Sparse Regression via Range Counting

Jean Cardinal

Université libre de Bruxelles (ULB), Brussels, Belgium
jcardin@ulb.ac.be

Aurélien Ooms

BARC, University of Copenhagen, Denmark
aurelien.ooms@di.ku.dk

─── **Abstract** ───

The sparse regression problem, also known as best subset selection problem, can be cast as follows: Given a set S of n points in $\mathbb{R}^d$, a point $y \in \mathbb{R}^d$, and an integer $2 \leq k \leq d$, find an affine combination of at most k points of S that is nearest to y. We describe a $O(n^{k-1} \log^{d-k+2} n)$-time randomized $(1 + \varepsilon)$-approximation algorithm for this problem with d and ε constant. This is the first algorithm for this problem running in time $o(n^k)$. Its running time is similar to the query time of a data structure recently proposed by Har-Peled, Indyk, and Mahabadi (ICALP'18), while not requiring any preprocessing. Up to polylogarithmic factors, it matches a conditional lower bound relying on a conjecture about affine degeneracy testing. In the special case where $k = d = O(1)$, we provide a simple $O_\delta(n^{d-1+\delta})$-time deterministic exact algorithm, for any $\delta > 0$. Finally, we show how to adapt the approximation algorithm for the sparse linear regression and sparse convex regression problems with the same running time, up to polylogarithmic factors.

1 Introduction

Searching for a point in a set that is the closest to a given query point is certainly among the most fundamental problems in computational geometry. It motivated the study of crucial concepts such as multidimensional search data structures, Voronoi diagrams, dimensionality reduction, and has immediate applications in the fields of databases and machine learning. A natural generalization of this problem is to search not only for a single nearest neighbor, but rather for the nearest *em combination* of a bounded number of points. More precisely, given an integer k and a query point y, we may wish to find an affine combination of k points of the set that is the nearest to y, among all possible such combinations. This problem has a natural interpretation in terms of sparse approximate solutions to linear systems, and is known as the *sparse regression*, or *sparse approximation* problem in the statistics and machine learning literature. Sparsity is defined here in terms of the ℓ_0 pseudonorm $\|.\|_0$, the number of nonzero components. The *sparse affine regression* problem can be cast as follows:

▶ **Problem 1** (Sparse affine regression)**.** *Given a matrix $A \in \mathbb{R}^{d \times n}$, a vector $y \in \mathbb{R}^d$, and an integer $2 \leq k \leq d$, find $x \in \mathbb{R}^n$ minimizing $\|Ax - y\|_2$, and such that $\|x\|_0 \leq k$, and $\sum_{i=1}^n x_i = 1$.*

By interpreting the columns of A as a set of n points in $\mathbb{R}^d$, the problem can be reformulated in geometric terms as the *nearest induced flat* problem.

▶ **Problem 2** (Nearest induced flat)**.** *Given a set S of n points in $\mathbb{R}^d$, an additional point $y \in \mathbb{R}^d$, and an integer k such that $2 \leq k \leq d$, find k points of S such that the distance from y to their affine hull is the smallest.*

Here the distance from a point to a flat is the distance to the closest point on the flat. Note that if we allow $k = 1$ in the definition above, we have the nearest neighbor problem as a special case. We consider the setting in which the dimension d of the ambient space as well as the number k of points in the sought combination are constant, and study the asymptotic complexity of the problem with respect to n. As observed recently by Har-Peled, Indyk, and Mahabadi [25], the problem is closely related to the classical *affine degeneracy testing* problem, defined as follows.

▶ **Problem 3** (Affine degeneracy testing)**.** *Given a set S of n points in $\mathbb{R}^d$, decide whether there exist $d + 1$ distinct points of S lying on an affine hyperplane.*

The latter can be cast as deciding whether a point set is in so-called *general position*, as is often assumed in computational geometry problems. In the special case $d = 2$, the problem is known to be 3SUM-hard [24, 9]. In general, it is not known whether it can be solved in time $O(n^{d-\delta})$ for some positive δ [21, 3], even for randomized algorithms. Supposing it cannot, we directly obtain a conditional lower bound on the complexity of the nearest induced flat problem. This holds even for approximation algorithms, which return an induced flat whose distance is within some bounded factor of the distance of the actual nearest flat.

▶ **Lemma 1** (Har-Peled, Indyk, and Mahabadi [25])**.** *If the nearest induced flat problem can be approximated within any multiplicative factor in time $O(n^{k-1-\delta})$ for some positive δ, then affine degeneracy testing can be solved in time $O(n^{d-\delta})$.*

Proof. Suppose we have an approximation algorithm for the nearest induced flat problem. Then given an instance of affine degeneracy testing, we can go through every point $y \in S$ and run this algorithm on an instance composed of the set $S \setminus \{y\}$, the point y, and $k = d$. The answer to the degeneracy testing instance is positive if and only if for at least one of these instances, the distance to the approximate nearest flat is zero. The running time is $O(n^{d-\delta})$. ◀

Motivations and previous works

Sparse regression is a cornerstone computational task in statistics and machine learning, and comes in a number of flavors. It is also referred to as *best subset selection* or, more generally, as *feature selection* problems [31, 11]. In practice, it is often useful to allow for the sparsity constraint by including a penalty term in the objective function, hence writing the problem in a Lagrangian form. If the ℓ_1 norm is used instead of the ℓ_0 norm, this method is known as the LASSO method [32], to which a tremendous amount of research has been dedicated in the past twenty years. In the celebrated k-SVD algorithm for sparse dictionaries design [2], the sparse coding stage consists of a number of sparse regression steps. In this context, they are typically carried out using greedy methods such as the matching pursuit algorithm [29]. Efficient sparse regression is also at the heart of *compressed sensing* techniques [13, 18].

Aiming at an exhaustive survey of the variants and applications of sparse regression is futile; instead, we refer to Hastie, Tibshirani, and Friedman [26] (Chapter 3), Miller [30], and references therein. We also point to Bertsimas, Pauphilet, and Van Parys [12] for a recent survey on practical aspects of sparse regression methods.

The computational complexity of sparse regression problems is also well-studied [31, 17, 23, 22]. In general, when a solution x is sought that minimizes the number of nonzero components while being at bounded distance from y, the problem is known to be NP-hard [31]. However, the complexity of the sparse regression problem when the sparsity constraint k is taken as a fixed parameter has not been thoroughly characterized. In particular, no algorithm with running time $o(n^k)$ is known.

Recently, Har-Peled, Indyk, and Mahabadi [25] showed how to use approximate nearest neighbor data structures for finding approximate solutions to the sparse affine regression problem. They mostly consider the *online* version of the problem, in which we allow some preprocessing time, given the input point set S, to construct a data structure, which is then used to answer queries with input y. They also restrict to *approximate* solutions, in the sense that the returned solution has distance at most $(1 + \varepsilon)$ times larger than the true nearest neighbor distance for any fixed constant ε. They show that if there exists a $(1 + \varepsilon)$-approximate nearest neighbor data structure with preprocessing time $S(n, d, \varepsilon)$ and query time $Q(n, d, \varepsilon)$, then we can preprocess the set S in time $n^{k-1}S(n, d, \varepsilon)$ and answer regression queries in time $n^{k-1}Q(n, d, \varepsilon)$. Plugging in state of the art results on approximate nearest neighbor searching in fixed dimension [8], we obtain a preprocessing time of $O(n^k \log n)$ with query time $O(n^{k-1} \log n)$ for fixed constants d and ε.

