Proceedings of Machine Learning Research vol 195:1–41, 2023 36th Annual Conference on Learning Theory

Quadratic Memory is Necessary for Optimal Query Complexity in Convex Optimization: Center-of-Mass is Pareto-Optimal

Moïse Blanchard MOISEB@MIT.EDU
Junhui Zhang JUNHUIZ@MIT.EDU
Patrick Jaillet JAILLET@MIT.EDU
Massachusetts Institute of Technology

Editors: Gergely Neu and Lorenzo Rosasco

Abstract

We give query complexity lower bounds for convex optimization and the related feasibility problem. We show that quadratic memory is necessary to achieve the optimal oracle complexity for first-order convex optimization with deterministic algorithms. In particular, this shows that center-of-mass cutting-planes algorithms in dimension d which use $\tilde{\mathcal{O}}(d^2)$ memory and $\tilde{\mathcal{O}}(d)$ queries are Pareto-optimal for both convex optimization and the feasibility problem, up to logarithmic factors. Precisely, building upon techniques introduced in [23], we prove that to minimize 1-Lipschitz convex functions over the unit ball to $1/d^4$ accuracy, any deterministic first-order algorithms using at most $d^{2-\delta}$ bits of memory must make $\tilde{\Omega}(d^{1+\delta/3})$ queries, for any $\delta \in [0, 1]$. For the feasibility problem, in which an algorithm only has access to a separation oracle, we show a stronger trade-off: for at most $d^{2-\delta}$ memory, the number of queries required is $\tilde{\Omega}(d^{1+\delta})$. This resolves a COLT 2019 open problem of Woodworth and Srebro.

Keywords: Convex optimization, feasibility problem, first-order methods, cutting-planes, center-of-mass, memory lower bounds, query complexity

1. Introduction

We consider the canonical problem of first-order convex optimization in which one aims to minimize a convex function $f : \mathbb{R}^d \to \mathbb{R}$ with access to an oracle that for any query $\boldsymbol{x}$ returns $(f(\boldsymbol{x}), \nabla f(\boldsymbol{x}))$ the value of the function and a subgradient of f at $\boldsymbol{x}$. Arguably, this is one of the most fundamental problems in optimization, mathematical programming and machine learning.

A classical question is how many oracle queries are required to find an ϵ-approximate minimizer for any 1-Lipschitz convex functions $f : \mathbb{R}^d \to \mathbb{R}$ over the unit ball. We denote by $B_d(\boldsymbol{x}, r) = \{\boldsymbol{x}' \in \mathbb{R}^d : \|\boldsymbol{x} - \boldsymbol{x}'\|_2 \leq r\}$ the ball centered in $\boldsymbol{x}$ of radius r. There exist methods that given first-order oracle access only need $\mathcal{O}(d \log 1/\epsilon)$ queries and this query complexity is worst-case optimal [29] when $\epsilon \ll 1/\sqrt{d}$. Known methods achieving the optimal $\mathcal{O}(d \log 1/\epsilon)$ query complexity fall in the broad class of cutting plane methods, that build upon the well-known ellipsoid method [47; 38] which uses $\mathcal{O}(d^2 \log 1/\epsilon)$ queries. These include the inscribed ellipsoid [41; 31], volumetric center or Vaidya's method [3; 42], approximate center-of-mass via sampling techniques [20; 6] and recent improvements [19; 17]. Unfortunately, all these methods suffer from at least $\Omega(d^3 \log 1/\epsilon)$ time complexity and further require storing all subgradients, or at least an ellipsoid in $\mathbb{R}^d$, therefore at least $\Omega(d^2 \log 1/\epsilon)$ bits of memory. These limitations are prohibitive for large-scale optimization, hence cutting plane methods are viewed as rather impractical and less frequently used for high-dimensional applications. On the other hand, the simplest, perhaps most commonly used and practical gradient descent requires $\mathcal{O}(1/\epsilon^2)$ queries, which is not optimal for $\epsilon \ll 1/\sqrt{d}$, but only needs $\mathcal{O}(d)$ time per query and $\mathcal{O}(d \log 1/\epsilon)$ memory.

A natural question is whether one can preserve the optimal query lower bounds from cutting-planes methods with simpler methods, for instance, inspired by gradient descent techniques. Such hope is largely motivated by the fact that in many different theoretical settings, cutting plane methods have achieved state-of-the-art runtimes including semidefinite programming [1; 19], submodular optimization [24; 14; 19; 16] or equilibrium computation [34; 15]. Towards this goal, [43] first posed this question in terms of query complexity / memory trade-off: given a certain number of bits of memory, which query complexity is achievable? While cutting planes methods require $\Omega(d^2 \log 1/\epsilon)$ memory, gradient descent only requires storing one vector and as a result, uses $\mathcal{O}(d \log 1/\epsilon)$ memory, which is information-theoretically optimal [43][1]. Understanding this trade-off could pave the way for the design of more efficient methods in convex optimization.

The first result in this direction was provided in [23], where they showed that it is impossible to be both optimal in query complexity and in memory. Specifically, any potentially randomized algorithm that uses at most $d^{1.25-\delta}$ memory must make at least $\tilde{\Omega}(d^{1+4/3\delta})$ queries. Thus, a super-linear amount of memory $d^{1.25}$ is required to achieve the optimal rate of convergence (that is achieved by algorithms using more than quadratic memory). However, this leaves open the fundamental question of whether one can improve over the memory of cutting-plane methods while keeping optimal query complexity.

Question (COLT 2019 [43]). Is it possible for a first-order algorithm that uses at most $\mathcal{O}(d^{2-\delta})$ bits of memory to achieve query complexity $\tilde{\mathcal{O}}(d \,\mathrm{polylog}\, 1/\epsilon)$ when $d = \Omega(\log^c 1/\epsilon)$ but $d = o(1/\epsilon^c)$ for all $c > 0$?

In this paper, building upon the techniques introduced in [23], we provide a negative answer to this question: quadratic memory is necessary to achieve optimal query complexity with deterministic algorithms. As a result, cutting plane methods including the standard center-of-mass algorithm are Pareto-optimal up to logarithmic factors within the query complexity / memory trade-off. Our main result for convex optimization is the following.

Theorem 1 *For $\epsilon = 1/d^4$ and any $\delta \in [0,1]$, a deterministic first-order algorithm guaranteed to minimize 1-Lipschitz convex functions over the unit ball with ϵ accuracy uses at least $d^{2-\delta}$ bits or makes $\tilde{\Omega}(d^{1+\delta/3})$ queries.*

A key component of cutting plane methods is that they merely rely on the subgradient information at each query to restrict the search space. As a result, these can be used to solve the larger class of feasibility problems that are essential in mathematical programming and optimization. In a feasibility problem, one aims to find an ϵ-approximation of an unknown vector $\boldsymbol{x}^\star$, and has access to a separation oracle. For any query $\boldsymbol{x}$, the separation oracle either returns a separating hyperplane $\boldsymbol{g}$ from $\boldsymbol{x}$ to $B_d(\boldsymbol{x}^\star, \epsilon)$—such that $\langle \boldsymbol{g}, \boldsymbol{x} - \boldsymbol{z} \rangle > 0$ for any $\boldsymbol{z} \in B_d(\boldsymbol{x}^\star, \epsilon)$—or signals that $\|\boldsymbol{x} - \boldsymbol{x}^\star\| \leq \epsilon$. This class of problems is broader than convex optimization since the negative subgradient always provides a separating hyperplane from a suboptimal query to the optimal set. Hence, feasibility and convex minimization problem are closely related and it is often the case that obtaining query lower bounds for the feasibility problem simplifies the analysis while still providing key insights for the

1. $\Omega(d \log 1/\epsilon)$ bits of memory are already required just to represent the answer to the optimization problem.

more restrictive convex optimization problem [29; 32]. Thus, a similar fundamental question is to understand the query complexity / memory trade-off for the feasibility problem. As noted above, any lower bound for convex optimization yields the same lower bound for the feasibility problem. Here, we can significantly improve over the previous trade-off.

Theorem 2 *For $\epsilon = 1/(48d^2\sqrt{d})$ and any $\delta \in [0,1]$, a deterministic algorithm guaranteed to solve the feasibility problem over the unit ball with ϵ accuracy uses at least $d^{2-\delta}$ bits of memory or makes at least $\tilde{\Omega}(d^{1+\delta})$ queries.*

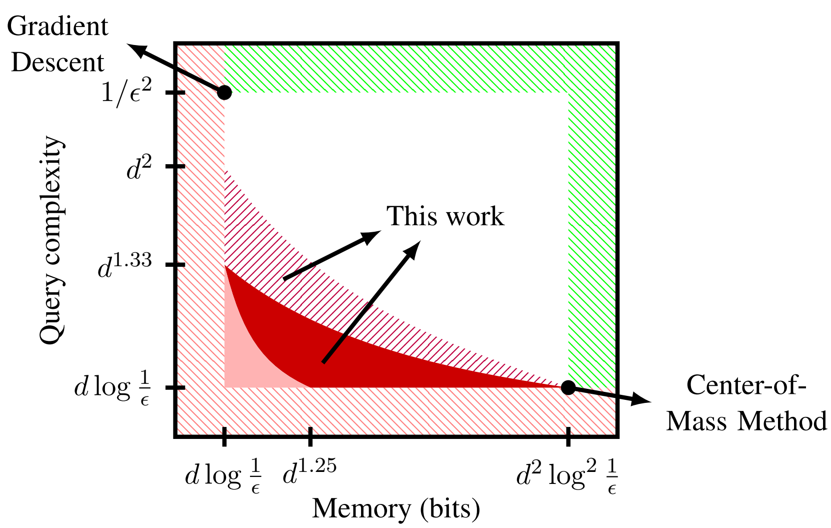

Figure 1: Trade-offs between memory and oracle complexity for minimizing 1-Lipschitz convex functions over the unit ball (adapted from [43; 23]). The dashed pink (resp. green) region corresponds to historical information-theoretic lower bounds (resp. upper bounds) on the memory and query-complexity. The solid pink region corresponds to the recent lower bound trade-off from [23], which holds for randomized algorithms. In our work, we show that the solid red region is not achievable for any deterministic algorithm. For the feasibility problem, we also show that the dashed red region is not achievable either for any deterministic algorithm.

1.1. Literature review

Recently, there has been a series of studies exploring the trade-offs between sample complexity and memory constraints for learning problems, such as linear regression [39; 37], principal component analysis (PCA) [25], learning under the statistical query model [40] and other general learning problems [7; 8; 26; 27; 5; 13; 18; 10; 11].

For parity problems that meet certain spectral (mixing) requirements, [36] that an exponential number of random samples is needed if the memory is sub-quadratic. Subsequently, similar trade-offs have then been obtained for various other discrete learning problems [35; 26; 18; 27; 5; 13] (finite concept class). For continuous problems, [37] was the first work to show sample complexity

/ memory lower bounds in the case of linear regression, building upon a computation tree argument introduced in [35]. They show that for accuracy $\epsilon \leq 1/d^{\mathcal{O}(\log d)}$, sub-quadratic memory algorithms require $\mathcal{O}(d \log\log 1/\epsilon)$, instead of d samples with full quadratic memory.

It should also be pointed out that [11] studied linear prediction problems under the streaming model by analyzing the *Approximate Null-Vector Problem* (ANVP). Both ANVP and the *Orthogonal Vector Game* proposed in [23] (which we build upon in this work) aim at finding vectors that lie approximately in the null space of a stream of vectors, but under different settings. A major difference is that ANVP considers a streaming setting whereas in the Orthogonal Vector Game (and the game introduced in this work), the player has access to the complete input in the beginning, then fixes a memory-constrained message based on the input.

In contrast to learning with random samples, there is limited understanding of the memory-constrained optimization and feasibility problems. [30] demonstrated that, in the absence of memory constraints, finding an ϵ-approximate solution for Lipschitz convex functions requires $\Omega(d \log 1/\epsilon)$ queries, which can be achieved by the center-of-mass method using $O(d^2 \log^2 1/\epsilon)$ bits of memory. At the other extreme, gradient descent needs $\Omega(1/\epsilon^2)$ queries but only $O(d \log 1/\epsilon)$ bits of memory, the minimum memory needed to represent a solution. These two extreme cases are represented by dashed pink "impossible region" and dashed green "achievable region" in Figure 1. Since then, [23] showed that there is a trade-off between memory and query for convex optimization: it is impossible to be both optimal in query complexity and memory. Their lower bound is represented by the solid pink "impossible region" in Figure 1. In this paper, we significantly improve these results to match the quadratic upper bound of cutting plane methods. Additionally, there has been recent progress in the study of query complexity for randomized algorithms [45; 44], and communication complexity for convex optimization in the distributed setting [2; 46].

