Local Search Methods for k-Means with Outliers

Shalmoli Gupta*
University of Illinois
201 N. Goodwin Ave.
Urbana, IL, 61801
sgupta49@illinois.edu

Ravi Kumar
Google
1600 Amphitheater Parkway
Mountain View, CA 94043
ravi.k53@gmail.com

Kefu Lu†
Washington University
1 Brookings Drive
St. Louis, MO 63130
kefulu@wustl.edu

Benjamin Moseley‡.
Washington University
1 Brookings Drive
St. Louis, MO 63130
bmoseley@wustl.edu

Sergei Vassilvitskii
Google
76, 9th Ave
New York, NY 10011
sergeiv@google.com

ABSTRACT

We study the problem of k-means clustering in the presence of outliers. The goal is to cluster a set of data points to minimize the variance of the points assigned to the same cluster, with the freedom of ignoring a small set of data points that can be labeled as outliers. Clustering with outliers has received a lot of attention in the data processing community, but practical, efficient, and provably good algorithms remain unknown for the most popular k-means objective.

Our work proposes a simple local search-based algorithm for k-means clustering with outliers. We prove that this algorithm achieves constant-factor approximate solutions and can be combined with known sketching techniques to scale to large data sets. Using empirical evaluation on both synthetic and large-scale real-world data, we demonstrate that the algorithm dominates recently proposed heuristic approaches for the problem.

1. INTRODUCTION

Clustering is a fundamental data mining task that has been extensively studied [3]. In a typical clustering problem the input is a set of points with a notion of similarity between every pair of points, and a parameter k, which specifies the desired number of clusters. The goal is to partition the points into k clusters such that points assigned to the same cluster are similar. One way to obtain this partition is to select a set of k centers and then assign each point to its nearest center. Different objectives can be used to determine the quality of the clustering solution. One of the most popular objectives is the k-means cost function, which minimizes the sum of squared distances between every point and its nearest center. In the Euclidean space this is equivalent to minimizing the variance of the points assigned to the same cluster.

Clustering using the k-means objective is one of the most widely studied data mining problems. It is well known that finding the optimal solution is NP-hard. In practice, the dominant algorithm used for this problem is *Lloyd's* algorithm, also known as the k-means method [29]. Underscoring the importance of the k-means problem, Lloyd's algorithm has been identified as one of the top ten algorithms used in data mining [39]. However, it is at best a heuristic and has no guarantees on the quality of the solution (other than being a local minimum).

In addition to heuristic approaches such as Lloyd's, there has been a lot of algorithmic work on the k-means problem. If the data lies in the Euclidean space, an algorithm that can achieve an approximation of $1+\epsilon$ for any $\epsilon > 0$ is known [27, 16]; unfortunately, it runs in time exponential in k and $1/\epsilon$ and hence is not practical. If the data resides in an arbitrary metric space, the algorithm with the best known theoretical guarantees is a local search algorithm that achieves a constant-factor approximation [20, 24].

Clustering with outliers. Although the k-means problem is well-studied, algorithms developed for it can perform poorly on real-world data. This is because the k-means objective assumes that all of the points can be naturally partitioned into k distinct clusters, which is often an unrealistic assumption in practice. Real-world data typically has background noise, and the k-means method is extremely sensitive to it. Noise can drastically change the quality of the clustering solution and it is important to take this into account in designing algorithms for the k-means objective.

To handle noisy data sets, we propose the problem of k-means with outliers. In this version of the problem, the clustering objective remains the same, but the algorithm is additionally allowed to discard a small set of data points from the input. These discarded points are labeled as *outliers* and are ignored in the objective, thus allowing the clustering algorithm to focus on correctly partitioning the bulk

*Supported in part by NSF grant CCF-1319376.
†Supported in part by NSF Grant CCF-1617724
‡Supported in part by a Google Research Award, a Yahoo Research Award and NSF Grant CCF-1617724

Proceedings of the VLDB Endowment, Vol. 10, No. 7

of the dataset that is potentially noise-free.

The k-means with outliers problem is far less well-understood than clustering under the vanilla k-means objective. While constant-factor polynomial time approximation algorithms are known [11, 13], these approaches require solving sophistical mathematical programs and the algorithms themselves are highly complicated and do not scale. To the best of our knowledge, these algorithms have never been implemented.

Developing fast practical algorithms for clustering with outliers remains an active area of research. For example, several constant-factor algorithms are known for the (easier) k-center objective[1] [11, 31, 32]. However, the more widely-used k-means objective remains elusive. The work of [12] developed an extension of Lloyd's algorithm to the case where there are outliers, but no guarantees are known on the quality of the solution obtained by the algorithm. The question remains: does there exists an efficient, practical algorithm with theoretical guarantees on its performance for the k-means with outliers problem?

Our contributions. In this paper we give the first algorithm for k-means clustering with outliers that has provable approximation guarantees and is practically efficient. It is also the first known local search method for this problem with performance guarantees.

Informally speaking, given the desired number of clusters (k) and an upper bound (z) on the number of outliers, our algorithm computes a constant-factor approximation to the k-means objective but may identify slightly more than z points as outliers. This bicriteria approximation is at the heart of the analysis: by allowing the algorithm to discard a few more outliers than specified, we gain analytic tractability. Complementing the analysis, in our experimental evaluation we demonstrate that the bound we prove is a worst-case theoretical bound and the algorithm typically discards close to the specified number of outliers. It is important to keep in mind that in practice the exact number of outliers is rarely known and labeling a few extra points as outliers typically does not affect the quality of the clustering solution.

Our algorithm is based on a local search process wrapped by an outlier removal step summarized as follows. Starting with an arbitrary set of k centers, the algorithm iteratively checks if swapping one of the current centers with a non-center would improve the objective, and makes the local step if it is profitable. Once the algorithm has converged on a locally optimal solution, it checks if any further local swap, with the additional removal of z outliers, can significantly improve the solution. If such a swap exists, it performs the swap and removal, and goes back to the simple local search mode. While the algorithm itself is very simple and easy to implement, proving that it has performance guarantees is more involved, especially with the outlier-based objective.

In addition to proving the method has theoretical guarantees, we show that this local search algorithm works well in practice. In particular, we compare our algorithm to the recently published adaptation of Lloyd's algorithm to the case of noisy data [12], which serves as a strong baseline. On both synthetic and large real datasets, we demonstrate that our algorithm consistently out-performs this baseline in both running time and in the quality of the solution obtained. We obtain similarly strong results when comparing the local search algorithm against other algorithms for related clustering problems, such as the recent Lagrangian relaxation algorithm [36].

Finally, we show how our algorithm can be combined with known sketching techniques to handle large data sets. In particular, we show how to use the fast k-means++ method [5] to develop a linear-time algorithm for large values of k and z. This algorithm has the same strong theoretical guarantees as our main algorithm.

[1]In the k-center problem the goal is to choose centers to minimize the maximum distance of a point to a center

2. BACKGROUND

2.1 Notation

Let U be a given set of points. For two points $u, v \in U$, let $d(u, v)$ denote the distance between u and v. We assume $d(\cdot, \cdot)$ is a metric, i.e., $d(u, u) = 0$ for all $u \in U$, $d(u, v) = d(v, u)$ for all $u, v \in U$, and $d(u, v) + d(v, w) \geq d(u, w)$ for all $u, v, w \in U$. For a set $S \subseteq U$ and a point $v \in U$, let $d(v, S)$ denote the distance of v to the closest point in S, i.e., $d(v, S) = \min_{u \in S} d(u, v)$. By scaling, we can assume that the minimum distance between points is 1. Let Δ denote the maximum distance between the points.

Let S be a set of *centers* of size k. Let

$$cost_{\mathrm{KM}}(S; U') = \sum_{v \in U'} d(v, S)^2,$$

be the *k-means* cost of the set of centers S on a subset $U' \subseteq U$ of points. (Other cost measures studied in the literature include the *k-center* cost, which is $\max_{v \in U'} d(v, S)$ and the *k-median* cost, which is $\sum_{v \in U'} d(v, S)$.)

For a given subset $Z \subseteq U$ of *outliers*, let

$$cost(S, Z) = cost_{\mathrm{KM}}(S; U \setminus Z) = \sum_{v \in U \setminus Z} d(v, S)^2, \qquad (1)$$

be the k-means cost for a set S of centers with respect to Z.

