Approximate Convex Hull of Data Streams

Avrim Blum[1]
TTI-Chicago, Chicago, United States
avrim@ttic.edu

Vladimir Braverman[2]
Johns Hopkins University, Baltimore, United States
vova@cs.jhu.edu

Ananya Kumar[3]
Carnegie Mellon University, Pittsburgh, United States
skywalker94@gmail.com

Harry Lang[4]
Johns Hopkins University, Baltimore, United States
hlang8@math.jhu.edu

Lin F. Yang[5]
Princeton University, Princeton, United States
lin.yang@princeton.edu

Abstract

Given a finite set of points $P \subseteq \mathbb{R}^d$, we would like to find a small subset $S \subseteq P$ such that the convex hull of S approximately contains P. More formally, every point in P is within distance ϵ from the convex hull of S. Such a subset S is called an ϵ-hull. Computing an ϵ-hull is an important problem in computational geometry, machine learning, and approximation algorithms.

In many applications, the set P is too large to fit in memory. We consider the streaming model where the algorithm receives the points of P sequentially and strives to use a minimal amount of memory. Existing streaming algorithms for computing an ϵ-hull require $O(\epsilon^{(1-d)/2})$ space, which is optimal for a worst-case input. However, this ignores the structure of the data. The minimal size of an ϵ-hull of P, which we denote by OPT, can be much smaller. A natural question is whether a streaming algorithm can compute an ϵ-hull using only $O(\mathsf{OPT})$ space.

We begin with lower bounds that show, under a reasonable streaming model, that it is not possible to have a single-pass streaming algorithm that computes an ϵ-hull with $O(\mathsf{OPT})$ space. We instead propose three relaxations of the problem for which we can compute ϵ-hulls using space near-linear to the optimal size. Our first algorithm for points in $\mathbb{R}^2$ that arrive in random-order uses $O(\log n \cdot \mathsf{OPT})$ space. Our second algorithm for points in $\mathbb{R}^2$ makes $O(\log(\epsilon^{-1}))$ passes before outputting the ϵ-hull and requires $O(\mathsf{OPT})$ space. Our third algorithm, for points in $\mathbb{R}^d$ for any fixed dimension d, outputs, with high probability, an ϵ-hull for all but δ-fraction of directions and requires $O(\mathsf{OPT} \cdot \log \mathsf{OPT})$ space.

[1] This work was supported in part by the National Science Foundation under grant CCF-1525971. Work was done while the author was at Carnegie Mellon University.
[2] This material is based upon work supported in part by the National Science Foundation under Grants No. 1447639, 1650041 and 1652257, Cisco faculty award, and by the ONR Award N00014-18-1-2364.
[3] Now at DeepMind.
[4] This research was supported by the Franco-American Fulbright Commission and supported in part by National Science Foundation under Grant No. 1447639, 1650041 and 1652257. The author thanks INRIA (l'Institut national de recherche en informatique et en automatique) for hosting him during the writing of this paper.
[5] This material is based upon work supported in part by National Science Foundation under Grant No. 1447639, 1650041 and 1652257. Work was done while the author was at Johns Hopkins University.

45th International Colloquium on Automata, Languages, and Programming (ICALP 2018).
Editors: Ioannis Chatzigiannakis, Christos Kaklamanis, Dániel Marx, and Donald Sannella;
Article No. 21; pp. 21:1–21:13
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

2012 ACM Subject Classification Theory of computation → Computational geometry, Theory of computation → Sketching and sampling, Theory of computation → Streaming models

Keywords and phrases Convex Hulls, Streaming Algorithms, Epsilon Kernels, Sparse Coding

Digital Object Identifier 10.4230/LIPIcs.ICALP.2018.21

Related Version A full version of the paper is available at https://arxiv.org/abs/1712.04564.

Acknowledgements We want to thank our anonymous reviewers for their useful feedback in preparing this manuscript, and Rachel Holladay, Drew Bagnell, Maya Rau-Murthy, Satya Prateek Tangirala, and Sunny Nahar for discussion of ideas and feedback on paper drafts.

1 Introduction

The question addressed by this paper is: *Can we compute approximate convex hulls of data streams using near-optimal space?* Approximate convex hulls are fundamental in computational geometry, computer vision, data mining, and many more (see e.g. [2]), and computing them in a streaming manner is important in the big data regime.

Our notion of approximate convex hulls is the commonly used ϵ-hull. Let P be a set of n points in $\mathbb{R}^d$. Let $\mathcal{C}(P)$ denote the convex hull of P, the smallest convex set containing P. We want a small subset S of P such that all points in P are inside $\mathcal{C}(S)$ or within distance ϵ from $\mathcal{C}(S)$. Since every point in P can be approximated by a sparse convex combination of points in S, S is also called a generating set [7]. For an example motivation of this particular definition, consider two far-away sensors rapidly collecting data: one of positive examples and the other of negative examples; if it is expected that these should be linearly separable with some margin ϵ, then an appropriate small summary of their data would be an ϵ-hull.

ϵ-hulls and their variants have been studied extensively in the literature. In the multiplicative error variant, ϵ-kernels, one requires that any directional width (the diameter of S in a particular direction) of S is a $(1 \pm \epsilon)$ approximation to that of P. ϵ-hulls and ϵ-kernels are intimately connected: algorithms for ϵ-kernels typically apply a transformation to the data, and then use algorithms for ϵ-hulls. For more details, we refer the reader to [2].

Existing work focuses on worst case bounds, which scale poorly with the dimension d. The worst case lower bound for the size of an ϵ-hull is $\Omega(\epsilon^{-(d-1)/2})$. Recently, it has been shown in [7] that one can do much better than the worst case bound if the size of the smallest ϵ-hull for P (which we denote as $\mathtt{OPT}$) is small. In their paper, they show that one can efficiently obtain S of size nearly linear in $\mathtt{OPT}$ and at most linear in the dimension d.

One concern of the algorithms in [7] is that they require storing all points of P in memory. The huge size of real-world datasets limits the applicability of these algorithms. A natural question to ask is whether it is possible to efficiently maintain an ϵ-hull of P when P is presented as a data stream while using a small amount of memory.

We provide both negative and positive results, summarized below.

