Efficient Clustering Algorithms for Computing Better Local Optima

Ainesh Bakshi
Department of Computer Science
Rutgers University
New Brunswick, NJ 08901
ainesh.bakshi@rutgers.edu

Pranjal Awasthi
Department of Computer Science
Rutgers University
New Brunswick, NJ 08901
pranjal.awasthi@cs.rutgers.edu

Abstract

Clustering with most objective functions is NP-Hard, even to approximate well. However, in many settings it is often sufficient to find a good locally optimal solution than a true global one. This is evident from the vast empirical success of heuristics such as the k-means algorithm. In this paper, we formally initiate study of the problem of efficiently computing a good local optimum to a non-convex problem. We study this in the context of clustering and design polynomial time algorithms that can indeed achieve the desired output. Furthermore, our model naturally takes into account the fact that often there is uncertainty in choosing the right value of k. We design algorithms that can output a good locally optimal solution even when the value of k is unknown.

1 Introduction

Clustering is one of the most fundamental and widely studies problems in machine learning and data mining. It has numerous applications and is often the first step towards more fine grained machine learning tasks such as classification. The most popular approach towards clustering is to formulate it as the optimal solution of an optimization problem such as k-median and k-means clustering [14, 10]. Within this framework various algorithms exist to approximately optimize these objective. These range from approximation algorithms with provable guarantees on the quality of the optimal solution, to fast and popular heuristics such as the k-means method that often work well on real world data [10, 1, 13, 15, 11]. The motivation for our work comes from the striking fact that in almost all application of clustering, heuristics such as the k-means method are preferred over more sophisticated methods that have provable guarantees. This suggests that often in machine learning, even though we are optimizing a non-convex objective to solve our problem, in most cases a good local optimum is good enough. For instance it is easy to show that the k-means method always returns a local optimum [13]. This fact has also observed in recent work on training deep networks where the networks that generalize well often converge to a good local optimum during the training phase [8]. Typically, a non convex problem has many local optima and if one's goal is to find a "good" one, can we design algorithms to address this question? This is the problem that we consider in this paper:

Can one design efficient algorithms for finding a good local optimum?

We study the above problem in the context of clustering and provide efficient algorithms. To the best of our knowledge, ours is the first work to directly address this problem. We believe that the above question is extremely relevant in many settings and our work will lead to further research in designing algorithms for the above setting. In order to formally answer the question, we need to define what it means to be a good local optimum. In this paper we work with a natural notion defined in Section 2. For clustering problems, a local optimum is one in which every point prefers its own center over any other center. This is also known as a *Voronoi* partitioning [9]. We define a good local optimum to be the one in which every point prefers its own center over any point outside the cluster. This is a

natural notion of clustering stability that has been studied previously. We show how to efficiently recover such good local optima.

The case of unknown k: We also address a common issue in clustering problems that concerns the choice of k, the total number of clusters. In many settings it is often unclear to know in advance how many clusters the data has. However, it is often easier to prescribe a minimum cluster size such that any clustering algorithm should output clusters of at least that size. This captures the intuition that very small clusters do not capture a global property of the data. Our framework of good local solutions can naturally handle such settings. In other words, given just a minimum cluster size and no bound on the number of clusters, our algorithm can output a good local solution (provided one exists) that meet the minimum size requirement.

1.1 Related Work

Recently there has been a significant interest in understanding properties of real data that facilitate design of optimal algorithms for clustering problems. These properties boil down to assuming that the centers in the optimal solution are well separated and points prefer their own center to other centers by a noticeable amount [7, 4, 2, 3, 6]. The works most closely related to us are that of Balcan et. al [4] and Awasthi et. al [2] on approximation stability. They show that if a clustering instance has the property that any good approximate solution is close to the optimal solution, structurally, then one can indeed recover a near optimal solution in polynomial time. Another closely related work is on that of perturbation resilience [3, 6]. Here one can show that if the instance is stable enough to perturbations in the distance metric, then one can again solve for the optimal k-median/k-means solutions. A different line of work studies conditions under which popular heuristics such as k-means can lead to optimal or near optimal solutions [15, 12]. Our definition of a good local solution is weaker than the data properties studies in these works. However, our end goal is also different in the sense that we are not looking for an optimal solution, but only a local optimum. The case of clustering with unknown k has also been studied in the clustering literature before. Typically this is handled via hierarchical clustering followed by an interactive process involving humans to pin down the right clustering [5].

2 Notation and Preliminaries

A clustering instance (S, d) is defined as a set S of n distinct points, where d is a distance metric. Given a clustering instance and an objective function Φ, the ground truth clustering is defined as a list of optimal clusters such that the overall cost, Φ is minimized. The optimal clusters are denoted by $\mathbb{C}_1^*, \mathbb{C}_2^*, ...\mathbb{C}_k^*$ and their centers are denoted by $c_1^*, c_2^*, ...c_k^*$. The notion of a good locally optimal solution that we study is the following

Definition 2.1 (Good Locally Optimal Solution)**.** Every point is closer to the center of its optimal cluster than to any point outside. $\forall p \in \mathbb{C}_j^*, \forall q \notin \mathbb{C}_j^*$:

$$d(p, c_j^*) < d(p, q)$$

This is a natural and desirable clustering property that is implied by many previously studied clustering notions such as approximation stability and perturbation resilience [4, 3]. Given a clustering, we say that a cluster $\mathbb{C}_i$ is *stable* if every point in $\mathbb{C}_i$ satisfies the above property. In order to obtain our technical results we also assume that each cluster has enough points that "behave" like centers. We call these the *good* points. For every cluster $\mathbb{C}_i$ the good points are the closest $\frac{|\mathbb{C}_i|}{2}$ points to the center. As mentioned before we assume that they behave like center in the following sense