Let k be the desired number of centers and let z be the target number of outliers. Let $C = \{a_1, \ldots, a_k\}$ be the set of centers chosen by an algorithm and let Z denote the set of points marked as outliers by the algorithm; we allow $|Z| \geq z$. Let $C^* = \{b_1, \ldots, b_k\}$ denote the optimal set of centers and let $Z^*, |Z^*| = z$, denote the outliers with respect to the optimal solution. Let OPT be the cost of the optimum solution. We assume that OPT ≥ 1. (The other case is trivial: by the assumption on the minimum distance, if OPT < 1, it must be 0. This in turn implies that $k = n - z$ and any set of k centers and z outliers gives an optimal solution.)

For $S, S' \subseteq U$ and a non-negative integer α, we define the α *farthest* points in S from S' to be the points $v_1, \ldots, v_\alpha$ in S that maximize $\sum_{i=1}^{\alpha} d(v_i, S')$. Let *outliers*$(S, X)$ be the z farthest points from S in $U \setminus X$; for simplicity, we let *outliers*$(S) =$ *outliers*$(S, \emptyset)$.

2.2 A local search algorithm

Our algorithm builds on the known local search algorithm for the k-means problem with no outliers [20, 24]. We call this algorithm LS. The algorithm starts with some set C of centers, given as input, and a universe U of points. It checks for each center $v \in C$ and each non-center $u \in U$ if swapping them would result in a better clustering solution, i.e., whether $C \cup \{u\} \setminus \{v\}$ has lower k-means cost than C. If

so, the swap is performed by replacing C by $C \cup \{u\} \setminus \{v\}$. After checking all possible swaps, the algorithm updates the solution with the best possible swap. The algorithm terminates once the solution value does not decrease significantly by performing a swap, for instance, once the solution value improves by less than a $(1 - \epsilon/k)$ factor. A complete description is given in Algorithm 1.

Algorithm 1 The local search algorithm for k-means with *no* outliers

LS (U', C, k)

$\alpha \leftarrow \infty$
{while the solution improves by performing swaps}
while $\alpha(1 - \frac{\epsilon}{k}) > cost_{\mathrm{KM}}(C; U')$ **do**
  $\alpha \leftarrow cost_{\mathrm{KM}}(C; U')$
  $\overline{C} \leftarrow C$ {a temporary (improved) set of centers}
  {for each center and non-center, perform a swap}
  for each $u \in U'$ and $v \in C$ **do**
    {if this is the most improved swap found so far}
    if $cost_{\mathrm{KM}}((C \cup \{u\} \setminus \{v\}); U') < cost_{\mathrm{KM}}(\overline{C}; U')$ **then**
      $\overline{C} \leftarrow C \cup \{u\} \setminus \{v\}$ {update the temp. solution}
    end if
  end for
  {update the solution to the best swap found}
  $C \leftarrow \overline{C}$
end while
return C as the k centers

It is easy to see that this algorithm always reduces the k-means cost. It is known [20, 24] that Algorithm 1 returns a 25 approximation to the optimum solution. We can further improve the approximation ratio by considering swapping t points of the solution simultaneously. The approximation ratio decreases to $(3 + 2/t)^2$, however, since there are $O(n^t)$ such potential swaps, the running time increases accordingly.

3. THE ALGORITHM

In this section we present our main algorithm for k-means clustering with outliers. We analyze the running time and discuss how data sketching techniques can further speed up the algorithm especially for large n. Finally, we state the approximation ratio of the algorithm but defer the full technical analysis to Section 5.

The algorithm takes as input the set U of n points, the desired number k of centers, the target number z of outliers, and a constant $0 < \epsilon < 1/6$. (In practice ϵ simply should be set sufficiently small; our experiments use $\epsilon = 10^{-4}$.) It outputs a final set C of k centers and a set Z of outliers.

3.1 Local search with outliers

In this section we present a local search algorithm that handles outliers; we call our algorithm LS-Outlier. This algorithm generalizes the classical version LS by allowing points to be discarded as outliers. It maintains a set Z of outliers it has identified and a set C of centers. Initially C is an arbitrary set of k points and Z is the set of the z farthest points from C. The algorithm tries to locally converge on good sets C and Z by utilizing three procedures.

Algorithm 2 Our local search algorithm for k-means with outliers

LS-Outlier (U, k, z)

$C \leftarrow$ an arbitrary set of k points from U
$Z \leftarrow outliers(C)$
$\alpha \leftarrow \infty$
while $\alpha(1 - \frac{\epsilon}{k}) > cost(C, Z)$ **do**
  $\alpha \leftarrow cost(C, Z)$
  {(i) local search with no outliers}
  $C \leftarrow$ LS$(U \setminus Z, C, k)$
  $\overline{C} \leftarrow C$ {a temporary (improved) set of centers}
  $\overline{Z} \leftarrow Z$ {a temporary (improved) set of outliers}
  {(ii) cost of discarding z additional outliers}
  if $cost(C, Z)(1 - \frac{\epsilon}{k}) > cost(C, Z \cup outliers(C, Z))$ **then**
    $\overline{Z} \leftarrow Z \cup outliers(C, Z)$
  end if
  {(iii) for each center and non-center, perform a swap and discard additional outliers}
  for each $u \in U$ and $v \in C$ **do**
    {if this is the most improved swap found so far}
    if $cost(C \cup \{u\} \setminus \{v\}, Z \cup outliers(C \cup \{u\} \setminus \{v\})) < cost(\overline{C}, \overline{Z})$ **then**
      {update the temp. solution}
      $\overline{C} \leftarrow C \cup \{u\} \setminus \{v\}$
      $\overline{Z} \leftarrow Z \cup outliers(C \cup \{u\} \setminus \{v\})$
    end if
  end for
  {update the solution allowing additional outliers if the solution value improved significantly}
  if $cost(C, Z)(1 - \frac{\epsilon}{k}) > cost(\overline{C}, \overline{Z})$ **then**
    $C \leftarrow \overline{C}$
    $Z \leftarrow \overline{Z}$
  end if
end while
return C as the k centers and Z as the outliers

(i) The algorithm first runs LS with centers C and the set of points $U \setminus Z$, the input with the current outliers removed, to converge to a local minimum.

(ii) Next, it checks the improvement that can be achieved by removing z *additional* outliers, i.e., the improved cost of adding the z farthest points in $U \setminus Z$ to Z. We call this the *no-swap* operation.

(iii) Last, it checks for each pair of points $v \in C$ and $u \in U$, if swapping these two points in C and discarding z additional outliers will improve the solution. I.e., the algorithm checks the clustering cost of using the points in $C \setminus \{v\} \cup \{u\}$ as the centers and discarding the points in $Z \cup outliers(C \setminus \{v\} \cup \{u\}, Z)$. We will say that the outliers discarded in this set was due to a *swap*.

When running (ii) and (iii), the algorithm commits to the best set of z additional outliers to discard between these two operations over all possibilities. If the best improvement was a no swap, it adds z outliers to Z and leaves C the same. Otherwise the best improvement is a swap between a pair $v \in C$ and $u \in U$. In this case the algorithm updates C to be $C \setminus \{v\} \cup \{u\}$ and adds the z farthest points from

$C \setminus \{v\} \cup \{u\}$ in $U \setminus Z$ to Z. This continues until there is no possible operation that will reduce the cost of the algorithm by a $(1 - \frac{\epsilon}{k})$ factor. A complete formal description is given in Algorithm 2.

Note that the algorithm is designed to be as conservative as possible when discarding additional outliers. In particular, the algorithm will first try to converge to a local maximum by performing local search without removing outliers. Then, the algorithm only discards additional outliers if there is a significant improvement in the solution cost. In the experiments we show that by being conservative the algorithm rarely discards more than z outliers.

3.2 Running time

We next show the performance guarantees of LS-Outlier. First, we analyze its running time.

THEOREM 1. *Algorithm* LS-Outlier *runs in time* $O(\frac{1}{\epsilon}k^2n^2\log(n\Delta))$.

PROOF. First notice that any solution set of k centers, regardless of the outliers chosen, has total cost at most $O(n\Delta^2)$. This is because any point is distance at most Δ from the centers chosen (by definition of Δ) and therefore each of the n points contributes at most Δ^2 to the objective. Each iteration of the **while** loop in LS or LS-Outlier improves the solution cost by at least a $(1 - \frac{\epsilon}{k})$ factor. Starting with the worst possible solution of cost $O(n\Delta^2)$ results in at most $O(\log_{1-\frac{\epsilon}{k}}(n\Delta^2)) = O(\frac{k}{\epsilon}\log(n\Delta))$ iterations of the **while** loop before the cost is at most $1 \leq \text{OPT}$. In each iteration, there are at most $O(kn)$ swaps to check (each center with each possible non-center). Finally, each swap reassigns at most n points to centers giving an overall running time of $O(\frac{1}{\epsilon}k^2n^2\log(n\Delta))$, the number of iterations $O(\frac{k}{\epsilon}\log(n\Delta))$ multiplied by the number of swaps per iteration $O(kn)$ and finally the number of points that need to be reassigned per iteration $O(n)$. □

3.3 Improving the running time

When working with large datasets, a running time quadratic in the number of points is prohibitive. In this section we discuss how to use sketching techniques to reduce the running time, and make LS-Outlier practical, even for very large data sets.