On the algorithmic side, the afore-mentioned methods that achieve $O(poly(d))$ query complexity [47; 38; 41; 31; 3; 42; 20; 6; 19; 17] all require at least $\Omega(d^2 \log 1/\epsilon)$ bits of memory. There is also significant literature on memory-efficient optimization algorithms, such as the Limited-memory-Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm [33; 22]. However, the convergence behavior for even the original BFGS on non-smooth convex objectives is still a challenging, open question [21].

Comparison with [23] Our proof techniques build upon those introduced in [23]. We follow the proof strategy that they introduced to derive lower bounds for the memory/query complexity. Below, we delineate which ideas and techniques are borrowed from [23] and which are the novel elements that we introduce. Details on these proof elements are given in Section 2.1.

First, [23] define a class of difficult functions for convex optimization of the following form

$$\max \left\{ \|\boldsymbol{A}\boldsymbol{x}\|_\infty - \eta_0, \eta_1 \left(\max_{i \leq N} \boldsymbol{v}_i^\top \boldsymbol{x} - i\gamma \right) \right\}, \tag{1}$$

where $\boldsymbol{A} \sim \mathcal{U}(\{\pm 1\}^{d/2 \times d})$ is a matrix with ± 1 entries sampled uniformly, and $v_i \sim \mathcal{U}(d^{-1/2}\{\pm 1\}^d)$ are sampled independently, uniformly within the rescaled hypercube. To give intuition on this class, the term $\|\boldsymbol{A}\boldsymbol{x}\|_\infty - \eta_0$ acts as barrier : in order to observe subgradients from the other term, one needs to use queries $\boldsymbol{x}$ that are approximately within the nullspace of $\boldsymbol{A}$. The second term $\max_{i \leq N} \boldsymbol{v}_i^\top \boldsymbol{x} - i\gamma$ is the "Nemirovski" function, which was used in previous works [28; 4; 9] to obtain lower bounds in parallel convex optimization. At a high level, the limitation in the lower bounds from [23] comes from the fact that one is limited in the number N of vectors $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_N$

that can be used in the Nemirovski function. To resolve this issue, we introduce adaptivity within the choice of a modified Nemirovski function. At a high level, we choose the vectors $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_N$ depending on the queries of the algorithm which allows to fit in more terms. In turn, this allows to improve the lower bounds.

As a second step, [23] relate the optimization problem on the defined class of functions to an Orthogonal Vector Game. In this game, the goal is to find vectors that are approximately orthogonal to a matrix $\boldsymbol{A}$ with access to row queries of $\boldsymbol{A}$. The argument is as follows: because of the barrier term $\|\boldsymbol{A}\boldsymbol{x}\|_\infty - \eta_0$, optimizing the Nemirovski function requires exploring independent directions of the nullspace of $\boldsymbol{A}$, which is performed at *informative queries*. With our new class of functions, we can adapt this logic. However, the adaptivity in the vectors $\boldsymbol{v}_i$ provides information to the learner on $\boldsymbol{A}$ in addition to the queried rows of $\boldsymbol{A}$. We therefore need to modify the game by introducing an Orthogonal Vector Game with Hints, where hints encapsulate this extra information.

For the last step, [23] give an information-theoretic argument to provide a query complexity lower bound on the defined Orthogonal Vector Game. We show that a similar argument holds for our modified game. The main added difficulty resides in bounding the information leakage from the hints, and we show that these provide no more information than the memory itself.

As a last remark, the lower bounds provided in [23] hold for randomized algorithms, while the adaptivity of our procedure only applies to deterministic algorithms.

1.2. Outline of paper

In Section 2, we formally define our setup and give a brief overview of our proof techniques. A sketch of the proof of Theorem 1 for convex optimization is given in Section 3, with full details given in Appendix A. In Appendix B we consider the feasibility problem and prove Theorem 2.

2. Formal setup and overview of techniques

Standard results in oracle complexity give the minimal number of queries for algorithms to solve a given problem. However, this does not account for possible restrictions on the memory available to the algorithm. In this paper, we are interested in the trade-off between memory and query complexity for both convex optimization and the feasibility problem. Our results apply to a large class of *memory-constrained* algorithms. We give below a general definition of the memory constraint for algorithms with access to an oracle $\mathcal{O} : \mathcal{S} \to \mathcal{R}$ taking as input a query $q \in \mathcal{S}$ and returning a response $\mathcal{O}(q) \in \mathcal{R}$.

Definition 3 (M-bit memory-constrained deterministic algorithm) *Let $\mathcal{O} : \mathcal{S} \to \mathcal{R}$ be an oracle. An M-bit memory-constrained deterministic algorithm is specified by a query function $\psi_{query} : \{0,1\}^M \to \mathcal{S}$ and an update function $\psi_{update} : \{0,1\}^M \times \mathcal{S} \times \mathcal{R} \to \{0,1\}^M$. The algorithm starts with the memory state $\mathsf{Memory}_0 = 0^M$ and iteratively makes queries to the oracle. At iteration t, it makes the query $q_t = \psi_{query}(\mathsf{Memory}_{t-1})$ to the oracle, receives the response $r_t = \mathcal{O}(q_t)$ then updates its memory $\mathsf{Memory}_t = \psi_{update}(\mathsf{Memory}_{t-1}, q_t, r_t)$.*

The algorithm can stop making queries at any iteration and the last query is its final output. Notice that the memory constraint applies only between each query but not for internal computations: the computation of the update ψ_{update} and the query ψ_{query} can potentially use unlimited memory. This is a rather weak memory constraint on the algorithm; a fortiori, our negative results also apply to stronger notions of memory-constrained algorithms. In Definition 3, we ask the query and update functions to be time-invariant, which in our context is without loss of generality: any M-bit algorithm using T queries with time-dependent query and update functions [43; 23] can be turned into an $(M + \lceil \log T \rceil)$-bit time-invariant algorithm by storing the iteration number t as part of the memory. The query lower bounds we provide are at most $T \leq poly(d)$. Hence, an additional $\log T = O(\log d)$ bits to the memory size M does not affect our main results, Theorems 1 and 2.

In this paper, we use the above described framework to study the interplay between query complexity and memory for two fundamental problems in optimization and machine learning.

Convex optimization. We first consider convex optimization in which one aims to minimize a 1-Lipschitz convex function $f : B_d(\mathbf{0}, 1) \to \mathbb{R}$ on the unit ball. The goal is to output a point $\tilde{\boldsymbol{x}} \in B_d(\mathbf{0}, 1)$ such that $f(\tilde{\boldsymbol{x}}) \leq \min_{\boldsymbol{x} \in B_d(\mathbf{0},1)} f(\boldsymbol{x}) + \epsilon$, referred to as ϵ-approximate solutions. The optimization algorithm has access to a first order oracle $\mathcal{O}_{CO} : B_d(\mathbf{0}, 1) \to \mathbb{R} \times \mathbb{R}^d$, which for any query $\boldsymbol{x}$ returns the couple $(f(\boldsymbol{x}), \partial f(\boldsymbol{x}))$ where $\partial f(\boldsymbol{x})$ is a subgradient of f at the query point $\boldsymbol{x}$.

Feasibility problem. Second, we consider the trade-off between memory and query complexity for the feasibility problem, where the goal is to find an element $\tilde{\boldsymbol{x}} \in Q$ for a convex set $Q \subset B_d(\mathbf{0}, 1)$. Instead of a first-order oracle, the algorithm has access to a separation oracle $\mathcal{O}_F : B_d(\mathbf{0}, 1) \to \{\mathsf{Success}\} \cup \mathbb{R}^d$. For any query $\boldsymbol{x} \in B_d(\mathbf{0}, 1)$, the separation oracle either returns $\mathsf{Success}$ reporting that $\boldsymbol{x} \in Q$, or provides a separating vector $\boldsymbol{g} \in \mathbb{R}^d$, i.e., such that for all $\boldsymbol{x}' \in Q$,

$$\langle \boldsymbol{g}, \boldsymbol{x} - \boldsymbol{x}' \rangle > 0.$$

We say that an algorithm solves the feasibility problem with accuracy $\epsilon > 0$ if it can solve any feasibility problem for which the successful set contains a ball of radius ϵ, i.e., such that there exists $\boldsymbol{x}^\star \in B_d(\mathbf{0}, 1)$ satisfying $B_d(\boldsymbol{x}^\star, \epsilon) \subset Q$.

The feasibility problem is at least as hard as convex optimization in the following sense: an algorithm that solves the feasibility problem with accuracy ϵ/L can be used to solve L-Lipschitz convex optimization problems by feeding the subgradients from first-order queries to the algorithm as separating hyperplanes. Alternatively, from any 1-Lipschitz function f one can derive a feasibility problem, where the feasibility set is $Q = \{\boldsymbol{x} \in B_d(\mathbf{0}, 1), f(\boldsymbol{x}) \leq f^\star + \epsilon\}$ and the separating oracle at $\boldsymbol{x} \notin Q$ is a subgradient $\partial f(\boldsymbol{x})$ at $\boldsymbol{x}$.

Remark 4 *Although we consider the case of constrained optimization, one can efficiently reduce the problem of approximate Lipschitz convex optimization over the unit ball to unconstrained approximate Lipschitz convex optimization [23]. Hence, our results also apply to the latter setting at the expense of losing $poly(d)$ factors in the necessary accuracy ϵ in Theorem 1. For the feasibility problem, there is no loss, Theorem 2 applies directly for the unconstrained feasibility setting.*

2.1. Overview of proof techniques and innovations

We prove the two main Theorems 1 and 2 with similar techniques, hence for conciseness, we only give here the main ideas used to derive lower bounds for convex optimization. Although our proof borrows techniques from [23], we introduce key innovations involving adaptivity to improve the lower bounds up to the maximum quadratic memory for deterministic algorithms—up to logarithmic factors. We recall, however, that the bounds in [23] hold for randomized algorithms as well. In the proofs, we aim to optimize the dependence of the parameters in d. Constants, however, are not necessarily optimized.

An adaptive optimization procedure. At the high level, we design an *optimization procedure* which for any algorithm constructs a hard family of convex functions adaptively on its queries. To be precise, the procedure constructs functions from the following family of convex functions with appropriately chosen parameters $\eta, \gamma_1, \gamma_2, p_{max}, l_p, \delta$:

$$F_{\boldsymbol{A},\boldsymbol{v}}(\boldsymbol{x}) = \max\left\{\|\boldsymbol{A}\boldsymbol{x}\|_\infty - \eta, \eta \boldsymbol{v}_0^\top \boldsymbol{x}, \eta\left(\max_{p\leq p_{max}, l\leq l_p} \boldsymbol{v}_{p,l}^\top \boldsymbol{x} - p\gamma_1 - l\gamma_2\right)\right\}. \tag{2}$$

We take $\boldsymbol{A} \sim \mathcal{U}(\{\pm 1\}^{n\times d})$ and $\boldsymbol{v}_0 \sim \mathcal{U}(\mathcal{D}_\delta)$ uniformly sampled in the beginning, where $\mathcal{D}_\delta \subset \mathcal{S}^{d-1}$ is a (finite) discretization of the sphere. The first term $\|\boldsymbol{A}\boldsymbol{x}\|_\infty - \eta$ acts as a barrier term: in order to observe subgradients from the other terms, one needs the query $\boldsymbol{x}$ to satisfy $\|\boldsymbol{A}\boldsymbol{x}\|_\infty \leq 2\eta$. These are called *informative queries* as introduced in [23]. Hence, informative queries must lie approximately in the orthogonal space to the lines of $\boldsymbol{A}$. The second term $\eta\boldsymbol{v}_0^\top \boldsymbol{x}$ ensures that queries with low objective (in particular with objective at most $-\eta\gamma_1/2$) have norm bounded away from 0. Thus, these queries, once renormalized, will still belong approximately to the nullspace of $\boldsymbol{A}$ denoted $Ker(\boldsymbol{A})$.

The adaptivity to the algorithm is captured in the third term, which is constructed along the optimization process. This construction proceeds by periods $p = 1, 2, \ldots, p_{max}$ designed so that during each period p, the algorithm is forced to visit a subspace of $Ker(\boldsymbol{A})$ of dimension k. To do so, we iteratively construct vectors $\boldsymbol{v}_{p,1}, \ldots \boldsymbol{v}_{p,l_p}$ as follows. Suppose that at the beginning of step t of period p, one has defined vectors $\boldsymbol{v}_{p,1}, \ldots, \boldsymbol{v}_{p,l}$.

- The procedure first evaluates the explored subspace of the algorithm during this period. In practice, the procedure keeps in memory *exploratory* queries $\boldsymbol{x}_{i_{p,1}}, \ldots, \boldsymbol{x}_{i_{p,r}}$ during period p up to time t. The exploratory subspace is then $Span(\boldsymbol{x}_{i_{p,1}}, \ldots, \boldsymbol{x}_{i_{p,r}})$.
- If a query with a sufficiently low objective is queried, we sample a new vector $\boldsymbol{v}_{p,l+1}$ which is approximately orthogonal to the exploratory subspace. The corresponding new term in the objective is $\boldsymbol{v}_{p,l+1}^\top \boldsymbol{x} - p\gamma_1 - (l+1)\gamma_2$.