They also consider the *sparse convex regression* problem, in which the coefficients of the combination are not only required to sum to one, but must also be nonnegative. In geometric terms, this is equivalent to searching for the *nearest induced simplex*. They describe a data structure for the sparse convex regression problem having the same performance as in the affine case, up to a $O(\log^k n)$ factor. For $k = 2$, they also give a $(2 + \varepsilon)$-approximation subquadratic-time offline algorithm. When $d = O(1)$, the running time of this algorithm can be made close to linear.

A closely related problem is that of searching for the nearest flat in a set [27, 10, 28]. This was also studied recently by Agarwal, Rubin, and Sharir [1], who resort to polyhedral approximations of the Euclidean distance to design data structures for finding an approximate nearest flat in a set. They prove that given a collection of n $(k-1)$-dimensional flats in $\mathbb{R}^d$, they can construct a data structure in time $O(n^k \operatorname{polylog}(n))$ time and space that can be used to answer $(1 + \varepsilon)$-approximate nearest flat queries in time $O(\operatorname{polylog}(n))$. They also consider the achievable space-time tradeoffs. Clearly, such a data structure can be used for online sparse affine regression: We build the structure with all possible $\binom{n}{k}$ flats induced by the points of S. This solution has a very large space requirement and does not help in the offline version stated as Problem 2.

In this paper, we give an efficient algorithm for Problem 2, and bridge the gap between the trivial upper bound of $O(n^k)$ and the lower bound given by the affine degeneracy testing problem, without requiring any preprocessing.

Our results

Nearest induced line, flat, or hyperplane

We prove that the nearest induced flat problem (Problem 2), can be solved within a $(1 + \varepsilon)$ approximation factor for constant d and ε in time $O(n^{k-1} \log^{d-k+2} n)$, which matches the conditional lower bound on affine degeneracy testing, up to polylogarithmic factors. Har-

Peled, Indyk, and Mahabadi [25] gave a data structure to preprocess a set of data points to allow solving the nearest induced flat problem on this set for any query point. Their data structure requires $\tilde{O}(n^k)$ preprocessing and $\tilde{O}(n^{k-1})$ query time. We propose an algorithm that gets rid of the preprocessing for single queries: the overall running time of our algorithm is equal to the query time of their data structure, up to polylogarithmic factors. To the best of our knowledge, this is a near-linear improvement on all previous methods for this special case.

The two main tools that are used in our algorithms are on the one hand the approximation of the Euclidean distance by a polyhedral distance, as is done in Agarwal, Rubin, and Sharir [1], and on the other hand a reduction of the decision version of the problem to orthogonal range queries. Note that orthogonal range searching data structures are also used in [25], albeit in a significantly distinct fashion.

In §2, as warm-up, we focus on the special case of Problem 2 in which $d = 3$ and $k = 2$.

▶ **Problem 4** (Nearest induced line in $\mathbb{R}^3$). *Given a set S of n points in $\mathbb{R}^3$, and an additional point y, find two points $a, b \in S$ such that the distance from y to the line going through a and b is the smallest.*

Our algorithm for this special case already uses all the tools that are subsequently generalized for arbitrary values of k and d. The general algorithm for the nearest induced flat problem is described in §3.

In §4, we consider the special case of Problem 2 in which $k = d$, which can be cast as the *nearest induced hyperplane* problem.

▶ **Problem 5** (Nearest induced hyperplane). *Given a set S of n points in $\mathbb{R}^d$, and an additional point y, find d points of S such that the distance from y to the affine hyperplane spanned by the d points is the smallest.*

For this case, we design an *exact* algorithm with running time $O(n^{d-1+\delta})$, for any $\delta > 0$. The solution solely relies on classical computational geometry tools, namely point-hyperplane duality and cuttings [16, 15].

Our algorithms can be adapted to perform *sparse linear regression*, instead of sparse affine regression. In the former, we drop the condition that the sum of the coefficients must be equal to one. This is equivalent to the nearest *linear* induced k-flat problem. It can be solved in the same time as in the affine case. To see this, realize that the problem is similar to the nearest induced flat problem where the first vertex is always the origin. The obtained complexity is the same as the one for the nearest induced flat problem.

Nearest induced simplex

Adapting our algorithm to sparse *convex* regression, which differs from sparse affine regression by requiring x to be positive, is a bit more involved.

Har-Peled, Indyk, and Mahabadi [25] augment their data structure for the nearest induced flat with orthogonal range searching data structures in $(k + 1)$-dimensional space to solve this problem with an extra $O(\log^k n)$ factor in both the preprocessing and query time. We show we can perform a similar modification.

The sparse convex regression problem can be cast as the problem of finding the nearest simplex induced by k points of S.

▶ **Problem 6** (Nearest induced simplex). *Given a set S of n points in $\mathbb{R}^d$, an additional point y, and an integer k such that $2 \le k \le d$, find k points of S such that the distance from y to their convex hull is the smallest.*

■ **Table 1** Results. For the approximation algorithms, the dependency on ε in the running time is of the order of $\varepsilon^{(1-d)/2}$.

Problem	Details	Approximation	Running Time
Problem 4: Nearest induced line in $\mathbb{R}^3$	§2	$1 + \varepsilon$	$O_\varepsilon(n \log^3 n)$
Problem 2: Nearest induced flat	§3	$1 + \varepsilon$	$O_{d,\varepsilon}(n^{k-1} \log^{d-k+2} n)$
Problem 5: Nearest induced hyperplane	§4	1	$O_{d,\delta}(n^{d-1+\delta}), \ \forall \delta > 0$
Problem 6: Nearest induced simplex	§5	$1 + \varepsilon$	$O_{d,\varepsilon}(n^{k-1} \log^d n)$

We prove that this problem can also be approximated within a $(1 + \varepsilon)$ approximation factor for constant d and ε in time $O(n^{k-1} \log^d n)$, hence with an extra $O(\log^{k-2} n)$ factor in the running time compared to the affine case. This is described in §5.

Our results and the corresponding sections are summarized in Table 1.

2 A $(1 + \varepsilon)$-approximation algorithm for the nearest induced line problem in $\mathbb{R}^3$

We first consider the nearest induced line problem (Problem 4). We describe a near-linear time algorithm that returns a $(1 + \varepsilon)$-approximation to the nearest induced line in $\mathbb{R}^3$, that is, a line at distance at most $(1 + \varepsilon)$ times larger than the distance to the nearest line.

▶ **Theorem 2.** *For any constant $\varepsilon > 0$, there is a randomized $(1+\varepsilon)$-approximation algorithm for the nearest induced line problem in $\mathbb{R}^3$ running in time $O_\varepsilon(n \log^3 n)$ with high probability.*

The sketch of our algorithm is as follows: First, reduce the problem of minimizing the Euclidean distance to that of minimizing the polyhedral distance for some well-chosen polyhedron depending on ε. Second, reduce the problem of minimizing the polyhedral distance to that of edge-shooting. Third, reduce the problem of edge-shooting to that of deciding whether an edge shot at a certain distance would hit any induced line through some sort of binary search. Fourth, efficiently solve this decision problem using orthogonal range counting data structures.

$(1 + \varepsilon)$-approximation via polyhedral distances

The *polyhedral distance* $d_Q(y, v)$ between two points y and v with respect to a polyhedron Q centered on the origin is the smallest λ such that the dilation λQ of Q centered on y contains v, hence such that $v \in y + \lambda Q$. Our proof uses the following result, of which a weaker variant due to Dudley [19] is a major ingredient in the design of the data structure described by Agarwal, Rubin, and Sharir [1].