Property 2.1. *Every point in an optimal cluster is closer to all its good points than to any good point of a different optimal cluster.* $\forall p \in \mathbb{C}_j^*, \forall g_j \in \mathbb{G}_j^*$,

$$d(p, g_j) < d(p, g_k)$$

Below we state our main results

Theorem 2.1. *Suppose we are given a set of n points such that there exists a good locally optimal clustering in which the size of each cluster is at least $r(n)$. Furthermore, assume that this local solution is minimal in the sense that no cluster cannot be broken down into sub clusters of size greater*

than $r(n)$ such that the resulting clustering is still locally good. Then there exists polynomial time algorithm that recovers a good locally optimal solution.

Theorem 2.2. *Given k and an instance guaranteed to have a locally good k clustering, there exits a polynomial time algorithm that outputs a set of k clusters $\mathbb{C}_1$, $\mathbb{C}_2$, ... $\mathbb{C}_k$, such that the resulting is locally good.*

In the next two sections, we outline the algorithms and the analysis behind our main results. We first consider the case when k is unknown.

3 Algorithm and Analysis for Theorem 2.1

In this section, while describing the algorithm, we introduce the following notation :

Definition 3.1. (CandidateClusters) $CandidateClusters$ at the i^{th} iteration are defined a set of clusters where each cluster consists of a clique of size $i/2$ around the center and the center has $deg \geq i$.

Since we want the clusters we extract at the i^{th} iteration to be laminar w.r.t. the clusters extracted previously, we define :

Definition 3.2. (StableLaminarClusters) $StableLaminarClusters$ at the i^{th} iteration are defined as a subset of $CandidateClusters$, where each cluster is laminar and stable with respect to $StableLaminarClusters$ extracted prior to the i^{th} iteration.

We also define $LaminarSubclusters$ at the i^{th} iteration to be the union of the $StableLaminarClusters$ formed prior to the i^{th} iteration.

As opposed to a standard clustering instance, we consider the case where k is not explicitly known. In this setting, the input is a set S consisting of n points and a constant $r(n)$, such that there exists a locally optimal solution as defined above where each cluster has size at least $r(n)$. We also assume that any cluster has no subset of size greater than $r(n)$, such that the subset is stable.

For ease of notation we refer to clusters in this locally good solution as optimal clusters. The algorithm we present is an extension of the ball link algorithm presented by Liang and Balcan [6]. The linkage algorithm we use connects two points p and q, in the i^{th} iteration, if $\mathbb{B}_p \cap \mathbb{B}_q \neq \phi$, where $\mathbb{B}_p$ and $\mathbb{B}_q$ is the set of the $i/2$ closest point to p and q respectively.

Intuitively, the linkage is determined by a point's nearest neighbors. The most straightforward case for the algorithm we present is when all the clusters are of the same size. In the i^{th} iteration, where $i = |\mathbb{C}_j^*|$, the center of each optimal cluster forms a clique with its closest $\mathbb{C}_j^*/2$ points. Further, each point in an optimal cluster is connected to the center and no point from another optimal cluster is connected to the center. Therefore, it is easy to identify and extract optimal clusters that are stable.

However, in the general case, the linkage procedure only guarantees that no edge with an optimal cluster of larger cardinality will be formed. In order to ensure than no edge with a smaller optimal cluster is formed, we introduce laminarity and stability checks. These checks prune out the smaller clusters that may have erroneously formed edges with optimal clusters currently being extracted. The assumption that optimal clusters don't contain a subset of size greater than $r(n)$ keeps them intact and prevents them from being pruned out by the stability check.

The main routine of the algorithm loops from $r(n)$ to n, and at each iteration, generates a graph using the linkage procedure. From this graph, it extracts clusters that are stable and laminar w.r.t the previous iteration.

The BallLinkage procedure in line 3 of Algorithm 1 is a straight forward linkage algorithm. For every iteration i, an edge is created between two points x, y if the closest $i/2$ points to x and y have a nonzero intersection.

3.1 Analysis

We begin by showing an interesting property of the linkage procedure in the i^{th} iteration, where i is the size of an optimal cluster. The center of the optimal cluster forms a clique with it's closest $i/2$ points and every point in the optimal cluster forms an edge with the center.

Algorithm 1: Main

Input: Set of all points S, constant r(n)

i $\leftarrow$ r(n);
while $i \leq n$ **do**
  $G \leftarrow$ BallLinkage(i);
  $StableLaminarClusters_i \leftarrow$ ExtractStableLaminarClusters(G, i);
  $StableLaminarClusters_i \leftarrow StableLaminarClusters_i \cup StableLaminarClusters$;
  i $\leftarrow$ i +1 ;
end

