cluster accurately in cases where the data satisfies a number of natural properties and where the traditional agglomerative algorithms fail. We also show how to adapt our algorithm to the inductive setting where our given data is only a small random sample of the entire data set.

## 1 Introduction

Many data mining and machine learning applications ranging from computer vision to biology problems have recently faced an explosion of data. As a consequence it has become increasingly important to develop effective, accurate, robust to noise, fast, and general clustering algorithms, accessible to developers and researchers in a diverse range of areas.

One of the oldest and most commonly used methods for clustering data, widely used in many scientific applications, is hierarchical clustering [5, 6, 9, 8, 11, 12, 14, 10, 13]. In hierarchical clustering the goal is not to find a single partitioning of the data, but a hierarchy (generally represented by a tree) of partitions which may reveal interesting structure in the data at multiple levels of granularity. The most widely used hierarchical methods are the agglomerative clustering techniques; most of these techniques start with a separate cluster for each point and then progressively merge the two closest clusters until only a single cluster remains. In all cases, we assume that we have a measure of similarity between pairs of objects, but the different schemes are distinguished by how they convert this into a measure of similarity between two clusters. For example, in single linkage the similarity between two clusters is the maximum similarity between points in these two different clusters. In complete linkage, the similarity between two clusters is the minimum similarity between points in these two different clusters. Average linkage has various variants, for example, a common one defines the similarity between two clusters as the average similarity between points in these two different clusters [8, 12].

Such algorithms have been used in a wide range of application domains ranging from biology applications to social sciences to computer vision applications mainly because they are quite fast and the output is quite easy to interpret. It is well known, however, that one of the main limitations of the agglomerative clustering algorithms is that they are *not* robust to noise [14]. In this paper we propose and analyze a robust algorithm for bottom-up agglomerative clustering. We show that our algorithm satisfies formal robustness guarantees and it will be successful in many cases where the traditional agglomerative algorithms fail.

In order to formally analyze correctness of our algorithm we use the framework introduced by Balcan et. al [2]. In this framework, we assume there is some target clustering (much like a k-class target function in the multi-class learning setting) and we say that an algorithm correctly clusters data satisfying property  $P$  if on any data set having property  $P$ , the algorithm produces a tree such that the target is some pruning of the tree. For example if all points are more similar to points

in their own target cluster than to points in any other cluster (this is called the strict separation property), then any of the standard agglomerative algorithms will succeed. See Figure 1. However, with just tiny bit of noise, for example if each point has even just one point from a different cluster that it is similar too, then the standard algorithms will all fail (we elaborate on this in Section 2.2). See Figure 2. This brings up the question: is it possible to design an agglomerative algorithm that is robust to these types of situations and more generally can tolerate a substantial degree of noise? The contribution of our paper is to provide a strong positive answer to this question; we develop a robust, linkage based algorithm that will succeed in many interesting cases where standard agglomerative algorithms will fail. At a high level, our new algorithm is robust to noise in two different and important ways. First, it uses more global information for creating an interesting starting point for a linkage procedure (a set of not too small, but also not too large blobs that are mostly “pure”); second, it uses a robust linkage procedure for merging large enough blobs.

## 1.1 Our Results

We present a new and robust algorithm for agglomerative clustering and we show that our algorithm will be successful in many cases where standard agglomerative algorithms will fail.

In particular, we show that if the data satisfies a natural good neighborhood property, then our algorithm can be used to cluster well in the tree model (i.e., to output a hierarchy such that the target clustering is a pruning of that hierarchy). The good neighborhood property roughly says after a small number of malicious points have been removed, for the remaining points, most of their nearest neighbors are from their target cluster. We also show how to adapt our algorithm to the inductive setting, where our given data is only a small random sample of the entire data set. Based on such a sample, our algorithm outputs an implicit hierarchy of clusterings of the full domain, that is evaluated with respect to the underlying distribution. A nice property of the condition and of the algorithm we analyze is that they are insensitive to any monotone transformation of the similarities.

It is worth noting that the good neighborhood property is much broader than the  $\nu$ -strict separation property, a generalization of the simple strict separation property discussed above, requiring that after a small number of outliers have been removed all points are strictly more similar to points in their own cluster than to points in other clusters. Balcan et. al [2] also analyzed the  $\nu$ -strict separation condition and provided an algorithm for producing a hierarchy with the desired property, but via a much more computationally expensive (non-agglomerative) algorithm. Our algorithm is simpler, substantially faster, and much more generally applicable compared to the algorithm in [2] specifically designed for  $\nu$ -strict separation.

## 1.2 Related Work

In agglomerative hierarchical clustering [9, 8, 11, 12] the goal is not to find a single partitioning of the data, but a hierarchy (generally represented by a tree) of partitionings which may reveal interesting structure in the data at multiple levels of granularity. Traditionally, only clusterings at a certain level are considered, but as we argue in Section 2 it is more desirable to consider all the prunings of the tree, since this way we can then handle much more general situations. As mentioned above, it is well known that standard agglomerative hierarchical clustering techniques are not tolerant to noise.

## 2 Definitions. A Formal Setup

We consider a clustering problem  $(S, \ell)$  specified as follows. Assume we have a data set  $S$  of  $n$  objects. Each  $x \in S$  has some (unknown) “ground-truth” label  $\ell(x)$  in  $Y = \{1, \dots, k\}$ , where we will think of  $k$  as much smaller than  $n$ . We let  $C_i = \{x \in S : \ell(x) = i\}$  denote the set of points of label  $i$  (which could be empty), and denote the target clustering as  $\mathcal{C} = \{C_1, \dots, C_k\}$ . Given another proposed clustering  $h, h : S \rightarrow Y$ , we define the error of  $h$  with respect to the target clustering to be the fraction of points on which  $h$  and  $\mathcal{C}$  disagree under the optimal matching of clusters in  $h$  to clusters in  $\mathcal{C}$ ; i.e.,

$$err(h) = \min_{\sigma \in \mathcal{S}_k} \left[ \Pr_{x \in S} [\sigma(h(x)) \neq \ell(x)] \right],$$

where  $\mathcal{S}_k$  is the set of all permutations on  $\{1, \dots, k\}$ . Equivalently, the error of a clustering  $\mathcal{C}' = \{C'_1, \dots, C'_k\}$  can be expressed as

$$\min_{\sigma \in \mathcal{S}_k} \frac{1}{n} \sum_i |C_i - C'_{\sigma(i)}|.$$

We will be considering clustering algorithms whose only access to their data is via a pairwise similarity function  $\mathcal{K}(x, x')$  that given two examples outputs a number in the range  $[-1, 1]$ . We will say that  $\mathcal{K}$  is a symmetric similarity function if  $\mathcal{K}(x, x') = \mathcal{K}(x', x)$  for all  $x, x'$ . In this paper we

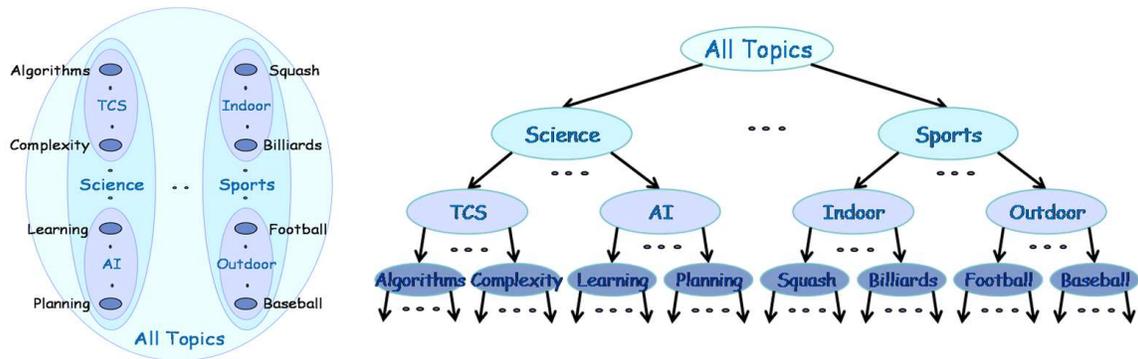


Figure 1: Consider a document clustering problem. Assume that data lies in multiple regions Algorithms, Complexity, Learning, Planning, Squash, Billiards, Football, Baseball. Suppose that  $\mathcal{K}(x, y) = 0.999$  if  $x$  and  $y$  belong to the same inner region;  $\mathcal{K}(x, y) = 3/4$  if  $x \in$  Algorithms and  $y \in$  Complexity, or if  $x \in$  Learning and  $y \in$  Planning, or if  $x \in$  Squash and  $y \in$  Billiards, or if  $x \in$  Football and  $y \in$  Baseball;  $\mathcal{K}(x, y) = 1/2$  if  $x$  is in (Algorithms or Complexity) and  $y$  is in (Learning or Planning), or if  $x$  is in (Squash or Billiards) and  $y$  is in (Football or Baseball); define  $\mathcal{K}(x, y) = 0$  otherwise. Both clusterings  $\{\text{Algorithms} \cup \text{Complexity} \cup \text{Learning} \cup \text{Planning}, \text{Squash} \cup \text{Billiards}, \text{Football} \cup \text{Baseball}\}$  and  $\{\text{Algorithms} \cup \text{Complexity}, \text{Learning} \cup \text{Planning}, \text{Squash} \cup \text{Billiards} \cup \text{Football} \cup \text{Baseball}\}$  satisfy the strict separation property.

assume that the similarity function  $\mathcal{K}$  is symmetric. For  $A \subseteq S$ , we denote by  $n_A$  the number of points in  $A$ .

