Streaming 1-Center with Outliers in High Dimensions

Hamid Zarrabi-Zadeh* Asish Mukhopadhyay†

Abstract

We study the 1-center problem with outliers in high-dimensional data streams. The problem definition is as follows: given a sequence of n points in d dimensions (with d arbitrarily large), enclose all but z points using a ball of minimum radius.

1 Introduction

The 1-center problem—finding the smallest ball enclosing a set of points—is a fundamental problem in computational geometry having many applications in areas like data mining, statistics, image processing, machine learning, etc.

In this paper, we consider the 1-center problem in a robust setting where at most z outliers can be dropped from the input before computing the smallest ball. In real-world applications, data is usually noisy, and this noise can hugely affect the quality of the output, if is not handled properly by the algorithm. In particular, for the 1-center problem, even one outlier can dramatically increase the size of the final solution.

We are interested in algorithms that work under the *data stream* model. In this model, the input data is given to the algorithm one at a time as a stream over time and the algorithm has only a limited amount of working space, so it cannot store all the input items received so far. This practical setting is in particular suitable for applications that involve massive data sets, as the algorithm can have only one pass over the input, and the whole data need not to be stored in memory.

In this paper, we focus on the high-dimensional version of the 1-center problem with outliers. The formal definition of the problem is as follows: given a sequence of n points in d dimensions (where d may be arbitrarily large), enclose all but z points using a ball of minimum radius.

The problem without outliers (i.e., $z = 0$) is equivalent to the well-known "minimum enclosing ball" problem, for which an offline algorithm with $O(d^{O(d)}n)$ time is available [9]. In high dimensions, Bădoiu and Clarkson [5] have given an elegant algorithm that computes a $(1 + \varepsilon)$-approximation to the minimum enclosing ball in $\lceil 2/\varepsilon \rceil$ passes using $O(nd/\varepsilon + (1/\varepsilon)^5)$ total time and $O(d/\varepsilon)$ space.

In the data stream model, a $(1 + \varepsilon)$-approximation to the minimum enclosing ball can be maintained in $O(1/\varepsilon^{(d-1)/2})$ space using coresets [1, 6] (indeed, it just suffices to keep extreme points along $O(1/\varepsilon^{(d-1)/2})$ directions). In high dimensions, where the dimension is arbitrarily large, a simple streaming algorithm is proposed by Zarrabi-Zadeh and Chan [16] that computes a 3/2-approximation to the minimum enclosing ball in any dimension using only $O(d)$ space and $O(d)$ update time. Agarwal and Sharathkuma [3] have recently shown how to maintain a $(\frac{1+\sqrt{3}}{2}+\varepsilon)$-approximation to the minimum enclosing ball in any dimension using $O((d/\varepsilon^3)\log(1/\varepsilon))$ space and $O(d/\varepsilon^5)$ update time.

For the related k-center problem, where the objective is to partition the points into k clusters of minimum maximum radius, Charikar *et al.* [7] gave an incremental algorithm that maintains a factor 8 approximation to the k-center in any metric space using $O(k)$ space. In their restricted setting, each input point is assigned to a cluster upon arrival, and once assigned, the point cannot be removed from its cluster at a later time (though clusters are free to merge at any time). Having relaxed this incremental restriction, Guha [12] and McCutchen and Khuller [14] have recently obtained $(2 + \varepsilon)$-approximation streaming algorithms for the k-center problem using $O((k/\varepsilon)\log(1/\varepsilon))$ space.

Charikar *et al.* [8] were the first who studied the k-center problem with outliers and proposed an offline algorithm with constant approximation factor in any metric space. The approximation factor of their algorithm is 3 if all center points that the algorithm is allowed to choose can be enumerated; and is 4, otherwise. Using this algorithm as a subroutine, McCutchen and Khuller [14] have recently obtained a streaming algorithm that computes a constant-factor approximation to the k-center problem with z outliers using $O(kz/\varepsilon)$ space. The approximation factor of their algorithm is $3+\varepsilon$ or $4+\varepsilon$, depending on which algorithm has been used as the subroutine.

For the 1-center problem with z outliers, one can use the "robust kernel" paradigm introduced in [13, 2] to obtain a streaming algorithm with $O(z/\varepsilon^{O(d)})$ space [4, 15]. In high dimensions, the only previous result with sub-exponential dependency on the dimension

*School of Computer Science, University of Waterloo, Waterloo, Ontario N2L 3G1, Canada, hzarrabi@uwaterloo.ca.