▶ **Lemma 3** (Arya, Arya, da Fonseca, Mount [4]). *For any positive integer d and positive real ε, there exists a d-dimensional polyhedron Q with $O(1/\varepsilon^{(d-1)/2})$ faces such that for every $y, v \in \mathbb{R}^d$:*

$$\|y - v\|_2 \leq d_Q(y, v) \leq (1 + \varepsilon) \cdot \|y - v\|_2 \,.$$

This bound is asymptotically optimal. See [5, 7, 6] for more details.

Next, we reduce Problem 4 to a counting problem in two steps.
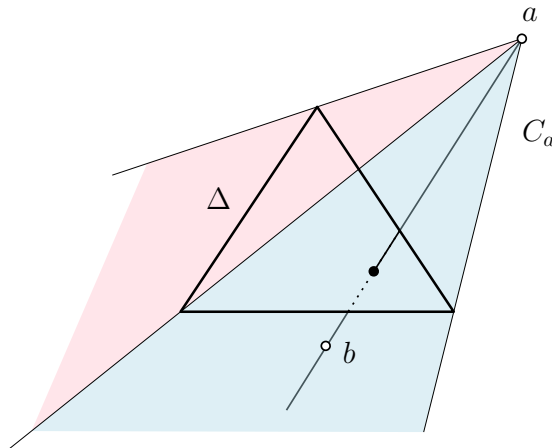
Figure 1 The cone C_a.

Edge-shooting

We use Lemma 3 for $d = 3$. We give an exact algorithm for computing the nearest induced line with respect to a polyhedral distance d_Q, where Q is defined from ε as in Lemma 3. Given a polyhedron Q, one can turn it into a simplicial polyhedron by triangulating it. Therefore, for constant values of ε, this reduces the problem to a constant number of instances of the *edge-shooting problem*, defined as follows: Given an edge e of Q, find the smallest value λ such that $y + \lambda e$ intersects a line through two points of S. We iterate this for all edges of Q, and pick the minimum value. This is exactly the polyhedral distance from y to its nearest induced line.

Binary search

Using a randomized binary search procedure, we reduce the edge-shooting problem to a *counting problem*, defined as follows: given the triangle Δ defined as the convex hull of y and $y + \lambda e$, count how many pairs of points $a, b \in S$ are such that the line $\ell(a, b)$ through them intersects Δ. Suppose there exists a procedure for solving this problem. We can use this procedure to solve the edge-shooting problem efficiently as follows.

First initialize λ to some upper bound on the distance (for instance, initialize λ to the distance to the closest data point $p \in S$: $\lambda = \min_{p \in S} \|p - y\|_2$). Then count how many lines $\ell(a, b)$ intersect Δ, using the procedure. If there is only one, then return its (polyhedral) distance to y. Otherwise, pick one such line uniformly at random and compute the value λ' such that this line intersects $y + \lambda' e$. Then iterate the previous steps with $\lambda \leftarrow \lambda'$, unless $\lambda' = 0$ in which case we return 0. Since we picked the line at random, and since there are $O(n^2)$ such lines at the beginning of the search, the number of iterations of this binary search is $O(\log n)$ with high probability.

We therefore reduced the nearest induced line problem to $O(\varepsilon^{-1} \log n)$ instances of the counting problem.

Orthogonal range counting queries

Data structures for *orthogonal range counting queries* store a set of points in $\mathbb{R}^g$ in such a way that the number of points in a given g-rectangle (cartesian product of g intervals) can be returned quickly. Known data structures for orthogonal range counting queries in $\mathbb{R}^g$
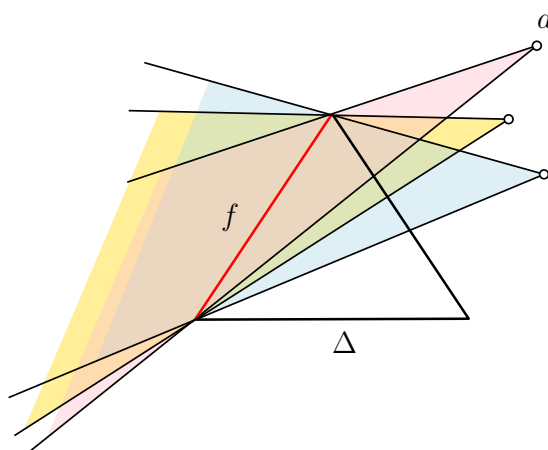
■ **Figure 2** The order of the points defined by the planes containing an edge f of Δ.

require $O(n \log^{g-1} n)$ preprocessing time and can answer queries in $O(\log^{g-1} n)$ time [34, 14].
Note that the actual coordinates of the points do not matter: We only need to know the
order of their projections on each axis. We now show how to solve the counting problem
using a data structure for orthogonal range queries in $\mathbb{R}^3$.

Let us fix the triangle Δ and a point $a \in \mathbb{R}^3$, and consider the locus of points $b \in \mathbb{R}^3$
such that the line $\ell(a, b)$ intersects Δ. This is a double simplicial cone with apex a and
whose boundary contains the boundary of Δ. This double cone is bounded by three planes,
one for each edge of Δ. In fact, we will only consider one of the two cones, because $\ell(a, b)$
intersects Δ if and only if either b is contained in the cone of apex a, or a is contained in the
cone of apex b. Let us call C_a the cone of apex a. This is illustrated on Figure 1.

Let us consider one edge f of Δ and all the planes containing f. These planes induce a
circular order on the points of S, which is the order in which they are met by a plane rotating
around the supporting line of f. This is illustrated on Figure 2. Now let us denote by H_f
the plane containing a and f and by H_f^+ the halfspace bounded by H_f and containing Δ.
The set of points of S contained in H_f^+ is an interval in the circular order mentioned above.
Hence the set of points contained in C_a is the intersection of three intervals in the three
circular orders defined by the three edges of Δ.

Proof of Theorem 2. Let Q be some polyhedron in $\mathbb{R}^3$, $\lambda \in \mathbb{R}$, $S \subset \mathbb{R}^3$, $y \in \mathbb{R}^3$, and e an
edge of Q. We use an orthogonal range counting data structure for storing the points of S
with coordinates corresponding to their ranks in each of the three permutations induced by
the three edges of $\Delta = \text{conv}(\{y, y + \lambda e\})$. We get those rank-coordinates by sorting S three
times, once for each induced permutation, in time $O(n \log n)$, then construct the orthogonal
range counting data structure with those coordinates in time $O(n \log^2 n)$. Then for each of
the n points $a \in S$, we count the number of points b in the cone C_a by querying the data
structure in $O(\log^2 n)$ time. Hence overall, the counting problem is solved in time $O(n \log^2 n)$.
Note that the circularity of the order can be easily handled by doubling every point.

This can be combined with the previous reductions provided we can choose a line
intersecting Δ uniformly at random within that time bound. This is achieved by first
choosing a with probability proportional to the number of points b such that $\ell(a, b) \cap \Delta \neq \emptyset$.
Then we can pick a point b uniformly at random in this set in linear time.

Combining with the previous reductions, we obtain an approximation algorithm running
in time $O_\varepsilon(n \log^3 n)$ for the nearest induced line problem in $\mathbb{R}^3$. ◀

3 A $(1 + \varepsilon)$-approximation algorithm for the nearest induced flat problem

This section is dedicated to proving our main result in full generality. We provide an efficient approximation algorithm for the nearest induced flat problem (Problem 2).