Our goal is to produce a hierarchical clustering that contains a pruning that is close to the target clustering. Formally, the goal of the algorithm is to produce a hierarchical clustering: that is, a tree on subsets such that the root is the set  $S$ , and the children of any node  $S'$  in the tree form a partition of  $S'$ . The requirement is that there must exist a *pruning*  $h$  of the tree (not necessarily using nodes all at the same level) that has error at most  $\epsilon$ . Balcan et. al [2] have shown that this type of output is necessary in order to be able to analyze non-trivial properties of the similarity function. For example, even if the similarity function satisfies the requirement that all points are more similar to all points in their own cluster than to any point in any other cluster (this is called the strict separation property) and even if we are told the number of clusters, there can still be multiple different clusterings that satisfy the property. In particular, one can show examples of similarity functions and two significantly different clusterings of the data satisfying the strict separation property. See Figure 1 for an example. However, under the strict separation property, there is a single hierarchical decomposition such that any consistent clustering is a pruning of this tree. This motivates clustering in the tree model and this is the model we consider in this work as well.

Given a similarity function satisfying the strict separation property (see Figure 1 for an example), we can efficiently construct a tree such that the ground-truth clustering is a pruning of this tree [2]. Moreover, almost any of the standard linkage based algorithms (*e.g.*, single linkage, average linkage, or complete linkage) would work well under this property. However, one can show that if the similarity function slightly deviates from the strict separation condition, then all the standard agglomerative algorithms will fail (we elaborate on this in section 2.2). In this context, the main question we address in this work is: Can we develop other more robust, linkage based algorithms that will succeed under more realistic and yet natural conditions on the similarity function?

**Note:** Note that strict separation does not guarantee that all the cutoffs for different points  $x$  are the same, so single linkage would not necessarily have the right clustering if just stopped once it has  $k$  clusters; however the target clustering will provably be a pruning of the final single linkage tree; this is why we define success based on prunings.

## 2.1 Properties of the similarity function

We describe here some natural properties of the similarity functions that we analyze in this paper. We start with a noisy version of the simple strict separation property (mentioned above) which was introduced in [2] and we then define an interesting and natural generalization of it.

**Property 1** *The similarity function  $\mathcal{K}$  satisfies  $\nu$ -strict separation for the clustering problem  $(S, \ell)$  if for some  $S' \subseteq S$  of size  $(1 - \nu)n$ ,  $\mathcal{K}$  satisfies strict separation for  $(S', \ell)$ . That is, for all  $x, x', x'' \in S'$  with  $x' \in C(x)$  and  $x'' \notin C(x)$  we have  $\mathcal{K}(x, x') > \mathcal{K}(x, x'')$ .*

So, in other words we require that the strict separation is satisfied after a number of bad points have been removed. A somewhat different condition is to allow each point to have some bad immediate neighbors as long as most of its immediate neighbors are good. Formally:

**Property 2** *The similarity function  $\mathcal{K}$  satisfies  $\alpha$ -good neighborhood property for the clustering problem  $(S, \ell)$  if for all points  $x$  we have that all but  $\alpha n$  out of their  $n_{C(x)}$  nearest neighbors belong to the cluster  $C(x)$ .<sup>1</sup>*

Note that  $\alpha$ -good neighborhood property is different from the  $\nu$ -strict separation property. For the  $\nu$ -strict separation property we can have up to  $\nu n$  bad points that can misbehave; in particular, these  $\nu n$  bad points can have similarity 1 to *all* the points in  $S$ ; however, once we remove these points the remaining points are more similar to points in their own cluster than to points in other cluster. On the other hand, for the  $\alpha$ -good neighborhood property we require that for all points  $x$  all but  $\alpha n$  out of their  $n_{C(x)}$  nearest neighbors belong to the cluster  $C(x)$ . (So we cannot have a point that has similarity 1 to all the points in  $S$ .) Note however that different points might misbehave on different  $\alpha n$  neighbors. We can also consider a property that generalizes both the  $\nu$ -strict separation property and the  $\alpha$ -good neighborhood. Specifically:

**Property 3** *The similarity function  $\mathcal{K}$  satisfies  $(\alpha, \nu)$ -good neighborhood property for the clustering problem  $(S, \ell)$  if for some  $S' \subseteq S$  of size  $(1 - \nu)n$ ,  $\mathcal{K}$  satisfies  $\alpha$ -good neighborhood property for  $(S', \ell)$ . That is, for all for all points  $x \in S'$  we have that all but  $\alpha n$  out of their  $n_{C(x) \cap S'}$  nearest neighbors in  $S'$  belong to the cluster  $C(x)$ .*

It is easy to see that:

**Fact 1** *If the similarity function  $\mathcal{K}$  satisfies the  $\alpha$ -good neighborhood property for the clustering problem  $(S, \ell)$ , then  $\mathcal{K}$  also satisfies the  $(\alpha, 0)$ -good neighborhood property for the clustering problem  $(S, \ell)$ .*

**Fact 2** *If the similarity function  $\mathcal{K}$  satisfies the  $\nu$ -strict separation property for the clustering problem  $(S, \ell)$ , then  $\mathcal{K}$  also satisfies the  $(0, \nu)$ -good neighborhood property for the clustering problem  $(S, \ell)$ .*

Balcan et. al [2] have shown that if  $\mathcal{K}$  satisfies the strict separation property with respect to the target clustering, then as long as the smallest target cluster has size  $5\nu n$ , one can in polynomial time construct a hierarchy with the guarantee that the ground-truth is  $\nu$ -close to a pruning of the hierarchy. Unfortunately the algorithm presented in [2] is computationally very expensive: it first generate a large list of  $\Omega(n^2)$  candidate clusters and repeatedly runs pairwise tests in order to laminarize these clusters; its running time is a large unspecified polynomial. Our new robust linkage algorithm can be used to get a simpler and much faster algorithm for clustering accurately under the  $\nu$ -strict separation property. Additionally, our algorithm is much more general as well.

As shown in [2], the  $(2, \epsilon)$  BBG-condition for k-median implies the  $\nu$ -strict separation condition [1], for  $\nu = 5\epsilon$ . One can show a similar result for the  $(\nu, c, \epsilon)$ -condition for k-median introduced by [3], and so the condition we analyze is strictly more general than these conditions.

As we show below, if the data satisfies the good neighborhood property, then most of the standard linkage based algorithms will fail. The contribution of our paper is to develop a robust, linkage based algorithm that will succeed under these natural conditions.

## 2.2 Standard linkage based algorithms are not robust

We can show an example where simple linkage based algorithm would perform very badly, but where our algorithm would work well. In particular, if we slightly modify the example in Figure 1, by adding a little bit of noise, to form links of high similarity between points in different inner blobs,

---

<sup>1</sup>Note that we assume that for any given point we have a canonical order for its neighbors, and so the set of  $t$  nearest neighbors of a given point is always well defined.