Algorithm 2: ExtractStableLaminarClusters

Input: Graph G, Current Iteration i

/* Extract Candidate Clusters */;
for $x \in S$ **do**
  $\mathbb{B}_x \leftarrow$ {closest $i/2$ points to x} $\cup$ { x };
  if $\mathbb{B}_x$ *forms a clique of size* $i/2$ *and deg(x)* $\geq i$ **then**
    $CandidateClusters \leftarrow$ {x} $\cup$ Neighbors(x) ;
  end
end
/*Laminars clusters that have already been extracted */;
$LaminarSubclusters \leftarrow StableLaminarClusters_{i-1}$;
/* Extract Stable Laminar Clusters from Candidate Clusters */;
for $CandidateCluster \in CandidateClusters$ **do**
  for $LaminarSubcluster \in LaminarSubclusters$ **do**
    $c_{lsc} \leftarrow$ Center($LaminarSubcluster$);
    /* Stability check for subsets of Candidate Cluster */;
    if $LaminarSubcluster \subset CandidateCluster$ **then**
      if $\forall p \in$ *LaminarSubcluster,* $\forall q \notin$ *LaminarSubcluster,* $d(p, c_{lsc}) < d(p, q)$ **then**
        $CandidateCluster \leftarrow CandidateCluster \setminus LaminarSubcluster$
      end
    else
      /* Laminarity Check for Candidate Cluster */;
      $CandidateCluster \leftarrow CandidateCluster \setminus LaminarSubcluster$
    end
  end
  $c \leftarrow$ Center($CandidateCluster$);
  if $\forall p \in$ *CandidateCluster,* $\forall q \notin$ *CandidateCluster,* $d(p, c) < d(p, q)$ **then**
    Extract CandidateCluster;
  end
end

Lemma 3.1. *Let $\mathbb{C}_j^*$ be a an optimal cluster and c_j^* be it's center. At $i = |\mathbb{C}_j^*|$, $\mathbb{B}_{c_j^*}$ forms a clique and there exists an edge p-c_j^*, $\forall p \in \mathbb{C}_j^*$.*

Proof. Let x = c_j^* and y $\in \mathbb{B}_{c_j^*}$. Assume, for sake of contradiction x,y is not an edge. Then $\mathbb{B}_x \cap \mathbb{B}_y = \phi$. By definition, y $\in \mathbb{B}_x$ and therefore the intersection is non zero. This is a contradiction. Therefore, there is an edge from c_j^* to every other point in $\mathbb{B}_{c_j^*}$. Let x,y $\in \mathbb{B}_{c_j^*} \setminus c_j^*$. Assume, for sake of contradiction x,y is not an edge. Then $\mathbb{B}_x \cap \mathbb{B}_y = \phi$. There are two cases, $c_j^* \notin \mathbb{B}_x \cup \mathbb{B}_y$ or $c_j^* \in \mathbb{B}_x \cup \mathbb{B}_y$. Consider the first case. Then, $\mathbb{B}_x$ or $\mathbb{B}_y$ contain a point $\notin \mathbb{C}_j^*$. Wlog, let z $\in \mathbb{B}_x$ such that z $\notin \mathbb{C}_j^*$. Then, x is closer to z than to c_j^* which is a contradiction, since Definition 2.1 is violated. Now consider the second case. Wlog, let $c_j^* \in \mathbb{B}_x$. Then, $\mathbb{B}_y$ picked a point $\notin \mathbb{C}_j^*$ before it picked c_j^*. This is a contradiction. Hence, $\mathbb{B}_{c_j^*}$ forms a clique.

Now consider the case where y $\in \mathbb{C}_j^* \setminus \mathbb{B}_{c_j^*}$. Assume for sake of contradiction, the edge x,y doesn't exist. Then, $\mathbb{B}_x \cap \mathbb{B}_y = \phi$. Intuitively, the ball around x consists of atleast half the points in $\mathbb{C}_j^*$. Since y is disjoint, it must contain a point not in $\mathbb{C}_j^*$. This implies that y is closer to a point $\notin \mathbb{C}_j^*$ than to c_j^*. This is a contradiction as it violates Definition 2.1. Hence, there is an edge between all points in $\mathbb{C}_j^*$ and it's center. □

Lemma 3.2. *No cluster of size smaller than $r(n)$ is extracted by the Algorithm.*

Proof. The algorithm begins at $i = r(n)$. Therefore, any candidate cluster formed has to be of size at least $r(n)$. Also, in the first iteration of the algorithm no point is pruned out since there are no $LaminarSubclusters$ extracted at a previous iteration. During any other iteration, if an optimal cluster is formed, it is completely extracted since it cannot be broken down in to stable clusters of size $> r(n)$. Therefore, no cluster of size $< r(n)$ would be extracted. □

Thus far we have showed that at the i^{th} iteration of Algorithm 1, all points in an optimal cluster form an edge with the center and any cluster that is extracted has to be atleast of size $r(n)$. However, the complete the argument, we need to show that at $i = \mathbb{C}_j^*$, no other cluster forms an edge with c_j^* and the clusters we extract are stable. We also have to argue that at any other i, only pure clusters are extracted and laminarity always holds. We give an inductive proof for Theorem 2.1.

In order to prove Theorem 2.1 by induction, we will first assume the inductive hypothesis is true, show that Theorem 2.1 is implied by it and then prove the inductive hypothesis. The inductive hypothesis is that at any iteration i, all optimal clusters smaller than i have been extracted. Using this, we show that the algorithm terminates, all points have been extracted and the extracted clusters are stable.