Once this new term is added to the objective, the algorithm is constrained to make queries with an additional component along the direction $-\boldsymbol{v}_{p,l+1}$. Since this vector is approximately orthogonal to all previous queries, this forces the algorithm to query vectors linearly independent from all previous queries in period p. The period then ends once the dimension of the exploratory subspace reaches k, having defined l_p vectors $\boldsymbol{v}_{p,1}, \ldots, \boldsymbol{v}_{p,l_p}$. As discussed above, the exploratory subspace must increase dimension for any additional such vector. Thus, after $l_p \leq k$ vectors, period p ends.

The constructed family of convex functions in Eq (2) is similar to the family described in Eq (1) that were considered in [23]. However, by sampling the vectors $\boldsymbol{v}_{p,l}$ adaptively, the *optimization procedure* is able to fit in more terms, thereby providing a significant improvement in the lower bounds.

Benefits of adaptivity. We now expand on how the adaptive terms allow improving the lower bound of [23] to match the quadratic upper bound of cutting plane methods. The limitation in the functions of the form Eq (1) comes from the fact that the offset in the Nemirovski function is $\gamma = \Omega(\sqrt{k\log d/d})$. This offset is necessary to ensure that with high probability, 1: subgradients $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_N$ are discovered exactly in this order and 2: that any query which visits a new vector $\boldsymbol{v}_i$ must not lie in the subspace formed by the last k last informative vectors. Indeed, for the last claim, from high-dimensional concentration, for a random unit vector $\boldsymbol{v}$ and a k dimensional subspace E, $\|P_E(\boldsymbol{v})\| = \Theta(\sqrt{k \log d/d})$. This offset is not necessary for our procedure, since by construction, at each period, a k-dimensional subspace of $Ker(\boldsymbol{A})$ is forced to be explored. As a result, we can take $\gamma_1 = \Theta(\sqrt{\log d/d})$. This offset is still necessary to ensure that vectors $\boldsymbol{v}_{p,l}$ are discovered in their order of construction (lexicographic order on (p, l)) with high probability.

An Orthogonal Vector Game with Hints. The next step of the proof involves linking the optimization of the above-mentioned constructed functions with an Orthogonal Vector Game with Hints. Similarly to the game introduced by [23], the goal for the player is to find k linearly-independent vectors approximatively in $Ker(\boldsymbol{A})$. To do so, the player can access an M-bit message Message and make m queries, where $M = ckd$ for a small constant $c > 0$. In the game introduced by [23], the queries are lines of the matrix $\boldsymbol{A}$. They then show that to find k dimensions of $Ker(\boldsymbol{A})$, where $\boldsymbol{A}$ is taken uniformly at random $\boldsymbol{A} \sim \{\pm 1\}^{d/2\times d}$, (nearly) all the lines of $\boldsymbol{A}$ must be queried. The argument is information-theoretic: each new dimension of $Ker(\boldsymbol{A})$ must be (approximately) orthogonal to all lines of $\boldsymbol{A}$. Hence, this provides additional mutual information $O(k)$ for every line of $\boldsymbol{A}$, including the $d/2-m$ lines that were not observed through queries. This extra information on $\boldsymbol{A}$ can only be explained by the message, which has M bits. Hence, $M \geq O(k)(d/2-m)$. Setting the constant $c > 0$ appropriately, this shows that $m = \Omega(d)$.

In our case, the optimization procedure ensures that the algorithm needs to explore k dimensions of $Ker(\boldsymbol{A})$ in each period. However, each query yields a response from the optimization oracle that can either be a line of $\boldsymbol{A}$ (corresponding to the term $\|\boldsymbol{A}\boldsymbol{x}\|_\infty - \eta$ of Eq (2)) or $\boldsymbol{v}_0$ (term $\eta \boldsymbol{v}_0^\top \boldsymbol{x}$ of Eq (2)), or previously defined vectors $\boldsymbol{v}_{p,'l,'}$. Since the vectors $\boldsymbol{v}_{p',l'}$ have been constructed adaptively on the queries of the algorithm, which themselves may depend on lines of $\boldsymbol{A}$, during a period p, responses $\boldsymbol{v}_{p',l'}$ for $p' < p$ are a source of information leakage for $\boldsymbol{A}$ from previous periods. As a result, the query lower bound on the game introduced by [23] is not sufficient for our purposes. Instead, we introduce an Orthogonal Vector Game with Hints, where hints correspond exactly to these vectors $\boldsymbol{v}_{p',l'}$ from previous periods. Informally, the game corresponds to a simulation of one of the periods of the optimization procedure: for each query $\boldsymbol{x}$, the oracle returns the subgradient that would have been returned in the optimization procedure, up to minor details.

Bounding the information leakage. Once the link is settled, the goal is to prove lower bounds on the number of queries needed to solve the Orthogonal Vector Game with Hints. The main difficulty is to bound the information leakage from these hints. We recall that hints are of the form $\boldsymbol{v}_{p',l'}$, which have been constructed adaptively on the queries of the algorithm during period p'. In particular, these contain information on the lines of $\boldsymbol{A}$ queried during period $p' < p$, which may be complementary with those queried during period p. If this total information leakage through the hints yields a mutual information with $Ker(\boldsymbol{A})$ significantly higher than that of the M bits of Message, obtained lower bounds cannot possibly reflect any trade-off with memory constraints. It is therefore essential to obtain information leakage at most $\tilde{\mathcal{O}}(M) = \tilde{\mathcal{O}}(dk)$.

To solve this issue, we introduce a discretization $\mathcal{D}_\delta$ of the unit sphere where the vectors $\boldsymbol{v}_{p,l}$ take value. Next, we show that each individual vector $\boldsymbol{v}_{p',l'}$ from previous periods can only provide information $\tilde{\mathcal{O}}(k)$ on the matrix $\boldsymbol{A}$. To have an intuition on this, note that for any (at most) k vectors $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k$, the volume of the subset of the unit sphere S^{d-1} of vectors approximately orthogonal to $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k$, say $S(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k) = \{\boldsymbol{y} \in S^{d-1} : |\boldsymbol{y}^\top \boldsymbol{x}_i| \leq d^{-3}, i \leq k\}$ is $q_k = \Omega(1/d^{3k})$. Hence, since the vector $\boldsymbol{v}$ is roughly taken uniformly at random within $\mathcal{D}_\delta \cap S(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k)$, we can show that the mutual information of $\boldsymbol{v}$ with the initial vectors $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k$ is at most $\mathcal{O}(-\log q_k) =$

$\mathcal{O}(k\log d)$. As a result, even if $m = d$, the total information leakage through the vectors $\boldsymbol{v}_{p',l'}$ from previous periods, is at most $\mathcal{O}(kd\log d)$. The formal proof involves an anti-concentration bounds on the distance of a random unit vector to a linear subspace of dimension k, as well as a more involved discretization procedure than the one presented above. In summary, by introducing adaptive functions through the optimization procedure, we show that the same memory-sample trade-off holds for the Orthogonal Vector Game with Hints and the game without hints introduced in [23], up to logarithmic factors.

3. Memory-constrained convex optimization

To prove our results we need to use discretizations of the unit sphere S^{d-1}, which we construct by first partitioning S^{d-1} into $N(\delta) = (\mathcal{O}(1)/\delta)^d$ regions of equal area and diameter at most δ, i.e. $\mathcal{V}_\delta = \{V_i(\delta), i \in [N(\delta)]\}$ (the existence of which is guaranteed by Lemma 10). Here $\delta > 0$ is taken as parameter. Then we take one point as the representative of each region, i.e. $\mathcal{D}_\delta = \{\boldsymbol{b}_i(\delta), i \in [N(\delta)]\} \subset S^{d-1}$, where for all $i \in [N(\delta)]$, $\boldsymbol{b}_i(\delta) \in V_i(\delta)$. With these notations we define the discretization function ϕ_δ such that for any $\boldsymbol{x} \in S^{d-1}$, $\phi_\delta(\boldsymbol{x}) = \boldsymbol{b}_i(\delta)$ where $\boldsymbol{x} \in V_i(\delta)$.

3.1. Definition of the difficult class of optimization problems

In this section we present the class of functions that we use to prove our lower bounds. Throughout the paper, we pose $n = \lceil d/4 \rceil$. We first define some useful functions. For any $\boldsymbol{A} \in \mathbb{R}^{n\times d}$, we define $\boldsymbol{g}_{\boldsymbol{A}}$ as follows

$$\boldsymbol{g}_{\boldsymbol{A}}(\boldsymbol{x}) = \boldsymbol{a}_{i_{\min}}, \qquad i_{\min} = \min\{i \in [n], |\boldsymbol{a}_i^\top \boldsymbol{x}| = \|\boldsymbol{A}\boldsymbol{x}\|_\infty\}.$$

With this function we can define a subgradient function for $\boldsymbol{x} \mapsto \|\boldsymbol{A}\boldsymbol{x}\|_\infty$,

$$\tilde{\boldsymbol{g}}_{\boldsymbol{A}}(\boldsymbol{x}) = \epsilon \boldsymbol{g}_{\boldsymbol{A}}(\boldsymbol{x}), \qquad \epsilon = sign(\boldsymbol{g}_{\boldsymbol{A}}(\boldsymbol{x})^\top \boldsymbol{x}).$$

We are now ready to introduce the class of functions which we use for our lower bounds. These are of the following form.

$$F_{\boldsymbol{A},\boldsymbol{v}}(\boldsymbol{x}) = \max\left\{\|\boldsymbol{A}\boldsymbol{x}\|_\infty - \eta, \eta\boldsymbol{v}_0^\top \boldsymbol{x}, \eta\left(\max_{p\le p_{max}} \max_{l\le l_p} \boldsymbol{v}_{p,l}^\top \boldsymbol{x} - p\gamma_1 - l\gamma_2\right)\right\}.$$

Here, $\boldsymbol{A} \in \{\pm 1\}^{n\times d}$ is a matrix. Also, $\boldsymbol{v}_0$ and the terms $\boldsymbol{v}_{p,l}$ are vectors in $\mathbb{R}^d$. More precisely, these vectors will lie in the discretization $\mathcal{D}_\delta$ for $\delta = 1/d^3$. We postpone the definition of p_{max} and l_p for $p \le p_{max}$. Last, we use the following choice for the remaining parameters: $\eta = 2/d^3$, $\gamma_1 = 12\sqrt{\frac{\log d}{d}}$ and $\gamma_2 = \frac{\gamma_1}{4d}$. For convenience, we also define the functions

$$F_{\boldsymbol{A}}(\boldsymbol{x}) = \max\{\|\boldsymbol{A}\boldsymbol{x}\|_\infty - \eta, \eta\boldsymbol{v}_0^\top \boldsymbol{x}\}$$
$$F_{\boldsymbol{A},\boldsymbol{v},p,l}(\boldsymbol{x}) = \max\left\{\|\boldsymbol{A}\boldsymbol{x}\|_\infty - \eta, \eta\boldsymbol{v}_0^\top \boldsymbol{x}, \eta\left(\max_{(p',l')\le_{lex}(p,l), l'\le l_{p'}} \boldsymbol{v}_{p',l'}^\top \boldsymbol{x} - p'\gamma_1 - l'\gamma_2\right)\right\},$$

with the convention $F_{\boldsymbol{A},\boldsymbol{v},1,0} = F_{\boldsymbol{A}}$. The functions $F_{\boldsymbol{A},\boldsymbol{v},p,l}$ will encapsulate the current state of the function to be minimized: it will be updated adaptively on the queries of the algorithm. We also define a subgradient function for $F_{\boldsymbol{A},\boldsymbol{v},p,l}$ by first favoring lines of $\boldsymbol{A}$, then vectors from $\boldsymbol{v}$ in case of ties, as follows,

$$\partial F_{\boldsymbol{A},\boldsymbol{v},p,l}(\boldsymbol{x}) = \begin{cases} \tilde{\boldsymbol{g}}_{\boldsymbol{A}}(\boldsymbol{x}_t) & \text{if } F_{\boldsymbol{A},\boldsymbol{v},l,p}(\boldsymbol{x}) = \|\boldsymbol{A}\boldsymbol{x}\|_\infty - \eta, \\ \eta \boldsymbol{v}_0 & \text{otherwise and if } F_{\boldsymbol{A},\boldsymbol{v},l,p}(\boldsymbol{x}) = \eta \boldsymbol{v}_0^\top \boldsymbol{x}, \\ \eta \boldsymbol{v}_{p,l} & \text{otherwise and if } (p,l) = \arg\max_{(p',l') \leq_{lex} (p,l)} \boldsymbol{v}_{p',l'}^\top \boldsymbol{x} - p'\gamma_1 - l'\gamma_2. \end{cases}$$

In the last case, ties are broken by lexicographic order. We also pose $\partial F_{\boldsymbol{A},\boldsymbol{v}} = \partial F_{\boldsymbol{A},\boldsymbol{v},p_{max},l_{p_{max}}}$.