1.1 Our Contributions

Let $\mathtt{OPT}$ be the optimal size of an ϵ-hull. In Section 3, we show, under a reasonable streaming model, that no streaming algorithm can achieve space bounds comparable to $\mathtt{OPT}$. In particular, no streaming algorithm can have space complexity competitive with $f(\mathtt{OPT}, d)$ in 3 dimensions or higher for any $f : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$. This lower bound guides us to consider variants on this problem. Note that the lower bound applies specifically to streaming algorithms; for the batch setting, [7] gives a polynomial-time algorithm that computes an ϵ-hull with space $O(d\mathsf{OPT} \log \mathsf{OPT})$.

We devise and prove the correctness of streaming algorithms for three relaxations of the problem. In Section 4, we show the first relaxation, in which the points are from $\mathbb{R}^2$ and come in a random order. In Section 5, we relax the problem (again in $\mathbb{R}^2$) by allowing the algorithm to make multiple passes over the stream. In Section 6, we show the third relaxation, in which the points come in an arbitrary order and from d-dimensional space, but we only require to approximate the convex hull in a large fraction of all directions.

In the first relaxation, our algorithm maintains an initially empty point set S. When our algorithm sees a new point p, it adds p to S if p is at least distance ϵ away from the convex hull of S. Additionally, our algorithm keeps removing points $p' \in S$ when some p' is contained inside the convex hull of $S \setminus \{p'\}$, that is, removing p' does not change the convex hull of S. Surprisingly, for any point stream P, with high probability this algorithm keeps an ϵ-hull of size $O(\mathsf{OPT} \cdot \log n)$, where n is the size of P.

In the second relaxation, we permit the algorithm to make a small number of passes over the stream. Our algorithm begins the first pass by taking $O(1)$ directions and storing the point with maximal dot product with each direction. In each of $O(\log(\frac{1}{\epsilon}))$ subsequent passes, we refine the solution by adding a new direction in sectors that incurred too much error while potentially deleting old directions that become no longer necessary. The algorithm computes an ϵ-hull of size $O(\mathsf{OPT})$.

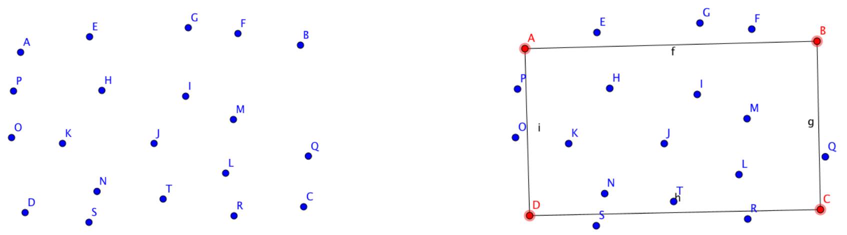
In the third relaxation, we only need to be correct in most directions (all but a δ fraction of directions). Our algorithm picks $O_d(\frac{\mathsf{OPT}}{\delta^2} \log \frac{\mathsf{OPT}}{\delta})$ random unit vectors. For each of these vectors v, we keep the point in the stream that has maximal dot product with v.

To the best of our knowledge, this is the first work that gives lower bounds and streaming algorithms for ϵ-hulls with space complexity comparable to the optimal approximation.

1.2 Related Work

Batch Algorithms. We use the term batch algorithm for an algorithm that stores the entire set of points in memory. In the batch setting, Bentley, Preparata, and Faust [5] give a $O_d(1/\epsilon^{(d-1)})$ space algorithm for computing an ϵ-hull of a set of points (assuming constant dimension d). Agarwal, Har-Peled, and Varadarajan [1] improve the result to give a $O_d(1/\epsilon^{(d-1)/2})$ space algorithm for ϵ-kernels, a multiplicative approximation of convex hulls. The running time bounds were further improved in [8, 10, 12]. Recently, Blum, Har-Peled, and Raichel [7] give the only known batch algorithms for an ϵ-hull that are competitive with the optimal ϵ-hull size of the given point set.

Streaming Algorithms with Worst Case Guarantees. Hershberger and Suri [11] and Agarwal & Yu [3] give 2D one-pass streaming algorithms for ϵ-hulls that uses $O(1/\sqrt{\epsilon})$ space. Agarwal, Har-Peled, and Varadarajan [1] give a one-pass streaming algorithm for ϵ-kernels that uses $O_d((1/\epsilon^{\frac{d-1}{2}}) \log^d n)$ space. Chan [8] removes the dependency on n and gives a streaming algorithm for ϵ-kernels that uses $O_d((1/\epsilon^{d-3/2}) \log^d 1/\epsilon)$ space. This was then improved to $O_d((1/\epsilon^{\frac{d-1}{2}}) \log \frac{1}{\epsilon})$ [13] and the time complexity was further improved by Arya and Chan [4]. Chan [9] also gives a dynamic streaming (allowing deletions in the stream) algorithm based on polynomial methods. All of these space bounds assume a constant dimension d, and focus on worst case guarantees.



(a) Set of points. **(b)** ϵ-hull in red (for suitable ϵ).

Figure 1 ϵ-hull of a set of points.

ϵ-kernels vs ϵ-hulls. Past work focuses on both ϵ-hulls and ϵ-kernels, a multiplicative error variant. ϵ-kernels can be trickier to compute, but are closely related to ϵ-hulls, and often use algorithms for ϵ-hulls as a core subroutine. We focus on ϵ-hulls, but extending this work to ϵ-kernels is an exciting (non-trivial) avenue for future research. In particular, typical reductions from ϵ-kernels to ϵ-hulls (e.g. see [2]) are not compatible with the notion of OPT.

Our Techniques. The proof of our 2D random order algorithm exposes an elegant connection between our 2D result, and a classic 1D result. Our multipass algorithm and (ϵ, δ)-hull algorithm are built on existing methods (e.g. [1, 3]) in that a core subroutine involves preserving the maximal point along certain directions.

2 Preliminaries

▶ **Definition 2.1.** For any bounded set $C \subseteq \mathbb{R}^d$, a point q is *ϵ-close* to C if $\inf_{x \in C} \|q - x\|_2 \leq \epsilon$.