†School of Computer Science, University of Windsor, Windsor, Ontario N9B 3P4, Canada, asishm@uwindsor.ca. Research supported by an NESRC Discovery Grant.

is the $(4+\varepsilon)$-approximation algorithm of McCutchen and Khuller [14] that requires $O(zd/\varepsilon)$ space.

In this paper, we propose a simple streaming algorithm that maintains a 2-approximation to the 1-center problem with z outliers using $O(zd^2)$ space. Moreover, using a general framework, we achieve a streaming algorithm whose approximation factor is $\sqrt{2}$ times the approximation factor of any streaming algorithm for the minimum enclosing ball problem in high dimensions. Combined with the result of [3], our framework yields an approximation factor of $(\sqrt{3}+1)/\sqrt{2}+\varepsilon \approx 1.93+\varepsilon$ for the 1-center problem with z outliers in any dimension using $O(d^3z+(d/\varepsilon^3)\log(1/\varepsilon))$ space. For the special case of $z=1$, we propose a better streaming algorithm that yields an approximation factor of $\frac{1}{4}(1+\sqrt{2})(1+\sqrt{3})+\varepsilon \approx 1.65+\varepsilon$ in any dimension using $O((d/\varepsilon^3)\log(1/\varepsilon))$ space.

2 A Buffering Framework

The main difficulty in handling outliers in the data stream model is that no secure information about the points arriving in future is available to the algorithm, and thus, we have no way to determine which point is an outlier, and which one is not, only based on the current information.

In particular, we cannot hope for any bounded approximation factor if we require each point to be marked as outlier or non-outlier upon arrival. To see this, just consider the case where there is only one outlier, and the first two points p_1 and p_2 of the stream have arrived. If the algorithm classifies both points as non-outliers, then the approximation factor would be unbounded as one of the points could be outlier and the other one could form a cluster of zero radius. If the algorithm marks one of the points, say p_1 as outlier, then the adversary can give the third point at p_1 which again makes the approximation factor unbounded.

To overcome this deficiency, we postpone the decision about the type of the input points until we get enough information. To this end, we use a buffering framework as shown in Figure 1. At each time, our framework maintains an approximate ball B, and a buffer containing at most m points of the stream. Whenever a new input point p arrives, it is first added to the buffer. If buffer is full, a "suitable" point q is extracted from the buffer and is fed to a streaming algorithm that maintains an approximate ball B for the points not in the buffer.

Two main things to be specified in this framework are the size of the buffer, and the strategy for extracting a suitable point from the buffer in line 3. In the ideal scenario, we keep all outliers in the buffer, and extract only non-outlier points. However, this is not possible in practice because we cannot precisely distinguish between outliers and non-outliers without having all the points in advance. Therefore, we need to relax this restriction and allow outliers to be extracted from the buffer as well, provided we have a mechanism to bound the total error incurred by the wrongly extracted outliers.

```
INSERT($p$):
1:  add $p$ to the buffer
2:  if buffer is full then
3:      extract a "suitable" point $q$ from the buffer
4:      update approximate ball B by adding $q$
5:  return $B$ + buffer
```

Figure 1: The buffering framework.

Let S be the set of all points in the input stream, O be the set of outliers, and $P = S \setminus O$ be the set of non-outliers. We denote by O_x $(\subseteq O)$ the set of those outliers that are wrongly classified as non-outliers (i.e., extracted from the buffer). We call the underlying algorithm that maintains an approximate minimum enclosing ball for the points out of the buffer (in line 4 of our framework) by MEB, and denote its approximation factor by α. If we can show that the radius of the minimum ball enclosing $P \cup O_x$ is at most β times the radius of the minimum ball enclosing P, then we achieve an $(\alpha\beta)$-approximation algorithm for our problem, since

$$\tilde{r}(P \cup O_x) \leqslant \alpha r(P \cup O_x) \leqslant \alpha\beta r(P),$$

where $r(P)$ is the radius of the minimum ball enclosing P, and $\tilde{r}(\cdot)$ denotes the radius of the approximate ball computed by MEB. The best upper bound we already have for α is 3/2 [16]. In the following sections, we provide some upper bounds on β.

3 1-Center with 1 Outlier

It is easy to observe that for $m = 1$ or 2, the approximation ratio of our framework is unbounded (see the argument at the beginning of the previous section). In this section, we provide an upper bound for $m > 2$ when there is only one outlier in the system. We start with a simple lemma.