A natural notion of a data sketch for clustering problems is that of a *coreset*. Observe that it is easy to extend the $cost_{\text{KM}}$ objective function to weighted point sets. Given a weight w_v associated with every point, v, we have:

$$cost_{\text{KM}}(S; U'; w) = \sum_{v \in U'} w_v \cdot d(v, S)^2.$$

We can then formally define the notion of a coreset [21]. A weighted point set $Y \subseteq U'$ is an *α-coreset*, if for any set of cluster centers C,

$$cost_{\text{KM}}(C, U') \leq \alpha \cdot cost_{\text{KM}}(C; Y; w).$$

While there has been a lot of work in understanding tradeoffs between the size of the coreset Y and the approximation ratio α, we focus on fast and efficient coreset constructions. Perhaps the most well known such technique is k-means++ [5]. The k-means++ algorithm is a simple randomized sampling procedure for the vanilla k-means problem (i.e., with no outliers).

The algorithm is parametrized by the number of points to be chosen ℓ. It works as follows. First it selects an arbitrary point to add to the set Y. Then for $\ell - 1$ steps it samples a single point from U and adds it to Y. The points are not chosen uniformly, rather a point $u \in U$ is chosen with probability proportional to $d(u, Y)^2$. The algorithm runs in time $O(\ell n)$. It is known that if we use k-means++ to sample k points, then it achieves an $O(\log k)$ approximation to the (vanilla) k-means objective. Further if $2k$ points are sampled, then it leads to an $O(1)$-approximation [1].

Notice that k-means++ is designed for a problem with no outliers. To adapt it to our setting, the key is to consider an instance of the k-means problem with no outliers, but with the number of centers set to $k + z$. In this case it is easy to see that the cost of the optimal solution to this problem is only smaller than the optimal solution to the problem where k centers and z outliers can be chosen. Indeed, one could set the $k + z$ centers to be the same k centers and z outliers. Therefore, by selecting $2(k + z)$ points as centers using k-means++, the final cost will be an $O(1)$-approximation to our problem. This, however, does not immediately solve our problem since we have chosen an *extra* $k + 2z$ centers.

To convert the selected set to a coreset we must assign weights to each point. Each point $v \in Y$ is given a weight $w_v = |\{u \mid d(u, v) = \min_{v' \in Y} d(u, v')\}|$, breaking ties arbitrarily. The value of w_v is the number of points in U where v is their closest point Y. This set can then be used as an input to the local search algorithm where the weights are treated as if there are w_v copies of v; this reduces the size of the input to $2(k + z)$.

As shown in [17, 21], this weighted point set forms an $O(1)$-coreset. If we cluster this weighted point set using an $O(1)$-approximation for the k-means with z outliers problem, then we will get an $O(1)$-approximation for the original input. Putting it together, we obtain the following result:

THEOREM 2. *By combining* LS-Outlier *with k-means++, the running time is* $O(\frac{1}{\epsilon}k^2(k + z)^2\log(n\Delta) + nz)$.

For handling extraordinarily large data sets, one can combine the algorithm with the scalable k-means++ method [6] for a fast distributed computing algorithm that computes a solution similar to k-means++.

3.4 Clustering quality

We next focus on the quality guarantees of LS-Outlier. This is our main technical result.

THEOREM 3. *Given k, z, Algorithm* LS-Outlier *outputs a set $C, |C| = k$ of centers and a set Z of outliers such that $cost(C, Z) \leq c \cdot OPT$ for some constant $c \geq 1$ and $|Z| \leq O(zk\log(n\Delta))$.*

Note that the algorithm discards at most $O(z \log n)$ outliers when k is fixed and Δ is bounded by a polynomial in n. Our analysis presented later in the paper shows that the constant c in Theorem 3 is 274. The experimental performance of the algorithm is good and the above statement backs this up by showing that the algorithm has provable guarantees.

For the related ℓ_2-norm objective [20], which is the square root of the k-means objective, the approximation factor we obtain is 17. For this objective, the best known analysis of local search *without outliers* yields a 5-approximation [24, 20]. Even without outliers, it is known that local search cannot be better than a 3-approximation in the worst case

for the ℓ_2-norm objective [24]. Our results are in line with what is previously known for the easier special case of the problem where there are no outliers.

First we bound the number of points marked as an outlier by LS-Outlier.

LEMMA 4. $|Z| \leq O(\frac{1}{\epsilon} zk \log n\Delta)$.

PROOF. The bound on the outliers follows from the number of local steps taken by Algorithm LS-Outlier. The algorithm initially sets C to be an arbitrary set of k centers. The worst case cost of such a solution is $O(n\Delta^2)$. In each iteration of the **while** loop, the algorithm removes at most z outliers and the cost of the solution decreases by an $1 - \frac{\epsilon}{k}$ factor. Knowing that the cost must be at least 1, the total number of iterations can be at most $O(\frac{1}{\epsilon} k \log(n\Delta))$. Thus the total number of outliers chosen by the algorithm is at most $O(\frac{1}{\epsilon} zk \log(n\Delta))$. □

It remains to bound the cost of the solution obtained by LS-Outlier. To prove the bound, we carefully string together the inequalities that hold due to local optimality of the solution. The immediate difference from the no-outlier proof by [20, 24] is in the notion of capture, where a point in the algorithm's solution cannot capture those points labeled as outliers. This plays a critical role in bounding the cost of any potential swap, where we first chose to ignore points labeled as outliers by either the algorithm or the optimal solution, and use the relaxed triangle inequality to bound the contribution of the other points. Indeed, it is the mechanics of the proof that motivate the third procedure in the main algorithm, that of checking the magnitude of the improvement using a swap *and* an additional z outliers. Intuitively, this is a reasonable check to make, since the algorithm may not be using the same z outliers as the optimum. However proving that there are no unexpected consequences to this check requires a formal analysis, which we present in Section 5.

4. EXPERIMENTS

We evaluate the performance of our algorithm on both synthetic and real datasets. As described in Section 3.2, we can use k-means++ to reduce the size of the input our algorithm runs on and improve its running time (Theorem 2). When running our algorithm, we start by using k-means++ to construct a weighted set of points of size $k + z$. We run Algorithm 2 on this weighted point-set. Throughout this section, abusing terminology slightly, we refer to this entire procedure as the LS-Outlier algorithm. We use $\epsilon = 10^{-4}$ in our experiments.

Baselines. As our main baseline we have implemented the algorithm in [12], modulo a small modification: instead of choosing the initial seed points uniformly at random, we use k-means++ to determine the initial k points to seed the algorithm. Our experiments showed that this dramatically improves the performance of the algorithm, making the baseline very strong. Henceforth, we call this algorithm Lloyd-Outlier.

Recall that our algorithm may discard more than z outliers. To make the comparison for cost fair, we first run LS-Outlier and then allow Lloyd-Outlier to throw out the same number of outliers.

Apart from Lloyd-Outlier, we also consider three well-known algorithms from the broader literature on clustering noisy data:

(i) A two-step process of outlier detection using LOF [10], followed by clustering using k-means++. We refer to this as LOF-Outlier. In order to detect the outliers, we compute LOF score for each point, and remove the top z points as outliers.

(ii) An integrated clustering with outlier method proposed in [36]. Optimized for the facility location problem, it uses Lagrangian relaxation and hence we refer to the method as LR. While the facility location problem is similar, the key difference is that the algorithm is allowed to chose any number of clusters, depending on the facility cost. In some practical settings, having a cost to open a new cluster is natural. In other cases the number of clusters is a more natural parameter.

(iii) DBSCAN from [14]. While DBSCAN is a popular clustering heuristic, we found it unsuitable for this problem. First, DBSCAN can have an arbitrary number of clusters and outliers, and it is very hard to set parameters to achieve a precise trade-off that we seek. Secondly, the DBSCAN method requires the knowledge of the number of points in each cluster and the radius of the clusters, which is not available for traditional unsupervised clustering; especially when the cluster sizes vary. Both of these concerns have been echoed in previous work, see for example [9, 18]. Finally, DBSCAN does not come with any theoretical guarantees on its performance. The results of the experiments were incomparable to the other methods, and thus we chose to omit them in the experimental evaluation.