We consider a so-called *optimization procedure* described in Procedure 1, which will construct the sequence of vectors $\boldsymbol{v} = (\boldsymbol{v}_{p,l})$ adaptively on the responses of the considered algorithm. Throughout this section, we use a parameter $1 \leq k \leq d/3 - 1$ — which will be taken as $k = \tilde{\Theta}(M/d)$ where M is the memory of the algorithm — and let p_{max} be the largest number which satisfies the following constraint.

$$p_{max} \leq \min\{c_{d,1}(d-1)/k,\ c_{d,2}(d/k)^{1/3} - 1\}, \tag{3}$$

where $c_{d,1} = 1/(90^2 \log^2 d)$ and $c_{d,2} = 1/(81 \log^{2/3} d)$.

The optimization procedure is described in Procedure 1. First, we sample independently $\boldsymbol{A} \sim \mathcal{U}(\{\pm 1\}^{n \times d})$ and $\boldsymbol{v}_0 \sim \mathcal{U}(\mathcal{D}_\delta)$. The matrix $\boldsymbol{A}$ and vector $\boldsymbol{v}_0$ are then fixed for the rest of the learning procedure. Next, we describe the adaptive procedure to return subgradients. It proceeds by periods, until p_{max} periods are completed, unless the total number of iterations reaches d^2, in which case the construction procedure ends as well. First, we say that a query is informative if $F_{\boldsymbol{A}}(\boldsymbol{x}) \leq \eta$. The procedure proceeds by periods $p \in [p_{max}]$ and in each period constructs the vectors $\boldsymbol{v}_{p,1}, \ldots, \boldsymbol{v}_{p,k}$ iteratively. We are now ready to describe the procedure at time t when the new query $\boldsymbol{x}_t$ is queried. Let $p \geq 1$ be the index of the current period and $\boldsymbol{v}_{p,1}, \ldots, \boldsymbol{v}_{p,l}$ be the vectors of this period constructed so far: the first period is $p = 1$ and we allow $l = 0$ here. As will be seen in the construction, we always have $l \geq 1$ except at the very beginning for which we use the notation $F_{\boldsymbol{A},\boldsymbol{v},1,0} = F_{\boldsymbol{A}}$. Together with these vectors, the oracle keeps in memory indices $i_{p,1}, \ldots, i_{p,r}$ with $r \leq k$ of *exploratory* queries. The constructed vectors from previous periods are $\boldsymbol{v}_{p',l'}$ for $p' < p$ and $l' \leq l_{p'}$.

1. If $\boldsymbol{x}_t$ is not informative, i.e. $F_{\boldsymbol{A}}(\boldsymbol{x}_t) > \eta$, then procedure returns $(\|\boldsymbol{A}\boldsymbol{x}_t\|_\infty - \eta, \tilde{\boldsymbol{g}}_{\boldsymbol{A}}(\boldsymbol{x}_t))$.

2. Otherwise, we follow the next steps. If $r \leq k-1$ and

$$F_{\boldsymbol{A},\boldsymbol{v},p,l}(\boldsymbol{x}_t) \leq -\frac{\eta\gamma_1}{2} \qquad \text{and} \qquad \frac{\|P_{Span(\boldsymbol{x}_{i_{p,r'}}, r' \leq r)^\perp}(\boldsymbol{x}_t)\|}{\|\boldsymbol{x}_t\|} \geq \frac{\gamma_2}{4},$$

   we set $i_{p,r+1} = t$ and increment r. In this case, we say that $\boldsymbol{x}_t$ is *exploratory*. Next,

   (a) Recalling that $F_{\boldsymbol{A},\boldsymbol{v},p,l}$ is constructed so far, if $F_{\boldsymbol{A},\boldsymbol{v},p,l}(\boldsymbol{x}_t) \geq \eta(-p\gamma_1 - l\gamma_2 - \gamma_2/2)$, we do not do anything.

   (b) Otherwise, and if $r < k$, let $\boldsymbol{b}_{p,1}, \ldots, \boldsymbol{b}_{p,r}$ be the result from the Gram-Schmidt decomposition of $\boldsymbol{x}_{i_{p,1}}, \ldots, \boldsymbol{x}_{i_{p,r}}$. Then, let $\boldsymbol{y}_{p,l+1}$ be a sample of the distribution obtained by the uniform distribution $\boldsymbol{y}_{p,l+1} \sim \mathcal{U}(S^{d-1} \cap \{\boldsymbol{z} \in \mathbb{R}^d : |\boldsymbol{b}_{p,r'}^\top \boldsymbol{z}| \leq \frac{1}{d^3}, \forall r' \leq r\})$. We then pose $\boldsymbol{v}_{p,l+1} = \phi_\delta(\boldsymbol{y}_{p,l+1})$. Having defined this new vector, we increment l.

Procedure 1: The optimization procedure for algorithm alg

Input: d, k, p_{max}, algorithm alg

Part 1: Procedure to adaptively construct $\boldsymbol{v}$

Sample $\boldsymbol{A} \sim \mathcal{U}(\{\pm 1\}^{n\times d})$ and $\boldsymbol{v}_0 \sim \mathcal{U}(\mathcal{D}_\delta)$.
Initialize the memory of alg to **0** and let $p = 1$, $r = l = 0$.
for $t \geq 1$ **do**
  if $t > d^2$ **then** Set $(P, L) = (p, l)$ and break the **for** loop ;
  Run alg with current memory to obtain a query $\boldsymbol{x}_t$
  if $F_{\boldsymbol{A}}(\boldsymbol{x}_t) > \eta$ **then** `// Non-informative query`
    return $(\|\boldsymbol{A}\boldsymbol{x}_t\|_\infty - \eta, \tilde{\boldsymbol{g}}_{\boldsymbol{A}}(\boldsymbol{x}_t))$ as response to alg.
  else `// Informative query`
    if $r \leq k-1$ *and* $F_{\boldsymbol{A},\boldsymbol{v},p,l}(\boldsymbol{x}_t) \leq -\eta\gamma_1/2$ *and* $\|P_{Span(\boldsymbol{x}_{i_{p,r'}}, r' \leq r)^\perp}(\boldsymbol{x}_t)\|/\|\boldsymbol{x}_t\| \geq \frac{\gamma_2}{4}$ **then**
      Set $i_{p,r+1} = t$ and increment $r \leftarrow r+1$.
    if $F_{\boldsymbol{A},\boldsymbol{v},p,l}(\boldsymbol{x}_t) < -\eta(p\gamma_1 + l\gamma_2 + \gamma_2/2)$ *and* $r < k$ **then**
      Compute Gram-Schmidt decomposition $\boldsymbol{b}_{p,1}, \ldots, \boldsymbol{b}_{p,r}$ of $\boldsymbol{x}_{i_{p,1}}, \ldots, \boldsymbol{x}_{i_{p,r}}$.
      Sample $\boldsymbol{y}_{p,l+1}$ uniformly on $\mathcal{S}^{d-1} \cap \{\boldsymbol{z} \in \mathbb{R}^d : |\boldsymbol{b}_{p,r'}^\top \boldsymbol{z}| \leq d^{-3}, \forall r' \leq r\}$.
      Define $\boldsymbol{v}_{p,l+1} = \phi_\delta(\boldsymbol{y}_{p,l+1})$ and increment $l \leftarrow l+1$.
    else if $F_{\boldsymbol{A},\boldsymbol{v},p,l}(\boldsymbol{x}_t) < -\eta(p\gamma_1 + l\gamma_2 + \gamma_2/2)$ *and* $p+1 \leq p_{max}$ **then**
      Set $l_p = l$ and $i_{p+1,1} = t$.
      Compute the Gram-Schmidt decomposition $\boldsymbol{b}_{p+1,1}$ of $\boldsymbol{x}_{i_{p+1,1}}$.
      Sample $\boldsymbol{y}_{p+1,1}$ uniformly on $\mathcal{S}^{d-1} \cap \{\boldsymbol{z} \in \mathbb{R}^d : |\boldsymbol{b}_{p+1,1}^\top \boldsymbol{z}| \leq d^{-3}\}$.
      Define $\boldsymbol{v}_{p+1,1} = \phi_\delta(\boldsymbol{y}_{p+1,1})$, increment $p \leftarrow p+1$ and reset $l = r = 1$.
    else if $F_{\boldsymbol{A},\boldsymbol{v},p,l}(\boldsymbol{x}_t) < -\eta(p\gamma_1 + l\gamma_2 + \gamma_2/2)$ **then** `// End of the construction`
      Set $l_{p_{max}} = l$, $i_{p_{max}+1,1} = t$.
      Set $(P, L) = (p_{max}, l)$ and break the **for** loop.
    return $(F_{\boldsymbol{A},\boldsymbol{v},p,l}(\boldsymbol{x}_t), \boldsymbol{\partial} F_{\boldsymbol{A},\boldsymbol{v},p,l}(\boldsymbol{x}_t))$ as response to alg.
end

Part 2: Procedure once $\boldsymbol{v}$, P, L are constructed
for $t' \geq t$ **do** **return** $(F_{\boldsymbol{A},\boldsymbol{v},P,L}(\boldsymbol{x}_{t'}), \partial F_{\boldsymbol{A},\boldsymbol{v},P,L}(\boldsymbol{x}_{t'}))$ as response to the query $\boldsymbol{x}_{t'}$;

(c) Otherwise, if $r = k$, this ends period p. We write the total number of vectors defined during period p as $l_p := l$. If $p+1 \leq p_{max}$, period $p+1$ starts from $t = i_{p+1,1}$. Similarly to above, let $\boldsymbol{b}_{p+1,1}$ be the result of the Gram-Schmidt procedure on $\boldsymbol{x}_{p+1,1}$, and we sample $\boldsymbol{y}_{p+1,1}$ according to a uniform distribution $\boldsymbol{y}_{p+1,1} \sim \mathcal{U}(S^{d-1} \cap \left\{\boldsymbol{z} \in \mathbb{R}^d : |\boldsymbol{b}_{p+1,1}^\top \boldsymbol{z}| \leq \frac{1}{d^3}\right\})$. Then, we pose $\boldsymbol{v}_{p+1,1} = \phi_\delta(\boldsymbol{y}_{p+1,1})$, increment p, and reset $l = r = 1$.

After these steps, with the current values of p and l, we return $(F_{\boldsymbol{A},\boldsymbol{v},p,l}(\boldsymbol{x}_t), \partial F_{\boldsymbol{A},\boldsymbol{v},l,p}(\boldsymbol{x}_t))$.

If we finish the last period $p = p_{max}$, or if we reach a total number of iterations d^2, the construction phase of the function ends. In both cases, let us denote by P, L the last defined period and vector $\boldsymbol{v}_{P,L}$. In particular, we have $p \leq p_{max}$ From now on, the final function to optimize is $F_{\boldsymbol{A},\boldsymbol{v},P,L}$ and the oracle is a standard first-order oracle for this function, using the subgradient function $\partial F_{\boldsymbol{A},\boldsymbol{v},P,L}$.

3.2. Sketch of proof for Theorem 1

We relate Procedure 1 to the standard convex optimization problem and prove query lower bounds under memory constraints for this procedure. Before doing so, we formally define what we mean by solving this optimization procedure.

Definition 5 *Let alg be an algorithm for convex optimization. We say that an algorithm alg is successful for the optimization procedure with probability $q \in [0,1]$ and accuracy $\epsilon > 0$, if taking $\boldsymbol{A} \sim \mathcal{U}(\{\pm 1\}^{n\times d})$, running alg with the responses given by the procedure, and denoting by $\boldsymbol{x}^\star(alg)$ the final answer returned by alg, with probability at least q over the randomness of $\boldsymbol{A}$ and of the procedure, one has*

$$F_{\boldsymbol{A},\boldsymbol{v},P,L}(\boldsymbol{x}^\star(alg)) \leq \min_{\boldsymbol{x}\in B_d(\mathbf{0},1)} F_{\boldsymbol{A},\boldsymbol{v},P,L}(\boldsymbol{x}) + \epsilon.$$

The optimization procedure is designed such that with probability at least $1 - C\sqrt{\log d}/d^2$, the procedure returns responses that are consistent with a first-order oracle of the function $F_{\boldsymbol{A},\boldsymbol{v},P,L}$ where $\boldsymbol{v}_{P,L}$ is the last vector to have been defined.