Lemma 1 *Let P be a set of $d+1$ equally distanced points in $\mathbb{R}^d$ $(d \geqslant 2)$. For each $p \in P$, the distance from p to the center of the minimum ball B enclosing $P \setminus \{p\}$ is equal to $\sqrt{\frac{d+1}{d-1}}r(B)$.*

Proof. This is immediate from the fact that the circumradius of a regular n-simplex of unit side length is $\sqrt{n/(2n+2)}$. $\square$

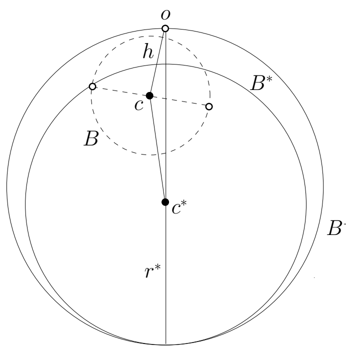

Figure 2: Handling 1 outlier.

Algorithm 1 *Let Q be the set of points in the buffer (including the new point). If buffer is full, extract a point $q \in Q$ so that $r(Q \backslash \{q\})$ is maximized.*

Theorem 2 *Algorithm 1 yields an approximation factor of $\beta = \frac{1}{2} + \frac{\sqrt{2}}{2}\sqrt{1 + \frac{1}{m-2}}$ in any dimension, using a buffer of size m (> 2).*

Proof. Let S be the set of all points in the stream, and o be the single outlier in S. If o is not extracted from the buffer, then we are done. Otherwise, let Q be the set of points in the buffer just before o is extracted. Consider the minimum ball B enclosing $Q\backslash\{o\}$. Let c and r be the center and the radius of B respectively, and let h be the distance from o to c. It is easy to verify that the ratio h/r is maximized when the distances between all the points in Q are the same. (In this case, any point of Q is equally likely to be extracted from the buffer.) Thus, we have from Lemma 1 that $h \leqslant \sqrt{\frac{m}{m-2}}r$.

Now, consider the minimum ball B^* enclosing $S\backslash\{o\}$ (i.e., the optimal solution). Let c^* and r^* be the center and the radius of B^*, respectively. Consider a point $q \in Q\backslash\{o\}$ on the boundary of B which is furthest from c^*. The angle $\angle c^*cq$ cannot be acute; otherwise all the points of $Q\backslash\{o\}$ can be enclosed by a ball smaller than B, which is a contradiction. Therefore

$$\|cc^*\| \leqslant \sqrt{\|qc^*\|^2 - \|qc\|^2} \leqslant \sqrt{r^{*2} - r^2}.$$

Let B^+ be the smallest ball enclosing both B^* and o. Obviously, $r(B^+) \leqslant \frac{1}{2}(r^* + h + \|cc^*\|)$. Therefore

$$\begin{aligned}\frac{r(S)}{r(S\backslash\{o\})} \leqslant \frac{r(B^+)}{r(B^*)} &\leqslant \frac{r^* + h + \sqrt{r^{*2} - \frac{m-2}{m}h^2}}{2r^*} \\ &\leqslant \frac{1}{2} + \frac{\sqrt{2}}{2}\sqrt{1 + \frac{1}{m-2}},\end{aligned}$$

where the maximum is attained when $r^* = h/m \times \sqrt{2(m-1)(m-2)}$. $\square$

Corollary 3 *For any fixed $\varepsilon > 0$, Algorithm 1 yields an approximation factor of $\beta = (\sqrt{2}+1)/2 + \varepsilon$ in any dimension using a buffer of size $O(1/\varepsilon^2)$.*

Combined with the streaming algorithm of [3], Corollary 3 immediately gives a factor-$(1.65 + \varepsilon)$ streaming algorithm for the 1-center problem with 1 outlier using $O((d/\varepsilon^3)\log(1/\varepsilon))$ space and $O(d/\varepsilon^5)$ update time.

4 1-Center with z Outliers

In this section, we provide a generalized algorithm for handling any number of outliers. Our algorithm uses the concept of centerpoint which is defined as follows [11]: given an n-point set P in d dimensions, a point $c \in \mathbb{R}^d$ is called a *centerpoint* of P if any halfspace containing c contains at least $\lceil n/(d+1)\rceil$ points of P. In other words, any halfspace (or convex set) that avoids a centerpoint can contain at most $\lfloor dn/(d+1)\rfloor$ points of P.