Metrics. For each algorithm we compute the k-means objective cost, given by (1). To further quantify the detection of outliers, we define two quantities: precision and recall. Assume that we know the set of true outliers Z^* and that an algorithm returns the set Z as outliers. The *precision* for the algorithm is the fraction of true outliers within the set of outliers claimed by the algorithm (formally, $\frac{|Z^* \cap Z|}{|Z|}$). Similarly *recall* is the fraction of true outliers that are returned by the algorithm, $\frac{|Z^* \cap Z|}{|Z^*|}$. These metrics together measure the accuracy of outlier detection.

4.1 Synthetic data

In this section we describe our experiments on synthetic datasets. To generate the synthetic dataset, we choose k *real centers* uniformly at random from a hypercube of side length 100. Centered at each of these real centers, we add points from a Gaussian distribution with unit variance. This gives us k well separated clusters. Finally, we sample z outliers uniformly at random from the hypercube of side length 100. Some of these outlier points may fall in the middle of the already selected Gaussians. Due to this, once the entire data has been generated we consider the points furthest from the real centers as the *true outliers*.

We generated the dataset for $n = 10000$ and $d = 2, 15$. Since both algorithms are initialized with a random seeding procedure, we run 10 trials for each case. We report the average cost, as well as, precision and recall for $d = 15$ dataset (similar trends were observed for $d = 2$ dataset). The precision and recall comparisons are presented in Table 2. In Figure 1 we show the cost of the algorithm. We note LS-Outlier

consistently outperforms Lloyd-Outlier in terms of cost and also in terms of precision and recall.

We highlight that the number of outliers thrown out by LS-Outlier, denoted by $|Z|$, is exactly z in all the cases. This leads to precision and recall being identical in this dataset .

Table 1: Results on synthetic data, $n = 10^4, d = 15$.

k	z	$\lvert Z\rvert$	LS-Outlier		Lloyd-Outlier	
			Prec.	Rec.	Prec.	Rec.
10	25	25	1	1	0.984	0.984
	50	50	1	1	0.993	0.993
	100	100	1	1	0.994	0.994
20	25	25	1	1	0.987	0.987
	50	50	1	1	0.992	0.992
	100	100	1	1	0.967	0.967

Next, we compare the results of our algorithms to the two-step method of LOF-Outlier. Computing LOF for each point is considerably slower, and hence for this experiment we generate a smaller dataset of $n = 1000$, and $d = 2$. The results are shown in Table 3. Observe, the solution returned by our algorithm is significantly better than LOF-Outlier both in terms of outlier detection accuracy, as well as clustering quality. We also note that when $z = 100$ the algorithm marked $|Z| = 120$ points of outliers, demonstrating the bicriteria approximation behavior. That said, it was still significantly better than the LOF-Outlier with the same number of outliers.

Table 2: Comparison with LOF-Outlier on synthetic data, $n = 1000, d = 2, k = 20$.

z	$\lvert Z\rvert$	LS-Outlier			LOF-Outlier		
		Prec.	Rec.	cost	Prec.	Rec.	cost
25	25	0.94	0.94	2047	0.84	0.84	2434
50	50	0.91	0.91	2158	0.91	0.91	6419
100	120	0.72	0.91	2077	0.58	0.67	10853

The final comparison we perform is against the LR method. We generate the synthetic data in the same manner as for the previous experiments. The LR algorithm accepts a cost for center creation and also a number of outliers. As output the algorithm generates a clustering and the number of centers. To ensure a fair comparison, our algorithms are run with k equal to the number of clusters generated by LR, even though the proper number of clusters is known when generating the synthetic data. These results are shown in Table 4. We note that our algorithm rarely reports more outliers than given. Additionally, the LS-Outlier consistently performs better than the LR algorithm in terms of cost.

4.2 Real data

Next we discuss performance of our algorithm on real datasets. The datasets we consider are: `SUSY, Power, Skin, Shuttle`, and `Covertype` available in the UCI Machine Learning Repository [28]. The datasets have 5M, 2M, 245K, 43K, and 11K instances respectively. We run tests by considering outliers in two different ways discussed below.

Table 3: Cost comparison with LR on synthetic data.

k	z	$\lvert Z\rvert$	LS-Out. cost	Lloyd-Out. cost	LR cost
21	25	25	16.54	26.21	16.29
23	50	50	16.54	40.32	28.79
25	100	163	14.79	48.74	35.28

Unfortunately the LOF-Outlier and LR algorithms scale poorly to larger datasets, therefore we only include comparisons to Lloyd-Outlier algorithm.

Small clusters as outliers. The `Shuttle` training dataset contains 43,500 instances. In total there are 7 classes. 99.6% of the data belongs to the four largest classes, while the two smallest classes together contain only 17 data points. We aim to detect these points as outliers. Thus we set the number of outliers to 17.

The results of our algorithm on this is shown in Table 5. As noted in [12], small clusters in this dataset are inherently difficult to separate from the large clusters. Given that, the precision and recall scores are fairly good (especially for $k = 15$), and the method consistently outperforms that of [12].

Table 4: Results on `Shuttle` dataset for $z = 17$.

k	$\lvert Z\rvert$	LS-Outlier		Lloyd-Outlier	
		Prec.	Rec.	Prec.	Rec.
5	21	0.176	0.212	0.159	0.176
10	34	0.176	0.353	0.132	0.294
15	51	0.182	0.553	0.158	0.494

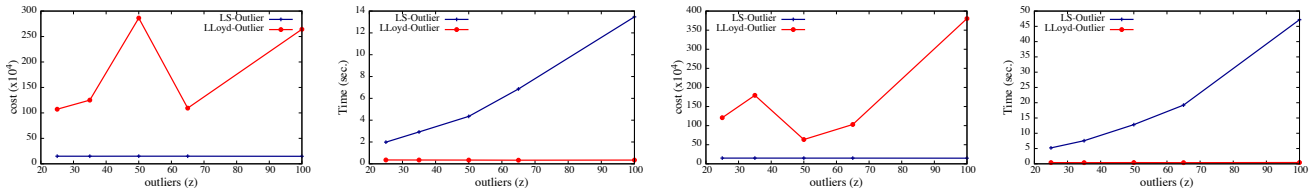
Random noise outliers. We test the performance of our algorithm when random noise is added to real data. We perform this experiment on `SUSY`, `Power`, `Skin`, and `Covertype`. We generate outliers as follows: randomly choose z points from the dataset. For each of those points, we add a uniform random noise on each of its dimensions.

The results of the precision and recall of our algorithm on the different datasets is shown in Tables 6–9. Notice that the precision and recall are very high for all the datasets and are consistently better than the baseline.

Table 5: Results on `SUSY` dataset.

k	z	$\lvert Z\rvert$	LS-Outlier		Lloyd-Outlier	
			Prec.	Rec.	Prec.	Rec.
10	25	25	1	1	0.964	0.964
	50	50	1	1	0.974	0.974
	100	100	1	1	0.981	0.981
20	25	25	1	1	0.936	0.936
	50	50	1	1	0.964	0.964
	100	100	1	1	0.958	0.958

The cost of the algorithm's solution can be found in Figure 2 and Figure 3. Notice the strong performance in the objective over the baseline method. The running time comparisons are shown in Figure 4 and Figure 5. Although



(a) Cost: $k = 10$ (b) Time: $k = 10$ (c) Cost: $k = 20$ (d) Time: $k = 20$

Figure 1: Comparison on synthetic data, $n = 10^4, d = 15$.

Table 6: Results on `Power` dataset.

k	z	$\lvert Z\rvert$	LS-Outlier		Lloyd-Outlier	
			Prec.	Rec.	Prec.	Rec.
10	25	25	1	1	0.960	0.960
	50	50	1	1	0.964	0.964
	100	100	0.994	0.994	0.982	0.982
20	25	25	0.984	0.984	0.864	0.864
	50	50	0.992	0.992	0.948	0.948
	100	100	0.994	0.994	0.962	0.962

Table 7: Results on `Skin` dataset.

k	z	$\lvert Z\rvert$	LS-Outlier		Lloyd-Outlier	
			Prec.	Rec.	Prec.	Rec.
10	25	25	0.904	0.904	0.848	0.848
	50	50	0.886	0.886	0.826	0.826
	100	120	0.768	0.864	0.752	0.846
20	25	25	0.92	0.92	0.824	0.824
	50	50	0.864	0.864	0.832	0.832
	100	100	0.878	0.878	0.856	0.856

LS-Outlier is slower than Lloyd-Outlier, it runs in reasonable time even for significantly large dataset like `SUSY`.