Proposition 6 *Let $\boldsymbol{A} \in \{\pm 1\}^{n\times d}$ and $\boldsymbol{v}_0 \in \mathcal{D}_\delta$. On an event $\mathcal{E}$ of probability at least $1 - C\sqrt{\log d}/d^2$ on the randomness of the procedure for some universal constant $C > 0$, all responses of the optimization procedure are consistent with a first-order oracle for the function $F_{\boldsymbol{A},\boldsymbol{v},P,L}$: for any $t \geq 1$, if $(f_t, \boldsymbol{g}_t)$ is the response of the procedure at time t for query $\boldsymbol{x}_t$, then $f_t = F_{\boldsymbol{A},\boldsymbol{v},P,L}(\boldsymbol{x}_t)$ and $\boldsymbol{g}_t = \partial F_{\boldsymbol{A},\boldsymbol{v},P,L}(\boldsymbol{x}_t)$.*

Now observe that for any constructed vectors $\boldsymbol{v}$, the function $F_{\boldsymbol{A},\boldsymbol{v},P,L}$ is $\sqrt{d}$-Lipschitz. As a result, if there exists an algorithm for convex optimization that guarantees precision ϵ for 1-Lipschitz functions, by rescaling, there exists an algorithm alg which is successful for the optimization procedure with probability $1 - C\sqrt{\log d}/d^2$ and precision $\epsilon\sqrt{d}$. In the next proposition, we show that to be successful, such an algorithm needs to properly define the complete function $F_{\boldsymbol{A},\boldsymbol{v}}$, i.e., to complete all periods until p_{max}.

Proposition 7 *Let alg be a successful algorithm for the optimization procedure with probability $q \in [0,1]$ and precision $\eta/(2\sqrt{d})$. Suppose that alg performs at most d^2 queries during the optimization procedure. Then when running alg with the responses of the optimization procedure, alg succeeds and ends the period p_{max} with probability at least $q - C\sqrt{\log d}/d$ for some universal constant $C > 0$.*

Next, we introduce an Orthogonal Vector Game with Hints, Game 2, where the main difference with the game introduced in [23] is that the player can provide additional hints. Using Proposition 7, we prove that solving the optimization procedure implies solving Game 2.

Proposition 8 *Let $m \leq d$. Suppose that there is an M-bit algorithm that is successful for the optimization procedure with probability q for accuracy $\epsilon = \eta/(2\sqrt{d})$ and uses at most mp_{max} queries. Then, there is an algorithm for Game 2 for parameters $(d,k,m,M,\alpha = \frac{2\eta}{\gamma_1}, \beta = \frac{\gamma_2}{4})$, for which the Player wins with probability at least $q - C\sqrt{\log d}/d$ for some universal constant $C > 0$.*

Last, we give a $m = \tilde{\Omega}(d)$ query lower bound for Game 2.

Game 2: Orthogonal Vector Game with Hints

Input: d, k, m, M, α, β

Oracle: Set $n \leftarrow \lfloor d/4 \rfloor$, sample $\boldsymbol{A} \sim \mathcal{U}(\{\pm 1\}^{n \times d})$.

Player: Observe $\boldsymbol{A}$

for $l \in [d]$ **do**

  Player: Based on $\boldsymbol{A}$ and any previous queries and responses, submit at most k vectors $\boldsymbol{x}_{l,1}, \ldots, \boldsymbol{x}_{l,r_l}$.

  Oracle: Perform the Gram-Schmidt decomposition $\boldsymbol{b}_{l,1}, \ldots, \boldsymbol{b}_{l,r_l}$ of $\boldsymbol{x}_{l,1}, \ldots, \boldsymbol{x}_{l,r_l}$. Then, sample a vector $\boldsymbol{y}_l \in S^{d-1}$ according to a uniform distribution $\mathcal{U}(S^{d-1} \cap \{\boldsymbol{z} \in \mathbb{R}^d : \forall r \leq r_l, |\boldsymbol{b}_{l,r}^\top \boldsymbol{z}| \leq d^{-3}\})$. As response to the query, return $\boldsymbol{v}_l = \phi_\delta(\boldsymbol{y}_l)$ to the player.

end

Player: Based on $\boldsymbol{A}$, all previous queries and responses, store an M-bit message Message.

Player: Based on $\boldsymbol{A}$, all previous queries and responses, submit a function $\boldsymbol{g} : B_d(\boldsymbol{0}, 1) \to (\{\boldsymbol{a}_j, j \leq n\} \cup \{\boldsymbol{v}_l, l \leq d\}) \times [d^2]$ to the Oracle.

for $i \in [m]$ **do**

  Player: Based on Message, any previous queries $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{i-1}$ and responses $\boldsymbol{g}_1, \ldots, \boldsymbol{g}_{i-1}$ from this loop phase, submit a query $\boldsymbol{x}_i \in \mathbb{R}^d$.

  Oracle: As the response to query $\boldsymbol{z}_i$, return $\boldsymbol{g}_i = \boldsymbol{g}(\boldsymbol{z}_i)$.

end

Player: Based on all queries and responses from this phase $\{\boldsymbol{z}_i, \boldsymbol{g}_i, i \in [m]\}$, and on Message, return some vectors $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_k$ to the oracle.

The player wins if the returned vectors have unit norm and satisfy for all $i \in [k]$

1. $\|\boldsymbol{A}\boldsymbol{y}_i\|_\infty \leq \alpha$
2. $\|P_{Span(\boldsymbol{y}_1, \ldots, \boldsymbol{y}_{i-1})^\perp}(\boldsymbol{y}_i)\|_2 \geq \beta$.

Proposition 9 *Let $k \geq 20\frac{M+3d\log(2d)+1}{c_H n}$. And let $0 < \alpha, \beta \leq 1$ such that $\alpha(\sqrt{d}/\beta)^{5/4} \leq \frac{1}{2}$. If the Player wins the Orthogonal Vector Game with Hints (Game 2) with probability at least $1/2$, then $m \geq \frac{c_H}{8(30\log d + c_H)} d$.*

Putting everything together, we prove our main result.

Proof of Theorem 1 We set $n = \lceil d/4 \rceil$ and $k = \lceil 20\frac{M+3d\log(2d)+1}{c_H n} \rceil$. By Proposition 6, with probability at least $1 - C\sqrt{\log d}/d^2$, the procedure is consistent with a first-order oracle for convex optimization. Hence, since the functions $F_{\boldsymbol{A},\boldsymbol{v},P,L}$ are $\sqrt{d}$-Lipschitz, any M-bit algorithm guaranteed to solve convex optimization within accuracy $\epsilon = \eta/(2d) = 1/d^4$ for 1-Lipschitz functions, yields an algorithm that is successful for the optimization procedure with probability at least $1 - C\sqrt{\log d}/d^2$ and precision $\epsilon\sqrt{d} = \eta/(2\sqrt{d})$. Suppose that it uses at most Q queries. Then, by Proposition 8, there is a strategy for Game 2 for parameters $(d, k, \lceil Q/p_{max} \rceil + 1, M, \alpha = \frac{2\eta}{\gamma_1}, \beta = \frac{\gamma_2}{4})$ in which the Player wins with probability at least $1 - C'\sqrt{\log d}/d$. For d large enough, this probability is at least $1/2$. Further, $\frac{2\eta}{\gamma_1}\left(\frac{4\sqrt{d}}{\gamma_2}\right)^{5/4} \leq \frac{(4/3)^{5/4}}{3}\eta d^3 \leq \frac{1}{2}$. Hence, by Proposition 9, one has

$\lceil Q/p_{max} \rceil + 1 \geq \frac{c_H}{8(30\log d + c_H)} d$. Because one has $p_{max} = \Theta((d/k)^{1/3}\log^{-2/3} d)$, this implies

$$Q = \Omega\left(\frac{(d/k)^{1/3} d}{\log^{5/3} d}\right) = \Omega\left(\frac{d^{5/3}}{(M + \log d)^{1/3}\log^{5/3} d}\right).$$

In particular, if $M = d^{1+\delta}$ for $\delta \in [0,1]$, the number of queries is $Q = \tilde{\Omega}(d^{1+(1-\delta)/3})$. ■

Acknowledgments

This work was partly funded by ONR grant N00014-18-1-2122 and AFOSR grant FA9550-19-1-0263.

References

[1] Kurt M Anstreicher. The volumetric barrier for semidefinite programming. *Mathematics of Operations Research*, 25(3):365–380, 2000.

[2] Yossi Arjevani and Ohad Shamir. Communication complexity of distributed convex learning and optimization. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, page 1756–1764, Cambridge, MA, USA, 2015. MIT Press.

[3] David S Atkinson and Pravin M Vaidya. A cutting plane algorithm for convex programming that uses analytic centers. *Mathematical programming*, 69(1-3):1–43, 1995.

[4] Eric Balkanski and Yaron Singer. Parallelization does not accelerate convex optimization: Adaptivity lower bounds for non-smooth convex minimization. *arXiv preprint arXiv:1808.03880*, 2018.

[5] Paul Beame, Shayan Oveis Gharan, and Xin Yang. Time-space tradeoffs for learning finite functions from random evaluations, with applications to polynomials. In *Proceedings of the 31st Conference On Learning Theory*, pages 843–856. PMLR, 2018.

[6] Dimitris Bertsimas and Santosh Vempala. Solving convex programs by random walks. *Journal of the ACM (JACM)*, 51(4):540–556, 2004.

[7] Gavin Brown, Mark Bun, Vitaly Feldman, Adam Smith, and Kunal Talwar. When is memorization of irrelevant training data necessary for high-accuracy learning? In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2021, page 123–132. Association for Computing Machinery, 2021.

[8] Gavin Brown, Mark Bun, and Adam Smith. Strong memory lower bounds for learning natural models. In *Proceedings of Thirty Fifth Conference on Learning Theory*, pages 4989–5029. PMLR, 2022.

[9] Sébastien Bubeck, Qijia Jiang, Yin-Tat Lee, Yuanzhi Li, and Aaron Sidford. Complexity of highly parallel non-smooth convex optimization. *Advances in neural information processing systems*, 32, 2019.

[10] Yuval Dagan and Ohad Shamir. Detecting correlations with little memory and communication. In Sébastien Bubeck, Vianney Perchet, and Philippe Rigollet, editors, *Proceedings of the 31st Conference On Learning Theory*, volume 75 of *Proceedings of Machine Learning Research*, pages 1145–1198. PMLR, 06–09 Jul 2018. URL https://proceedings.mlr.press/v75/dagan18a.html.

[11] Yuval Dagan, Gil Kur, and Ohad Shamir. Space lower bounds for linear prediction in the streaming model. In Alina Beygelzimer and Daniel Hsu, editors, *Proceedings of the Thirty-Second Conference on Learning Theory*, volume 99 of *Proceedings of Machine Learning Research*, pages 929–954. PMLR, 25–28 Jun 2019. URL https://proceedings.mlr.press/v99/dagan19b.html.

[12] Uriel Feige and Gideon Schechtman. On the optimality of the random hyperplane rounding technique for max cut. *Random Structures & Algorithms*, 20(3):403–440, 2002.

[13] Sumegha Garg, Ran Raz, and Avishay Tal. Extractor-based time-space lower bounds for learning. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2018, page 990–1002. Association for Computing Machinery, 2018.

[14] Martin Grötschel, László Lovász, and Alexander Schrijver. *Geometric algorithms and combinatorial optimization*, volume 2. Springer Science & Business Media, 2012.

[15] Albert Xin Jiang and Kevin Leyton-Brown. Polynomial-time computation of exact correlated equilibrium in compact games. In *Proceedings of the 12th ACM conference on Electronic commerce*, pages 119–126, 2011.

[16] Haotian Jiang. Minimizing convex functions with integral minimizers. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 976–985. SIAM, 2021.

[17] Haotian Jiang, Yin Tat Lee, Zhao Song, and Sam Chiu-wai Wong. An improved cutting plane method for convex optimization, convex-concave games, and its applications. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, pages 944–953, 2020.

[18] Gillat Kol, Ran Raz, and Avishay Tal. Time-space hardness of learning sparse parities. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2017, page 1067–1080. Association for Computing Machinery, 2017.

[19] Yin Tat Lee, Aaron Sidford, and Sam Chiu-wai Wong. A faster cutting plane method and its implications for combinatorial and convex optimization. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, pages 1049–1065. IEEE, 2015.

[20] Anatoly Yur'evich Levin. An algorithm for minimizing convex functions. In *Doklady Akademii Nauk*, volume 160, pages 1244–1247. Russian Academy of Sciences, 1965.

[21] Adrian S. Lewis and Michael L. Overton. Nonsmooth optimization via quasi-Newton methods. *Mathematical Programming*, 141(1):135–163, October 2013.

[22] Dong C. Liu and Jorge Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45(1):503–528, August 1989.

[23] Annie Marsden, Vatsal Sharan, Aaron Sidford, and Gregory Valiant. Efficient convex optimization requires superlinear memory. In *Conference on Learning Theory*, pages 2390–2430. PMLR, 2022.

[24] S Thomas McCormick. Submodular function minimization. *Handbooks in operations research and management science*, 12:321–391, 2005.

[25] Ioannis Mitliagkas, Constantine Caramanis, and Prateek Jain. Memory limited, streaming pca. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, page 2886–2894, Red Hook, NY, USA, 2013. Curran Associates Inc.