Proof. It is clear from Algorithm 1 that it terminates after n iterations. Let us assume a set of points C was not extracted after the algorithm terminated. Let p be any point in C. Then p must belong to some optimal cluster $\mathbb{C}_j^*$. At i = $|\mathbb{C}_j^*|$, p must have been removed from the part of $\mathbb{C}_j^*$ that was extracted. However, only clusters that have previously been extracted are pruned. Hence p would have to belong to cluster that was extracted at a previous iteration. Therefore, the algorithm terminates and all the points in S get extracted. Lines 25-26 of Algorithm 2 ensure that any cluster extracted is stable w.r.t. Definition 2.1. This completes the proof of Theorem 1. □

Next, we prove the inductive hypothesis : At any i, all optimal clusters $\mathbb{C}_j^*$, such that $|\mathbb{C}_j^*| \leq i$ have been extracted. We need to show that this holds for $i + 1$. There are two cases to consider, there is atleast one optimal cluster $\mathbb{C}_k^*$ such that $i + 1 = |\mathbb{C}_k^*|$ or there is no optimal cluster of cardinality $i + 1$. We prove the following lemmas to assist in the proof.

Lemma 3.3. *At $i = |\mathbb{C}_j^*|$, where $\mathbb{C}_j^*$ is a stable cluster, the algorithm extracts $\mathbb{C}_j^*$.*

Proof. By lemma 3.2, $\mathbb{B}_{c_j^*}$ forms a clique and c_j^* has a degree $\geq |\mathbb{C}_j^*|$. Assuming the cluster extracted is not pure, there exists at least one impure edge with another optimum cluster $\mathbb{C}_k^*$. Consider the case where $|\mathbb{C}_k^*| > |\mathbb{C}_j^*|$. Let x = c_j^* and y $\in \mathbb{C}_k^*$ such that edge x-y exists. Therefore, $|\mathbb{B}_x \cap \mathbb{B}_y| \geq 1$. Also, $\mathbb{B}_x \subset \mathbb{G}_j^*$. This implies y is closer to a good point in $\mathbb{C}_j^*$ than a good point in $\mathbb{C}_k^*$. This is a contradiction as it violates Property 2.1.
Now consider the other case where $|\mathbb{C}_k^*| \leq |\mathbb{C}_j^*|$. By the inductive hypothesis, $\mathbb{C}_k^*$ has already been completely extracted. If only a subset of $\mathbb{C}_k^*$ is present, the laminary check prunes it out. If entire $\mathbb{C}_k^*$ is present, the stablity check prunes it out. Therefore, the algorithm extracts exactly $\mathbb{C}_j^*$. □

Lemma 3.4. *$\forall$ i, j, s.t. $i \leq |\mathbb{C}_j^*|$, if an extracted cluster is centered at c_x, s.t. $c_x \in \mathbb{C}_j^*$, there is no impure edge c_x-y, where y belongs to another optimal cluster.*

Proof. Let us assume otherwise. Let the current extracted cluster be $\mathbb{C}_x$, centered at c_x, and y $\in \mathbb{C}_k^*$. Consider the case where $|\mathbb{C}_k^*| < i$. Now, either c_k^* is connected to c_x or c_k^* isn't connected to c_x. For the first case, c_k^* is the center of a cluster that we have extracted, since $|\mathbb{C}_k^*| < i$. This cluster cannot be partially included, since laminarity holds. Therefore, it forms a $LaminarSubcluster$ such that it satisfies Definition 2.1. Therefore, the $LaminarSubcluster$ is pruned and the edge $c_x - c_k^*$ is removed, reducing to the second case. In the second case, we reject the cluster since $d(y, c_k^*) < d(y, c_x)$. Therefore, there is no impure edge to a smaller cluster.

Consider the case where $|\mathbb{C}_k^*| \geq i$. Now, either c_k^* is connected to c_x or c_k^* isn't connected to c_x. For the first case, $|\mathbb{B}_{c_x} \cap \mathbb{B}_{c_k^*}| \geq 1$. But $\mathbb{B}_{c_k^*} \subseteq \mathbb{G}_{c_k^*}$. Therefore, c_x is closer to a good point $\mathbb{C}_k^*$ than all good points in $\mathbb{C}_j$. This is a contradiction as it violates Property 2.1. In the second case, $d(y, c_k^*) < d(y, c_x)$ and stability check rejects this cluster. Therefore there is no impure edge in any extracted cluster centered in a cluster that hasn't already been extracted. □

To complete the proof by induction, we consider the following cases:

Proof. **Case 1:** There exists at least one optimal cluster, $\mathbb{C}_k^*$, such that $i + 1 = |\mathbb{C}_k^*|$. By lemma 3.4, we know whatever part of $\mathbb{C}_k^*$ that has been extracted till now is pure. By lemma 3.3, at $i + 1$, $\mathbb{C}_k^*$ is completely extracted and is pure.
Case 2: There exists no optimal cluster such that its cardinality is $i + 1$. This case is trivially true, since by the inductive hypothesis we have already extracted all optimal clusters of size $\leq i$ and no optimal cluster was formed on the $i + 1$-th iteration. This completes the proof of the inductive hypothesis. □