▶ **Definition 2.2.** Given a set of points $P \subseteq \mathbb{R}^n$, $S \subseteq P$ is an *ϵ-hull* of P if for every $p \in P$, p is ϵ-close to $\mathcal{C}(S)$, the convex hull of S.

▶ **Definition 2.3.** Let $\mathsf{OPT}(P, \epsilon)$ denote the number of points in a (not necessarily unique) smallest ϵ-hull of P. We omit P and ϵ if it is clear from the context.

2.1 Streaming Model

Our streaming model, while simple, captures most streaming algorithms for ϵ-hulls in the literature. In our model, a streaming algorithm $\mathcal{A}$ is given ϵ in advance but not the size of the input point stream $P \in \mathbb{R}^d$. P is presented to an algorithm $\mathcal{A}$ sequentially:

$$P = (p_1, p_2, \ldots, p_t, \ldots),$$

where $p_t \in \mathbb{R}^d$ is the point coming at time t. Note that P may have duplicate points. For the ϵ-hull problem, we require Algorithm $\mathcal{A}$ to maintain a subset $S \subseteq P$. For each point $p \in P$, $\mathcal{A}$ can choose to add p to S (remembering p) or ignore p (therefore permanently forgetting p). $\mathcal{A}$ can also choose to delete points in S, in which case these points are permanently lost. After one-pass of the stream, we require S to be an ϵ-hull of the points set P. A trivial streaming algorithm could just keep all points it has seen. However, such an algorithm would not be feasible in the big data regime. Ideally, $\mathcal{A}$ should use space competitive with $\mathsf{OPT}(P, \epsilon)$.

3 Lower Bounds

An (f, r)-optimal algorithm in dimension d uses space competitive with $f(\mathtt{OPT}(P, \epsilon), d)$ and maintains an $(r\epsilon)$-hull where $r > 1$. Note that this definition is rather permissive, since it allows an arbitrary function of OPT and allows slack in ϵ as well.

▶ **Definition 3.1.** For $r \geq 1$, $f : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$, we say a streaming algorithm $\mathcal{A}$ is (f, r)-optimal if given arbitrary $\epsilon > 0$ and point stream $P \subseteq \mathbb{R}^d$, $\mathcal{A}$ keeps an $(r\epsilon)$-hull of P of size at most $f(\mathtt{OPT}(P, \epsilon), d)$.

▶ **Theorem 3.2.** *For all $r \geq 1$, $d \geq 3$, $f : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$, there does not exist an (f, r)-optimal streaming algorithm in $\mathbb{R}^d$.*

Proof. See Theorem A.5, Theorem A.6, and Corollary A.8 in the Appendix of the full version for the proof. Here we give high level intuition for the case $r = 1$, $d = 3$. In our proof, we assume for the sake of contradiction that there exists such a streaming algorithm A and function f. We construct 3 sequences of points P_1, P_2, P_3. Let $P_1 \circ P_2$ denote sequence P_1 followed by sequence P_2 (P_2 appended to P_1). We then show that if A keeps an ϵ-hull of size at most $f(\mathtt{OPT}(P_1 \circ P_2, \epsilon), 3)$ after receiving $P_1 \circ P_2$, then it cannot keep an ϵ-hull of size at most $f(\mathtt{OPT}(P_1 \circ P_2 \circ P_3, \epsilon), 3)$ after receiving $P_1 \circ P_2 \circ P_3$. This is a contradiction.

To do this, we ensure that $|P_2|$ is much larger than $|P_1|$ and that P_1 is an ϵ-hull of $P_1 \circ P_2$. This forces the algorithm to keep only a small proportion of points in $P_1 \circ P_2$. We then ensure that $|P_3|$ is much larger than $|P_2|$ and that P_2 is an ϵ-hull for $P_2 \circ P_3$. However, since the algorithm only kept a small number of points in P_2, it is forced to keep many points in P_3. See the appendix for precise details. The result extends easily for $d > 3$. For $r > 1$, we use a similar construction, but add more sets of points $P_4, P_5, \ldots$ ◀

We can also ask a slightly different question: what if an algorithm is given an additional parameter k in advance, and only needs to maintain an ϵ-hull at time t when OPT of the substream at time t falls below k. The algorithm we give for (ϵ, δ)-hulls in Section 6 is of this form. In the appendix of the full version (Definition A.9 and Theorem A.10), we formulate a lower bound for this case.

Our lower bounds guide future research by showing that we need to think beyond the current streaming models, add reasonable assumptions to the problem, or the space bounds of our algorithms must include some functions of ϵ or $|P|$ (along with OPT and d).

4 2D Random Order Algorithm (ROA)

In many cases, data points are generated i.i.d., for example in mixture models or topic models (e.g. [6]). In this section we assume a more general setup: that the points come in a random order. More precisely, for all sets of points P, every permutation of P must have equal probability density. The case where the data points are generated i.i.d. (*making no assumptions about the distribution*) is a special case. We assume the points are in 2D. To begin, we introduce the following definition.

▶ **Definition 4.1.** A point p is *interior* to P if p is in the convex hull of $P \setminus \{p\}$.

4.1 1D Algorithm

To motivate our 2D algorithm, we begin with a classic result in 1-dimension. Consider the algorithm **ROA-insertion**: Begin by keeping a set $S = \{\}$. For each point $p \in P$ that the algorithm sees, if the distance from p to the convex hull of S is at most ϵ, we discard p. Otherwise, we add p to S.

Algorithm 1 Pseudocode for ROA (2D random order algorithm).

```
1: $S \leftarrow \{\}$
2: **when** $p \in P$ **is received do:**
3:     **if** $\text{dist}(p, \mathcal{C}(\text{S})) \leq \epsilon$ **then**:
4:         // Discard $p$
5:     **else**:
6:         $S \leftarrow S \cup \{p\}$
7:         **for** $p' \in S$ **sequentially do:**
8:             If $p'$ is an interior point of $S$ then $S \leftarrow S \setminus \{p'\}$
9:         **end for**
10: **end when**
```

▶ **Lemma 4.2.** *There exists a constant $c > 0$ such that for any random order input stream P containing at most n points, ROA-insertion maintains a subset $S \subseteq P$ which is an ϵ-hull of P at all times. Moreover, if $P \subseteq \mathbb{R}^1$ then with probability at least $1 - 1/n^3$,*

$$|S| \leq c \cdot \log n \leq c \cdot \mathit{OPT}(P, \epsilon) \cdot \log n,$$

note that for any $\epsilon \geq 0$, if $P \subseteq \mathbb{R}^1$ then $1 \leq \mathit{OPT}(P, \epsilon) \leq 2$.