In conclusion, even in this random noise model our algorithm performs better than the baseline algorithm in almost all cases and in many cases significantly better.

4.3 Summary

Thus, our key experimental findings are the following.

- LS-Outlier has high precision and recall. On synthetic data both are always at least 90%. On real data, both precision and recall are better than the Lloyd-Outlier method. The algorithm also performs better than the LR method.
- The k-means objective cost of our algorithm is always strictly better than the baseline, even though both are local search methods, and both start with a solution seeded by k-means++. In fact, the cost is almost always 50% better on synthetic data and usually over 20% better on real data.
- On average, the algorithm discards fewer than $2z$ outliers. This demonstrates that in practice the number of outliers discarded is far better than Theorem 3 predicts.

Table 8: Results on `Covertype` dataset.

k	z	$\lvert Z\rvert$	LS-Outlier		Lloyd-Outlier	
			Prec.	Rec.	Prec.	Rec.
10	25	25	1	1	0.780	0.780
	50	50	1	1	0.878	0.878
	100	100	1	1	0.943	0.943
20	25	25	1	1	0.532	0.532
	50	50	1	1	0.764	0.764
	100	100	1	1	0.880	0.880

We believe these results strongly support using LS-Outlier for k-means clustering with outliers. It has significantly improved cost and more accurately identifies outliers in real data sets. Further, the number of outliers discarded is close to the input parameter.

5. ANALYSIS OF CLUSTERING QUALITY

In this section we present a complete analysis of the clustering quality of LS-Outlier. In particular, our goal is to prove the cost bound in Theorem 3, i.e., we show that the cost is at most a constant factor larger than the cost of the optimal solution.

The following fact will be useful throughout the analysis. It is known as the *relaxed triangle inequality*.

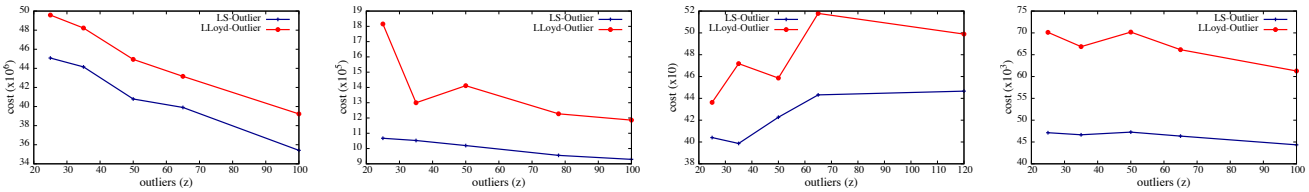
FACT 1 (RELAXED TRIANGLE INEQUALITY). *For any points u, v, w in a metric space U it is the case that $2(d(u,v)^2 + d(v,w)^2) \geq d(u,w)^2$.*

PROOF. Fix any $u, v, w \in U$. We have that $d(u,w)^2 \leq (d(u,v) + d(v,w))^2$ since we know that $d(u,w) \leq d(u,v) + d(v,w)$ by definition of a metric space. Further, $(d(u,v) + d(v,w))^2 \leq d(u,v)^2 + d(v,w)^2 + 2d(u,v)d(v,w)$. It is easy to verify that $2d(u,v)d(v,w) \leq d(u,v)^2 + d(v,w)^2$, which completes the proof. □

For the remainder of the proof fix $C = \{a_1, \ldots a_k\}$ to be the set of centers, and Z to be the set of outliers at the end of the algorithm. Let C^* and Z^* be the centers and outliers of a fixed optimal solution.

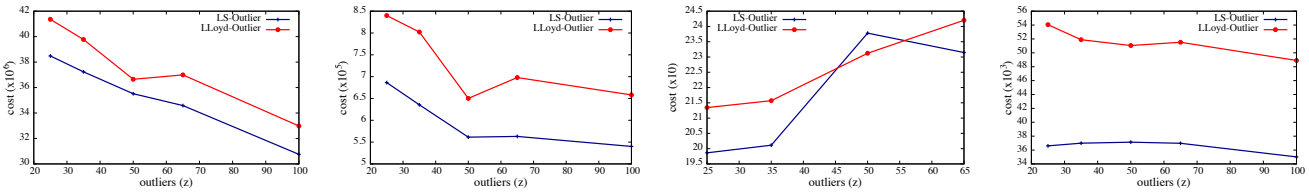
5.1 Local optimality

The key property of LS-Outlier and effectively the only inequality we can rely on in the proof is that of local optimality.



(a) SUSY (b) Power (c) Skin (d) Covertype

Figure 2: K-means cost of clustering real datasets for $k = 10$.



(a) SUSY (b) Power (c) Skin (d) Covertype

Figure 3: K-means cost of clustering real datasets for $k = 20$.

PROPOSITION 5 (LOCAL OPTIMALITY). *(i) It is the case that*

$$\Big(cost(C, Z \cup outliers(C)) - cost(C, Z)\Big) \geq -\frac{\epsilon}{k} cost(C, Z).$$

(ii) Further, for any $u \in U$ and $v \in C$ it is the case that

$$\Big(cost(C \cup \{u\} \setminus \{v\}, Z \cup outliers(C \cup \{u\} \setminus \{v\}))) - cost(C, Z)\Big) \geq -\frac{\epsilon}{k} cost(C, Z).$$

Property (i) states that the solution cannot be improved by more than a $(1 - \frac{\epsilon}{k})$ factor even if we allow an additional z outliers added to Z. Property (ii) states that for any swap pair $u \in U$ and $v \in C$, it is the case that the solution does not improve by a $(1 - \frac{\epsilon}{k})$ factor if we swap u and v as centers and remove an additional z outliers. Both properties follow from the local search termination conditions of the algorithm, the **while** loop in LS-Outlier (Algorithm 2).

Our goal is to find a meaningful way of applying Proposition 5 to bound the cost of LS-Outlier by the optimal solution's cost. Our analysis is inspired by the elegant analysis of the problem without outliers [20, 24].

5.2 Swap pairs and capture

We define a set of k *swap pairs* of the form (a_i, b_j). For each of these swap pairs, we will apply Proposition 5 and by summing the inequalities over the pairs we will be able to bound the cost of LS-Outlier by the optimal solution.

To define the swap pairs, we require the following definition. For $a \in C$ let

$$N(a) = \{v \mid a = \arg\min_{a' \in C} d(v, a')\},$$

denote all the points in the cluster corresponding to point a in the output of LS-Outlier. Similarly, for $b \in C^*$ let

$$N^*(b) = \{v \mid b = \arg\min_{b' \in C^*} d(v, b')\},$$

denote all the points in the cluster corresponding to point b in the optimum.

DEFINITION 6 (CAPTURE). *For $a \in C$ and $b \in C^*$ we say that a* captures *b if*

$$|N(a) \cap (N^*(b) \setminus Z)| > \frac{1}{2}|N^*(b) \setminus Z)|.$$

Note that $b \in C^*$ can be captured by at most one $a \in C$.

We define a set $\mathcal{P}$ of swap pairs as follows.

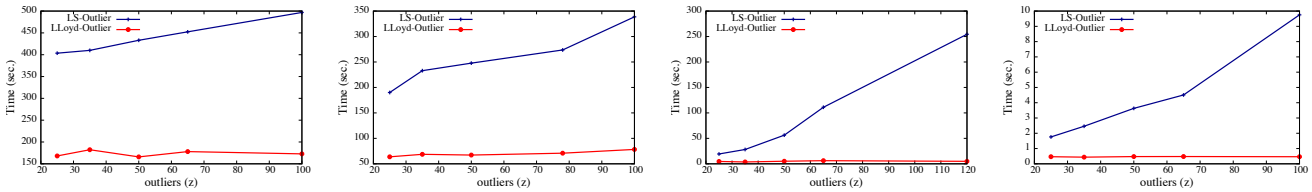
- If $a \in C$ captures exactly one $b \in C^*$, then add (a, b) to $\mathcal{P}$.
- For each $a \in C$ that captures no $b \in C^*$, add the pair (a, b) to $\mathcal{P}$ for any $b \in C^*$ such that (i) b is not already included in a swap pair and (ii) each $a \in C$ is involved in at most two swap pairs.

Properties of swap pairs. The key to this definition of swap pairs is ensuring that it has the following properties.

1. Each $b \in C^*$ is involved in exactly one swap pair;
2. Each $a \in C$ is involved in at most two swap pairs;
3. If $(a, b) \in \mathcal{P}$ then a captures no $b' \in C^* \setminus \{b\}$.