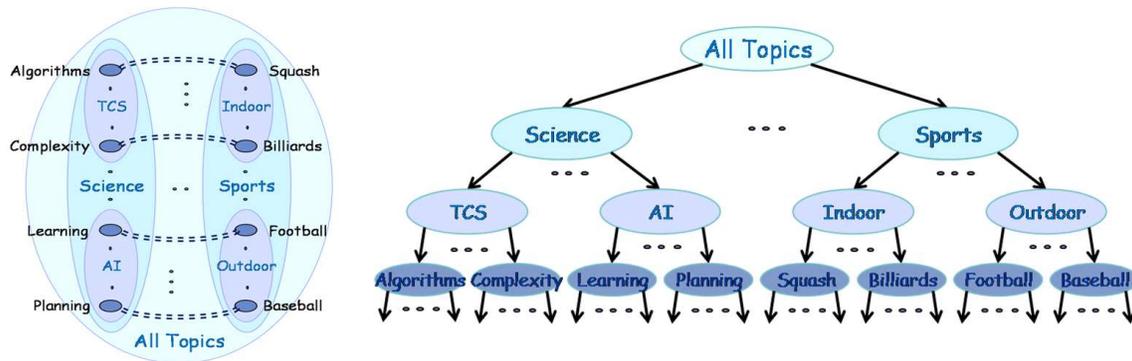


Figure 2: Same as Figure 1 except let us match each point in Algorithms with a point in Squash, each point in Complexity with a point in Billiards, each point in Learning with a point in Football, and each point in Planning with a point in region Baseball. Define the similarity measure to be the same as in Figure 1 except that we let  $\mathcal{K}(x, y) = 1$  if  $x$  and  $y$  are matched. Note that for  $\alpha = 1/n$  the similarity function satisfies the  $\alpha$ -good neighborhood property with respect to any of the prunings of the tree above. However, single linkage, average linkage, and complete linkage would initially link the matched pairs and produce clusters with very high error with respect to any such clustering.

we can show that many of the classic linkage based algorithms will perform poorly<sup>2</sup>. See Figure 2 for a precise description of the similarity measure.

In particular, the single linkage algorithm, the average linkage algorithm, and the complete linkage algorithm, would in the first  $n/2$  stages merge the matched pairs of points. From that moment on, no matter how they perform, none of the natural and desired clusterings will be even  $1/2$  close to any of the prunings of the hierarchy produced. Notice however, that the similarity  $\mathcal{K}$  satisfies  $\alpha$ -good neighborhood property with respect to any of the desired clusterings (for  $\alpha = 1/n$ ), and that our algorithm will be successful on this instance. The  $\nu$ -strict separation is not satisfied in this example either, for any constant  $\nu$ .

### 3 Robust Hierarchical Clustering

In this section we describe an algorithm that we prove is successful if the data satisfies the good neighborhood property. This procedure has two phases: first, it uses somewhat more global information for creating an interesting starting point for a linkage procedure – a set of not too small, but also not too large blobs that are mostly “pure”. In a second phase, it runs robust linkage procedure on this set of blobs. Both steps have to be done with care and we will describe in detail in the following sections both steps of our algorithm. In particular, in section 3.1 we describe the procedure for generating a set of interesting blobs and in section 3.2 we describe the linkage procedure.

---

#### Algorithm 1 Robust Agglomerative Hierarchical Clustering

---

**Input:** Similarity function  $\mathcal{K}$ , set of points  $S$ ,  $\nu > 0$ ,  $\alpha > 0$ .

1. Run Algorithm 2 with parameters  $\nu$ ,  $\alpha$  to generate an interesting list  $L$  of blobs that partitions the whole set  $S$ .
2. Run the linkage Algorithm 3 on these blobs to get the tree  $T$ .

**Output:** Tree  $T$  on subsets of  $S$ .

---

We start with a useful definition.

**Definition 1** For  $A \subseteq S$ ,  $B \subseteq S$  we define  $\mathcal{K}_{\text{median}}(A, B) = \text{median}\{\mathcal{K}(x, x'); x \in A, x' \in B\}$  and we call this the median similarity of  $A$  to  $B$ .

<sup>2</sup>Since, usually, the similarity function between pairs of objects is constructed based on heuristics, this can easily happen; for example we could have a similarity measure that puts a lot of weight on features such as date or names, and so we could easily have a document about Learning being more similar to a document about Football than to other documents about Learning.

For simplicity we denote  $\mathcal{K}_{\text{median}}(\{x\}, B)$  as  $\mathcal{K}_{\text{median}}(x, B)$ .

**Notation:** For the rest of this section we assume that the similarity function  $\mathcal{K}$  satisfies the  $(\alpha, \nu)$ -good neighborhood property for the clustering problem  $(S, \ell)$ . Let  $S' \subseteq S$  be the set of  $(1 - \nu)n$  points such that  $\mathcal{K}$  satisfies  $\alpha$ -good neighborhood property with respect to  $S'$ . We call the points in  $S'$  good points and the points in  $S \setminus S'$  bad points. Let  $G_i = C_i \cap S'$  be the good set of label  $i$ . Let  $G = \cup G_i$  the whole set of good points; so  $G = S'$ . Clearly  $|G| \geq n - \nu n$ . Denote by  $\mathcal{C}_G$  the restriction of the target clustering to the set  $G$ .

Note that the following is a useful consequence of the definition.

**Claim 2** *Let  $\mathcal{K}$  be a symmetric similarity function satisfying the  $(\alpha, \nu)$ -good neighborhood property for the clustering problem  $(S, \ell)$ . As long as  $t$  is smaller than  $n_{C_i}$  for any good point  $x \in C_i$  all but at most  $(\nu + \alpha)n$  out of its  $t$  nearest neighbors lie in its good set  $C_i(x) \cap G$ .*

### 3.1 Generating an interesting starting point

---

**Algorithm 2** Generate interesting blobs

---

**Input:** similarity function  $\mathcal{K}$ , set of points  $S$ ,  $\nu > 0$ ,  $\alpha > 0$ .

Let the initial threshold  $t = 6(\nu + \alpha)n + 1$ . Let  $L$  be empty. Let  $A_S = S$ .

- Step 1** Construct the graph  $F_t$  where we connect points  $x$  and  $y$  in  $A_S$  if they share at least  $t - 2(\nu + \alpha)n$  points in common out of their  $t$  nearest neighbors with respect to the whole set of points  $S$ .
- Step 2** Construct the graph  $H_t$  by connecting points  $x$  and  $y$  if they share at least  $3(\nu + \alpha)n$  neighbors in the graph  $F_t$ .
- Step 3** (i) Add to  $L$  all the components  $C$  of  $H_t$  with  $|C| \geq 3(\nu + \alpha)n$  and remove from  $A_S$  all the points in all these components.  
(ii) For all points  $x$  in  $A_S$  check if  $(\nu + \alpha)n$  out of their  $5(\nu + \alpha)n$  nearest neighbors are in  $L$ . If so, then assign point  $x$  to any of the blobs in  $L$  of highest median. Remove the points in all these components from  $A_S$ .
- Step 4** While  $|A_S| \geq 3(\nu + \alpha)n$  and  $t < n$ , increase the critical threshold and go to Step 1.
- Step 5** Assign all points  $x$  that do not belong to any of the blobs in  $L$  arbitrarily to one of the blobs.

**Output:** A list of blobs which form a partition of  $S$ .

---

We can show the following:

**Theorem 3** *Let  $\mathcal{K}$  be a symmetric similarity function satisfying the  $(\alpha, \nu)$ -good neighborhood property for the clustering problem  $(S, \ell)$ . So long as the smallest target cluster has size greater than  $9(\nu + \alpha)n$ , then we can use Algorithm 2 to create a list  $L$  of blobs each of size at least  $3(\nu + \alpha)n$  such that:*

- The blobs in  $L$  form a partition of  $S$ .
- Each blob in the list  $L$  contains good points from only one good set; i.e., for any  $C \in L$ ,  $C \cap G \subseteq G_i$  for some  $i \leq k$ .

**Proof:** In the following we denote by  $n_{C_i}$  the number of points in the target cluster  $i$ . Without loss of generality assume that  $n_{C_1} \leq n_{C_2} \leq \dots \leq n_{C_k}$ . We will show by inductions on  $i \leq k$  that:

- (a) For any  $t \leq n_{C_i}$ , any blob in the list  $L$  only contains good points from a single good set  $G_i$ ; all blobs have size at least  $3(\nu + \alpha)n$ .
- (b) At the beginning of the iteration  $t = n_{C_i} + 1$ , any good point  $x \in C_j \cap G$ ,  $j \in \{1, 2, \dots, i\}$  has already been assigned to a blob in the list  $L$  that contains points only from  $C_j \cap G$  and has more good points than bad points.

These two claims clearly imply that each blob in the list we output contains good points from only one good set. Moreover at  $t = n_{C_k}$  all good points have been assigned to one of the blobs in  $L$ . Since we assign the remaining points  $x$  that do not belong to any of the blobs in  $L$  (these can only

be bad points) arbitrarily to one of the blobs, we also get that the blobs in  $L$  form a partition of  $S$ , as desired.