4 Algorithm and Analysis for Theorem 2.2

In addition to the variables introduced in the previous section, we introduce :
$StableSubclusters$ at the i^{th} iteration : A set of clusters that are formed as a part of a $CandidateCluster$ but are stable and are pruned out.
$PrunedSubclusters$ at the i^{th} iteration : A list of $StableSubclusters$ for each $CandidateCluster$.
The linkage procedure is identical to one described in the previous subsection. In addition, we have to consider the case where an optimal cluster is formed in the i^{th} iteration but is entirely pruned out by the stability check. This case is highly detrimental to the algorithm as this optimal cluster would never be recovered. To circumvent this case, we always keep the clique around the center of a $CandidateCluster$ and prove that this clique is pure. The algorithm may still prune $StableSubclusters$ that belong the optimal cluster but don't lie in the clique. These clusters are stored in the $PrunedSubclusters$ list to be reassigned correctly towards the end. The next challenge is that the clique can eventually become large enough to contain multiple complete clusters that have previously been extracted. Thus, we form a forest where each node is a $StableLaminarCluster$. We also connect root nodes to their supersets in each iteration. As a post processing step, we consider the set $PrunedSubclusters \setminus R$, where R is the union of all the root nodes in the forest. We assign each cluster in this set to it's closest neighbor and propagate this assignment down the forest. We then show that this assignment is pure and provides a partition of set S. Using a greedy pruning algorithm on the forest, we can find a stable solution for this clustering instance.

Algorithm 3: Main

Input: Set of all points S, constant k

```
i ← 1;
while i ≤ n do
    G ← BallLinkage(i) /* Same as Algorithm 2 */;
    StableLaminarClusters_i ← ExtractStableLaminarClusters(G, i);
    for StableLaminarCluster ∈ StableLaminarClusters_i do
        T_j^i ← StableLaminarCluster;
        Create an edge with existing root nodes that are properly contained in T_j^i;
        j ← j + 1;
    end
    i ← i + 1 ;
end
PruneForest(T, k, PrunedSubclusters);
```

The BallLinkage routine in Algorithm 3 is identical to the one in Algorithm 1. The ExtractStableLaminarClusters routine is slightly modified. We only prune out the stable laminar clusters that don't lie in the clique around the center, i.e. the clique always remains intact. The intuition behind this, as mentioned previously, is that in an adverse instance, we may end up pruning out everything that is

formed in the i^{th} iteration where $i = \mathbb{C}_j^*$ thus erroneously excluding all of $\mathbb{C}_j^*$. However, we may also prune out stable components of an optimal cluster. Therefore, we add these pruned clusters to another list, $PrunedSubclusters$, to correctly assign them as a post processing step.

Algorithm 4: PruneForest

Input: Forest T, Number of Clusters k, $PrunedSubclusters$

$StableClustering \leftarrow$ Root Nodes of all Trees in Forest T;
$PrunedSubclusters \leftarrow PruneSubclusters \setminus StableClustering$;
Assign each PrunedSubcluster to closest point in $StableClustering$;
while $|StableClustering| < k$ **do**
  for $Cluster \in StableClustering$ **do**
    if *$Cluster$ is not stable w.r.t. $S \setminus Cluster$* **then**
      $StableClustering \leftarrow$ Children($Cluster$);
    end
  end
  /*Expand the current clustering by going one level deeper*/;
  if $|StableClustering| < k$ **then**
    $StableClustering \leftarrow$ { $Cluster | Cluster \in StableClustering$ and Children($Cluster$) are stable }
  end
end

4.1 Analysis

The following lemma follows from the previous section

Lemma 4.1. *Let $\mathbb{C}_j^*$ be a an optimal cluster and c_j^* be it's center. At $i = |\mathbb{C}_j^*|$, $\mathbb{B}_{c_j^*}$ forms a clique and there exists an edge p-c_j^*, $\forall p \in \mathbb{C}_j^*$.*

In order to show a proof by induction similar to the one for Theorem 2.1, we have to show additional properties hold true. Since we no longer remove the clique formed around the center, we argue that this clique is either pure or consists only of laminar extracted clusters. Also, since we remove stable components of the cluster, we must argue that at $i = |\mathbb{C}_j^*|$, all points belonging to $\mathbb{C}_j^*$ have been either extracted or added to $PrunedSubclusters$.

Lemma 4.2. *$\forall$ i, st $i \leq |\mathbb{C}_j^*|$, if an extracted cluster is centered at x, st $x \in \mathbb{C}_j^*$, $\mathbb{B}_x$ is pure.*

Proof. Let x $\in \mathbb{C}_j^*$. Let y $\in \mathbb{C}_k^*$ s.t $j \neq k$. There are two cases, either $c_k^* \in \mathbb{B}_x$ or $c_k^* \notin \mathbb{B}_x$. In the first case, x is closer to a good point in $\mathbb{C}_k^*$, namely c_k^* than all good points in $\mathbb{C}_j^*$ which is a contradiction as it violates Property 2.1. In the second case, c_k^* is a part of a stable subcluster and is pruned out. However, $d(y, c_k^*) < d(y, x)$, therefore, the stability check rejects this cluster. □

Lemma 4.3. *At $i = |\mathbb{C}_j^*|$, $\mathbb{B}_{c_j^*} \cup \{\mathbb{C}_j^* \setminus StableSubclusters\}$ is extracted and is pure.*

Proof. By lemma 4.1, $\mathbb{B}_{c_j^*}$ forms a clique, c_j^* has a degree of atleast $|\mathbb{C}_j^*|$ and there is no edge to a larger cluster. The algorithm prunes out all $StableSubclusters$ that are connected to c_j^* and are not in $\mathbb{B}_{c_j^*}$.
Let us assume that the cluster extracted is not pure. However, we know that $\mathbb{B}_{c_j^*}$ is pure (by lemma 4.2) and therefore an impure part exists in whatever remains. However, as a result of lemma 4.1, the impure part can only be a smaller cluster. By the inductive hypothesis, we know that at least the clique of the smaller cluster has been extracted. Therefore, the impure edges are pruned out by the laminarity and stability checks. □