We use the following notations: $\mathrm{aff}(X)$ denotes the affine hull of the set X and $\mathrm{conv}(X)$ denotes its convex hull. The set $\{1, 2, \ldots, n\}$ is denoted by $[n]$.

▶ **Theorem 4.** *For any constant positive real $\varepsilon > 0$ and constant positive integers d and k, there is a randomized $(1 + \varepsilon)$-approximation algorithm for the nearest induced flat problem in $\mathbb{R}^d$ running in time $O_\varepsilon(n^{k-1} \log^{d-k+2} n)$ with high probability.*

Proof. The algorithm is a generalization of the one in the previous section, in which the point a is replaced by a set composed of $k - 1$ points $a_1, a_2, \ldots, a_{k-1}$, and the edge e is now a (simplicial) $(d - k)$-face of Q. Given a $k - 1$-tuple of points $a_1, a_2, \ldots, a_{k-1}$, we characterize the locus of points a_k such that the affine hull of the points $a_1, a_2, \ldots, a_k$ intersects the convex hull of y and $y + \lambda e$. These hyperplanes are again such that counting all such points can be done using orthogonal range queries. More precisely, we perform the following steps.

### $(1 + \varepsilon)$-approximation and binary search

From Lemma 3, there exists a polyhedron with $O(1/\varepsilon^{(d-1)/2})$ faces such that the induced polyhedral distance $d_Q(., .)$ is a $(1 + \varepsilon)$-approximation of the Euclidean distance. We know that the distance d_Q from the point y to the nearest induced flat is attained at a point lying on a $(d - k)$-face of $y + \lambda Q$. We can therefore perform the same procedure as in the previous case, except that we now shoot a $(d - k)$-face e of Q, instead of an edge, in the same way as is done in Agarwal, Rubin, Sharir [1]. Δ still denotes the convex hull of y and $y + \lambda e$, which generalizes to a $(d - k + 1)$-simplex. The binary search procedure generalizes easily: start with a large enough λ, if there is more than one flat $\mathrm{aff}(\{a_1, a_2, \ldots, a_k\})$ intersecting $\Delta = \mathrm{conv}(\{y, y + \lambda e\})$, pick one such flat uniformly at random, and compute the value λ such that this flat intersects Δ. There are only $O(n^k)$ such flats at the beginning of the search, hence a search takes $O(\log n)$ steps with high probability. We can therefore reduce the problem to $O(\varepsilon^{(1-d)/2} \log n)$ instances of the following *counting problem*: given a $(d - k + 1)$-simplex Δ, count the number of k-tuples of points $a_1, a_2, \ldots, a_k \in S$ whose affine hull $\mathrm{aff}(a_1, a_2, \ldots, a_k)$ intersects Δ.

An intersection condition

We first make a simple observation that characterizes such k-tuples. Let A be a set of k points $\{a_1, a_2, \ldots, a_k\}$, and let $B = \{b_1, b_2, \ldots, b_{d-k+2}\}$ be the set of vertices of Δ. We assume without loss of generality that the points of A together with the vertices of Δ are in general position. We define $d - k + 2$ hyperplanes $H_i = \mathrm{aff}(A \cup B \setminus \{b_i, a_k\}), i \in [d - k + 2]$. We then let H_i^+ be the halfspace supported by H_i that contains b_i, and H_i^- the halfspace that does not contain b_i.

▶ **Lemma 5.**

$$\mathrm{aff}(A) \cap \Delta \neq \emptyset \iff a_k \in \left(\left(\bigcap_{i=1}^{d-k+2} H_i^+ \right) \cup \left(\bigcap_{i=1}^{d-k+2} H_i^- \right) \right).$$
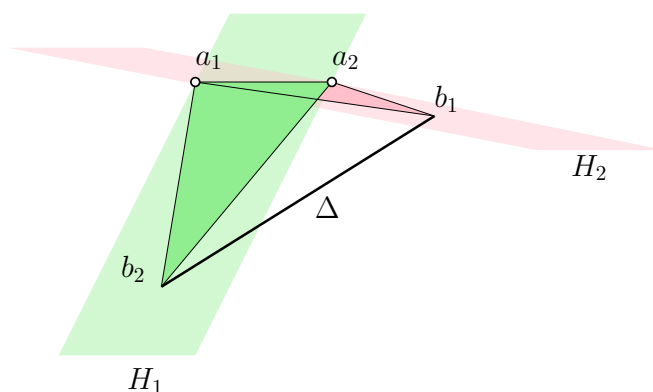
Figure 3 Illustration of Lemma 5 in the case $k = d = 3$. The plane through a_1, a_2, a_3 intersects the line segment Δ if and only if a_3 is located either above or below the two planes H_1, H_2.

Proof. ($\Rightarrow$) Suppose that $a_k \notin (\bigcap_i H_i^+) \cup (\bigcap_i H_i^-)$. Hence there exists $i \in [d - k + 2]$ such that $a_k \in H_i^-$, and $j \in [d - k + 2]$ such that $a_k \in H_j^+$. We show that $\text{aff}(A) \cap \Delta = \emptyset$. Let us consider the intersection of the two halfspaces H_i^- and H_j^+ with the $(k - 1)$-dimensional subspace $\text{aff}(A)$. In this subspace, both halfspaces have the points $a_1, a_2, \ldots, a_{k-1}$ on their boundary, and contain a_k. Hence it must be that $H_i^- \cap \text{aff}(A) = H_j^+ \cap \text{aff}(A)$. Therefore, every point $p \in \text{aff}(A)$ either lies in H_i^-, or in H_j^-. In both cases, it is separated from Δ by a hyperplane, and $p \notin \Delta$.

($\Leftarrow$) Suppose that $\text{aff}(A) \cap \Delta = \emptyset$. We now show that there exists $i \in [d - k + 2]$ such that $a_k \in H_i^-$, and $j \in [d - k + 2]$ such that $a_k \in H_j^+$. Since both $\text{aff}(A)$ and Δ are convex sets, if $\text{aff}(A) \cap \Delta = \emptyset$ then there exists a hyperplane H containing $\text{aff}(A)$ and having Δ on one side. Since the points of A are affinely independent, H can be rotated to contain all points of A except a_k, and separate a_k from Δ. After this rotation, H has $d - (k - 1)$ degrees of freedom left, and can be further rotated to contain a whole $(d - k)$-face of Δ, while still separating Δ from a_k. For some $i \in [d - k + 2]$, this is now the hyperplane H_i that separates some vertex b_i from a_k, and $a_k \in H_i^-$.

Similarly, the same hyperplane H can instead be rotated in order to contain all points of A except a_k, and have a_k and Δ this time on the same side. It can then be further rotated to contain a $(d - k)$-face of Δ, while still having Δ and a_k on the same side. Now for some $j \in [d - k + 2]$, this is now the hyperplane H_j that has b_j and a_k on the same side, and $a_k \in H_j^+$.   ◀

Note that for the case $k = 2$ and $d = 3$ the set $(\bigcap_i H_i^+) \cup (\bigcap_i H_i^-)$ is the double cone of apex a; the lower part $(\bigcap_i H_i^+)$ is the cone C_a in Figure 1. The case where $k = 3$ and $d = 3$ is illustrated on Figure 3.

Reduction to orthogonal range queries

We now show that in perfect analogy with the previous section, we can solve the counting problem efficiently using an orthogonal range counting data structure.