Claims (a) and (b) are clearly both true initially. We show now that as long as  $t \leq n_{C_1}$ , the graphs  $F_t$  and  $H_t$  have the following properties:

- (1) No good point in cluster  $i$  is connected in  $F_t$  to a good point in a different cluster  $j$ , for  $i, j > 1$ ,  $i \neq j$ . Since  $\mathcal{K}$  satisfies the  $(\alpha, \nu)$ -good neighborhood property for the clustering problem  $(S, \ell)$ , by Claim 2, we know that as long as  $t$  is smaller than  $n_{C_i}$  for any good point  $x \in C_i$  all but at most  $(\nu + \alpha)n$  out of its  $t$  nearest neighbors lie in its good set, i.e.,  $C_i(x) \cap G$ ; similarly, as long as  $t$  is smaller than  $n_{C_j}$  for any good point  $y \in C_j$ , all but at most  $(\nu + \alpha)n$  out of its  $t$  nearest neighbors lie in its good set, i.e.,  $|C_j(y) \cap G|$ ; so it cannot be the case that for  $6(\nu + \alpha)n \leq t \leq n_{C_1}$  two good points in two different clusters  $i, j \geq 1$  share  $t - 2(\nu + \alpha)n$  points in common out of their  $t$  nearest neighbors.
- (2) No bad point is connected in  $F_t$  to both a good point in cluster  $i$  and a good point in different cluster  $j$ , for  $i, j > 1$ ,  $i \neq j$ . This again follows from the fact that since  $t \leq n_{C_i}$  for all  $i$ , for any good point  $x$  all but at most  $(\nu + \alpha)n$  out of its  $t$  nearest neighbors lie in its good set  $C_i(x) \cap G$  (by Claim 2); so for a bad point  $z$  to share  $t - 2(\nu + \alpha)n$  points out of its  $t$  nearest neighbors in common with the  $t$  nearest neighbors of a good point  $x$  in  $G_i$  it must be the case that  $z$  has  $t - 3(\nu + \alpha)n$  points out of its  $t$  nearest neighbors in  $G_i$ ; but that means that there cannot be other good point  $y$  in  $G_j$ , where  $j \neq i$  such that  $z$  and  $y$  share  $t - 2(\nu + \alpha)n$  points among their  $t$  nearest neighbors, because we would need to have that  $t - 3(\nu + \alpha)n$  of points out of  $z$ 's  $t$  nearest neighbors lie in  $G_i$  and  $t - 3(\nu + \alpha)n$  of points out of  $z$ 's  $t$  nearest neighbors in  $G_j$ ; but this cannot happen if  $t > 6(\nu + \alpha)n$  since  $2(t - 3(\nu + \alpha)n) > t$  for  $t > 6(\nu + \alpha)n$ .
- (3) All the components of  $H_t$  of size at least  $3(\nu + \alpha)n$  will only contain good points from one cluster. Since in  $F_t$  bad points can only connect to one good set, we get that no two good points in the different clusters connect in  $H_t$ .

We can use (1), (2), and (3) to argue that as long as  $t \leq n_{C_1}$ , each blob in  $L$  contains good points from at most one target cluster. This is true at the beginning and by (3), for any  $t \leq n_{C_1}$ , anytime we insert a whole new blob in  $L$  in Step 3(i), that blob must contain good points from at most one target cluster. We now argue that this property is never violated as we assign points to blobs already in  $L$  based on the median test in Step 3(ii). Note that at all time steps all the blobs in  $L$  have size at least  $3(\nu + \alpha)n$ . Assume that a good point  $x$  has more than  $(\nu + \alpha)n$  out of its  $5(\nu + \alpha)n$  nearest neighbors in  $S$  in the list  $L$ . By Lemma 5, there must exist a blob in  $L$  that contains only good points from  $C(x)$ . By Lemma 4, if we assign  $x$  based on the median test in Step 3(ii), then we will add  $x$  to a blob containing good points only from  $C(x)$ , and so we maintain the invariant that each blob in  $L$  contains good points from at most one target cluster.

We now show that at the beginning of the iteration  $t = n_{C_1} + 1$ , all the good points in  $C_1$  have already been assigned to a blob in the list  $L$  that only contains good points from  $C_1 \cap G$ . There are a few cases. First, if prior to  $t = n_{C_1}$  we did not yet extract in the list  $L$  a blob with good points from  $C_1$ , then it must be the case that all good points in  $C_1$  connect to each other in the graph  $F_t$ ; so there will be a component of  $H_t$  that will contain all good points from  $C_1$  and potentially bad points, but no good points from another target cluster; moreover this  $|C_1| \geq 9(\nu + \alpha)n$ , this component will be output in Step 3(i). Second, if prior to  $t = n_{C_1}$  we did extract some, but still, more than  $3(\nu + \alpha)n$  points from the good set  $G_1$  do not belong to blobs in the list  $L$ , then more than  $3(\nu + \alpha)n$  of good points will connect to each other in  $F_t$ , and then in  $H_t$ , so we will add one blob to  $L$  containing these good points (plus at most  $\nu n$  bad points). Finally, it could be that by the time we reach  $t = n_{C_1}$  all but  $l < 3(\nu + \alpha)n$  good points in  $C_1$  have been assigned to a blob in the list  $L$  that has good points only from  $C_1$ . Since  $|C_1| \geq 9(\nu + \alpha)n$  we must have assigned at least  $9(\nu + \alpha)n - 3(\nu + \alpha)n - \nu n \geq 5(\nu + \alpha)n$  good points from  $C_1$  to the list  $L$ . This together with the  $(\alpha, \nu)$ -good neighborhood property implies that the good points in  $C_1$  that do not belong to the list  $L$  yet, must have  $(\nu + \alpha)n$  out of their  $5(\nu + \alpha)n$  nearest neighbors in  $S$  in the list  $L$  (at most  $\nu$  out of the  $5(\nu + \alpha)n$  nearest neighbors can be bad points, at most  $\alpha n$  can be good points from a different cluster, and at most  $3(\nu + \alpha)n$  can be good points in  $C_1$  that do not yet belong to  $L$ ). So we will assign these points to blobs in  $L$  based on the median test in Step 3(ii). By Lemma 4, when we assign them based on the median test in Step 3(ii), we will add them to a blob containing good points from  $C_1$  and no good points from other cluster  $C_j$ , as desired.

We then iterate the argument on the remaining set  $A_S$ . The key point is that for  $t \geq n_i$ ,  $i > 1$ , once we start analyzing good points in  $C_{n_i+1}$  we have that *all* the good points in  $C_{n_i}$ ,  $C_{n_i-1}$ , ...,  $C_{n_1}$  have already been assigned to blobs in  $L$ . ■

We prove below two useful lemmas used in the above proof.

**Lemma 4** *Let  $\mathcal{K}$  be a symmetric similarity function satisfying the  $(\alpha, \nu)$ -good neighborhood property for the clustering problem  $(S, \ell)$ . Assume that  $L$  is a list of disjoint clusters each of size at least  $3(\nu + \alpha)n$ . Assume also that each cluster in  $L$  intersects at most a good set; i.e., for any  $C$  in  $L$ , we have  $C \cap G \subseteq G_i$  for some  $i$ . Consider  $x \in G$  such that there exist  $C$  in  $L$  with  $C \cap G \subseteq C(x) \cap G$ . Let  $\tilde{C}$  be the blob in  $L$  of highest median similarity to  $x$ . Then  $\tilde{C} \cap G \subseteq C(x) \cap G$ .*

**Proof:** Let us fix a good point  $x$ . Let  $C'$  and  $C''$  be such that  $C' \cap G \subseteq C(x) \cap G$  and  $C'' \cap G \subseteq C_i \cap G$ , for  $C_i \neq C(x)$ . Since  $\mathcal{K}$  is a symmetric similarity function satisfying the  $(\alpha, \nu)$ -good neighborhood property, by Claim 2, we have that  $x$  can be more similar to at most  $\nu n + \alpha n$  points in  $C''$  than with any point in  $C' \cap G$ . Since  $|C'| \geq 3(\nu + \alpha)n$  and  $|C''| \geq 3(\nu + \alpha)n$  we get that

$$\mathcal{K}_{\text{median}}(x, C') \geq \mathcal{K}_{\text{median}}(x, C'').$$

This then implies that the blob  $\tilde{C}$  in  $L$  of highest median similarity to  $x$  must satisfy  $\tilde{C} \cap G \subseteq C(x) \cap G$ , as desired.  $\blacksquare$