A natural question is whether the space bound for this algorithm generalizes to higher dimensions. Our experiments suggest that it does not even generalize to 2D. In our experiments, we set $\epsilon = 0$ and gave ROA-insertion n equally spaced points inside a square. OPT is 4, since all the points are contained inside a square. However, experimentally, the number of points kept by ROA-insertion increases much faster than $\log n$.

4.2 2D Algorithm

We extend algorithm ROA-Insertion to get algorithm **ROA**. Let the points kept by ROA at the i^{th} step of the algorithm be S_i. At each step i, we iteratively delete interior points from S_i until S_i has no interior points. We summarize algorithm ROA in Algorithm 1.

The proof of ROA gives an interesting connection between our 2D algorithm, and the 1D classical result in the previous section. We begin with a technical lemma (see Lemma B.1 in the appendix of the full version for the proof), and then proceed to the main theorem.

▶ **Lemma 4.3.** *(Similar Boundaries) Suppose A and B are ϵ-hulls of P. Let $\mathcal{H}$ denote the (two-way) Hausdorff distance and $\partial\mathcal{C}(A)$ denote the boundary of the convex hull of A. Then $\mathcal{H}(\partial\mathcal{C}(A), \partial\mathcal{C}(B)) \leq \epsilon$.*

▶ **Theorem 4.4.** *There exists a constant $c > 0$ such that for any random order input stream P containing at most n points, ROA maintains a subset $S \subseteq P$ which is an ϵ-hull of P at all times. Moreover, if $P \subseteq \mathbb{R}^2$ then with probability at least $1 - 1/n^2$,*

$$|S| \leq c \cdot \mathit{OPT}(P, \epsilon) \cdot \log n.$$

Since the algorithm is deterministic, the probability is over the arrival order of P.

Proof. An inductive argument shows that at each iteration i, S is an ϵ-hull of P. We focus on the proof of the space bound.

Step 1: Consider an optimal ϵ-hull T of P. We show that all points in S are near the boundary $\partial\mathcal{C}(T)$. Note that S does not contain any interior points, so for all $s \in S$, $s \in \partial\mathcal{C}(S)$. Then by Lemma 4.3, for every point $s \in S$, $\text{dist}(s, \partial\mathcal{C}(T)) \leq \epsilon$.

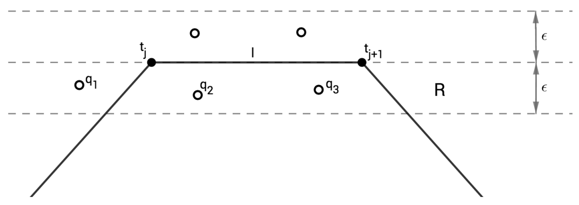


Figure 2 Figure for Theorem 4.4. Consider all points near segment $l = \overline{t_j t_{j+1}}$. Consider points $q_1, q_2, q_3 \in Q$ on one side of l. They are contained in a thin strip R of width ϵ.

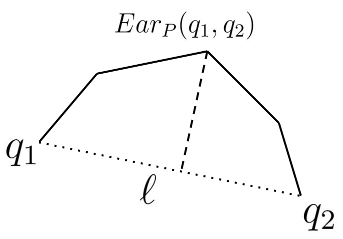


Figure 3 A diagram of $Ear_P(q_1, q_2)$. The dotted line is ℓ, and the length of the dashed line is $\mathsf{Error}_P(q_1, q_2)$.

Step 2: We split T into OPT sections, and show that with high probability our algorithm keeps $O(\log n)$ points for each section. Since T is optimal, it does not contain any interior points. Label the points in T: t_1, ..., t_k, clockwise along the boundary of the convex hull of T. For every $s \in S$, since $\text{dist}(s, \partial\mathcal{C}(T)) \leq \epsilon$, s is within distance ϵ from the line segment connecting some t_i and t_{i+1}. Now, referring to Figure 2, consider the line segment l connecting arbitrary t_j and t_{j+1}, and consider all points in P within distance ϵ from l. We group the points based on which side of the line segment they are on - consider the points Q on one side of the line segment.

The points Q are contained in some narrow strip R with width ϵ. Effectively, because Q is contained in a narrow strip, we can reduce to the 1D case and apply the proof from Lemma 4.2. To see this reduction, consider the projection of the points in Q onto the (infinite) line l' connecting t_j and t_{j+1}. Let $f(q_i)$ denote the projection of q_i onto line l'. If $f(q_k)$ is between $f(q_i)$ and $f(q_j)$, and q_k arrives after q_i and q_j, then our algorithm discards q_k because q_k is within distance ϵ from the line segment connecting q_i and q_j. Applying the 1D proof, we get that with high probability we keep $O(\log n)$ points for each segment.

Step 3: We take a union bound over the OPT sections to get the desired result, where we note that $\mathsf{OPT} \leq n$. ◀

5 2D Multipass Algorithm

In this section, we relax the problem by letting the algorithm pass over the stream P multiple times. Let $\text{diam}(P)$ refer to the diameter of the point-set P which is $\max_{x,y \in P} d(x, y)$. Our algorithm requires $\log(\frac{\text{diam}(P)}{\epsilon})$ passes and $O(\mathsf{OPT})$ memory. For convenience of exposition, we assume $\text{diam}(P) = 1$ and prove a bound of $\log(\frac{1}{\epsilon})$ passes. If $\text{diam}(P) \neq 1$, we can simply scale all the points, and ϵ, by $\frac{1}{\text{diam}(P)}$ and run the algorithm to get the desired bound.

By convention, we define the distance between a point p and a set A to be $d(p, A) = \min_{a \in A} d(p, a)$. In a slight abuse of notation, for a finite set P we define ∂P to be the subset of P that lies on the boundary of the convex hull of P. Formally:

▶ **Definition 5.1.** For a finite set $P \subset \mathbb{R}^2$, we define $\partial P = P \cap \partial\mathcal{C}(P)$. Here $\partial\mathcal{C}(P)$ means the boundary of the convex hull of P.