[26] Dana Moshkovitz and Michal Moshkovitz. Mixing implies lower bounds for space bounded learning. In *Proceedings of the 2017 Conference on Learning Theory*, pages 1516–1566. PMLR, 2017.

[27] Dana Moshkovitz and Michal Moshkovitz. Entropy Samplers and Strong Generic Lower Bounds For Space Bounded Learning. In *9th Innovations in Theoretical Computer Science Conference (ITCS 2018)*, volume 94 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 28:1–28:20. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018.

[28] Arkadi Nemirovski. On parallel complexity of nonsmooth convex optimization. *Journal of Complexity*, 10(4):451–463, 1994.

[29] Arkadij Semenovič Nemirovskij and David Borisovich Yudin. Problem complexity and method efficiency in optimization. 1983.

[30] A.S. Nemirovsky, D.B. Yudin, and E.R. Dawson. *Problem Complexity and Method Efficiency in Optimization*. A Wiley-Interscience publication. Wiley, 1983. ISBN 978-0471103455.

[31] Ju E Nesterov. Self-concordant functions and polynomial-time methods in convex programming. *Report, Central Economic and Mathematic Institute, USSR Acad. Sci*, 1989.

[32] Yurii Nesterov. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2003.

[33] Jorge Nocedal. Updating quasi-newton matrices with limited storage. *Mathematics of Computation*, 35(151):773–782, 1980.

[34] Christos H Papadimitriou and Tim Roughgarden. Computing correlated equilibria in multi-player games. *Journal of the ACM (JACM)*, 55(3):1–29, 2008.

[35] Ran Raz. A time-space lower bound for a large class of learning problems. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 732–742, 2017. doi: 10.1109/FOCS.2017.73.

[36] Ran Raz. Fast learning requires good memory: A time-space lower bound for parity learning. *Journal of the ACM (JACM)*, 66(1):1–18, 2018.

[37] Vatsal Sharan, Aaron Sidford, and Gregory Valiant. Memory-sample tradeoffs for linear regression with small error. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2019, page 890–901. Association for Computing Machinery, 2019.

[38] Naum Z Shor. Cut-off method with space extension in convex programming problems. *Cybernetics*, 13(1):94–96, 1977.

[39] Jacob Steinhardt and John Duchi. Minimax rates for memory-bounded sparse linear regression. In *Proceedings of The 28th Conference on Learning Theory*, pages 1564–1587. PMLR, 2015.

[40] Jacob Steinhardt, Gregory Valiant, and Stefan Wager. Memory, communication, and statistical queries. In *29th Annual Conference on Learning Theory*, pages 1490–1516. PMLR, 2016.

[41] Sergei Pavlovich Tarasov. The method of inscribed ellipsoids. In *Soviet Mathematics-Doklady*, volume 37, pages 226–230, 1988.

[42] Pravin M Vaidya. A new algorithm for minimizing convex functions over convex sets. *Mathematical programming*, 73(3):291–341, 1996.

[43] Blake Woodworth and Nathan Srebro. Open problem: The oracle complexity of convex optimization with limited memory. In *Conference on Learning Theory*, pages 3202–3210. PMLR, 2019.

[44] Blake E. Woodworth and Nathan Srebro. Lower bound for randomized first order convex optimization. *arXiv: Optimization and Control*, 2017.

[45] Blake E Woodworth and Nati Srebro. Tight complexity bounds for optimizing composite objectives. In *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

[46] Blake E Woodworth, Brian Bullins, Ohad Shamir, and Nathan Srebro. The min-max complexity of distributed stochastic convex optimization with intermittent communication. In Mikhail Belkin and Samory Kpotufe, editors, *Proceedings of Thirty Fourth Conference on Learning Theory*, volume 134 of *Proceedings of Machine Learning Research*, pages 4386–4437. PMLR, 15–19 Aug 2021. URL https://proceedings.mlr.press/v134/woodworth21a.html.

[47] David B Yudin and Arkadi S Nemirovskii. Informational complexity and efficient methods for the solution of convex extremal problems. *Matekon*, 13(2):22–45, 1976.

Appendix A. Complete proofs of Section 3

In this section, we give the complete details of our tradeoffs between memory and query complexity for convex optimization. Throughout the proof, we will use concentration bounds relegated to Appendix C. First, the construction of the discretizations $\mathcal{D}_\delta$ with equal-area partitions uses the following result.

Lemma 10 ([12] Lemma 21) *For any $0 < \delta < \pi/2$, the sphere S^{d-1} can be partitioned into $N(\delta) = (\mathcal{O}(1)/\delta)^d$ equal volume cells, each of diameter at most δ.*

A.1. Properties and validity of the optimization procedure

We begin with a simple lemma showing that during each period p at most $l_p \leq k$ vectors $\boldsymbol{v}_{p,1}, \ldots, \boldsymbol{v}_{p,l_p}$ are constructed.

Lemma 11 *At any time of the construction procedure, $l \leq r$. In particular, since $r \leq k$, we have $l_p \leq k$ for all periods $p \leq p_{max}$.*

Proof Fix a period p. We prove this by induction. The claim is satisfied for any $l = 1$ when $p \geq 2$ since in this case, at the first time $t = i_{p,1}$ of the period p we also construct the first vector $\boldsymbol{v}_{p,1}$. For $p = 1$, note that the first informative query t that falls in scenarios (2b) or (2c) is exploratory. Indeed, in these cases we have $F_{\boldsymbol{A},\boldsymbol{v},1,0}(\boldsymbol{x}_t) < \eta(-\gamma_1 - \gamma_2/2) \leq -\eta\gamma_1/2$, and the second criterion for an exploratory query is immediate $\|P_{Span(\boldsymbol{x}_{i_{1,r'}}, r' \leq 0)}(\boldsymbol{x}_t)\| = 0$ since no indices $i_{1,r}$ have been defined yet.

We now suppose that the claim holds for $l - 1 \geq 1$. Let $t_{p,l}$ be the time when $\boldsymbol{v}_{p,l}$ is constructed and $i_{p,1}, \ldots, i_{p,r}$ the indices constructed until the beginning of iteration $t_{p,l}$. If a new index $i_{p,r'}$ was constructed in times $(t_{p,l-1}, t_{p,l})$ then the claim holds immediately. Suppose that this is not the case. Note that $t_{p,l}$ falls in scenario (2b) which means in particular that

$$\eta(\boldsymbol{v}_{p,l-1}^\top \boldsymbol{x}_{t_{p,l}} - p\gamma_1 - (l-1)\gamma_2) \leq F_{\boldsymbol{A},\boldsymbol{v},p,l-1}(\boldsymbol{x}_{t_{p,l}}) < \eta(-p\gamma_1 - (l-1)\gamma_2 - \gamma_2/2).$$

As a result,

$$|\boldsymbol{y}_{p,l-1}^\top \boldsymbol{x}_{t_{p,l}}| \geq |\boldsymbol{v}_{p,l-1}^\top \boldsymbol{x}_{t_{p,l}}| - \delta > \frac{\gamma_2}{2} - \delta.$$

Next, when $r \geq l - 1$ is the number of indices constructed so far, we decompose $\boldsymbol{y}_{p,l-1} = \alpha_1 \boldsymbol{b}_{p,1} + \ldots + \alpha_r \boldsymbol{b}_{p,r} + \tilde{\boldsymbol{y}}_{p,l-1}$ where $\tilde{\boldsymbol{y}}_{p,l-1} \in Span(\boldsymbol{x}_{i_{p,r'}}, r' \leq r)^\perp$. Since by construction of $\boldsymbol{y}_{p,l-1}$ one has $|\alpha_{r'}| \leq d^{-3}$ for all $r' \leq r$, we have

$$\|\tilde{\boldsymbol{y}}_{p,l-1} - \boldsymbol{y}_{p,l-1}\| \leq \frac{\sqrt{r}}{d^3} \leq \frac{1}{d^2\sqrt{d}}.$$

Therefore,

$$\|P_{Span(\boldsymbol{x}_{i_{p,r'}}, r' \leq r)^\perp}(\boldsymbol{x}_{t_{p,l}})\| \geq |\tilde{\boldsymbol{y}}_{p,l-1}^\top \boldsymbol{x}_{t_{p,l}}| \geq |\boldsymbol{y}_{p,l-1}^\top \boldsymbol{x}_{t_{p,l}}| - \frac{1}{d^2\sqrt{d}} > \frac{\gamma_2}{2} - \frac{1}{d^2\sqrt{d}} - \delta \geq \frac{\gamma_2}{4}.$$

As a result, $t_{p,l}$ is exploratory, hence $i_{p,r+1} = t_{p,l}$. This ends the proof of the recursion and the lemma. ■

We recall that P and L denote the last defined period and vector $\boldsymbol{v}_{P,L}$. From Lemma 11, we have in particular $P \leq p_{max}$ and $L \leq k$. The next step involves showing that with high probability, the returned values and vectors returned by the above procedure are consistent with a first-order oracle for minimizing the function $F_{\boldsymbol{A},\boldsymbol{v},P,L}$, as stated in Proposition 6.

Proof of Proposition 6 Consider a given iteration t. We aim to show that we have $(f_t, \boldsymbol{g}_t) = (F_{\boldsymbol{A},\boldsymbol{v},P,L}(\boldsymbol{x}_t), \partial F_{\boldsymbol{A},\boldsymbol{v},P,L}(\boldsymbol{x}_t))$. By construction, if $t \geq d^2$, the result is immediate. Now suppose $t \leq d^2$. We first consider the case when $\boldsymbol{x}_t$ is non-informative (1). By definition, $F_{\boldsymbol{A}}(\boldsymbol{x}_t) > \eta$. Since for any $(p,l) \leq_{lex} (P,L)$ one has $|\boldsymbol{v}_{p,l}^\top \boldsymbol{x}_t| \leq \|\boldsymbol{v}_{p,l}\| \|\boldsymbol{x}_t\| \leq 1$, we have

$$F_{\boldsymbol{A},\boldsymbol{v},P,L}(\boldsymbol{x}_t) = \max\left\{F_{\boldsymbol{A}}(\boldsymbol{x}_t), \eta\left(\max_{(p,l)\leq_{lex}(P,L)} \boldsymbol{v}_{p,l}^\top \boldsymbol{x} - p\gamma_1 - l\gamma_2\right)\right\} = F_{\boldsymbol{A}}(\boldsymbol{x}_t).$$

As a result, the response of the procedure for $\boldsymbol{x}_t$ is consistent with $F_{\boldsymbol{A},\boldsymbol{v},P,L}$ and the returned subgradient is $\tilde{\boldsymbol{g}}_{\boldsymbol{A}}(\boldsymbol{x}_t) = \partial F_{\boldsymbol{A},\boldsymbol{v},P,L}(\boldsymbol{x}_t)$. Therefore, it suffices to focus on informative queries (2). We will denote by $t_{p,l}$ the index of the iteration when $\boldsymbol{v}_{p,l}$ has been defined, for $(p,l) \leq_{lex} (P,L)$. Consider a specific couple $(p,l) \leq_{lex} (P,L)$, and let r denote the number of constructed indices on or before $t_{p,l}$. Let $\boldsymbol{b}_{p,1}, \ldots, \boldsymbol{b}_{p,r}$ the corresponding vectors resulting from the Gram-Schmidt procedure on $\boldsymbol{x}_{i_{p,1}}, \ldots, \boldsymbol{x}_{i_{p,r}}$. Then, conditionally on the history until time $t_{p,l}$, the vector $\boldsymbol{v}_{p,l}$ was defined as $\boldsymbol{v}_{p,l} = \phi_\delta(\boldsymbol{y}_{p,l})$, where $\boldsymbol{y}_{p,l}$ is sampled as $\sim \mathcal{U}(S^{d-1} \cap \{\boldsymbol{z} \in \mathbb{R}^d : |\boldsymbol{b}_{p,r'}^\top \boldsymbol{z}| \leq d^{-3}, \forall r' \leq r\})$. As a result, from Lemma 21, for any $t \leq t_{p,l}$, we have

$$\mathbb{P}\left(|\boldsymbol{x}_t^\top \boldsymbol{v}_{p,l}| \geq 3\sqrt{\frac{2\log d}{d}} + \frac{2}{d^2}\right) \leq \frac{6\sqrt{2\log d}}{d^6}.$$