**Lemma 5** *Let  $\mathcal{K}$  be a symmetric similarity function satisfying the  $(\alpha, \nu)$ -good neighborhood property for the clustering problem  $(S, \ell)$ . Assume that  $L$  is a list of clusters each of size at least  $3(\nu + \alpha)n$ . Assume also that each cluster in  $L$  intersects at most a good set; i.e., for any  $C$  in  $L$ , we have  $C \cap G \subseteq G_i$  for some  $i$ . Consider  $x \in G$  such that there is no cluster  $C$  in  $L$  with  $C \cap G \subseteq C(x) \cap G$ . Then at most  $(\alpha + \nu)n$  of its  $t$  nearest neighbors for any  $t \leq n_{C(x)}$  can be in  $L$  and all the rest are outside.*

**Proof:** Since  $\mathcal{K}$  satisfies the  $(\alpha, \nu)$ -good neighborhood property, by Claim 2, for all  $x \in G$ , for all  $t \leq n_{C(x)}$  at most  $(\alpha + \nu)n$  of its  $t$  nearest neighbors are not from  $C(x)$ . Consider  $x \in G$  such that there is no cluster  $C$  in  $L$  with  $C \cap G \subseteq C(x) \cap G$ . So, the list  $L$  contains no good points from  $C$ , which implies that at most  $(\alpha + \nu)n$  of its  $t$  nearest neighbors for any  $t \leq n_{C(x)}$  can be in  $L$ .  $\blacksquare$

### 3.2 A robust linkage procedure

Our linkage procedure is given by Algorithm 3.

---

#### Algorithm 3 A robust linkage procedure

---

**Input:** A list  $L$  of blobs; similarity function  $\mathcal{K}$  on pairs of points.

- Repeat till only one cluster remains in  $L$ :
  - (a) Find clusters  $C, C'$  in the current list which maximize  $\text{score}(C, C')$
  - (b) remove  $C$  and  $C'$  from  $L$  merge them into a single cluster and add that cluster to  $L$ .
- Let  $T$  be the tree with single elements as leaves and internal nodes corresponding to all the merges performed.

**Output:** Tree  $T$  on subsets of  $S$ .

---

We describe in the following the notion of similarity between pairs of blobs used in Algorithm 3.

**Definition 6** *Let  $L = \{A_1, \dots, A_l\}$  be a list of disjoint subsets of  $S$ . For each  $i$ , for each point  $x$  in  $A_i$  we compute  $\mathcal{K}_{\text{median}}(\{x\}, A_j)$ ,  $j \neq i$ , sort them in increasing order, and define  $\text{rank}(x, A_j)$  as the rank of  $A_j$  in the order induced by  $x$ . We define*

$$\text{rank}(A_i, A_j) = \text{median}_{x \in A_i} [\text{rank}(x, A_j)].$$

For example, if  $A_{j_1}$  is the subset of highest median similarity to  $x$  out of all  $A_j$ ,  $j \neq i$ , then  $\text{rank}(x, A_{j_1}) = l$ . Similarly, if  $A_{j_2}$  is the subset of smallest median similarity to  $x$  out of all  $A_j$ ,  $j \neq i$ , then  $\text{rank}(x, A_{j_2}) = 1$ .

**Definition 7** *Let  $L = \{A_1, \dots, A_l\}$  be a list of disjoint subsets of  $S$ . We define the score between  $A_i$  and  $A_j$  as*

$$\text{score}(A_i, A_j) = \min[\text{rank}(A_i, A_j), \text{rank}(A_j, A_i)].$$

Note that while the  $\text{rank}(\cdot, \cdot)$  might be asymmetric,  $\text{score}(\cdot, \cdot)$  is designed to be symmetric. We now present a useful lemma.

**Lemma 8** *Let  $\mathcal{K}$  be a symmetric similarity function satisfying the  $(\alpha, \nu)$ -good neighborhood property for the clustering problem  $(S, \ell)$ . Let  $L$  be a list of disjoint clusters, all of size at least  $2(\alpha + \nu)n$ . Assume that  $B \cap G \subseteq G_i$ ,  $B' \cap G \subseteq G_i$ , and  $(B'' \cap G) \cap G_i = \emptyset$ . Then we have both*

$$\text{score}(B, B') < \text{score}(B, B'') \quad \text{and} \quad \text{score}(B, B') < \text{score}(B', B'').$$

**Proof:** Let  $x$  be a good point. The  $(\alpha, \nu)$ -good neighborhood property implies that there exists  $c_x$  such that at most  $\alpha n$  points  $z \in G$ ,  $z \notin C(x)$  can have similarity  $K(x, z)$  greater or equal to  $c_x$  and at most  $\alpha n$  points  $y \in G \cap C(x)$  can have similarity  $K(x, z)$  strictly smaller than  $c_x$ . Since each of the blobs has size at least  $2(\alpha + \nu)n$  and since each blob contains at most  $\nu n$  bad points, we get that for all blobs  $B'$  and  $B''$  such that  $B' \cap G \subseteq C(x) \cap G$  and  $(B'' \cap G) \cap (C(x) \cap G) = \emptyset$  we have

$$\mathcal{K}_{\text{median}}(\{x\}, B') > \mathcal{K}_{\text{median}}(\{x\}, B'').$$

So a good point  $x$  will rank blobs  $B'$  s.t.  $B' \cap G \subseteq C(x) \cap G$  later than blobs  $B''$  such that  $(B'' \cap G) \cap (C(x) \cap G) = \emptyset$  in the order it induces. Assume that there are exactly  $r$  blobs  $B$  in  $L$  such that  $(B \cap G) \cap (C(x) \cap G) = \emptyset$ . Since there are at most  $\nu n$  bad points and each of the blobs has size at least  $2(\alpha + \nu)n$ , we obtain that for all  $B, B'$  in  $L$  such that  $B \cap G \subseteq C_i \cap G$  and  $B' \cap G \subseteq C_i \cap G$ , and for all  $B''$  in  $L$  with  $(B'' \cap G) \cap (C_i \cap G) = \emptyset$  we have both

$$\text{rank}(B, B') > r \geq \text{rank}(B, B'') \quad \text{and} \quad \text{rank}(B', B) \geq r \geq \text{rank}(B', B'').$$

This then implies that

$$\text{score}(B, B') = \text{score}(B', B) = \min[\text{rank}(B, B'), \text{rank}(B', B)] > r.$$

Similarly,

$$\text{score}(B, B'') = \text{score}(B'', B) = \min[\text{rank}(B, B''), \text{rank}(B'', B)] > r.$$

Finally, we have

$$\text{score}(B', B'') = \text{score}(B'', B') = \min[\text{rank}(B', B''), \text{rank}(B'', B')] \leq r.$$

These imply:

$$\text{score}(B, B') > \text{score}(B, B'') \quad \text{and} \quad \text{score}(B, B') > \text{score}(B', B''),$$

as desired. ■

We now show that is the similarity function we have satisfies the good neighborhood property, given a good starting point, Algorithm 3 will be successful in outputting a good hierarchy.

**Theorem 9** *Let  $\mathcal{K}$  be a symmetric similarity function satisfying the  $(\alpha, \nu)$ -good neighborhood property for the clustering problem  $(S, \ell)$ . Assume that  $L$  is a list of clusters each of size at least  $3(\nu + \alpha)n$  that partition the entire set of points. Assume also that each cluster in  $L$  intersects at most a good set; i.e., for any  $C$  in  $L$ , we have  $C \cap G \subseteq G_i$  for some  $i$ . Then Algorithm 3 constructs a binary tree such that the ground-truth clustering is  $\nu$ -close to a pruning of this tree.*

**Proof:** First note that at each moment the list  $L$  of clusters is a partition of the whole dataset and that all clusters in  $L$  have size at least  $3(\nu + \alpha)n$ . We prove by induction that at each time step the list of clusters restricted to  $G$  is laminar w.r.t.  $\mathcal{C}_G$ .