Consider a vertex b_i of Δ and a $(k - 2)$-subset T of points of S, denoted by $T = \{a_1, a_2, \ldots, a_{k-2}\}$. Let us denote by f the facet of Δ that is induced by the vertices b_j such that $j \neq i$. Now consider the hyperplane containing f together with T, and one additional point p of S. These hyperplanes all contain $\text{aff}(f \cup T)$, which is a $(d - 2)$-flat. Let us consider the unit normal vectors to these hyperplanes centered on some point contained in

this $(d-2)$-flat. These vectors lie in the orthogonal flat of dimension $d - (d-2) = 2$, hence in a plane. Therefore, they induce a circular order on the points of S. Hence for a fixed set of $k-2$ points of S and a fixed facet f of Δ, we can assign a rank to each other point of S. These will play the role of the coordinates of the points in the range counting data structure.

We now observe that counting the number of k-tuples whose affine hull intersects Δ amounts to orthogonal range counting with respect to these coordinates. Indeed, fix the first $(k-2)$-subset of points $T = \{a_1, a_2, \ldots, a_{k-2}\}$, and compute the rank of each other point of S with respect to the circular order of the hyperplanes defined above, around each facet f of Δ. Now consider a $(k-1)$th point a_{k-1}. From Lemma 5, all points a_k contained in the range $(\bigcap_i H_i^+) \cup (\bigcap_i H_i^-)$ are such that $\mathrm{aff}(a_1, a_2, \ldots, a_k)$ intersects Δ. But this range is the union of two $(d-k+2)$-rectangles in the space of coordinates that we defined. The coordinates of these two $(d-k+2)$-rectangles are defined by the coordinates of a_{k-1}. We can therefore set up a new orthogonal range counting data structure for each $(k-2)$-subset T, and perform $2n$ queries in it, two for each additional point $a_{k-1} \in S$.

We can now outline our algorithm for solving the counting problem:

1. For each $(k-2)$-subset T of points $a_1, a_2, \ldots, a_{k-2}$ in $\binom{S}{k-2}$:
   a. For each vertex b_i of Δ, compute the rank of each point of S with respect to the hyperplanes containing $f = \mathrm{conv}(\{b_j : j \neq i\})$ and T.
   b. Build a $(d-k+2)$-dimensional range counting data structure on S using these ranks as coordinates.
   c. For each other point $a_{k-1} \in S$:
      i. Perform two range counting queries using the rectangular ranges corresponding to $\bigcap_i H_i^+$ and $\bigcap_i H_i^-$, respectively.
   d. Return the sum of the values returned by the range counting queries.

Note that there are a few additional technicalities which we have to take care of. First, the orders defined by the hyperplanes are circular, hence we are really performing range queries on a torus. This can be easily fixed, as mentioned previously, by doubling each point. Then we have to make sure to avoid double counting, since any permutation of the a_i in the enumeration of k-tuples yields the same set A, and hence, the same flat $\mathrm{aff}(A)$. (Note that in §2 we avoided double counting by observing that only one of $a \in C_b$ and $b \in C_a$ can be true.) This only affects the counting problem and is not problematic if we consider *ordered* subsets T; it causes each intersecting flat to be counted exactly $k!$ times.[1] The termination condition for the binary search can be changed to when the range count is $k!$ and the sampling method for finding a uniform random binary search pivot is unaffected since each candidate flat is represented an equal number of times.

As for the running time analysis, step 1b costs $O(n \log^{d-k+1} n)$, while step 1(c)i costs $O(\log^{d-k+1} n)$ and is repeated $n - k + 2$ times, hence costs $O(n \log^{d-k+1} n)$ overall as well [34, 14]. These are multiplied by the number of iterations of the main loop, yielding a complexity of $O(n^{k-1} \log^{d-k+1} n)$ for the counting procedure.

Finally, this counting procedure can be combined with the binary search procedure provided we can choose a flat intersecting Δ uniformly at random within that time bound. This is achieved by first choosing a set prefix $\{a_1, a_2, \ldots, a_{k-1}\} \in \binom{S}{k-1}$ with probability proportional to the number of points $a_k \in S$ such that $\mathrm{aff}(\{a_1, a_2, \ldots, a_k\}) \cap \Delta \neq \emptyset$. Then we can pick a point a_k uniformly at random in this set in linear time. Multiplying by the number of edge-shooting problems we have to solve, the counting procedure is invoked $O(\varepsilon^{(1-d)/2} \log n)$ times, yielding the announced running time.   ◀

---

[1] Enumerating each subset T exactly once as $(k-2)$-tuples in lexicographic order and only constructing the orthogonal range searching data structure on the points of S that come after a_{k-2} reduces this overcounting to 2 times per flat. In our case, this is unnecessary since k is constant.
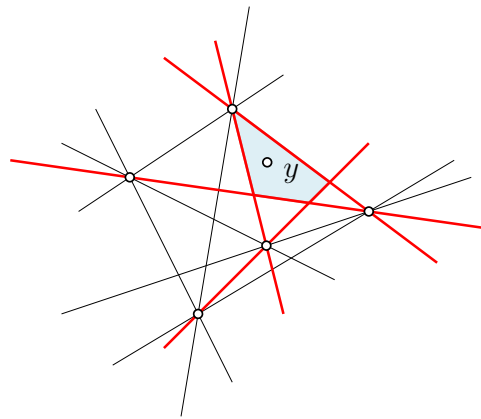
■ **Figure 4** The candidate nearest hyperplanes.

4   An exact algorithm for the nearest induced hyperplane problem

In this section we consider the special case $k = d$, the nearest induced hyperplane problem (Problem 5). The previous result gives us a randomized $(1 + \varepsilon)$-approximation algorithm running in time $O_\varepsilon(n^{d-1} \log^2 n)$ for this problem. We describe a simple deterministic $O(n^{d-1+\delta})$-time exact algorithm using only standard tools from computational geometry.

▶ **Theorem 6.** *The nearest induced hyperplane problem can be solved in deterministic time $O(n^{d-1+\delta})$ for any $\delta > 0$.*

The first tool we need is point-hyperplane duality. Let $\bar{H}$ be the hyperplane arrangement that is dual to S, in which each point of S is now a hyperplane. Note that every vertex of this arrangement is the dual of a hyperplane induced by d points of S.

Unfortunately, while some dualities preserve vertical distances, there does not exist a duality that preserves euclidean distances. To overcome this obstacle, we make a topological observation. Recall that the *zone* of a hyperplane h in an arrangement $\bar{H}$ (not including h) is the union of the d-cells of $\bar{H}$ intersected by h. Similarly, we define the *refined zone* of a hyperplane h in an arrangement $\bar{H}$ (not including h) to be the union of the d-simplices of the bottom-vertex decomposition of $\bar{H}$ intersected by h.

▶ **Lemma 7.** *Let $\bar{H}$ be the hyperplane arrangement that is dual to S, and $\bar{y}$ the hyperplane dual to the point y. The induced hyperplane that is nearest to y corresponds to a vertex of the* refined zone *of $\bar{y}$ in the arrangement $\bar{H}$.*

Proof. Consider the arrangement of all $\binom{n}{k}$ hyperplanes induced by subsets of k points in S. Then clearly, the induced hyperplane nearest to y must be one of the hyperplanes bounding the cell of this arrangement that contains y (see Figure 4 for an illustration with $d = 2$). Consider a rectilinear motion of y towards this nearest hyperplane. In the dual arrangement $\bar{H}$, this corresponds to a continuous motion of the hyperplane $\bar{y}$ that at some point hits a vertex of the arrangement. Because it is the first vertex that is hit, it must belong to a cell of the bottom vertex decomposition of $\bar{H}$ that $\bar{y}$ intersects, hence to the refined zone of $\bar{y}$.   ◀

We refer to chapter 28 of the Handbook of Discrete and Computational Geometry [33] for background on hyperplane arrangements and their decompositions.