Given any two points $q_1, q_2 \in \partial P$, define $\ell = \mathcal{C}(\{q_1, q_2\})$ to be the line segment with endpoints q_1 and q_2. Observe that the set $\mathcal{C}(P) \setminus \ell$ has at most two connected components. Define $Ear_P(q_1, q_2)$ to be the component that lies to the left of the vector from q_1 to q_2. We define $\mathsf{Error}_P(q_1, q_2) = \max_{x \in Ear_P(q_1,q_2)} d(x, \ell)$ to be the maximum distance of a point in this

Algorithm 2 Input: a stream of points $P \subset \mathbb{R}^2$ and a value $\epsilon \in (0,1]$. Output: an ϵ-approximate hull of P

```
1: $t_1 \leftarrow (1,0)$, $t_2 \leftarrow (-1,0)$
2: $T \leftarrow$ an ordered list $(t_1, t_2)$
3: For $i = \{1,2\}$, associate $q_i \leftarrow \mathsf{GetMax}_P(t_i)$ with $t_i$
4: Initialize Flag to down position
5: for all $1 \le i \le |T|$ (in parallel) do
6:     Compute $\mathsf{Error}_P(q_i, q_{i+1})$
7:     Compute $\mathsf{Error}_P(q_{i-1}, q_{i+1})$
8:     $t'_i \leftarrow$ direction bisecting $t_i$ and $t_{i+1}$
9:     $q'_i \leftarrow \mathsf{GetMax}(t'_i)$
10: for all $1 \le i \le |T|$ (in parallel) do
11:     if $\mathsf{Error}_P(q_{i-1}, q_{i+1}) \le \epsilon$ and neither $t_{i+1}$ or $t_{i-1}$ have been deleted then
12:         Remove $t_i$ from $T$
13:     if $\mathsf{Error}_P(q_i, q_{i+1}) > \epsilon$ then
14:         Add $t'_i$ to $T$ and associate $q'_i$ with $t'_i$
15:         Raise Flag
16: Recompute indices of $T$ to preserve clockwise-order
17: Delete any points/vectors except $t_i \in T$ and their associated $q_i$
18: if Flag is up then
19:     Go to Line 4
20: else
21:     Output $\{q_1, \dots, q_{|T|}\}$
```

component from ℓ. See Figure 3 for an example. Note that we can compute $\mathsf{Error}_P(q_1, q_2)$ in a single pass (see Algorithm 2 and Lemma C.1 in the Appendix of the full version).

Let t be a unit vector. Define $\mathsf{GetMax}_P(t)$ to be $\arg\max_{p \in P} p \cdot t$. It is clear that $\mathsf{GetMax}_P(t)$ can be computed in a single pass. Algorithm 2 is the main multipass algorithm, using Error and GetMax as blackboxes. We always maintain a set of directions T. On Lines 5-9 we run $3|T|$ single-pass algorithms completely in parallel, therefore requiring only a single pass. By the phrase "associating a point with a direction", we mean to keep this point as piece of satellite data.

Our main result for this section is the behavior of Algorithm 2. We define a word as the space required to store a single point in $\mathbb{R}^2$.

We begin with some preliminary statements. We defer the proofs of these lemmas to the Appendix of the full version (see Lemmas C.2, C.3, C.4, and C.6). Throughout this section, we use the convention of incrementing subscripts modulo n (for example $q_{n+1} = q_1$).

▶ **Lemma 5.2.** *If Algorithm 2 terminates, it outputs an ϵ-hull to P.*

▶ **Lemma 5.3.** *Algorithm 2 terminates in $3 + \lceil \log_2(1/\epsilon) \rceil$ passes.*

▶ **Lemma 5.4.** *Let $p, p', q', q \in \partial P$ be in clockwise order along $\partial\mathcal{C}(P)$. Then $\mathsf{Error}_P(p', q') \le \mathsf{Error}_P(p, q)$.*

▶ **Lemma 5.5.** *There exists an ϵ-hull for P using only points from ∂P of cardinality at most $2\mathsf{OPT}(P, \epsilon)$.*

Note that Lemma 5.5 is not trivial. By definition, there exists an ϵ-hull for P of size $\mathsf{OPT}(P, \epsilon)$ using points from P. It may be that an ϵ-hull of optimal size must use a point from the interior of P. For example, consider a square of side length $r \in (\sqrt{2}\epsilon, 2\epsilon)$, where

∂P consists of the four corners. It is possible, due to interior points, that $\mathsf{OPT}(P, \epsilon) = 2$ and yet an ϵ-hull using only points from ∂P must use all four corners. Note that this example also shows that the bound in Lemma 5.5 is tight.

On Line 8, t'_i is defined as the direction that bisects t_i and t_{i+1}. We define the bisection of unit vectors a and b to be the unit vector obtained by rotating a clockwise through through half of the rotation required to point in the direction of b.

▶ **Theorem 5.6.** *Given a stream of points $P \subset \mathbb{R}^2$ and a value $\epsilon \in (0, 1]$, Algorithm 2 terminates within $3 + \lceil \log_2(1/\epsilon) \rceil$ passes, stores at most $24\mathit{OPT}(P, \epsilon) + O(1)$ words, and returns an ϵ-hull of P of cardinality $6\mathit{OPT}(P, \epsilon)$.*

Proof. By Lemma 5.3, Algorithm 2 terminates after $3 + \lceil \log_2(1/\epsilon) \rceil$ passes. By Lemma 5.2, Algorithm 2 outputs an ϵ-hull to P. It only remains to bound the space usage and cardinality of the set returned

Let $W \subset \partial P$ be an ϵ-approximation of P such that $n = |W| \leq 2\mathsf{OPT}(P, \epsilon)$. Lemma 5.5 guarantees that such a W exists. Let $W = \{w_1, \ldots, w_n\}$ be an ordering of W that is clockwise in $\partial\mathcal{C}(P)$. By definition, $\mathsf{Error}_P(w_i, w_{i+1}) \leq \epsilon$ for every $i \in \mathbb{Z}$ (recall the convention of indexing modulo n). Consider the state of the algorithm at the beginning of a pass; for notation let T contain the directions $\{t_i\}_{i=1}^{|T|}$ associated respectively with $\{q_i\}_{i=1}^{|T|}$.