In particular, assume that our current list of clusters restricted to  $G$  is laminar with respect to  $\mathcal{C}_G$  (which is true at the start). This implies that for each cluster  $C$  in our current clustering and each  $C_r$  in the ground truth, we have either

$$C \cap G \subseteq G(C_r) \quad \text{or} \quad G(C_r) \subseteq C \cap G \quad \text{or} \quad (C \cap G) \cap G(C_r) = \emptyset.$$

Now, consider a merge of two clusters  $C$  and  $C'$ . The only way that laminarity could fail to be satisfied after the merge is if for one of the two clusters, say,  $C'$ , we have that  $C' \cap G$  is strictly contained inside  $C_{r'} \cap G$ , for some ground-truth cluster  $C_{r'}$  (so,  $(C_{r'} \cap G) \setminus (C' \cap G) \neq \emptyset$ ,  $(C' \cap G) \subset C_{r'}$ ) and yet  $C \cap G$  is disjoint from  $C_{r'} \cap G$ . But there must exist  $C''$  in the list  $L$  such that  $(C'' \cap G) \subset C_{r'} \setminus (C' \cap G)$ ,  $|C''| \geq 3(\nu + \alpha)n$ . By Lemma 8 we know that

$$\text{score}(C', C'') > \text{score}(C', C).$$

However, this contradicts the specification of the algorithm, since by definition it merges the pair  $C, C'$  such that  $\text{score}(C', C)$  is greatest. ■

### 3.3 The Main Result

Our main result is the following:

**Theorem 10** *Let  $\mathcal{K}$  be a symmetric similarity function satisfying the  $(\alpha, \nu)$ -good neighborhood property for the clustering problem  $(S, \ell)$ . As long as the smallest target cluster has size greater than  $9(\nu + \alpha)n$ , then we can use Algorithm 1 in order to produce a tree such that the ground-truth clustering is  $\nu$ -close to a pruning of this tree in  $O(n^{\omega+1})$  time, where  $O(n^\omega)$  is the state of the art for matrix multiplication.*

**Proof:** The correctness follows immediately from Theorems 3 and 9. The running time follows from Lemma 14 in Appendix A. ■

### 3.4 The Inductive Setting

In this section we consider an *inductive* model in which  $S$  is merely a small random subset of points from a much larger abstract instance space  $X$ . Based on such a sample, our algorithm outputs a hierarchy over the sample, which also *implicitly* represents a hierarchy of the whole space which is evaluated with respect to the underlying distribution. Let us assume for simplicity that  $X$  is finite and that the underlying distribution is uniform over  $X$ .

Our goal is to design an algorithm that based on the sample produces a tree of small error with respect to the whole distribution. Formally, we assume that each node in the tree derived over the sample  $S$  induces a cluster (a subset of  $X$ ) which is implicitly represented as a function  $f : X \rightarrow \{0, 1\}$ . For a fixed tree  $T$  and a point  $x$ , we define  $T(x)$  as the subset of nodes in  $T$  that contain  $x$  (the subset of nodes  $f \in T$  with  $f(x) = 1$ ). We say that a tree  $T$  has error at most  $\epsilon$  if  $T(X)$  has a pruning  $f_1, \dots, f_{k'}$  of error at most  $\epsilon$ .

---

#### Algorithm 4 Inductive Robust Agglomerative Hierarchical Clustering

---

Input: Similarity function  $\mathcal{K}$ , parameters  $\alpha, \nu, k \in \mathbb{Z}^+$ ;  $n = n(\alpha, \nu, k, \delta)$ ;

- Pick a set  $S = \{x_1, \dots, x_n\}$  of  $n$  random examples from  $X$ .
  - Run Algorithm 1 with parameters  $2\alpha, 2\nu$  on the set  $S$  and obtain a tree  $T$  on the subsets of  $S$ . Let  $Q$  be the set of leaves of this tree.
  - Associate each node  $u$  in  $T$  a function  $f_u$  (which induces a cluster) specified as follows:  
 Consider  $x \in X$ , and let  $q(x) \in Q$  be the leaf given by  $\operatorname{argmax}_{q \in Q} \mathcal{K}_{\text{median}}(x, q)$ ; if  $u$  appears on the path from  $q(x)$  to the root, then set  $f_u(x) = 1$ , otherwise set  $f_u(x) = 0$ .
  - Output the tree  $T$ .
- 

Let  $N = |X|$ . For the rest of this section we assume that the similarity function  $\mathcal{K}$  satisfies the  $(\alpha, \nu)$ -good neighborhood property for the clustering problem  $(X, \ell)$ . Let  $S' \subseteq S$  be the set of  $(1 - \nu)N$  points such that  $\mathcal{K}$  satisfies  $\alpha$ -good neighborhood property with respect to  $S'$ . We call the points in  $S'$  good points and the points in  $S \setminus S'$  bad points. Let  $G_i = C_i \cap S'$  be the good set of label  $i$ . Let  $G = \cup G_i$  the whole set of good points; so  $G = S'$ .

Our main result in this section is the following:

**Theorem 11** *Let  $\mathcal{K}$  be a symmetric similarity function satisfying the  $(\alpha, \nu)$ -good neighborhood property for the clustering problem  $(X, \ell)$ . As long as the smallest target cluster has size greater than  $18(\nu + \alpha)N$ , then using Algorithm 4 with parameters  $\alpha, \nu, k$ , and  $n = \Theta\left(\frac{1}{\min(\alpha, \nu)} \ln \frac{k}{\delta \cdot \min(\alpha, \nu)}\right)$ , we can produce a tree with the property that the ground-truth is  $\nu + \delta$ -close to a pruning of this tree with probability  $1 - \delta$ . Moreover, the size of this tree is  $O\left(\frac{1}{\min(\alpha, \nu)} \ln \frac{k}{\delta \cdot \min(\alpha, \nu)}\right)$ .*

**Proof:** Note that  $n$  is large enough so that with probability at least  $1 - \delta/2$  we have that  $S$  contains at most  $2\nu n$  bad points and  $\mathcal{K}$  satisfies the  $(2\alpha, 2\nu)$ -good neighborhood property with respect to the clustering induced over the sample (by Lemma 12) and each target cluster has at least  $9(\nu + \alpha)n$  points in the sample (by Chernoff bounds).

Assume below that this happens. So, by Theorem 10 we get that the tree  $T$  induced over the sample has error at most  $2\nu$  over the sample. Let  $L$  be the list of leaves of  $T$ . By Theorem 3, we

know that  $L$  forms a partition of  $S$  and that each element of  $L$  has size at least  $6(\nu + \alpha)n$  and it contains good points from only one good set i.e., for any  $C \in L$ ,  $C \cap G \subseteq G_i$  for some  $i \leq k$ . Let us fix a good point  $x$ . By Lemma 13, with probability at least  $1 - \delta^2/2$  at most  $2\alpha n$  of the  $n_{\tilde{C}_G(x)}$  nearest points to  $x$  from  $G \cap S$  can be outside  $C(x) \cap G$ , where  $n_{\tilde{C}_G(x)}$  is  $|C(x) \cap G \cap S|$ . We can show that  $\tilde{C}$  be the blob in  $L$  of highest median similarity to  $x$  satisfies  $\tilde{C} \cap G \subseteq C(x) \cap G$ . To see this, let  $C'$  and  $C''$  in  $L$  be such that  $C' \cap G \subseteq C(x) \cap G$  and  $C'' \cap G \subseteq C_i \cap G$ , for  $C_i \neq C(x)$ . By the above facts we know that  $x$  can be more similar to at most  $2\nu n + 2\alpha n$  points in  $C''$  than with any point in  $C' \cap G$ . Since  $|C'| \geq 6(\nu + \alpha)n$  and  $|C''| \geq 6(\nu + \alpha)n$  we get that

$$\mathcal{K}_{\text{median}}(x, C') \geq \mathcal{K}_{\text{median}}(x, C'').$$

This then implies that the blob  $\tilde{C}$  in  $L$  of highest median similarity to  $x$  must satisfy  $\tilde{C} \cap G \subseteq C(x) \cap G$ , as desired. So, for any given point  $x$ , with probability  $1 - \delta^2/2$  over the draw of the random sample, the leaf in  $T$  of highest median similarity to  $x$  has the property that all its good points are from  $C(x)$ . Since this is true for any  $x$ , by Markov inequality, we get that with probability  $1 - \delta/2$  a  $1 - \delta$  fraction of the good points connect to a leaf that contain good points from their own cluster only.