Lemma 4.4. *Greedy pruning terminates and outputs a good locally optimal k clustering of the data.*

Proof. By lemma 1.5, $\forall\ \mathbb{C}_j^*$, at $i = |\mathbb{C}_j^*|$, $\{\mathbb{C}_j^* \setminus StableSubclusters\}$ gets extracted and forms a node in the tree. By Lemma 4.3, any parent of this node in the forest, would contain all of $\{\mathbb{C}_j^* \setminus StableSubclusters\}$. Therefore, the forest contains a node for each $\{\mathbb{C}_j^* \setminus StableSubclusters\}$, s.t

it is stable. Therefore, the greedy pruning algorithm always maintains a list of stable clusterings and terminates when k such stable clustering are found. □

Finally, consider the case where we want to extract the clustering that minimizes the objective function Φ, given that the forest we extracted has been processed into a tree. In order to argue correctness, we need to show that the optimal clustering is a k pruning of the tree.

Lemma 4.5. *The optimal clustering is laminar with respect to the the tree obtained by processing the forest constructed by Algorithm 3.*

Proof. Observe that the root of the tree definitely contains all the k optimal centers $c_1^*, c_2^*, ...c_k^*$, since we always retain the clique for each $CandidateCluster$. Taking the set difference between $PrunedSubclusters$ and $StableClustering$ gives us the stable components of the optimal clustering that were pruned out by the stability check. The assignment of these clusters is guaranteed to be pure since the centers exist in the $StableClustering$ and each point is closer to its center than any point not in the cluster. By lemma 4.3, at $i = |\mathbb{C}_j^*|$, $\mathbb{B}_{c_j^*} \cup \{\mathbb{C}_j^* \setminus StableSubclusters\}$ forms a node T_j^i and by laminarity these points stay together higher up in the tree. Therefore, propagating the $StableSubclusters$ of $\mathbb{C}_j^*$ down completes $\mathbb{C}_j^*$ at node T_j^i. This is true for all optimal clusters and thus the optimal clustering is a k pruning of the tree. □

The proof of the inductive hypothesis is similar to that of Algorithm 1 and has been omitted due to space constraints. Essentially, the only difference in the algorithm is to keep the cliques formed in every iteration, which are either pure or contain complete extracted clusters.

5 Limitations of prior work

Our algorithm is inspired by the linkage procedure of [6], however we need to perform several additional steps in order to make sure that the output is correct. Here we indicate the necessity of this by demonstrating that the algorithm from [6] in itself will not work.

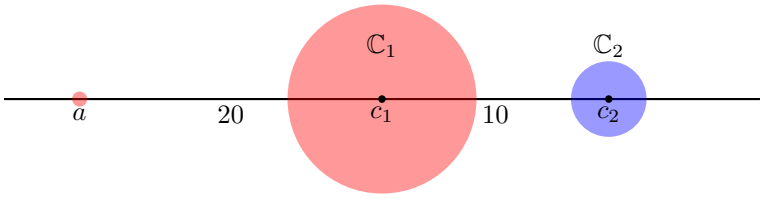

Figure 1: A 1-D Clustering instance that has a good locally stable solution

Consider the clustering instance where we have two optimal clusters $\mathbb{C}_1$ and $\mathbb{C}_2$, centered at c_1 and c_2. The point $a \in \mathbb{C}_1$ and is 20 units away from c_1. The center of $\mathbb{C}_2$, c_2, is 10 units away from c_1. All other points in $\mathbb{C}_1$ are ϵ-close to c_1. Similarly, all points in $\mathbb{C}_2$ are ϵ-close to c_2. Also, $|\mathbb{C}_1| >> |\mathbb{C}_2|$. It is clear that this instance is 2-stable. However, $\forall p \in \mathbb{C}_1, \forall q \in \mathbb{C}_2\ d(p, c_1) < d(q, c_1)$ is not true as $d(a, c_1) > d(c_1, c_2)$. Using the linkage procedure of [6], it is possible that a ball $B(c_1, r)$, where $10 < r < 20$, is formed such that $\{\mathbb{C}_1 \setminus a\} \cup \mathbb{C}_2 \nsubseteq B(c_1, r)$. This ball also satisfies the margin property. However, as a result laminarity no longer holds and the optimal clustering is not a k pruning of the tree that would be formed.

6 Conclusion and Future Work

In this work we initiated the study of computing good locally optimal solutions for non-convex problems in machine learning. Although the results we presented are in the context of clustering, our framework is general enough to be applicable to many other problems. It would be very interesting if viewing non-convex problems from this lens can shed new light in areas such as deep learning. For clustering itself, several open problems remain. Although our algorithms are polynomial time, the runtime is high. Can one provide similar guarantees on the solution if, for instance, the algorithm is

run on a random sample. Can the algorithm be extended to work when the number of good points in each cluster is not that large. We believe that these are intriguing questions for future work.

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 J. Comput.*, 33(3):544–562, 2004.

[2] Pranjal Awasthi, Avrim Blum, and Or Sheffet. Stability yields a ptas for k-median and k-means clustering. In *Proceedings of the 2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, FOCS '10, 2010.