The second tool is an upper bound on the complexity of a zone in an arrangement [20]. The *complexity of a zone* is the sum of the complexities of its cells, and the complexity of a cell is the number of faces of the cell (vertices, edges, . . .). The upper bound is as follows:

▶ **Theorem 8** (Zone Theorem [20]). *The complexity of a zone in an arrangement of n hyperplanes in $\mathbb{R}^d$ is $O(n^{d-1})$.*

In particular, this result gives an upper bound of $O(n^{d-1})$ vertices for a given zone. Since the complexity of a refined zone is not more than the complexity of the corresponding zone, this bound also holds for the complexity of a given refined zone.

The third tool is Chazelle's efficient construction of cuttings [15]. A *cutting* of $\mathbb{R}^d$ is a partition of $\mathbb{R}^d$ into disjoint regions. Given a set of hyperplanes H in $\mathbb{R}^d$, a $\frac{1}{r}$-*cutting* for H is a cutting of $\mathbb{R}^d$ such that each region is intersected by no more than $\frac{|H|}{r}$ hyperplanes in H. In particular, we are interested in Chazelle's construction when r is constant. In that case, only a single step of his construction is necessary and yields regions that are the simplices of the bottom-vertex decomposition of some subset of H.

▶ **Theorem 9** (Chazelle [15, Theorem 3.3]). *Given a set H of n hyperplanes in $\mathbb{R}^d$, for any real constant parameter $r > 1$, we can construct a $\frac{1}{r}$-cutting for those hyperplanes consisting of the $O(r^d)$ simplices of the bottom-vertex decomposition of some subset of H in $O(n)$ time.*

More details on cuttings can be found in chapters 40 and 44 of the Handbook [33].

▶ **Lemma 10.** *For any positive constant δ, given a hyperplane h and an arrangement of hyperplanes $\bar{H}$ in $\mathbb{R}^d$, the vertices of the refined zone of h in $\bar{H}$ can be computed in time $O(n^{d-1+\delta})$.*

Proof. Using Theorem 9 with some constant r, we construct, in linear time, a $\frac{1}{r}$-cutting of the arrangement consisting of $O(r^d)$ simplicial cells whose vertices are vertices of $\bar{H}$. To find the vertices of the refined zone, we only need to look at those cells that are intersected by $\bar{y}$. If such a cell is not intersected by any hyperplane of $\bar{H}$ then its vertices are part of the refined zone of $\bar{y}$. Otherwise, we recurse on the hyperplanes intersecting that cell. From Theorem 8, there are at most $O(r^{d-1})$ such cells. The overall running time for the construction is therefore:

$$T(n) \leq O\left(r^{d-1}\right) T\left(\frac{n}{r}\right) + O(n).$$

For all constant $\delta > 0$, we can choose a sufficiently large constant r, such that $T(n) = O(n^{d-1+\delta})$, as claimed. ◀

Proof of Theorem 6. From Lemma 10, we find the vertices of the refined zone of $\bar{y}$ in the arrangement $\bar{H}$ in time $O(n^{d-1+\delta})$. Then we compute the distance from y to each of the induced hyperplanes corresponding to vertices of the refined zone in time $O(n^{d-1})$. From Lemma 7, one of them must be the nearest. ◀

5 A $(1 + \varepsilon)$-approximation algorithm for the nearest induced simplex problem

We now consider the nearest induced simplex problem (Problem 6). The algorithm described in §2 for the case $k = 2$ and $d = 3$ can be adapted to work for this problem.

As in §2, consider the computation of the nearest induced segment under some polyhedral distance d_Q approximating the Euclidean distance. The reduction from this computation to edge-shooting still works with some minor tweak: if we shoot edges to find the nearest induced segment under d_Q, we may miss some of the segments. Fortunately, the points of these missed segments that are nearest to our query point under d_Q must be endpoints of

those segments. We can take those into account by comparing the nearest segment found by edge-shooting to the nearest neighbor, found in linear time. As before, edge-shooting is reduced to a counting problem.

Referring to the proof of Theorem 2 and Figure 1, the analogue of the counting problem in §2 for the nearest induced segment problem amounts to searching for the points b lying in the intersection of the cone C_a with the halfspace bounded by $\text{aff}(\Delta)$ that does not contain a. In dimension d, the affine hull of Δ is a hyperplane, and we restrict b to lie on one side of this hyperplane.

We therefore get a $(1 + \varepsilon)$-approximation $O(n \log^d n)$-time algorithm for the *nearest induced segment* problem in any fixed dimension d. This compares again favorably with the $(2 + \varepsilon)$-approximation $O(n \log n)$-time algorithm proposed in [25].

We generalize this to arbitrary values of k and prove the following result.

▶ **Theorem 11.** *For any constant positive real $\varepsilon > 0$ and constant positive integers d and k, there is a randomized $(1 + \varepsilon)$-approximation algorithm for the nearest induced simplex problem in $\mathbb{R}^d$ running in time $O(n^{k-1} \log^d n)$ with high probability.*

Again, we compute the nearest induced simplex under some polyhedral distance d_Q. As in the case $k = 2$, $(d - k)$-face-shooting can be adapted to take care of missed simplices: for each $2 \le k' \le k$, shoot $(d - k')$-faces of Q to find the nearest $(k' - 1)$-simplex. For $k' = 1$, find the nearest neighbor in linear time. For any $(k - 1)$-simplex, let $0 \le k' \le k$ be the smallest natural number such that no $(d - k')$-face of Q hits the simplex when shot from the query point. It is obvious that, for all $t < k'$, some $(d - t)$-face of Q hits the simplex, and that, for all $t \ge k'$, no $(d - t)$-face of Q hits the simplex. For the sake of simplicity, we hereafter focus on solving the face-shooting problem when $k' = k$, thus ignoring the fact a simplex can be missed. Because the obtained running time will be of the order of $\tilde{O}(n^{k'-1})$, the running time will be dominated by this case.

In order to reduce face-shooting to range counting queries, we need an analogue of Lemma 5 for convex combinations. Let A be a set of k points $\{a_1, a_2, \ldots, a_k\}$, and let Δ be a $(d - k + 1)$-simplex with vertices in $B = \{b_1, b_2, \ldots, b_{d-k+2}\}$. We suppose that these points are in general position. We define the hyperplanes $H_i = \text{aff}(A \cup B \setminus \{b_i, a_k\})$, for $i \in [d-k+2]$, and $G_j = \text{aff}(A \cup B \setminus \{a_j, a_k\})$, for $j \in [k-1]$. We let H_i^+ be the halfspace supported by H_i that contains b_i, and G_j^- the halfspace supported by G_j that does not contain a_j.