Adding back the  $\delta/2$  chance of failure due to either  $\mathcal{K}$  not satisfying the  $(2\alpha, 2\nu)$ -good neighborhood property or having more than  $2\nu n$  bad points in  $S$ , we get that with probability  $1 - \delta$  the error rate of the hierarchy implied by  $T$  over the whole set  $X$  is at most  $\nu + \delta$ .  $\blacksquare$

**Lemma 12** *Let  $\mathcal{K}$  be a symmetric similarity function satisfying the  $(\alpha, \nu)$ -good neighborhood property for the clustering problem  $(X, \ell)$ . If we draw a set  $S$  of  $n = \Theta\left(\frac{1}{\min(\alpha, \nu)} \ln \frac{1}{\delta \min(\alpha, \nu)}\right)$ , then with probability  $1 - \delta$ ,  $S$  contains at most  $2\nu n$  bad points and the similarity function  $\mathcal{K}$  satisfies the  $(2\alpha, 2\nu)$ -good neighborhood property with respect to the target clustering restricted to the sample  $S$ .*

**Proof:** Since  $n \geq \frac{3}{\nu} \ln \frac{2}{\delta}$ , by Chernoff bounds, we have that with probability  $1 - \delta/2$  at most  $2\nu n$  bad points fall into the sample. Let us fix a good point  $x$  in  $S$  and let us denote by  $n_{\tilde{C}_G(x)}$  the number of points in  $C(x) \cap G \cap S$ . By Lemma 13 we have that for  $n = \Theta\left(\frac{1}{\alpha} \ln \frac{n}{\delta}\right)$ , with probability at least  $1 - \delta/(2n)$  (over the draw of the other points in the sample) we have that all but  $2\alpha n$  of the  $n_{\tilde{C}_G(x)}$  nearest neighbors of  $x$  from  $S \cap G$  are points from the set  $C(x) \cap G$ . By union bound over all points  $x$  in  $S$  we have that simultaneously for all good points  $x$  in  $S$ , all but  $2\alpha n$  of their  $n_{\tilde{C}_G(x)}$  nearest neighbors in  $S \cap G$  come from  $C(x) \cap G$ .

These together imply that if  $n = \Theta\left(\frac{1}{\min(\alpha, \nu)} \ln \frac{n}{\delta}\right)$ , then with probability  $1 - \delta$  at most  $2\nu n$  bad points fall into the sample and the similarity function  $\mathcal{K}$  satisfies the  $(2\alpha, 2\nu)$ -good neighborhood property with respect to the target clustering restricted to the sample  $S$ . Using the inequality  $\ln x \leq \alpha x - \ln \alpha - 1$  for  $\alpha, x > 0$ , we then get the desired result.  $\blacksquare$

**Lemma 13** *Let  $\mathcal{K}$  be a symmetric similarity function satisfying the  $(\alpha, \nu)$ -good neighborhood property for the clustering problem  $(X, \ell)$ . Consider  $x \in G$ . If we draw a set  $S$  of  $n = \Theta\left(\frac{1}{\alpha} \ln \frac{1}{\delta}\right)$  random points from  $X$ , then with probability at least  $1 - \delta$  we have that at most  $2\alpha n$  of the  $n_{\tilde{C}_G(x)}$  nearest points to  $x$  from  $G \cap S$  can be outside  $C(x) \cap G$ , where  $n_{\tilde{C}_G(x)}$  is  $|C(x) \cap G \cap S|$ .*

**Proof:** Let  $C_G(x)$  denote  $C(x) \cap G$  and let  $n_{C_G(x)} = |C(x) \cap G|$ . Let us define  $NN(x)$  to be the nearest  $n_{C_G(x)}$  points to  $x$  in  $G$ . Since  $\mathcal{K}$  satisfies the  $(\alpha, \nu)$ -good neighborhood property for the clustering problem  $(X, \ell)$  we have:

$$Pr_{z \sim X}[z \in NN(x) \setminus C_G(x)] \leq \alpha.$$

Since  $NN(x)$  and  $C_G(x)$  have the same size, this is equivalent to the statement:

$$Pr_{z \sim X}[z \in C_G(x) \setminus NN(x)] \leq \alpha.$$

So, by Chernoff bounds applied to both of the above, with probability at least  $1 - \delta$  we have that at  $2\alpha n$  points are in  $(NN(x) \setminus C_G(x)) \cap S$  and at most  $2\alpha n$  points are in  $(C_G(x) \setminus NN(x)) \cap S$ .

We now argue that at most  $2\alpha n$  of the  $n_{\tilde{C}_G(x)}$  nearest points to  $x$  in  $G \cap S$  can be outside  $C(x) \cap G$ , where  $n_{\tilde{C}_G(x)} = |C(x) \cap G \cap S|$ . Let  $n_1$  be the number of points in  $(NN(x) \setminus C_G(x)) \cap S$ . Let  $n_2$  be the number of points in  $(C_G(x) \setminus NN(x)) \cap S$ . Let  $n_3$  be the number of points in  $(C_G(x) \cap NN(x)) \cap S$ . By construction, we have

$$n_{\tilde{C}_G(x)} = n_2 + n_3,$$

and we are given that  $n_1, n_2 \leq 2\alpha n$ . We now distinguish two cases.

The first case is  $n_1 \geq n_2$ . In this case we have

$$n_1 + n_3 \geq n_2 + n_3 = n_{\tilde{C}_G(x)}.$$

This implies that the nearest  $n_{\tilde{C}_G(x)}$  points to  $x$  in  $G \cap S$  all lie inside  $NN(x)$ , since by definition all points inside  $NN(x)$  are closer to  $x$  than any point in  $G$  outside  $NN(x)$ . Since we are given that at most  $n_1 \leq 2\alpha n$  of them can be outside  $C_G(x)$ , we get that at most  $2\alpha n$  of the  $n_{\tilde{C}_G(x)}$  nearest neighbors of  $x$  are not from  $C_G(x)$ , as desired.

The second case is  $n_1 \leq n_2$ . This implies that the nearest  $n_{\tilde{C}_G(x)}$  good points to  $x$  in the sample include *all* the points in  $NN(x)$  in the sample, plus possibly some others too. But this implies in particular that it includes all the  $n_3$  points in  $C_G(x) \cap NN(x)$  in the sample. So, it can include at most

$$n_{\tilde{C}_G(x)} - n_3 \leq 2\alpha \cdot n$$

points not in  $C_G(x) \cap NN(x)$ , and even if all those are not in  $C_G(x)$ , it is still  $\leq 2\alpha n$ ; so at most  $2\alpha n$  of the  $n_{\tilde{C}_G(x)}$  nearest neighbors of  $x$  are not from  $C_G(x)$ , as desired. ■

**Note:** Note that if we are willing to lose a bit in the accuracy the analysis in this section allows us to speed up the algorithm in Theorem 10.

## 4 Conclusions

In this paper we propose and analyze a new robust algorithm for bottom-up agglomerative clustering. We show that our algorithm can be used to cluster accurately in cases where the data satisfies a number of natural properties and where the traditional agglomerative algorithms fail. We also show how to adapt our algorithm to the inductive setting where our given data is only a small random sample of the entire data set.

It would be interesting to see if our algorithmic approach can be shown to work for other natural properties. For example, it would be particularly interesting to analyze a relaxation of the max stability property in [2] which was shown to be a necessary and sufficient condition for single linkage, or the average stability property which was shown to be a sufficient condition for average linkage.

**Acknowledgments:** We thank Avrim Blum for numerous useful discussions. This work was supported in part by NSF grant CCF-0953192.

## References

- [1] M. F. Balcan, A. Blum, and A. Gupta. Approximate clustering without the approximation. In *ACM-SIAM Symposium on Discrete Algorithms*, 2009.
- [2] M. F. Balcan, A. Blum, and S. Vempala. A discriminative framework for clustering via similarity functions. In *40th ACM Symposium on Theory of Computing*, 2008.
- [3] M. F. Balcan, H. Roeglin, and S. Teng. Agnostic clustering. In *The 20th International Conference on Algorithmic Learning Theory*, 2009.
- [4] M. Blum, R. W. Floyd, V. R. Pratt, R. L. Rivest, and R. Endre Tarjan. Linear time bounds for median computations. In *Proceedings of the fourth Annual ACM symposium on Theory of computing*, 1972.
- [5] D. Bryant and V. Berry. A structured family of clustering and tree construction methods. *Advances in Applied Mathematics*, 27(4):705–732, 2001.
- [6] D. Cheng, R. Kannan, S. Vempala, and G. Wang. A divide-and-merge methodology for clustering. *ACM Trans. Database Syst.*, 31(4):1499–1525, 2006.
- [7] D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. In *Proceedings of the 19-th Annual ACM conference on Theory of computing*, 1987.
- [8] S. Dasgupta and P. Long. Performance guarantees for hierarchical clustering. *Journal of Computer and System Sciences*, 70(4):555 – 569, 2005.
- [9] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley, 2000.
- [10] S. Gollapudi, R. Kumar, and D. Sivakumar. Programmable clustering. In *Symposium on Principles of Database Systems*, pages 348 – 354, 2006.
- [11] A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data*. Prentice Hall, 1988.
- [12] A. K. Jain, M. N. Murthy, and P. J. Flynn. Data clustering: A review. In *ACM Computing Reviews*, 1999.