[3] Pranjal Awasthi, Avrim Blum, and Or Sheffet. Center-based clustering under perturbation stability. *Inf. Process. Lett.*, 112(1-2), January 2012.

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

[5] M. F. Balcan, A. Blum, and S. Vempala. A discrimantive framework for clustering via similarity functions. In *Proceedings of the 40th ACM Symposium on Theory of Computing*, 2008.

[6] Maria-Florina Balcan and Yingyu Liang. Clustering under perturbation resilience. *ICALP*, 2012.

[7] S. Ben-David, U. von Luxburg, and D. Pal. A sober look at stability of clustering. In *Proceedings of the Annual Conference on Computational Learning Theory*, 2006.

[8] Ian Goodfellow Yoshua Bengio and Aaron Courville. Deep learning. Book in preparation for MIT Press, 2016.

[9] Mary Inaba, Naoki Katoh, and Hiroshi Imai. Applications of weighted voronoi diagrams and randomization to variance-based k-clustering: (extended abstract). In *Proceedings of the tenth annual symposium on Computational geometry*, SCG '94, 1994.

[10] K. Jain and V. V. Vazirani. Approximation algorithms for metric facility location and k-median problems using the primal-dual schema and lagrangian relaxation. *JACM*, 48(2):274 – 296, 2001.

[11] Tapas Kanungo, David M. Mount, Nathan S. Netanyahu, Christine D. Piatko, Ruth Silverman, and Angela Y. Wu. A local search approximation algorithm for k-means clustering. In *Proceedings of the eighteenth annual symposium on Computational geometry*, SCG '02, New York, NY, USA, 2002. ACM.

[12] A. Kumar and R. Kannan. Clustering with spectral norm and the k-means algorithm. In *Proceedings of the 51st Annual IEEE Symposium on Foundations of Computer Science*, 2010.

[13] S.P. Lloyd. Least squares quantization in PCM. *IEEE Trans. Inform. Theory*, 28(2):129–137, 1982.

[14] M. Meila. The uniqueness of a good clustering for k-means. In *Proceedings of the 23rd International Conference on Machine Learning*, 2006.

[15] R. Ostrovsky, Y. Rabani, L. Schulman, and C. Swamy. The effectiveness of lloyd-type methods for the k-means problem. In *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science*, 2006.

Algorithm 5: ExtractStableLaminarClusters

Input: Graph G, Current Iteration i

1 /* Extract Candidate Clusters as done in Algorithm 3*/;
2 **for** $x \in S$ **do**
3 $\quad \mathbb{B}_x \leftarrow$ {closest $i/2$ points to x} $\cup$ { x };
4 $\quad$ **if** $\mathbb{B}_x$ *forms a clique of size* $i/2$ *and deg(x)* $\geq i$ **then**
5 $\quad\quad CandidateClusters \leftarrow$ {x} $\cup$ Neighbors(x) ;
6 $\quad$ **end**
7 **end**
8 /*Laminars clusters that have already been extracted */;
9 $LaminarSubclusters \leftarrow StableLaminarClusters_{i-1}$;
10 /* Extract Stable Laminar Clusters from Candidate Clusters */;
11 **for** $CandidateCluster \in CandidateClusters$ **do**
12 $\quad c \leftarrow$ Center($CandidateCluster$);
13 $\quad$ **for** $LaminarSubcluster \in LaminarSubclusters$ **do**
14 $\quad\quad c_{lsc} \leftarrow$ Center(LaminarSubcluster);
15 $\quad\quad$ /* Stability check for subsets of Candidate Cluster */;
16 $\quad\quad$ **if** $LaminarSubcluster \subset CandidateCluster$ *and* $LaminarSubcluster \nsubseteq \mathbb{B}_c$ **then**
17 $\quad\quad\quad$ **if** $\forall p \in$ ***LaminarSubcluster,*** $\forall q \notin$ ***LaminarSubcluster,*** $d(p, c_{lsc}) < d(p, q)$ **then**
18 $\quad\quad\quad\quad CandidateCluster \leftarrow CandidateCluster \setminus LaminarSubcluster$
$\quad\quad\quad\quad\quad PrunedSubcluster \leftarrow LaminarSubcluster$;
19 $\quad\quad\quad$ **end**
20 $\quad\quad$ **else**
21 $\quad\quad\quad$ /* Laminarity Check for Candidate Cluster */;
22 $\quad\quad\quad CandidateCluster \leftarrow CandidateCluster \setminus LaminarSubcluster$
23 $\quad\quad$ **end**
24 $\quad$ **end**
25 $\quad$ **if** $\forall p \in CandidateCluster$, $\forall q \notin$ ***CandidateCluster,*** $d(p, c) < d(p, q)$ **then**
26 $\quad\quad$ Extract $CandidateCluster$;
27 $\quad$ **end**
28 **end**

A Appendix

In order to prove Theorem 3.1 by induction, we will first assume the inductive hypothesis is true, show that Theorem 3.1 is implied by it and then prove the inductive hypothesis. The inductive hypothesis is that at any iteration i, $\forall j$, $\mathbb{B}_{c_j} \cup \{\mathbb{C}_j \setminus StableSubclusters\}$ is extracted. Using this, we show that the algorithm terminates, the forest, after the assignment of $PrunedSubclusters$ forms a partitioning of S, and the extracted clusters are stable.