Lemma 4 *Given a point set S with z outliers, the centerpoint of any subset $P \subseteq S$ of size $(d+1)(z+1)$ lies in the minimum ball enclosing all non-outliers.*

Proof. Let c be the centerpoint of P. By the properties of centerpoints, any ball that avoids c can contain at most $\lfloor d/(d+1)|P|\rfloor = d(z+1)$ points of P, while there are at least $(d+1)(z+1) - z = d(z+1) + 1$ non-outliers in P. It means that c is contained in any ball that encloses non-outliers, and in particular, is contained in the minimum enclosing ball. $\square$

Algorithm 2 *Let $m = (d+1)(z+1)$ and Q be the set of points in the buffer. If $|Q| \geqslant m$, extract from Q a point which is closest to the centerpoint of Q.*

Theorem 5 *Algorithm 2 yields an approximation factor of $\beta = \sqrt{2}$ in any dimension.*

Proof. Let P be the set of non-outliers in the stream, and let B^* be the minimum ball enclosing P centered at c^*. Suppose that a subset O_x of the outliers is wrongly extracted from the buffer by the algorithm. Fix a point $q \in O_x$, and let Q be the set of m points in the buffer just before q is extracted. If c is the centerpoint of Q, then there is a non-outlier point $p \in Q$ so that $\angle c^*cp \geqslant \pi/2$; because otherwise, all non-outliers in Q should be contained in an open halfspace defined by the hyperplane $\mathcal{H}$ passing through c normal to $\overleftrightarrow{cc^*}$ (see Figure 3). But this is impossible by the fact that any halfspace avoiding c can contain at most $d(z+1)$ points of Q, while there are at least $d(z+1)+1$ non-outliers in Q. Therefore, $\|cc^*\| \leqslant \sqrt{\|pc^*\|^2 - \|pc\|^2}$.

Now, let r^* be the radius of B^*, and let $r = \|qc\|$. By the algorithm's selection, we have $\|pc\| \geqslant r$. Moreover, by the fact that $p \in B^*$ we have $\|pc^*\| \leqslant r^*$. Therefore,

$$\|qc^*\| \leqslant \|qc\| + \|cc^*\| \leqslant r + \sqrt{r^{*2} - r^2}.$$

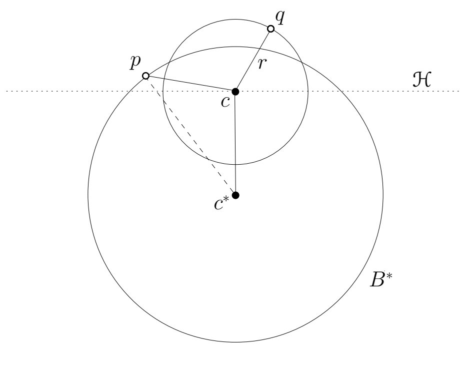

Figure 3: Proof of Theorem 5.

Since the above inequality holds for all points $q \in O_x$, we have

$$\frac{r(P \cup O_x)}{r(P)} \leqslant \frac{r + \sqrt{r^{*2} - r^2}}{r^*} \leqslant \sqrt{2},$$

where the maximum is attained when $r^* = \sqrt{2}r$. $\square$

The space used by Algorithm 2 is $O(d^2 z)$. Each update requires computing the centerpoint of a set of $O(dz)$ points in d dimensions. Computing the exact centerpoint is usually expensive (exponential in d). However, we can use the approximation algorithm of Clarkson *et al.* [10] to compute an approximate $O(1/d^2)$-centerpoint (instead of the exact $1/(d+1)$-centerpoint) in time polynomial in both d and the number of input points. Using this approximation algorithm, we get an update time polynomial in d and z, at the expense of increasing the space bound slightly to $O(d^3 z)$. Combined with the streaming algorithm of [3], we achieve an approximation factor of $\sqrt{2}((\sqrt{3}+1)/2) + \varepsilon \approx 1.93 + \varepsilon$ for the 1-center problem with z outliers in any dimension using $O(d^3 z + (d/\varepsilon^3)\log(1/\varepsilon))$ space and $\mathrm{poly}(d, z, 1/\varepsilon)$ update time.

5 A Simple 2-Approximation Algorithm

In this section, we propose a simple 2-approximation algorithm for the 1-center problem with z outliers. The algorithm is self-contained and does not depend on an underlying streaming algorithm for the minimum enclosing ball.