We then define the following event

$$\mathcal{E} = \bigcap_{(p,l)\leq_{lex}(P,L)} \bigcap_{t\leq t_{p,l}} \left\{|\boldsymbol{x}_t^\top \boldsymbol{v}_{p,l}| < 3\sqrt{\frac{2\log d}{d}} + \frac{2}{d^2}\right\},$$

which by the union bound has probability $\mathbb{P}(\mathcal{E}) \geq 1 - 3\sqrt{2\log d}/d^2$. We are now ready to show that the construction procedure is consistent with optimizing $F_{\boldsymbol{A},\boldsymbol{v},P,L}$ on the event $\mathcal{E}$. As seen before, we can suppose that $\boldsymbol{x}_t$ is informative (2). Using the same notations as before, because $\mathcal{E}$ is met, for any $p < p' \leq P$ and $l' \leq l_{p'}$, we have for $d \geq 2$,

$$\boldsymbol{v}_{p',l'}^\top \boldsymbol{x}_t - p'\gamma_1 - l'\gamma_2 < 3\sqrt{\frac{2\log d}{d}} + \frac{1}{d} - p\gamma_1 - \gamma_1 \leq -p\gamma_1 - \frac{\gamma_1}{2} \leq -p\gamma_1 - d\gamma_2 - \frac{\gamma_2}{2},$$

where we used $3\sqrt{2}+1 \leq 6$ and $2d\gamma_2 \leq \gamma_1/2$. As a result, we obtain that

$$\max_{(p',l')\leq_{lex}(P,L), p'>p} \boldsymbol{v}_{p',l'}^\top \boldsymbol{x}_t - p'\gamma_1 - l'\gamma_2 < -p\gamma_1 - l\gamma_2 - \frac{\gamma_2}{2}.$$

Next, we consider the case of vectors $\boldsymbol{v}_{p,l'}$ where $l \leq l' \leq l_p$ and $t_{p,l'} \geq t$ (this also includes the case when we defined $\boldsymbol{v}_{p,l}$ at time $t = t_{p,l}$). We write $\tilde{l}$ for the smallest such index l. As a remark, $\tilde{l} \in \{l, l+1\}$. Note that if such indices exist, this means that before starting iteration t, the procedure has not yet reached $r = k$. There are two cases. If $\boldsymbol{x}_t$ was exploratory, we have $t = i_{p,r}$ hence $\|P_{Span(\boldsymbol{b}_{p,r'},r'\leq r)^\top}(\boldsymbol{x}_t)\| = 0$. If $\boldsymbol{x}_t$ is not exploratory, either

$$\|P_{Span(\boldsymbol{b}_{p,r'},r'\leq r)^\top}(\boldsymbol{x}_t)\| < \frac{\gamma_2}{4}\|\boldsymbol{x}_t\| \leq \frac{\gamma_2}{4}, \tag{4}$$

or we have $F_{\boldsymbol{A},v,p,l}(\boldsymbol{x}_t) > -\eta\gamma_1/2$. We start with the last scenario when $F_{\boldsymbol{A},v,p,l}(\boldsymbol{x}_t) > -\eta\gamma_1/2$. Then, on $\mathcal{E}$, one has

$$\max_{(p,l)<_{lex}(p',l')\leq_{lex}(P,L)} \boldsymbol{v}_{p',l'}^\top \boldsymbol{x}_t - p'\gamma_1 - l'\gamma_2 \leq -\gamma_1 + 3\sqrt{\frac{2\log d}{d}} + \frac{1}{d} \leq \frac{\gamma_1}{2}$$

As a result, this shows that $F_{\boldsymbol{A},\boldsymbol{v},P,L}(\boldsymbol{x}_t) = F_{\boldsymbol{A},\boldsymbol{v},p,l}(\boldsymbol{x}_t)$. Hence using a first-order oracle from $F_{\boldsymbol{A},\boldsymbol{v},l,p}$ at $\boldsymbol{x}_t$ is already consistent with $F_{\boldsymbol{A},\boldsymbol{v},P,L}$. Thus, for whichever step (2a), (2b) or (2c) is performed, since these can only increase the knowledge on $\boldsymbol{v}$, the response given by the construction procedure is consistent with minimizing $F_{\boldsymbol{A},\boldsymbol{v}}$.

It remains to treat the first two scenarios in which we always have Eq (4). In particular, when writing $\boldsymbol{x}_t = \alpha_1\boldsymbol{b}_{p,1} + \ldots + \alpha_r\boldsymbol{b}_{p,r} + \tilde{\boldsymbol{x}}_t$ where $\tilde{\boldsymbol{x}}_t = P_{Span(\boldsymbol{b}_{p,r'},r'\leq r)^\perp}(\boldsymbol{x}_t)$, we have $\|\tilde{\boldsymbol{x}}_t\| < \frac{\gamma_2}{4}$. As a result, for $\tilde{l} \leq l' \leq l_p$, one has for

$$\begin{aligned}
|\boldsymbol{v}_{p,l'}^\top \boldsymbol{x}_t| \leq |\boldsymbol{y}_{p,l'}^\top \boldsymbol{x}_t| + \delta &\leq |\alpha_1||\boldsymbol{y}_{p,l'}^\top \boldsymbol{b}_{p,1}| + \ldots + |\alpha_r||\boldsymbol{y}_{p,l'}^\top \boldsymbol{b}_{p,r}| + \|\tilde{\boldsymbol{x}}_t\| + \delta \\
&< \|\boldsymbol{\alpha}\|_1 \frac{1}{d^3} + \frac{\gamma_2}{4} + \delta \\
&\leq \frac{\gamma_2}{4} + \frac{1}{d^2\sqrt{d}} + \frac{1}{d^3} \leq \frac{\gamma_2}{2},
\end{aligned}$$

where in the last inequality we used $d \geq 3$. As a result, provided that $\tilde{l}$ exists, this shows that

$$\max_{\tilde{l}\leq l'\leq l_p} \boldsymbol{v}_{p,l'}^\top \boldsymbol{x}_t - p\gamma_1 - l'\gamma_2 = \boldsymbol{v}_{p,\tilde{l}}^\top \boldsymbol{x}_t - p\gamma_1 - \tilde{l}\gamma_2 < -p\gamma_1 - \tilde{l}\gamma_2 + \frac{\gamma_2}{2}. \tag{5}$$

On the other hand, if $t = i_{p+1,1}$, the same reasoning works for t viewing it as in period $p+1$, which shows for this case that

$$\max_{l'\leq l_{p+1}} \boldsymbol{v}_{p+1,l'}^\top \boldsymbol{x}_t - (p+1)\gamma_1 - l'\gamma_2 = \boldsymbol{v}_{p+1,1}^\top \boldsymbol{x}_t - (p+1)\gamma_1 - \gamma_2 < -(p+1)\gamma_1 - \frac{\gamma_2}{2}. \tag{6}$$

As a conclusion of these estimates, we showed that on $\mathcal{E}$, we have

$$F_{\boldsymbol{A},\boldsymbol{v},P,L}(\boldsymbol{x}_t) = \max\left\{F_{\boldsymbol{A},\boldsymbol{v},p,l}(\boldsymbol{x}_t), \eta(\boldsymbol{v}_{p',l'}^\top \boldsymbol{x}_t - p'\gamma_1 - l'\gamma_2)\right\} := \tilde{F}_{\boldsymbol{A},\boldsymbol{v},t}(\boldsymbol{x}_t)$$

where (p', l') is the very next vector that is defined after starting iteration t (potentially, it has $t_{p',l'} = t$ if we defined a vector at this time). It then suffices to check that the value and vector returned by the procedure are consistent with the right-hand side. By construction, if we constructed $\boldsymbol{v}_{p',l'}$ at step t: case (2b) or (2c), then the procedure directly uses a first-order oracle for $\tilde{F}_{\boldsymbol{A},\boldsymbol{v},t}$. Further, by construction of the subgradients since they break ties lexicographically in (p, l), the returned subgradient is exactly $\partial F_{\boldsymbol{A},\boldsymbol{v},P,L}(\boldsymbol{x}_t)$. It remains to check that this is the case when no vector $\boldsymbol{v}_{p',l'}$ is defined at step t: case (2a). This corresponds to the case when $F_{\boldsymbol{A},\boldsymbol{v},p,l}(\boldsymbol{x}_t) \geq \eta(-p\gamma_1 - l\gamma_2 - \gamma/2)$. In this case, the upper bound estimates from Eq (5) and Eq (6) imply that

$$\boldsymbol{v}_{p',l'}^\top \boldsymbol{x}_t - p'\gamma_1 - l'\gamma_2 < -p\gamma_1 - l\gamma_2 - \gamma/2,$$

and as a result, $F_{\boldsymbol{A},\boldsymbol{v},P,L}(\boldsymbol{x}_t) = F_{\boldsymbol{A},\boldsymbol{v},p,l}(\boldsymbol{x}_t)$. Therefore, using a first-order oracle of $F_{\boldsymbol{A},\boldsymbol{v},p,l}$ at $\boldsymbol{x}_t$ is valid, and the break of ties of the subgradient of $\tilde{F}_{\boldsymbol{A},\boldsymbol{v},t}$ is the same as the break of ties of $\partial F_{\boldsymbol{A},\boldsymbol{v},P,L}(\boldsymbol{x}_t)$. This ends the proof that on $\mathcal{E}$ the procedure gives responses consistent with an optimization oracle for $F_{\boldsymbol{A},\boldsymbol{v},P,L}$ with subgradient function $\partial F_{\boldsymbol{A},\boldsymbol{v},P,L}$. Because $\mathbb{P}(\mathcal{E}) \geq 1 - C\sqrt{\log d}/d^2$ for some constant $C > 0$, this ends the proof of the proposition. ■

Last, we provide an upper bound on the optimal value of $F_{\boldsymbol{A},\boldsymbol{v},P,L}$.

Proposition 12 *Let $\boldsymbol{A} \sim \mathcal{U}(\{\pm 1\}^{n\times d})$ and $\boldsymbol{v}_0 \sim \mathcal{U}(\mathcal{D}_\delta)$. For any algorithm alg for convex optimization, let $\boldsymbol{v}$ be the resulting set of vectors constructed by the randomized procedure. With probability at least $1 - C\sqrt{\log d}/d$ over the randomness of $\boldsymbol{A}$, $\boldsymbol{v}_0$ and $\boldsymbol{v}$, we have*

$$\min_{\boldsymbol{x}\in B_d(\boldsymbol{0},1)} F_{\boldsymbol{A},\boldsymbol{v}}(\boldsymbol{x}) \leq -\frac{\eta}{30\sqrt{(kp_{max}+1)\log d}},$$

for some universal constant $C > 0$.

Proof For simplicity, let us enumerate all the constructed vectors $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_{l_{max}}$ by order of construction. Hence, $l_{max} \leq p_{max}k$. We use the same enumeration for $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_{l_{max}}$. Next, let $C_d = \sqrt{40(l_{max}+1)\log d}$ and consider the following vector,

$$\bar{\boldsymbol{x}} = -\frac{1}{C_d}\sum_{l=0}^{l_{max}} P_{Span(\boldsymbol{a}_i, i\leq n)^\perp}(\boldsymbol{v}_l).$$

In particular, note that we included $\boldsymbol{v}_0$ in the sum. For convenience, we write $P_{\boldsymbol{A}^\perp}$ instead of $P_{Span(\boldsymbol{a}_i, i\leq n)^\perp}$. Also, for convenience let us define $\boldsymbol{z}_l = \sum_{l'\leq l} P_{\boldsymbol{A}^\perp}(\boldsymbol{v}_l)$. Fix an index $1 \leq l \leq l_{max}$. Then, by Lemma 21, with $t_0 := \sqrt{\frac{6\log d}{d}} + \frac{2}{d^2}$, we have

$$\begin{aligned}
\mathbb{P}\left(|P_{\boldsymbol{A}^\perp}(\boldsymbol{v}_{l+1})^\top \boldsymbol{z}_l| > t_0\|\boldsymbol{z}_l\|\right) &= \mathbb{P}\left(|\boldsymbol{v}_{l+1}^\top P_{\boldsymbol{A}^\perp}(\boldsymbol{z}_l)| > t_0\|\boldsymbol{z}_l\|\right) \\
&\leq \mathbb{P}\left(|\boldsymbol{v}_{l+1}^\top P_{\boldsymbol{A}^\perp}(\boldsymbol{z}_l)| > t_0\|P_{\boldsymbol{A}^\perp}(\boldsymbol{z}_l)\|\right) \\
&\leq \frac{2\sqrt{6\log d}}{d^2}.
\end{aligned}$$

Similarly, we have that

$$\mathbb{P}\left(|\boldsymbol{v}_{l+1}^\top \boldsymbol{z}_l| > t_0\|\boldsymbol{z}_l\|\right) \leq \frac{2\sqrt{6\log d}}{d^2}.$$

We consider the event $\mathcal{E} = \bigcap_{l\leq l_{max}} \{|\boldsymbol{v}_l^\top \boldsymbol{z}_{l-1}|, |P_{\boldsymbol{A}^\perp}(\boldsymbol{v}_l)^\top \boldsymbol{z}_{l-1}| \leq t_0\|\boldsymbol{z}_l\|\}$, which since $l_{max} \leq d$, by the union bound has probability at least $1 - 4\sqrt{6\log d}/d$. Then, on $\mathcal{E}$, for any $l < l_{max}$,

$$\|\boldsymbol{z}_{l+1}\|^2 \leq \|\boldsymbol{z}_l\|^2 + \|P_{\boldsymbol{A}^\perp}(\boldsymbol{v}_{l+1})\|^2 + 2|P_{\boldsymbol{A}^\perp}(\boldsymbol{v}_{l+1})^\top \boldsymbol{z}_l| \leq \|\boldsymbol{z}_l\|^2 + 1 + 2t_0\|\boldsymbol{z}_l\|.$$