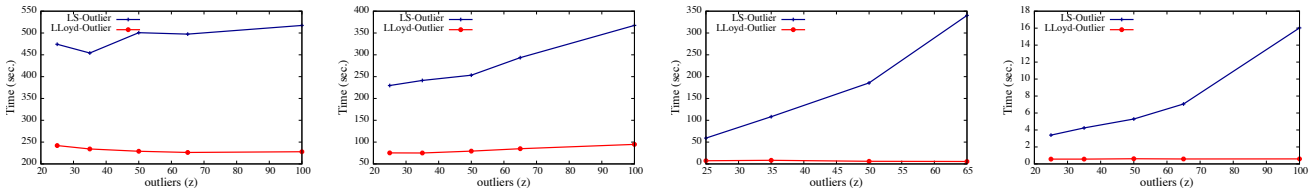
The second and third properties follow by definition of the swap pairs. The first property is non-obvious. We will show that indeed there are enough centers in C that capture no center in C^* to ensure that every $b \in C^*$ is involved in one swap pair. To do so, let β denote the number of centers in C that capture exactly one center in C^*.

LEMMA 7. *There are at least $\frac{k-\beta}{2}$ centers in C that capture no center in C^*, where β is the number of of centers in C that capture exactly one center in C^*.*



(a) SUSY (b) Power (c) Skin (d) Covertype

Figure 4: Running Time of clustering real datasets for $k = 10$.



(a) SUSY (b) Power (c) Skin (d) Covertype

Figure 5: Running Time of clustering real datasets for $k = 20$.

PROOF. Let t be the number of centers in C that capture two or more centers in C^* and x be the number that capture none. We know that $2t + \beta \leq k$ since each center in C^* can be captured by at most one center in C. This implies that $t \leq \frac{k-\beta}{2}$. We further know that $t + x + \beta = k$ as this counts all centers in C. Hence we have that

$$k = t + x + \beta \geq \frac{k-\beta}{2} + x + \beta.$$

This implies $x \geq \frac{k-\beta}{2}$ proving the lemma. □

Lemma 7 thus states that there are $\frac{k-\beta}{2}$ centers in C that capture no center in C^*. Since we allow each a to be involved in two swap pairs, this ensures that every center in b will be included in some swap pair. This can be use to derive that every center in C^* will be included in some swap pair, since we allow each center in C to be involved in at most two swap pairs. However, it may not be the case that every center in C is involved in a swap pair.

Now that we have swap pairs with the desirable properties, we will now focus on using Proposition 5 to bound the cost of LS-Outlier. For brevity, we define

$$cost(a, b) := cost(C \cup \{b\} \setminus \{a\}, Z \cup outliers(C \cup \{b\} \setminus \{a\})).$$

The following is implied by applying Proposition 5 and summing over all k swap pairs.

$$\sum_{(a,b)\in\mathcal{P}} (cost(a, b) - cost(C, Z)) \geq -\epsilon \cdot cost(C, Z).$$

5.3 Bounding the cost of a swap

Our goal is to bound $cost(a, b)$ in a meaningful way. It is challenging to argue about this value directly, instead, we define a clustering using centers $C \cup \{b\} \setminus \{a\}$ that has cost larger than $cost(a, b)$, which closely resembles the clustering of LS-Outlier. By doing this and the fact that the swap pairs include every center in the optimal solution, we will be able to show that the solution of LS-Outlier closely resembles the optimal solution. This is where we use the notion of capture (Definition 6). Intuitively, the definition gives us control on which clusters obtained by LS-Outlier closely resemble some cluster in the optimal solution and which clusters resemble no cluster in the optimal solution. We show that $\sum_{(a,b)\in\mathcal{P}} cost(a, b)$ is bounded by $O(cost(C^*, Z^*))$, the optimal solution's cost. Knowing that $cost(C, Z)$ is cost of LS-Outlier, the inequality implies that the algorithm is a constant-factor approximation, for constant ϵ. The full proof is much more involved because the assignment of points to centers after doing a swap is delicate.

Defining a permutation. To begin we first require another crucial definition. We define a permutation π on the points in $U \setminus (Z \cup Z^*)$. Consider ordering all of the points in $U \setminus (Z \cup Z^*)$ such that those belonging to $N^*_{b_i}$ come before $N^*_{b_{i+1}}$ for all $i = 1, 2, \ldots, k-1$. For points in $N^*_{b_i}$ for all i we further define an ordering on them where the points N_{a_j} come before $N_{a_{j+1}}$ for $j = 1, 2, \ldots, k-1$. This defines the permutation.

Using the permutation, we define a mapping between the points. For a point $v \in N^*_{b_i}$ let $\pi(v)$ denote its diametrically opposite point in $N^*_{b_i}$ according the above permutation.

Cost of a swap. We are ready to show an upper bound on $cost(a, b)$. To do this, using the centers in $C \cup \{b\} \setminus \{a\}$ we will define a suboptimal assignment of points in $U \setminus Z$ to centers and choose z additional outliers. The cost of this assignment is clearly only larger than $cost(a, b)$, since $cost(a, b)$ is defined using the optimal assignment and the optimal z additional outliers.

Using the above permutation definition, we can define how to bound the cost of a solution after doing a swap. Consider a swap pair $(a, b) \in \mathcal{P}$. We map points in U to centers in $C \cup \{b\} \setminus \{a\}$ in the following way. Note that there are at most $|Z| + z$ outliers in this clustering.

- Each $v \in Z \setminus N^*_b$ and each $v \in Z^*$ is an outlier.
- Each $v \in N^*_b$ is mapped to the center b.

- Each $v \in N_{a',a' \neq a} \setminus (Z^* \cup N_b^*)$ is mapped to a'.
- Each $v \in N_a \setminus (Z^* \cup N_b^*)$ is mapped to the center for $\pi(v)$ in the solution obtained by LS-Outlier. We will show that this center must not be a and therefore is in $C \cup \{b\} \setminus \{a\}$.

First we show that this mapping is feasible. In particular, we show that for all $v \in N_a \setminus (Z^* \cup N_b^*)$ it is the case that $\pi(v)$'s center in the solution obtained by LS-Outlier is not a.

LEMMA 8. *For any point $v \in N_a \setminus (Z^* \cup N_b^*)$ we have $\pi(v) \in N_{a'}$ for some $a' \neq a$.*

PROOF. By definition π is permutation on points in $U \setminus (Z \cup Z^*)$. In particular, this implies that $\pi(v)$ is not an outlier either in LS-Outlier or in the optimum. We know that $v \in N_{b'}^*$ for some $b' \neq b$ by assumption. Using the permutation π, v is mapped to the diametrically opposite point $\pi(v)$ in $N_{b'}^*$. Knowing that a does not capture any $b' \neq b$ by definition of swap pairs, it is the case that $\pi(v)$ must not be in N_a. Since $\pi(v)$ is not an outlier, this implies the lemma. □

Having concluded that the above definition of the clustering is feasible, we now bound its cost in the following manner; we use the fact that outliers have no cost. In the following, let $\pi^c(v)$ denote the center for $\pi(v)$ and let v^c denote v's center in the solution obtained by LS-Outlier.

$$\begin{aligned}
& cost(a,b) - cost(C,Z) \\
\leq & \sum_{v \in N_b^*} d(v,b)^2 + \sum_{v \in N_{a',a' \neq a}} d(v,a')^2 \\
& \quad + \sum_{v \in N_a \setminus (Z^* \cup N_b^*)} d(v,\pi^c(v))^2 - \sum_{v \in U \setminus Z} d(v,v^c)^2 \\
\leq & \sum_{v \in N_b^*} d(v,b)^2 - \sum_{v \in N_b^* \setminus Z} d(v,v^c)^2 \\
& \quad + \sum_{v \in N_{a',a' \neq a} \setminus (Z^* \cup N_b^*)} (d(v,a')^2 - d(v,v^c)^2) \\
& \quad + \sum_{v \in N_a \setminus (Z^* \cup N_b^*)} (d(v,\pi^c(v))^2 - d(v,v^c)^2).
\end{aligned}$$

We will bound each of these terms separately in the following lemmas over *all swap pairs*. For the third term, $\sum_{v \in N_{a',a' \neq a} \setminus (Z^* \cup N_b^*)} (d(v,a')^2 - d(v,v^c)^2)$, notice that v^c is a' for all v's in the summation. Thus, this term drops out. Now consider the first two terms.