Algorithm 3 *Pick a centerpoint c of the first $(d+1)(z+1)$ points of the stream S, and initialize $r = 0$. For each input point $p \in S$: (i) insert p into a buffer Q; (ii) if $|Q| > z$, extract from Q a point q which is closest to c, and update r to the distance from c to q. When the end of the input is reached, return the ball B of radius r centered at c.*

Theorem 6 *Algorithm 3 is a 2-approximation streaming algorithm for handling z outliers in any dimension.*

Proof. By Lemma 4, c is contained in the minimum ball enclosing non-outliers. Let q_f be the last point extracted from the buffer, and let r_f be the last value of r. If q_f is a non-outlier, then $\overline{cq_f}$ of length r_f is contained in the minimum ball. Otherwise, if q_f is an outlier, then there is a non-outlier q in the buffer whose distance to c is at least r_f, and $\overline{cq}$ is contained in the minimum ball. It implies that in both cases, the radius of the minimum ball is at least $r_f/2$, while the radius of the ball returned by our algorithm is r_f. $\square$

The space complexity of Algorithm 3 is $O(d^2 z)$, as we need to accumulate the first $(d+1)(z+1)$ points of the stream and compute their centerpoint in the initialization step. The update time of the algorithm is the same as that of Algorithm 2 in the initialization step, and is $O(dz)$ afterwards.

References

[1] P. K. Agarwal, S. Har-Peled, and K. R. Varadarajan. Approximating extent measures of points. *J. ACM*, 51(4):606–635, 2004.

[2] P. K. Agarwal, S. Har-Peled, and H. Yu. Robust shape fitting via peeling and grating coresets. *Discrete Comput. Geom.*, 39(1-3):38–58, 2008.

[3] P. K. Agarwal and R. Sharathkumar. Streaming algorithms for extent problems in high dimensions. In *Proc. 21st ACM-SIAM Sympos. Discrete Algorithms*, 2010, to appear.

[4] P. K. Agarwal and H. Yu. A space-optimal data-stream algorithm for coresets in the plane. In *Proc. 23rd Annu. ACM Sympos. Comput. Geom.*, pages 1–10, 2007.

[5] M. Bădoiu and K. L. Clarkson. Smaller core-sets for balls. In *Proc. 14th ACM-SIAM Sympos. Discrete Algorithms*, pages 801–802, 2003.

[6] T. M. Chan. Faster core-set constructions and data stream algorithms in fixed dimensions. *Comput. Geom. Theory Appl.*, 35(1–2):20–35, 2006.

[7] M. Charikar, C. Chekuri, T. Feder, and R. Motwani. Incremental clustering and dynamic information retrieval. *SIAM J. Comput.*, 33(6):1417–1440, 2004.

[8] M. Charikar, S. Khuller, D. M. Mount, and G. Narasimhan. Algorithms for facility location problems with outliers. In *Proc. 12th ACM-SIAM Sympos. Discrete Algorithms*, pages 642–651, 2001.

[9] B. Chazelle and J. Matoušek. On linear-time deterministic algorithms for optimization problems in fixed dimension. *J. Algorithms*, 21(3):579–597, 1996.

[10] K. L. Clarkson, D. Eppstein, G. L. Miller, C. Sturtivant, and S.-H. Teng. Approximating center points with iterated radon points. In *Proc. 9th Annu. ACM Sympos. Comput. Geom.*, pages 91–98, 1993.

[11] H. Edelsbrunner. *Algorithms in combinatorial geometry.* Springer-Verlag, 1987.

[12] S. Guha. Tight results for clustering and summarizing data streams. In *Proc. 12th Internat. Conf. Database Theory*, pages 268–275, 2009.

[13] S. Har-Peled and Y. Wang. Shape fitting with outliers. *SIAM J. Comput.*, 33(2):269–285, 2004.

[14] R. M. McCutchen and S. Khuller. Streaming algorithms for k-center clustering with outliers and with anonymity. In *Proc. 11th Internat. Workshop Approx. Algorithms*, pages 165–178, 2008.

[15] H. Zarrabi-Zadeh. An almost space-optimal streaming algorithm for coresets in fixed dimensions. In *Proc. 16th Annu. European Sympos. Algorithms*, volume 5193 of *Lecture Notes Comput. Sci.*, pages 817–829, 2008.

[16] H. Zarrabi-Zadeh and T. M. Chan. A simple streaming algorithm for minimum enclosing balls. In *Proc. 18th Canad. Conf. Computat. Geom.*, pages 139–142, 2006.