- [13] S. C. Johnson. Hierarchical clustering schemes. *Psychometrika*, 32(3):241–254, 1967.
- [14] M. Narasimhan, N. Jojic, and J. Bilmes. Q-clustering. In *Neural Information Processing Systems*, 2005.

## A Run Time Analysis

**Lemma 14** *Algorithm 1 has a running time of  $O(n^{\omega+1})$ , where  $O(n^\omega)$  is the state of the art for matrix multiplication. (The current value of  $\omega$  is 2.376 [7]).*

**Proof:** We analyze the running times of Algorithms 2 and 3 separately. For Algorithm 2, we also discuss certain data structures which are utilized throughout the algorithm for speed up.

In the preprocessing step, we construct a list of nearest neighbors for each point in  $S$  by sorting  $n - 1$  other points in decreasing order of similarity. This takes  $O(n \log n)$  time for each point and thus the entire preprocessing step costs  $O(n^2 \log n)$  time.

Now, think of a directed  $t$ -regular graph  $E_t$ , where, for each point  $j$  in the  $t$  nearest neighbors of a point  $i$ , there is a directed edge from  $i$  to  $j$  in  $E_t$ . We construct two  $n \times n$  matrices  $Adj^E$  and  $Nbrs^E$ .  $Adj^E$  is the adjacency matrix for the graph  $E_t$  and  $Nbrs^E = Adj^E \times (Adj^E)^T$ .  $Nbrs^E_{ij}$  gives the number of common neighbors between  $i$  and  $j$  in  $E_t$  and from this, we can know whether to draw an edge between them in  $F_t$ . Constructing  $Nbrs^E$  for the first time takes  $O(n^\omega)$  time. Notice, however, we do not require to recompute  $Nbrs^E$  from scratch in every iteration over  $t$  as the graph  $E_t$  is monotonically increasing. In iteration  $t + 1$ , for each point  $i$  exactly one new edge is added to another point  $k$  in the graph  $E_t$  where  $k$  is the  $(t + 1)$ th nearest neighbor of  $i$ . If there was already an edge from  $k$  to  $i$  in  $E_t$ , the value  $Nbrs^E_{ik}$  (and  $Nbrs^E_{ki}$ ) increases by 1. For every point  $j$  that has a directed edge to  $k$ ,  $Nbrs^E_{ij}$  (and  $Nbrs^E_{ji}$ ) increases by 1. This can be computed by comparing the row  $Adj^E_i$  of  $Adj^E$  with column  $Adj^E_k$  of  $Adj^E$ . This requires  $O(n)$  time for each point  $i$ . Since there are  $n$  such points the total cost of the iteration is  $O(n^2)$ . There can be a maximum of  $O(n)$  such iterations. Hence, the total cost of updating  $Nbrs^E$  over all iterations is  $O(n^3)$ . Now, let us consider the case when a new blob of size at least  $3(\nu + \alpha)n$  is found and added to  $L$ . We remove this blob from  $A_s$  and correspondingly all the points in the blob from  $Adj^E$  and  $Adj^E$ . For each such removal we need to recompute  $Adj^E$  and  $Nbrs^E$ . There can be at most  $n/3(\nu + \alpha)n = 1/3(\nu + \alpha)$  such iterations. Thus, we take at most  $O(n^\omega/(\nu + \alpha))$  time. Therefore, the total cost of constructing the graph  $F_t$  over all iterations is  $O(n^\omega(1/(\nu + \alpha) + n^{3-\omega}))$ .

Similarly, let us have two  $n \times n$  matrices  $Adj^F$  and  $Nbrs^F$ , where  $Adj^F$  is the adjacency matrix of the undirected graph  $F_t$  and  $Nbrs^F = Adj^F \times Adj^F$  (notice,  $F_t$  is undirected). Note that the same trick does not hold while constructing the graph  $H_{t+1}$  from  $H_t$  as the graph  $F_t$  is not monotonically increasing. e.g. two points  $x$  and  $y$  which had an edge in  $F_t$  as they had exactly  $t - 2(\nu + \alpha)n$  neighbors in common may not have an edge in  $F_{t+1}$  any more as they might not have  $t + 1 - 2(\nu + \alpha)n$  neighbors in common. Thus, for each iteration we need to recompute the matrix  $Nbrs^F$ . Thus, to construct  $H_t$ , it takes a total of  $O(n^{\omega+1})$  time over all iterations.

In Step 3(i), we can find all the components of  $H_t$  in at most  $O(|V_{H_t}| + |E_{H_t}|)$  time where  $|V_{H_t}| = n$  and  $|E_{H_t}| = O(n^2)$ . Thus, we can do this in at most  $O(n^2)$  time. Now let's look at the Step 3(ii). It is easy to see that we do this Step at most  $1/3(\nu + \alpha)$  times, i.e. once for each new blob. For the first stage of this Step, we maintain a set  $P_i$  for each point  $i$  that is a set of points from its first  $5(\nu + \alpha)$  neighbors that are not yet in the list  $L$  and a count  $c_i$  of the points already present in  $L$ . Every time a new blob  $B$  is added to  $L$  or we get a non-empty set  $T$  (construction explained below) after an iteration of Step 3(ii), we check for the common points between  $P_i$  and  $B$  or  $T$ , then remove all such points from  $P_i$  and add the number of these points to  $c_i$ . Since, there can be at most  $O(n)$  such iterations, this step can take  $O(n^3)$  time over all iterations for all points. For each point  $x$  that makes it to the next stage of Step 3(ii), we compute the median to each blob  $B$  present in  $L$ . This can be done in  $O(|B|)$  time for each blob [4]. For all the blobs, this can be done in  $\sum_{B_i \in L} O(|B_i|) \leq O(n)$  time. Once we have the median for all blobs, we can find the highest of these ( $\#blobs \leq 1/3(\nu + \alpha)$ ) medians which requires at most  $O(1/(\nu + \alpha))$  time. Notice, that once a point  $x$  gets to this stage, it will be added to one of the blobs in  $L$  and not be considered again, which means we hit this stage at most  $O(n)$  times. Thus, over all iterations, for all points, this stage of Step 3(ii) can be done in  $O(n^2)$  time. Notice, we need to do an additional post-processing step. We add all such  $x$  to a set  $T$ . This set  $T$  is compared with  $P_i$  (as explained above) at the beginning of the next iteration. The addition of these points to  $L$  could result in more points satisfying the first condition of Step 3(ii). Therefore, the time required for Step 3 is  $O(n^3)$ .

The costliest Steps in the Algorithm 2 is Step 2 and causes the algorithm to have  $O(n^{\omega+1})$  running time.

Algorithm 3 has a running time of  $O(n/(\nu + \alpha)(n + 1/(\nu + \alpha) \log(1/(\nu + \alpha))))$ . At each level, we compute the score of each cluster with respect to all other clusters. Firstly, we compute the median similarity of each cluster  $C_i$  with respect to a point  $x$  in  $O(|C_i|)$  time [4]. This is done for all clusters except  $C(x)$ . The total running time is  $\sum_{C_i \in L: C_i \neq C(x)} O(|C_i|) < O(n)$ . Secondly, we compute the rank by sorting these similarity values in ascending order. Since there can be at most  $1/3(\nu + \alpha)$  clusters this can be done in  $O(1/(\nu + \alpha) \log 1/(\nu + \alpha))$  time. This is done for all points, hence the total cost of computing ranks of all clusters with respect to all points is  $O(n(n + 1/(\nu + \alpha) \log(1/(\nu + \alpha))))$ . Finally, we compute the rank for a cluster  $C_i$  with respect to the cluster  $C_j$  by taking the median of the ranks given by each point  $x \in C_j$  to the cluster  $C_i$  and this takes  $O(|C_j|)$  time. This is done by  $C_j$  for all other clusters. Thus, the total time taken by  $C_j$  to compute ranks for all other clusters is  $O(|C_j|/(\nu + \alpha))$ . Since we do it for all clusters it requires  $\sum_{C_j \in L} O(|C_j|/(\nu + \alpha)) \leq O(n/(\nu + \alpha))$  time. Thus, we can be done with each level in  $O(n(n + 1/(\nu + \alpha) \log(1/(\nu + \alpha))))$  time. The number of levels is at most the number of clusters which is  $O(1/(\nu + \alpha))$ . Therefore, the total running time of the algorithm is  $O(n/(\nu + \alpha)(n + 1/(\nu + \alpha) \log(1/(\nu + \alpha))))$ .

For Algorithm 1, the costliest step is Algorithm 2 and thus has a running time of  $O(n^{\omega+1})$ . ■