LEMMA 9. *It is the case that $\sum_{(a,b) \in \mathcal{P}} \left(\sum_{v \in N_b^*} d(v,b)^2 - \sum_{v \in N_b^* \setminus Z} d(v,v^c)^2 \right) \leq cost(C^*,Z^*) - (1 - \frac{\epsilon}{k}) cost(C,Z)$.*

PROOF.

$$\begin{aligned}
& \sum_{(a,b) \in \mathcal{P}} \left(\sum_{v \in N_b^*} d(v,b)^2 - \sum_{v \in N_b^* \setminus Z} d(v,v^c)^2 \right) \\
\leq & \quad cost(C^*,Z^*) - \sum_{(a,b) \in \mathcal{P}} \sum_{v \in N_b^* \setminus Z} d(v,v^c)^2 \\
& (\because \text{each } b \text{ appears once in a swap pair in } \mathcal{P})
\end{aligned}$$

We can further bound this as:

$$\begin{aligned}
\leq & \quad cost(C^*,Z^*) - \sum_{v \in U \setminus Z} d(v,v^c)^2 + \sum_{v \in Z^* \setminus Z} d(v,v^c)^2 \\
= & \quad cost(C^*,Z^*) - cost(C,Z) + \sum_{v \in Z^* \setminus Z} d(v,v^c)^2 \\
\leq & \quad cost(C^*,Z^*) - cost(C,Z) + \frac{\epsilon}{k} cost(C,Z) \\
& (\because |Z^* \setminus Z| \leq z \text{ and Proposition 5 }). \quad \square
\end{aligned}$$

We can now consider the last term. The following algebraic fact will be useful in bounding the complex expression.

FACT 2. *For any positive real numbers x, y, z and parameter $0 < \delta \leq 1$ it is the case that $(x+y+z)^2 \leq (1+\frac{2}{\delta})(x+y)^2 + (1+2\delta)z^2$.*

PROOF.

$$\begin{aligned}
& (x+y+z)^2 \\
& = (x+y)^2 + z^2 + 2z(x+y) \\
& \leq \left(1 + \frac{2}{\delta}\right)(x+y)^2 + (1+2\delta)z^2 \\
& (\because \text{either } x+y \leq \delta z \text{ or } x+y > \delta z \text{ }). \quad \square
\end{aligned}$$

Now we are ready to bound the final term over all swap pairs.

LEMMA 10. *$\sum_{(a,b) \in \mathcal{P}} \sum_{v \in N_a \setminus (Z^* \cup N_b^*)} (d(v,\pi^c(v))^2 - d(v,v^c)^2) \leq (8 + \frac{16}{\delta}) cost(C^*,Z^*) + 4\delta cost(C,Z)$ for any $0 < \delta \leq 1$.*

PROOF. Fix any $(a,b) \in \mathcal{P}$. Notice that $d(v,\pi^c(v)) \leq d(v,\pi(v)) + d(\pi(v),\pi^c(v)) \leq d(v,b) + d(b,\pi(v)) + d(\pi(v),\pi^c(v))$ where both inequalities follow from the triangle inequality.

Consider any $v \in U \setminus (Z \cup Z^*)$. For any such point v, $\pi(v)$ is well defined. Notice that $d(v,b)+d(b,\pi(v))+d(\pi(v),\pi^c(v))-d(v,v^c) \geq 0$. This is because the first three terms form an upper bound on the distance from v to a center in $C \setminus \{v^c\}$ and v is assigned to center v^c in the clustering obtained by LS-Outlier (the closest center to v in C).

Using the above, we have the following.

$$\begin{aligned}
& \sum_{(a,b) \in \mathcal{P}} \sum_{v \in N_a \setminus (Z^* \cup N_b^*)} (d(v,\pi^c(v))^2 - d(v,v^c)^2) \\
\leq & \sum_{(a,b) \in \mathcal{P}} \sum_{v \in N_a \setminus (Z^* \cup N_b^*)} \Big(\big(d(v,b) + d(b,\pi(v)) \\
& \qquad + d(\pi(v),\pi^c(v)) \big)^2 - d(v,v^c)^2 \Big) \\
\leq & \, 2 \sum_{v \in U \setminus (Z^* \cup Z)} \Big(\big(d(v,b) + d(b,\pi(v)) \\
& \qquad + d(\pi(v),\pi^c(v)) \big)^2 - d(v,v^c)^2 \Big) \\
& (\because \text{each } a \in C \text{ appears at most twice and every term is positive from the above})
\end{aligned}$$

Continuing, we bound the sum as:

$$\leq 2 \sum_{v \in U \setminus (Z^* \cup Z)} \left(\left(1 + \frac{2}{\delta}\right) \left(d(v,b) + d(b,\pi(v))\right)^2 + (1+2\delta) d(\pi(v), \pi^c(v))^2 - d(v,v^c)^2 \right) \quad (\because \text{Fact 2})$$

$$\leq 2 \sum_{v \in U \setminus (Z^* \cup Z)} \left(\left(2 + \frac{4}{\delta}\right) (d(v,b)^2 + d(b,\pi(v))^2) + (1+2\delta) d(\pi(v), \pi^c(v))^2 - d(v,v^c)^2 \right) \quad (\because \text{Fact 1}).$$

To complete the lemma consider the value of $\sum_{v \in U \setminus (Z^* \cup Z)} \left(d(\pi(v), \pi^c(v))^2 - d(v,v^c)^2\right)$. This summation is over all the points considered in the permutation π and each point v in the permutation is mapped to by exactly one other point that is diametrically opposite. Due to this, each point $v \in U \setminus (Z^* \cup Z)$ contributes $d(v,v^c)^2$ once in the first term and once in the second term. Thus,

$$\sum_{v \in U \setminus (Z^* \cup Z)} \left(d(\pi(v), \pi^c(v))^2 - d(v,v^c)^2\right) = 0.$$

This argument implies the following.

$$\begin{aligned} & 2 \sum_{v \in U \setminus (Z^* \cup Z)} \left(\left(2 + \frac{4}{\delta}\right) (d(v,b)^2 + d(b,\pi(v))^2) + (1+2\delta) d(\pi(v), \pi^c(v))^2 - d(v,v^c)^2 \right) \\ \leq \; & 2 \sum_{v \in U \setminus (Z^* \cup Z)} \left(\left(2 + \frac{4}{\delta}\right) (d(v,b)^2 + d(b,\pi(v))^2) \right) + 2 \sum_{v \in U \setminus (Z^* \cup Z)} 2\delta d(\pi(v), \pi^c(v))^2 \\ \leq \; & \left(8 + \frac{16}{\delta}\right) cost(C^*, Z^*) + 4\delta cost(C, Z). \quad \square \end{aligned}$$

5.4 The final step

We can now put all of these arguments together and bound the cost of LS-Outlier.

THEOREM 11. LS-Outlier *is an $O(1)$-approximation algorithm for any fixed $0 < \epsilon \leq 1/4$.*

PROOF. Recall the main bound we have:

$$\sum_{(a,b) \in \mathcal{P}} (cost(a,b) - cost(C,Z)) \geq -\epsilon cost(C,Z).$$

By Lemmas 9 and 10 we have the following.

$$\begin{aligned} & \left(8 + \frac{16}{\delta}\right) cost(C^*, Z^*) + 4\delta \cdot cost(C,Z) + cost(C^*, Z^*) - \left(1 - \frac{\epsilon}{k}\right) cost(C,Z) \\ & \geq -\epsilon \cdot cost(C,Z) \end{aligned}$$

Combing these inequalities gives

$$\frac{9 + \frac{16}{\delta}}{1 - 4\delta - \epsilon - \frac{\epsilon}{k}} cost(C^*, Z^*) \geq cost(C,Z). \quad \square$$

Choosing $\delta = \frac{2\sqrt{79} - 16}{9}$ and ϵ to be sufficiently small establishes that the algorithm obtains a 274-approximation for the k-means objective. If the objective is the ℓ_2-norm of the distances of the points to centers, then the algorithm obtains a 17-approximation.

6. OTHER RELATED WORK

Clustering algorithms have been studied for decades in several communities, even outside core computer science. A thorough review of the relevant literature, especially in the context of data management and data processing, is beyond the scope of our work. In addition to the book [3], there are many surveys available [8, 19, 23].

The k-means problem with outliers, which is the main topic of our paper, has been studied from both theoretical and heuristic points of view. Charikar et al. [11] and Chen [13] proposed constant-factor approximation algorithms for the problem. Unfortunately these algorithms are too complicated to be effective in practice. Our algorithm LS-Outlier, on the other hand, has similar theoretical guarantees while being simple to implement. Chawla and Gionis [12] extended Lloyd's algorithm to handle outliers, but they provide no guarantees on the quality of the clustering, only showing that it reaches a local optimum.