For $s \in \{1, 2\}$, suppose that $w_i, q_j, q_{j+s}, w_{i+1}$ are in clockwise order of $\partial\mathcal{C}(P)$. By Lemma 5.4, $\mathsf{Error}_P(q_j, q_{j+s}) \leq \mathsf{Error}_P(w_i, w_{i+1}) \leq \epsilon$. We draw two conclusions. The first conclusion ($s = 1$) is that on Line 13, t'_i will not be added to T. The second conclusion ($s = 2$) is that on Line 11, t_{i+1} is a candidate for deletion (i.e. t_{i+1} will be deleted unless t_i or t_{i+2} have already been deleted).

Using the clockwise ordering of $\partial\mathcal{C}(P)$, we say that a point $q \in \partial P$ is on edge (w_i, w_{i+1}) if it lies between w_i and w_{i+1} in the ordering. Suppose that $\{q_j\}_{j=1}^{|T|}$ contains m points on edge (w_i, w_{i+1}). By the reasoning in the preceding paragraph, it is easy to verify that all but $\lceil \frac{m-1}{2} \rceil + 1$ will be deleted on Line 11. As for points added on Line 13, this can only occur at the boundary (between q_j and q_{j+1} where q_j is the last point on some edge) and therefore adds at most 1 point per edge.

Combining these facts, we see that an edge which enters a pass with m points finishes that pass with at most $\lceil \frac{m-1}{2} \rceil + 2$ points. Inductively we begin with $m = \{0, 1, 2\}$ for each edge. This implies that $m \leq 3$ after each pass. Therefore $|T| \leq 3n \leq 6\mathsf{OPT}(P, \epsilon)$ at all times.

Finally, observe that the storage of $4|T| + O(1)$ points are used in a pass. To compute Error without precision issues, storing a single point suffices. Therefore for each i we store one point for each of the two Error computations, one point for GetMax, and the original point q_i and vector t_i. The $O(1)$ is just a workspace to carry out the calculations. ◀

6 (ϵ, δ)-Hull

In this section we give an algorithm for a relaxation of ϵ-hulls, which we call (ϵ, δ)-hulls. Our results hold for arbitrary point sets $P \subseteq \mathbb{R}^d$. Intuitively, an (ϵ, δ)-hull of P is within distance ϵ from the boundary of the convex hull of P in at least a $1 - \delta$ fraction of directions. We begin by building up the definition of an (ϵ, δ)-hull.

▶ **Definition 6.1.** Given a vector $v \in \mathbb{R}^d$ and a finite point set $P \subseteq \mathbb{R}^d$, we define the **directional extent** as

$$\omega_v(P) = \max_{p \in P} p \cdot v.$$

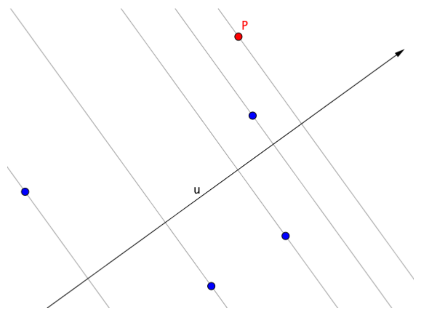


Figure 4 Point p maximizes the set of points in direction u because its projection onto u is the highest.

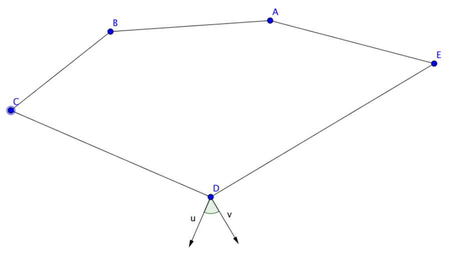


Figure 5 All vectors between u and v with Euclidean norm at most 1 are in E_D^T. See texts for details.

If $p \in \mathbb{R}^d$ is a point we define $\omega_v(p) = p \cdot v = \omega_v(\{p\})$. We say that S **maximizes** P in v if $\omega_v(P) = \omega_v(S)$ (see Figure 4). Note that S can be either a single vector or a set of vectors.

▶ **Definition 6.2.** Let $P \subset \mathbb{R}^d$ be a set of points and $S \subseteq P$. We say S **ϵ-maximizes** P in v if $v = 0$ or, letting $v' = v/\|v\|_2$, we have

$$|\omega_{v'}(P) - \omega_{v'}(S)| \leq \epsilon.$$

Note that as per definition 6.1, S can be either a single vector or a set of vectors.

▶ **Definition 6.3.** Given $P \subseteq \mathbb{R}^d$, an **(ϵ, δ)-hull** is a subset $S \subseteq P$ such that if we pick a vector v uniformly at random from the boundary of the unit sphere, $\mathcal{S}^{d-1}$, S ϵ-maximizes P in direction v with probability at least $1 - \delta$, that is,

$$\Pr_{v \sim \mathcal{S}^{d-1}}(|\omega_v(P) - \omega_v(S)| > \epsilon) \leq \delta.$$

Suppose we fix the dimension d. We give a randomized algorithm that uses m points and with probability at least $1 - \gamma$ gives us an (ϵ, δ)-hull of a point set P, where k is the optimal size for the ϵ-hull of P, and m satisfies:

$$m \in O_d\left(\frac{k}{\delta^2} \cdot \log \frac{k}{\gamma\delta}\right).$$

Note that m does not explicitly depend on ϵ. Our algorithm for d-dimensional space is as follows: Choose m uniformly random vectors in the unit ball B^d (or equivalently on the boundary of the unit ball, $\mathcal{S}^{d-1}$). For each chosen vector v we store a single point $p \in P$ that maximizes P in direction v, that is, $p \cdot v = \omega_v(P)$. This can easily be done in streaming. Note that the given complexity is for a fixed dimension d, the actual space complexity will be multiplied by some (exponential) function of d, but independent of ϵ.