▶ **Lemma 12.**

$$\text{conv}(A) \cap \Delta \ne \emptyset \iff a_k \in \left(\left(\bigcap_{i=1}^{d-k+2} H_i^+ \right) \cap \left(\bigcap_{j=1}^{k-1} G_j^- \right) \right).$$

Proof. ($\Leftarrow$) Suppose that $a_k \in (\bigcap_i H_i^+) \cap (\bigcap_j G_j^-)$. We have that $\text{conv}(A) \cap \Delta \ne \emptyset$ if and only if both $\text{aff}(A) \cap \Delta \ne \emptyset$ and $\text{conv}(A) \cap \text{aff}(\Delta) \ne \emptyset$ hold. From Lemma 5, we already have $\text{aff}(A) \cap \Delta \ne \emptyset$. It therefore remains to show that $\text{conv}(A) \cap \text{aff}(\Delta) \ne \emptyset$.

We first prove that $(\bigcap_j G_j) \cap \text{conv}(A) \ne \emptyset$. We proceed by induction on k. It can easily be shown to hold for $k = 2$. Let us suppose it holds for $k - 1$, and prove it for k. The hyperplane G_{k-1} separates a_{k-1} from a_k. Consider the point a'_{k-1} of the segment between a_{k-1} and a_k that lies on G_{k-1}. Let $A' = \{a_1, a_2, \ldots, a_{k-2}, a'_{k-1}\}$. Consider the intersection G'_j of all hyperplanes G_j for $j \in [k-2]$ with the subspace $\text{aff}(A')$. In the subspace $\text{aff}(A')$, The hyperplanes G'_j for $j \in [k-2]$ all separate a_j from a'_{k-1}. Hence we can apply induction on A' and the hyperplanes G' in dimension $k - 2$, and we have that $(\bigcap_{j \in [k-2]} G'_j) \cap \text{conv}(A') \ne \emptyset$. Now because $a'_{k-1} \in \text{conv}(\{a_{k-1}, a_k\})$, we also have that $(\bigcap_{j \in [k-1]} G_j) \cap \text{conv}(A) \ne \emptyset$.
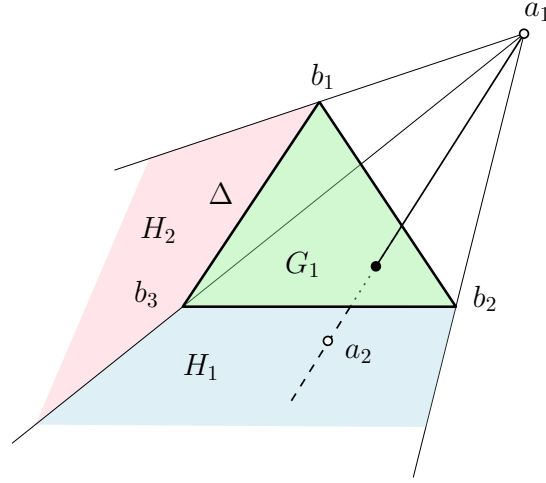
Figure 5 Illustration of Lemma 12 in the case $d = 3$ and $k = 2$. The segment $a_1 a_2$ intersects Δ if and only if a_2 is located in the colored region below Δ.

Now we also observe that $\bigcap_j G_j = \text{aff}(\Delta)$. The fact that $\text{aff}(\Delta) \subseteq \bigcap_j G_j$ is immediate since each G_j contains $\text{aff}(\Delta)$. To prove that $\bigcap_j G_j$ cannot contain more than $\text{aff}(\Delta)$ it suffices to show that those flats are of the same dimensions. Since the set $A \cup B$ is in general position, a_j (and a_k) cannot lie on G_j. Then we claim that the G_j are in general position. Indeed if they are not, then there must be some $1 \leq k' \leq k - 1$ where $\cap_{j \leq k'-1} G_j = \cap_{j \leq k'} G_j$. However, this is not possible since $a_{k'} \in \cap_{j \leq k'-1} G_j$ but $a_{k'} \notin \cap_{j \leq k'} G_j$. The dimension of $\bigcap_j G_j$ is thus $d - k + 1$, the same as the dimension of $\text{aff}(\Delta)$.

Therefore, $\text{conv}(A) \cap \text{aff}(\Delta) \neq \emptyset$, as needed.

($\Rightarrow$) Suppose that $a_k \notin (\bigcap_i H_i^+) \cap (\bigcap_j G_j^-)$. Then one of the halfspace does not contain a_k. It can be of the form H_i^+ or G_j^-. In both cases, all points of A are either contained in the hyperplane H_i or G_j, or lie in H_i^- or G_j^-. Hence the hyperplane H_i or G_j separates the interiors of the convex hulls. From the general position assumption, it also separates the convex hulls. ◀

The Lemma is illustrated on Figures 5 and 6 in the cases $d = 3$, $k = 2$, and $d = k = 3$.

Proof of Theorem 11. The algorithm follows the same steps as the algorithm described in the proof of Theorem 4, except that the ranges used in the orthogonal range counting data structure are different, and involve a higher-dimensional space.

We reduce the problem to that of counting the number of k-subsets A of S whose convex hull intersects a given $(d - k + 1)$-simplex Δ. We already argued that when fixing the first $k - 2$ points $a_1, a_2, \ldots, a_{k-2}$, the hyperplanes H_i induce a circular order on the points of S. Similarly, when the points $a_1, a_2, \ldots, a_{k-2}$ are fixed, the hyperplanes G_j all contain the $(d-2)$-flat $\text{aff}(A \cup B \setminus \{a_j, a_{k-1}, a_k\})$, hence also induce a circular order on the points of S. Thus for each $(k - 2)$-subset of S, we can assign $(d - k + 2) + (k - 1) = d + 1$ coordinates to each point of S, one for each family of hyperplanes. We then build an orthogonal range query data structure using these coordinates. For each point a_{k-1}, we query this data structure and count the number of points a_k such that $a_k \in (\bigcap_i H_i^+) \cap (\bigcap_j G_j^-)$. From Lemma 12, we can deduce the number of subsets A whose convex hull intersects Δ.

We can decrease by one the dimensionality of the ranges by realizing that the supporting hyperplane of G_{k-1}^- is unique as it does not depend on a_{k-1}, only the orientation of G_{k-1}^- does. To only output points a_k such that $a_k \in G_{k-1}^-$ we construct two data structures: one
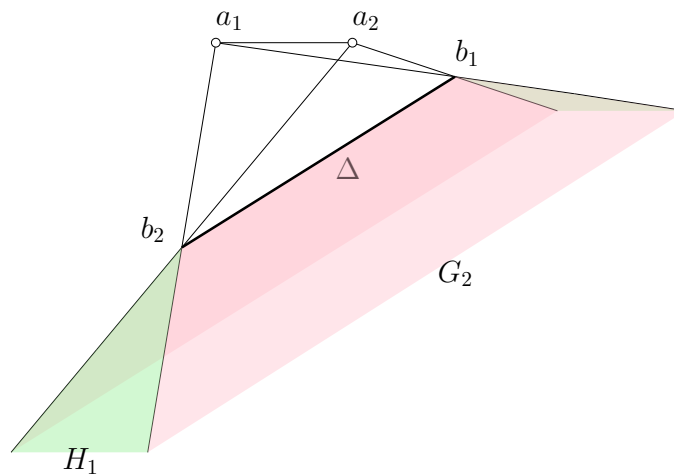
Figure 6 Illustration of Lemma 12 in the case $k = d = 3$. The triangle $a_1 a_2 a_3$ intersects Δ if and only if a_3 is located in the colored region.

with the points above G_{k-1} and one with the points below G_{k-1}. We query the relevant data structure depending whether a_{k-1} is above or below G_{k-1}. This spares a logarithmic factor and yields an overall running time of $O(n^{k-1} \log^{d-1} n)$ for the counting problem. Multiplying by the $O(\log n)$ rounds of binary search yields the claimed result. ◀

References

1   Pankaj K. Agarwal, Natan Rubin, and Micha Sharir. Approximate nearest neighbor search amid higher-dimensional flats. In *25th Annual European Symposium on Algorithms, ESA 2017, September 4-6, 2017, Vienna, Austria*, pages 4:1–4:13, 2017. `doi:10.4230/LIPIcs.ESA.2017.4`.