Some papers have focused on other clustering objectives while taking into account the effect of outliers. Ester et al. [14] proposed DBSCAN, a heuristic that combines clustering and outlier removal. However, DBSCAN suffers from a number of problems, as noted in earlier work [18]. It is very sensitive to parameters, not robust, can have an arbitrary number of clusters and outliers, and needs to know the radius of each cluster. Hence its requirements are outside the scope of our methods. Bohm et al. [9] extended independent component analysis to make clustering robust to outliers; once again, their method does not have any provable guarantees for the quality of the resulting clustering. There is a huge body of work dealing with first finding outliers, i.e., separating the outlier detection from the clustering objective [2, 4, 7, 25, 26, 30, 37, 38]. Our work, on the other hand, treats clustering and outlier removal in an integrated fashion as done in [22, 9, 14].

Implementations of k-means algorithms have been studied in the data engineering community, especially in the context of better integration with an existing relational database management system. Ordonez and Omiecinski [35] considered an efficient implementation of k-means that is based on disk. Subsequently, Ordonez [33, 34] studied an SQL implementation of k-means. We believe our algorithm LS-Outlier, owing to its simplicity, can be easily integrated in existing database management systems.

Scalability issues regarding k-means have also been studied in the past. Farnstrom et al. [15] used compression-based techniques to obtain a single-pass algorithm for k-means. Bahmani et al. [6] extended k-means++ to work in the MapReduce model. The initial portion of our algorithm LS-Outlier can use this extension, to make it scalable. The literature on the scalability and the implementation of k-means in non-traditional computational models such as streaming and message passing are too numerous to be listed and are outside the scope of our work.

7. CONCLUSIONS

In this paper we considered the problem of k-means with outliers and obtained a new algorithm that is based on local search. Our algorithm is simple, practical, and can be adapted to scale for large data; in addition, it has provable performance guarantees. Experiments indicate that this algorithm can outperform recently proposed heuristic approaches for the problem on both synthetic and real data.

There are several interesting future research directions. First, if the data points are from the Euclidean space (as opposed to a general metric space), can the guarantees be stronger or the analysis simpler? Second, can we close the gap between the desired number of outliers and the actual number of outliers found by the algorithm? Our experimental evaluation suggests that the algorithm performs much better than the analysis seems to indicate. Finally, are there good data structures and sketching techniques to further speed up the local search and update steps?

8. REFERENCES

[1] A. Aggarwal, A. Deshpande, and R. Kannan. Adaptive sampling for k-means clustering. In *RANDOM*, pages 15–28, 2009.

[2] C. Aggarwal and P. Yu. Outlier detection for high dimensional data. In *SIGMOD*, pages 37–46, 2001.

[3] C. C. Aggarwal and C. K. Reddy. *Data Clustering: Algorithms and Applications*. Chapman and Hall/CRC, 2013.

[4] A. Arning, R. Agrawal, and P. Raghavan. A linear method for deviation detection in large databases. In *KDD*, pages 164–169, 1996.

[5] D. Arthur and S. Vassilvitskii. k-means++: the advantages of careful seeding. In *SODA*, pages 1027–1035, 2007.

[6] B. Bahmani, B. Moseley, A. Vattani, R. Kumar, and S. Vassilvitskii. Scalable k-means++. *PVLDB*, 5(7):622–633, 2012.

[7] S. Bay and M. Schwabacher. Mining distance-based outliers in near linear time with randomization and a simple pruning rule. In *KDD*, pages 29–38, 2003.

[8] P. Berkhin. Survey of clustering data mining techniques. In J. Kogan, C. K. Nicholas, and M. Teboulle, editors, *Grouping Multidimensional Data: Recent Advances in Clustering*. Springer, 2006.

[9] C. Bohm, C. Faloutsos, and C. Plant. Outlier-robust clustering using independent components. In *SIGMOD*, pages 185–198, 2008.

[10] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander. LOF: Identifying density-based local outliers. *SIGMOD Record*, 29(2):93–104, 2000.

[11] M. Charikar, S. Khuller, D. M. Mount, and G. Narasimhan. Algorithms for facility location problems with outliers. In *SODA*, pages 642–651, 2001.

[12] S. Chawla and A. Gionis. k-means-: A unified approach to clustering and outlier detection. In *ICDM*, pages 189–197, 2013.

[13] K. Chen. A constant factor approximation algorithm for k-median clustering with outliers. In *SODA*, pages 826–835, 2008.

[14] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *AAAI*, pages 226–231, 1996.

[15] F. Farnstrom, J. Lewis, and C. Elkan. Scalability for clustering algorithms revisited. *SIGKDD Explor. Newsl.*, 2:51–57, 2000.

[16] D. Feldman, M. Monemizadeh, and C. Sohler. A PTAS for k-means clustering based on weak coresets. In *SOCG*, pages 11–18, 2007.

[17] S. Guha, A. Meyerson, N. Mishra, R. Motwani, and L. O'Callaghan. Clustering data streams: Theory and practice. *TKDE*, 15(3):515–528, 2003.

[18] S. Guha, R. Rastogi, and K. Shim. CURE: An efficient clustering algorithm for large databases. In *SIGMOD*, pages 73–84, 1998.

[19] S. Guinepain and L. Gruenwald. Research issues in automatic database clustering. *SIGMOD Record*, 34(1):33–38, 2005.

[20] A. Gupta and K. Tangwongsan. Simpler analyses of local search algorithms for facility location. *CoRR*, abs/0809.2554, 2008.

[21] S. Har-Peled and A. Kushal. Smaller coresets for k-median and k-means clustering. *Discrete & Computational Geometry*, 37(1):3–19, 2007.

[22] A. Hinneburg and D. A. Keim. An efficient approach to clustering in large multimedia databases with noise. In *KDD*, pages 58–65, 1998.

[23] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: A review. *ACM Computing Surveys*, 31:264–323, 1999.

[24] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu. A local search approximation algorithm for k-means clustering. *Comput. Geom.*, 28(2-3):89–112, 2004.

[25] E. Knorr and R. Ng. Algorithms for mining distance-based outliers in large datasets. In *VLDB*, pages 392–403, 1998.

[26] E. M. Knorr and R. T. Ng. A unified notion of outliers: Properties and computation. In *KDD*, pages 19–22, 1997.

[27] A. Kumar, Y. Sabharwal, and S. Sen. A simple linear time $(1+\epsilon)$-approximation algorithm for k-means clustering in any dimensions. In *FOCS*, pages 454–462, 2004.

[28] M. Lichman. UCI machine learning repository, 2013.

[29] S. P. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–136, 1982.

[30] K. E. M. and N. R. T. Finding intensional knowledge of distance-based outliers. In *VLDB*, pages 211–222, 1999.

[31] G. Malkomes, M. Kusner, W. Chen, K. Weinberger, and B. Moseley. Fast distributed k-center clustering with outliers on massive data. In *NIPS*, pages 1063–1071, 2015.

[32] R. M. McCutchen and S. Khuller. Streaming algorithms for k-center clustering with outliers and with anonymity. In *APPROX*, pages 165–178, 2008.

[33] C. Ordonez. Integrating k-means clustering with a relational DBMS using SQL. *TKDE*, 18:188–201, 2006.

[34] C. Ordonez and P. Cereghini. SQLEM: Fast clustering in SQL using the EM algorithm. In *SIGMOD*, pages 559–570, 2000.

[35] C. Ordonez and E. Omiecinski. Efficient disk-based k-means clustering for relational databases. *TKDE*, 16:909–921, 2004.

[36] L. Ott, L. Pang, F. T. Ramos, and S. Chawla. On integrated clustering and outlier detection. In *NIPS*, pages 1359–1367, 2014.

[37] S. Papadimitriou, H. Kitagawa, P. Gibbons, and C. Faloutsos. LOCI: Fast outlier detection using the local correlation integral. In *ICDE*, pages 315–326, 2003.

[38] S. Ramaswamy, R. Rastogi, , and K. Shim. Efficient algorithms for mining outliers from large data sets. In *SIGMOD*, pages 427–438, 2000.

[39] X. Wu, V. Kumar, J. Ross Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. J. McLachlan, A. Ng, B. Liu, P. S. Yu, Z.-H. Zhou, M. Steinbach, D. J. Hand, and D. Steinberg. Top 10 algorithms in data mining. *Knowl. Inf. Syst.*, 14:1–37, 2007.