6.1 Proof of (ϵ, δ)-hull Algorithm

Consider an arbitrary point set P, and suppose that our algorithm keeps a subset S of P. The core of our proof, leading up to lemma 6.11, shows that for each point that our algorithm picks, S gets closer to an (ϵ, δ)-hull. In particular, we define the set C of bad vectors in B^d, as vectors v s.t. S does not ϵ-maximize P in v. We want to bound the number of points we need to include in S so that C is small. Crucially, we show that C is a union of a small number of convex sets, and does not contain any vectors we selected (recall that our algorithm selects uniformly random vectors in B^d, and uses these to select certain points in P). Then, we can approximate C with a union of ellipsoids, which has small VC-dimension. This finally allows us to apply the machinery of ϵ-nets to get the desired result.

We begin with some definitions and lemmas.

▶ **Definition 6.4.** Let B^d denote the unit ball in d dimensions. Let $\mathcal{S}^{d-1}$ denote the unit sphere in d dimensions, which is ∂B^d (the boundary of B^d).

▶ **Definition 6.5.** Let $V^d(S)$ denote the d-dimensional volume (Lebesgue measure) of a measurable set S in d-dimensional space.

▶ **Definition 6.6.** Given $T \subseteq \mathbb{R}^d$ and $t \in \mathbb{R}^d$, we define E_t^T to be the set of all vectors $v \in B^d$ such that t maximizes T in v, that is,

$$E_t^T = \{v \mid v \cdot t = \omega_v(T) \wedge |v|_2 \leq 1\}.$$

Figure 5 shows a set of points T. All vectors between u and v with Euclidean norm at most 1, in the range indicated by the angle, are in E_D^T. Note that u is perpendicular to line segment CD and v is perpendicular to line segment DE. Only points $t \in T$ that lie on the boundary of the convex closure of T have non-empty E_t^T.

▶ **Definition 6.7.** Given a point stream P, and a set S, we say *the set of bad vectors* C (with respect to P, S) is the set of vectors v in B^d such that S does not ϵ-maximize P in v. An equivalent definition of (ϵ, δ)-hulls is that $V^d(C)/V^d(B^d) \leq \delta$.

We are now ready to present the following lemmas about the properties of E_t^T.

▶ **Lemma 6.8** (ϵ-Maximization Lemma). *Suppose $P \subseteq \mathbb{R}^d$ is a finite set of points and $T \subseteq P$ is an ϵ-hull of P, and $t \in T$. Then t ϵ-maximizes P for all vectors $v \in E_t^T$ (see Definition 6.6).*

▶ **Lemma 6.9** (Covering Lemma). *For all finite point sets $T \subseteq \mathbb{R}^d$, $\bigcup_{t \in T} E_t^T = B^d$.*

▶ **Lemma 6.10** (Convex Lemma). *For any point $t \in \mathbb{R}^d$ and finite set $T \subseteq \mathbb{R}^d$, E_t^T is convex and has finite volume.*

We want to show that the set of points S our algorithm chooses is ϵ-maximal in most directions. One way is to show that for each point our algorithm picks, the set of *bad vectors* (vectors that our stored points do not ϵ-maximize) shrinks. We present a crucial lemma that formalizes this notion under some assumptions.

▶ **Lemma 6.11.** *Given a finite point set $P \subseteq \mathbb{R}^d$ and a finite-volume convex set $C \subseteq \mathbb{R}^d$. Assume that there exists some $p \in P$ s.t. for all unit vectors $v \in C$, p ϵ-maximizes P in v. Suppose that we pick arbitrary vectors $v_1, ..., v_k \in C$ and corresponding points $p_1, ..., p_k \in P$ s.t. for all i, p_i maximizes P in v_i. Then there exists a finite-volume convex subset $C' \subseteq C$ s.t.*

1. *For all $i \in [k]$, $v_i \notin C'$.*
2. *For all vectors $v \in C \setminus C'$, $S = \{p_1, ..., p_k\}$ ϵ-maximizes P in v.*

Proof. Consider a vector v_i that we picked, and corresponding point p_i. If $p_i = p$, then $C' = \{\}$ satisfies the required conditions. Otherwise, let $H_i = \{v \mid p_i \cdot v \geq p \cdot v\}$. H_i is a half-space that contains the vector v_i. Furthermore for all vectors $v \in H_i \cap C$, S ϵ-maximizes P in v. So the set of vectors in C that p_i does not maximize are contained in $H_i^c \cap C$, where H_i^c does not contain v_i. Applying this argument for each vector v_i and corresponding point p_i, we can construct C' to be the intersection of C with the k (open) half-spaces H_i^c corresponding to each of the points p_i we selected. Our constructed C' is convex, because it is the intersection of convex sets, and it is bounded and measurable. ◀

For completeness, we include a standard lemma that is similar to the finite ϵ-net in computational geometry. Before we proceed, for a family of sets $\mathcal{H}$, we denote the simplified version of *VC-dimension* $d' = \widetilde{\mathrm{VC}}(\mathcal{H})$ as the smallest positive integer d' such that for every finite set $A \subseteq \mathbb{R}^d$, $|\{h \cap A : h \in \mathcal{H}\}| \le |A|^{d'}$ (that is, such that a simple variant of Sauer's Lemma holds). We then have the following lemma, which we prove in the appendix.

▶ **Lemma 6.12.** *Let $\tau, \gamma \in (0,1)$ be two parameters. Let $\mathcal{H}$ be a set of measurable sets in $\mathbb{R}^d$ such that $\widetilde{VC}(\mathcal{H}) \le d'$ for some integer d'. Given a measuable convex set $C \subseteq \mathbb{R}^d$, let $\mathcal{H}_C = \{c \in \mathcal{H} : c \subseteq C\}$ be the sets of subsets of H contained in C. Suppose we choose $m = \Theta(\frac{d'}{\tau^2}\log\frac{d'}{\tau\gamma})$ points uniformly random in C. Then, except with probability γ, **all** sets $u \in \mathcal{H}_C$ with $V^d(u)/V^d(C) \ge \tau$ contains some selected point, where $V^d(u)$ denotes the volume of u.*

Before we present the main theorem, we note that the set of unions of k ellipsoids is of small VC-dimension. The formal proof is presented in the Appendix.