2   Michal Aharon, Michael Elad, and Alfred Bruckstein. K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Transactions on Signal Processing*, 54(11):4311–4322, November 2006. `doi:10.1109/TSP.2006.881199`.

3   Nir Ailon and Bernard Chazelle. Lower bounds for linear degeneracy testing. *J. ACM*, 52(2):157–171, 2005. `doi:10.1145/1059513.1059515`.

4   Rahul Arya, Sunil Arya, Guilherme D. da Fonseca, and David M. Mount. Optimal bound on the combinatorial complexity of approximating polytopes. In *SODA*, pages 786–805. SIAM, 2020.

5   Sunil Arya, Guilherme D. da Fonseca, and David M. Mount. On the combinatorial complexity of approximating polytopes, April 2016. `arXiv:1604.01175v4`.

6   Sunil Arya, Guilherme D. da Fonseca, and David M. Mount. Approximate convex intersection detection with applications to width and minkowski sums, July 2018. `arXiv:1807.00484v1`.

7   Sunil Arya, Guilherme Dias da Fonseca, and David M. Mount. Near-optimal epsilon-kernel construction and related problems. In *Symposium on Computational Geometry*, volume 77 of *LIPIcs*, pages 10:1–10:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017.

8   Sunil Arya, David M. Mount, Nathan S. Netanyahu, Ruth Silverman, and Angela Y. Wu. An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *J. ACM*, 45(6):891–923, 1998. `doi:10.1145/293347.293348`.

9   Luis Barba, Jean Cardinal, John Iacono, Stefan Langerman, Aurélien Ooms, and Noam Solomon. Subquadratic algorithms for algebraic 3SUM. *Discrete & Computational Geometry*, 61(4):698–734, 2019. `doi:10.1007/s00454-018-0040-y`.

10   Ronen Basri, Tal Hassner, and Lihi Zelnik-Manor. Approximate nearest subspace search. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(2):266–278, 2011. `doi:10.1109/TPAMI.2010.110`.

11   Dimitris Bertsimas, Angela King, and Rahul Mazumder. Best subset selection via a modern optimization lens. *Ann. Statist.*, 44(2):813–852, April 2016. `doi:10.1214/15-AOS1388`.

12   Dimitris Bertsimas, Jean Pauphilet, and Bart Van Parys. Sparse Regression: Scalable algorithms and empirical performance. *arXiv e-prints*, February 2019. `arXiv:1902.06547`.

13   Emmanuel J. Candès, Justin K. Romberg, and Terence Tao. Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information. *IEEE Trans. Information Theory*, 52(2):489–509, 2006. `doi:10.1109/TIT.2005.862083`.

14   Bernard Chazelle. A functional approach to data structures and its use in multidimensional searching. *SIAM J. Comput.*, 17(3):427–462, 1988. `doi:10.1137/0217026`.

15   Bernard Chazelle. Cutting hyperplanes for divide-and-conquer. *Discrete & Computational Geometry*, 9:145–158, 1993. `doi:10.1007/BF02189314`.

16   Bernard Chazelle and Joel Friedman. A deterministic view of random sampling and its use in geometry. *Combinatorica*, 10(3):229–249, 1990. `doi:10.1007/BF02122778`.

17   G. Davis, S. Mallat, and M. Avellaneda. Adaptive greedy approximations. *Constructive Approximation*, 13(1):57–98, March 1997. `doi:10.1007/BF02678430`.

18   David L. Donoho. Compressed sensing. *IEEE Trans. Information Theory*, 52(4):1289–1306, 2006. `doi:10.1109/TIT.2006.871582`.

19   Richard M. Dudley. Metric entropy of some classes of sets with differentiable boundaries. *Journal of Approximation Theory*, 10(3):227–236, 1974. `doi:10.1016/0021-9045(74)90120-8`.

20   Herbert Edelsbrunner, Raimund Seidel, and Micha Sharir. On the zone theorem for hyperplane arrangements. *SIAM J. Comput.*, 22(2):418–429, 1993. `doi:10.1137/0222031`.

21   Jeff Erickson and Raimund Seidel. Better lower bounds on detecting affine and spherical degeneracies. *Discrete & Computational Geometry*, 13:41–57, 1995. `doi:10.1007/BF02574027`.

22   Dean P. Foster, Satyen Kale, and Howard J. Karloff. Online sparse linear regression. In *Proceedings of the 29th Conference on Learning Theory, COLT 2016, New York, USA, June 23-26, 2016*, pages 960–970, 2016. URL: `http://proceedings.mlr.press/v49/foster16.html`.

23   Dean P. Foster, Howard J. Karloff, and Justin Thaler. Variable selection is hard. In *Proceedings of The 28th Conference on Learning Theory, COLT 2015, Paris, France, July 3-6, 2015*, pages 696–709, 2015. URL: `http://proceedings.mlr.press/v40/Foster15.html`.

24   Anka Gajentaan and Mark H. Overmars. On a class of $o(n^2)$ problems in computational geometry. *Comput. Geom.*, 5:165–185, 1995. `doi:10.1016/0925-7721(95)00022-2`.

25   Sariel Har-Peled, Piotr Indyk, and Sepideh Mahabadi. Approximate sparse linear regression. In *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9-13, 2018, Prague, Czech Republic*, pages 77:1–77:14, 2018. `doi:10.4230/LIPIcs.ICALP.2018.77`.

26   Trevor Hastie, Robert Tibshirani, and Jerome H. Friedman. *The elements of statistical learning: data mining, inference, and prediction, 2nd Edition*. Springer series in statistics. Springer, 2009. URL: `http://www.worldcat.org/oclc/300478243`.

27   Avner Magen. Dimensionality reductions in ℓ_2 that preserve volumes and distance to affine spaces. *Discrete & Computational Geometry*, 38(1):139–153, 2007. `doi:10.1007/s00454-007-1329-4`.

28   Sepideh Mahabadi. Approximate nearest line search in high dimensions. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015*, pages 337–354, 2015. `doi:10.1137/1.9781611973730.25`.

29   Stéphane Mallat and Zhifeng Zhang. Matching pursuits with time-frequency dictionaries. *IEEE Trans. Signal Processing*, 41(12):3397–3415, 1993. `doi:10.1109/78.258082`.

30   Alan Miller. *Subset Selection in Regression*. Chapman and Hall/CRC, 2002.

31   Balas K. Natarajan. Sparse approximate solutions to linear systems. *SIAM J. Comput.*, 24(2):227–234, 1995. `doi:10.1137/S0097539792240406`.

32 Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1):267–288, 1996. URL: `http://www.jstor.org/stable/2346178`.

33 Csaba D. Tóth, Joseph O'Rourke, and Jacob E. Goodman, editors. *Handbook of Discrete and Computational Geometry*. Chapman and Hall/CRC, 3rd edition, 2017. `doi:10.1201/9781315119601`.

34 Dan E. Willard. New data structures for orthogonal range queries. *SIAM J. Comput.*, 14(1):232–253, 1985.