Proof. It is clear from Algorithm 5 that it terminates after n iterations. Let us assume a set of points C was not present in the forest after the $PrunedSubclusters$ had been assigned. Let p be any point in C. Then p must belong to some optimal cluster $\mathbb{C}^*_j$. At i = $|\mathbb{C}^*_j|$, p was either formed a node or was removed from the part of $\mathbb{C}^*_j$ that was pruned by the stability check and added to $PrunedSubclusters$. In the first case, the point p must exist in the forest. In the second case, point p gets added to the Forest once $PrunedSubclusters$ have been assigned. Hence p must eventually exist in the Forest. Therefore, the algorithm terminates and all the points in S exist in the forest. The algorithm also ensures that any cluster that is extracted is stable w.r.t. Definition 2.1, as is evident in lines 25-26 of Algorithm 5. This completes the proof of Theorem 3.1. □

Next, we prove the inductive hypothesis : At any i, all optimal clusters $\mathbb{B}_{c^*_j} \cup \{\mathbb{C}^*_j \setminus StableSubclusters\}$ is extracted , such that $|\mathbb{C}^*_j| \leq i$ have been extracted. We need to show that this holds for $i + 1$. There are two cases to consider, there is atleast one optimal cluster $\mathbb{C}^*_k$ such that $i + 1 = |\mathbb{C}^*_k|$ or there is no optimal cluster of cardinality $i + 1$. We prove the following lemmas to assist in the proof.

Lemma A.1. *At* $i = |\mathbb{C}^*_j|$, $\mathbb{B}_{c^*_j} \cup \{\mathbb{C}^*_j \setminus StableSubclusters\}$ *is extracted and is pure.*

Proof. By lemma 4.1, $\mathbb{B}_{c_j^*}$ forms a clique, c_j^* has a degree of atleast $|\mathbb{C}_j^*|$ and there is no edge to a larger cluster. The algorithm prunes out all $StableSubclusters$ that are connected to c_j^* and are not in $\mathbb{B}_{c_j^*}$.
Let us assume that the cluster extracted is not pure. However, we know that $\mathbb{B}_{c_j^*}$ is pure (by lemma 4.2) and therefore an impure part exists in whatever remains. However, as a result of lemma 4.1, the impure part can only be a smaller cluster. By the inductive hypothesis, we know that at least the clique of the smaller cluster has been extracted. Therefore, the impure edges are pruned out by the laminarity and stability checks.

□

Lemma A.2. *$\forall$ i, j s.t. $i \leq |\mathbb{C}_j^*|$, if an extracted cluster is centered at c_x, s.t. $c_x \in \mathbb{C}_j^*$, there is no impure edge c_x-y, where y belongs to another optimal cluster.*

Proof. Let us assume otherwise. Let the current extracted cluster be $\mathbb{C}_x$, centered at c_x, and y $\in \mathbb{C}_k^*$. Consider the case where $|\mathbb{C}_k^*| < i$. Now, either c_k^* is connected to c_x or c_k^* isn't connected to c_x. For the first case, c_k^* is the center of a cluster that we have extracted, since $|\mathbb{C}_k^*| < i$. This cluster cannot be partially included, since laminarity holds. Therefore, it forms a laminar subcluster such that $\forall p \in$ LaminarSubcluster, $\forall q \notin$ LaminarSubcluster, $d(p, c_{lsc}) < d(p, q)$. Therefore, the LaminarSubcluster is pruned and the edge $c_x - c_k^*$ is removed, reducing to the second case. In the second case, we reject the cluster since $d(y, c_k^*) < d(y, c_x)$. Therefore, there is no impure edge to a smaller cluster. Consider the second case where $|\mathbb{C}_k^*| \geq i$. Now, either c_k^* is connected to c_x or c_k^* isn't connected to c_x. For the first case, $|\mathbb{B}_{c_x} \cap \mathbb{B}_{c_k^*}| \geq 1$. But $\mathbb{B}_{c_k^*} \subseteq \mathbb{G}_{c_k^*}$. Therefore, c_x is closer to a good point $\mathbb{C}_k^*$ than all good points in $\mathbb{C}_j$. This is a contradiction as it violates Property 2.1. In the second case, $d(y, c_k^*) < d(y, c_x)$ and the check at lines 25-26 of Algorithm 5 rejects this cluster. Therefore there is no impure edge in any extracted cluster centered in a cluster that hasn't already been extracted. □

Lemma A.3. *$\forall$ i, s.t. $i > |\mathbb{C}_j^*|$, any extracted cluster $\mathbb{C}_x$ contains complete extracted clusters of size $< i$.*

Proof. By lemma A.2, x has no edge with a cluster larger than i. By the induction hypothesis, all clusters of size $\mathbb{C}_j^*$ and less have been extracted. Therefore, $\mathbb{C}_x$ contains extracted clusters of size $< i$, since laminarity holds. □

To complete the proof by induction, we consider the following cases:

Proof. **Case 1:** By lemma A.2, we know whatever part of $\mathbb{C}_k^*$ that has been extracted till now is pure. By lemma A.1, at $i + 1$, $\mathbb{B}_{c_k^*} \cup \{\mathbb{C}_k^* \setminus StableSubclusters\}$ is extracted and the $StableSubclusters$ are added to $PrunedSubclusters$.
Case 2: This case is trivially true, since by the inductive hypothesis we have already extracted all optimal clusters of size $\leq i$ and no optimal cluster was formed on the $i + 1$-th iteration and no impure component can be extracted. This completes the proof of the inductive hypothesis. □