▶ **Lemma 6.13.** *Let $\mathcal{E}$ be the sets of all ellipsoids in $\mathbb{R}^d$. Let $\mathcal{E}^k = \{e_1 \cup e_2 \cup e_3 \ldots \cup e_k : e_i \in \mathcal{E}\}$. Then $\widetilde{VC}(\mathcal{H}) \le 4kd^2$.*

Now we are ready to present the main theorem in this section.

▶ **Theorem 6.14.** *Let $\gamma, \delta \in (0,1), k \ge 1$ be parameters. Given a point stream P in $\mathbb{R}^d$ and $\epsilon \ge 0$. Suppose $\mathit{OPT}(P,\epsilon) \le k$. Then there exists a one-pass streaming algorithm, given P, γ, δ, k, stores a set $S \subseteq P$ of $m = \Theta_d\big(\frac{k}{\delta^2}\log\frac{k}{\gamma\delta}\big)$ points, such that, except with probability γ, S is an (ϵ,δ)-hull of P.*

Proof. To begin the proof, we recall the algorithm. We first pick uniformly at random $m = \Theta\big(\frac{d^{2d+2}k}{\delta^2}\log\frac{kd}{\gamma\delta}\big)$ directions from B^d, the d-dimensional unit ball. When the stream is coming, we maintain the extreme point from P in each direction. The output S is the set of extreme points in each direction.

Intuitively, S is an (ϵ,δ)-hull iff B^d only contains a small region of bad vectors (with respect to P, S). Let T be an optimal ϵ-hull of P, with $|T| = k$. Fix $t \in T$. Consider the set E_t^T. In our proof we will show that with high probability each set E_t^T only contains a small subset of bad vectors, C_t', such that, for all vectors $v \in E_t^T \setminus C_t'$, S ϵ-maximizes P in v. Then we show that $\sum_{t\in T} V^d(C_t') \le \delta$, which completes the proof.

Suppose the selected random set of vectors is $A \subseteq B^d$. Fix $t \in T$. By Lemma 6.8, $t \in T$ ϵ-maximizes P for all vectors $v \in E_t^T$. Then by Lemma 6.11, there exists a finite-volume convex subset $C_t' \subseteq E_t^T$ such that $C_t' \cap A = \emptyset$ and for all $v \in E_t^T \setminus C_t'$, S ϵ-maximizes v. Next, for each $t \in T$, we select a large ellipsoid u_t contained in C_t' such that $V^d(C_t') \le V^d(u_t)d^d$. Note that $\cup_{t\in T} C_t'$ is a member of the family $\mathcal{E}^{|T|} = \{h_1 \cup h_2 \cup \ldots \cup h_{|T|} : \forall i,\ h_i \text{ is an ellipsoid}\}$. By Lemma 6.13, $\widetilde{\mathrm{VC}}(\mathcal{E}^{|T|}) \le 4kd^2$. By Lemma 6.12, since all u_t do not contain any point from A, with probability at least $1-\gamma$, it must be the case that $V^d(\cup_{t\in T} u_t)/V^d(B^d) \le \delta/(d^d)$. Since the u_t are disjoint, this means that $V^d(\cup_{t\in T} C_t')/V^d(B^d) \le \delta$. Furthermore, by Lemma 6.9, $\bigcup_{t\in T} E_t^T = B^d$. Therefore, with probability at least $1-\gamma$, S ϵ-maximizes all vectors in B^d except for those in $\cup_{t\in T} C_t'$. Thus, S is an (ϵ,δ)-hull except with probability γ. ◀

References

1 Pankaj K. Agarwal, Sariel Har-Peled, and Kasturi R. Varadarajan. Approximating extent measures of points. *J. ACM*, 51(4):606–635, 2004. `doi:10.1145/1008731.1008736`.

2 Pankaj K. Agarwal, Sariel Har-Peled, and Kasturi R. Varadarajan. *Geometric approximation via coresets*, pages 1–30. University Press, 2005.

3 Pankaj K Agarwal and Hai Yu. A space-optimal data-stream algorithm for coresets in the plane. In *Proceedings of the twenty-third annual symposium on Computational geometry*, pages 1–10. ACM, 2007.

4 Sunil Arya and Timothy M. Chan. Better epsilon-dependencies for offline approximate nearest neighbor search, euclidean minimum spanning trees, and epsilon-kernels. In *Proceedings of the Thirtieth Annual Symposium on Computational Geometry*, SOCG'14, pages 416:416–416:425, New York, NY, USA, 2014. ACM. `doi:10.1145/2582112.2582161`.

5 Jon Louis Bentley, Franco P. Preparata, and Mark G. Faust. Approximation algorithms for convex hulls. *Commun. ACM*, 25(1):64–68, jan 1982. `doi:10.1145/358315.358392`.

6 David M Blei. Probabilistic topic models. *Communications of the ACM*, 55(4):77–84, 2012.

7 Avrim Blum, Sariel Har-Peled, and Benjamin Raichel. Sparse approximation via generating point sets. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 548–557, 2016.

8 Timothy M. Chan. Faster core-set constructions and data-stream algorithms in fixed dimensions. *Computational Geometry*, 35(1):20–35, 2006. `doi:10.1016/j.comgeo.2005.10.002`.

9 Timothy M. Chan. Dynamic streaming algorithms for epsilon-kernels. In *Proc. 32nd Annu. Sympos. Comput. Geom. (SoCG)*, 2016.

10 Timothy M. Chan. Applications of chebyshev polynomials to low-dimensional computational geometry. In *Proc. 33rd Annu. Sympos. Comput. Geom. (SoCG)*, 2017.

11 John Hershberger and Subhash Suri. Adaptive sampling for geometric problems over data streams. In *Proceedings of the Twenty-third ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '04, pages 252–262, New York, NY, USA, 2004. ACM. `doi:10.1145/1055558.1055595`.

12 David M. Mount Sunil Arya, Guilherme D. da Fonseca. Near-optimal epsilon-kernel construction and related problems. In *Proc. 33rd Annu. Sympos. Comput. Geom. (SoCG)*, 2017.

13 Hamid Zarrabi-Zadeh. An almost space-optimal streaming algorithm for coresets in fixed dimensions. *Algorithmica*, 60(1):46–59, 2011. `doi:10.1007/s00453-010-9392-2`.