We now prove by induction that $\|\boldsymbol{z}_l\|^2 \leq 40\log d \cdot (l+1)$, which is clearly true for $\boldsymbol{z}_0$ since $\|\boldsymbol{z}_0\| = \|P_{\boldsymbol{A}^\perp}(\boldsymbol{v}_0)\| \leq \|\boldsymbol{v}_0\| \leq 1$. Suppose this is true for $l < l_{max}$. Then, using the above equation and the fact that $t_0 \leq 3\sqrt{\frac{\log d}{d}}$ for $d \geq 4$,

$$\|\boldsymbol{z}_{l+1}\|^2 \leq 40\log d \cdot (l+1) + 1 + 6\sqrt{40}\log d\sqrt{\frac{l+1}{d}} \leq 40\log d \cdot (l+2),$$

where we used $l_{max} + 1 \leq d$, which completes the induction. In particular, on $\mathcal{E}$, we have that $\|\bar{\boldsymbol{x}}\| \leq 1$. Also, observe that by construction $\bar{\boldsymbol{x}} \in Span(\boldsymbol{a}_i, i \leq n)^\perp$ so that $\|\boldsymbol{A}\bar{\boldsymbol{x}}\|_\infty = 0$. Next, for any $0 \leq l \leq l_{max}$, we have

$$\boldsymbol{v}_l^\top \bar{\boldsymbol{x}} = -\frac{\boldsymbol{v}_l^\top \boldsymbol{z}_{l_{max}}}{C_d} = -\frac{1}{C_d}\left(\|P_{\boldsymbol{A}^\perp}(\boldsymbol{v}_l)\|^2 + \boldsymbol{v}_l^\top \boldsymbol{z}_{l-1} + \sum_{l<l'\leq l_{max}} \boldsymbol{v}_l^\top P_{\boldsymbol{A}^\perp}(\boldsymbol{v}_{l'})\right).$$

We will give estimates on each term of the above equation. First, if the indices $i_{p,1},\ldots,i_{p,r}$ were defined before defining $\boldsymbol{v}_l$, we denote $\tilde{\boldsymbol{y}} = P_{Span(\boldsymbol{x}_{i_{p,r'}}, r'\leq r)^\perp}(\boldsymbol{y}_l)$, the component of $\boldsymbol{y}_l$ which is perpendicular to the explored space at that time. Then, we can write $\boldsymbol{y}_l = \alpha_1^l \boldsymbol{b}_{p,1} + \ldots + \alpha_r^l \boldsymbol{b}_{p,1} + \tilde{\boldsymbol{y}}_l$, and note that

$$\|\tilde{\boldsymbol{y}}_l\| = \sqrt{\|\boldsymbol{y}_l\| - (\alpha_1^l)^2 - \ldots - (\alpha_r^l)^2} \geq \sqrt{1 - \frac{k}{d^6}} \geq 1 - \frac{1}{d^5}.$$

Then, we have

$$\begin{aligned}
\|P_{\boldsymbol{A}^\perp}(\boldsymbol{v}_l)\| &\geq \|P_{\boldsymbol{A}^\perp}(\boldsymbol{y}_l)\| - \delta \\
&\geq \|P_{Span(\boldsymbol{a}_i, i\leq n, \boldsymbol{b}_{p,r'}, r\leq r')^\perp}(\boldsymbol{y}_l)\| - \delta \\
&= \|P_{Span(\boldsymbol{a}_i, i\leq n, \boldsymbol{b}_{p,r'}, r\leq r')^\perp}(\tilde{\boldsymbol{y}}_l)\| - \delta \\
&\geq \left\|P_{Span(a_i, i\leq n, \boldsymbol{b}_{p,r'}, r'\leq r)^\perp}\left(\frac{\tilde{\boldsymbol{y}}_l}{\|\tilde{\boldsymbol{y}}_l\|}\right)\right\| - \frac{1}{d^5} - \delta.
\end{aligned}$$

As a result, since $\delta = d^{-3}$, this shows that

$$\|P_{\boldsymbol{A}^\perp}(\boldsymbol{v}_l)\|^2 \geq \left\|P_{Span(a_i, i\leq n, \boldsymbol{b}_{p,r'}, r'\leq r)^\perp}\left(\frac{\tilde{\boldsymbol{y}}_l}{\|\tilde{\boldsymbol{y}}_l\|}\right)\right\|^2 - 2\delta.$$

Now observe that $dim(Span(a_i, i\leq n, \boldsymbol{b}_{p,r'}, r'\leq r)^\perp) \geq d-n-k$, while $\frac{\tilde{\boldsymbol{y}}_l}{\|\tilde{\boldsymbol{y}}_l\|}$ is a uniformly random unit vector in $Span(\boldsymbol{b}_{p,r'}, r\leq r')^\perp$. Therefore, using Proposition 20 we obtain for $t < 1$,

$$\begin{aligned}
&\mathbb{P}\left(\|P_{\boldsymbol{A}^\perp}(\boldsymbol{v}_l)\|^2 + 2\delta - \frac{d-n-k}{d} \leq -t\right) \\
&\leq \mathbb{P}\left(\left\|P_{Span(a_i, i\leq n, \boldsymbol{b}_{p,r'}, r'\leq r)^\perp}\left(\frac{\tilde{\boldsymbol{y}}_l}{\|\tilde{\boldsymbol{y}}_l\|}\right)\right\|^2 - \frac{d-n-k}{d} \leq -t\right) \\
&\leq e^{-(d-k)t^2}.
\end{aligned}$$

As a result since $d-n-k \geq d/2$, we obtain

$$\mathbb{P}\left(\|P_{\boldsymbol{A}^\perp}(\boldsymbol{v}_l)\|^2 \leq \frac{1}{2} - 2\sqrt{\frac{\log d}{d}} - 2\delta\right) \leq \frac{1}{d^2}.$$

Nwxt, define $\mathcal{F} = \bigcap_{l \leq l_{max}} \{\|P_{\boldsymbol{A}^\perp}(\boldsymbol{v}_l)\|^2 \geq \frac{1}{2} - 2\sqrt{\frac{\log d}{d}} - 2\delta\}$, which since $l_{max}+1 \leq d$ and by the union bound has probability at least $\mathbb{P}(\mathcal{F}) \geq 1 - 1/d$. Next, we turn to the last term. For any $0 \leq l < l_{max}$, we focus on the sequence $(\sum_{l'=l+1}^{l+u} \boldsymbol{v}_l^\top P_{\boldsymbol{A}^\top}(\boldsymbol{y}_{l'}))_{1 \leq u \leq l_{max}-l}$ and first note that this is a martingale. These increments are symmetric (because $\boldsymbol{y}_{l'}$ is symmetric) even conditionally on $\boldsymbol{A}$ and $\boldsymbol{v}_l, \boldsymbol{y}_l, \ldots, \boldsymbol{y}_{l'-1}$. Next, let $t_1 = 2\sqrt{\frac{3\log d}{d}} + \frac{2}{d^2}$. Note that for $d \geq 4$, we have $t_1 \leq 4\sqrt{\frac{\log d}{d}}$. Further, by Lemma 21,

$$\mathbb{P}(|\boldsymbol{v}_l^\top P_{\boldsymbol{A}^\top}(\boldsymbol{y}_{l'})| > t_1) = \mathbb{P}(|P_{\boldsymbol{A}^\top}(\boldsymbol{v}_l)^\top \boldsymbol{y}_{l'}| > t_1) \leq \frac{4\sqrt{3\log d}}{d^4},$$

where we used the fact that $P_{\boldsymbol{A}^\perp}$ is a projection. Let $\mathcal{G}_l = \bigcap_{l < l' \leq l_{max}} \{|\boldsymbol{v}_l^\top P_{\boldsymbol{A}^\top}(\boldsymbol{v}_{l'})| \leq t_1\}$, which by the union bound has probability $\mathbb{P}(\mathcal{G}_l) \geq 1 - 4\sqrt{3\log d}/d^3$. Next, we define $I_{l,u} = (\boldsymbol{v}_l^\top P_{\boldsymbol{A}^\top}(\boldsymbol{y}_{l+u}) \wedge t_1) \vee (-t_1)$, the increments capped at absolute value t_1. Because $\boldsymbol{v}_l^\top P_{\boldsymbol{A}^\top}(\boldsymbol{y}_{l+u})$ is symmetric, so is $I_{l,u}$. As a result, these are bounded increments of a martingale, to which we can apply the Azuma-Hoeffding inequality.

$$\mathbb{P}\left(\left|\sum_{u=1}^{l_{max}-l} I_{l,u}\right| \leq 2t_1\sqrt{(l_{max}-l)\log d}\right) \geq 1 - \frac{2}{d^2}.$$

We denote by $\mathcal{H}_l$ this event. Observe that on $\mathcal{G}_l$, the increments $I_{l,u}$ and $\boldsymbol{v}_l^\top P_{\boldsymbol{A}^\top}(\boldsymbol{y}_{l+u})$ coincide for all $1 \leq u \leq l_{max} - l$. As a result, on $\mathcal{G}_l \cap \mathcal{H}_l$ we obtain

$$\begin{aligned}
\left|\sum_{l < l' \leq l_{max}} \boldsymbol{v}_l^\top P_{\boldsymbol{A}^\perp}(\boldsymbol{v}_{l'})\right| &\leq \left|\sum_{l < l' \leq l_{max}} \boldsymbol{v}_l^\top P_{\boldsymbol{A}^\perp}(\boldsymbol{y}_{l'})\right| + (l_{max}-1)\delta \\
&\leq \left|\sum_{u=1}^{l_{max}-l} I_{l,u}\right| + (d-2)\delta \\
&\leq 2t_1\sqrt{l_{max}\log d} + (d-2)\delta.
\end{aligned}$$

Then, on the event $\mathcal{E} \cap \mathcal{F} \cap \bigcap_{l \leq l_{max}} \mathcal{G}_l \cap \mathcal{H}_l$, for any $1 \leq l \leq l_{max}$ one has

$$\begin{aligned}
\boldsymbol{v}_l^\top \boldsymbol{z}_{l_{max}} &\geq \frac{1}{2} - 2\sqrt{\frac{\log d}{d}} - t_0\|\boldsymbol{z}_l\| - 2t_1\sqrt{l_{max}\log d} - \frac{1}{d^2} \\
&\geq \frac{1}{2} - 2\sqrt{\frac{\log d}{d}} - 3\log d\sqrt{40\frac{l_{max}+1}{d}} - 8\log d\sqrt{\frac{l_{max}}{d}} - \frac{1}{d^2} \\
&\geq \frac{1}{2} - 30\log d\sqrt{\frac{l_{max}+1}{d}} \\
&\geq \frac{1}{6},
\end{aligned}$$

where in the last inequalities we used the fact that $l_{max} \leq kp_{max} \leq c_{d,1}d-1$ where $c_{d,1} = \frac{1}{90^2 \log^2 d}$ as per Eq (3). As a result, we obtain that on $\mathcal{E} \cap \mathcal{F} \cap \bigcap_{l \leq l_{max}} \mathcal{G}_l \cap \mathcal{H}_l$, which has probability at most $1 - C\sqrt{\log d}/d$ for some constant $C > 0$,

$$\max_{p \leq p_{max}, l \leq k} \boldsymbol{v}_{p,l}^\top \bar{\boldsymbol{x}} \leq -\frac{1}{6C_d} \leq -\frac{1}{40\sqrt{(kp_{max}+1)\log d}}.$$

Since $\|\boldsymbol{A}\bar{\boldsymbol{x}}\|_\infty = 0$, and $\eta \geq \frac{\eta}{40\sqrt{(kp_{max}+1)\log d}}$, this shows that

$$F_{\boldsymbol{A},\boldsymbol{v}}(\bar{\boldsymbol{x}}) \leq -\frac{\eta}{40\sqrt{(kp_{max}+1)\log d}}.$$

This ends the proof of the proposition. ■

A.2. Reduction from convex optimization to the optimization procedure

Next, we prove Proposition 7 which shows that to be successful for the optimization procedure, an algorithm needs to properly define the function $F_{\boldsymbol{A},v}$, i.e., to complete all periods until p_{max}.

Proof of Proposition 7 Let $\boldsymbol{x}^\star(alg) = \boldsymbol{x}_T$ denote the final answer of alg when run with the optimization procedure. By hypothesis, we have $T \leq d^2$. As before, let $P \leq p_{max}$ and $L \leq k$ be the indices such that the last vector constructed by the optimization procedure is $v_{P,L}$. Let $\mathcal{E}$ be the event when alg run on the optimization procedure does not end period p_{max}. We focus on $\mathcal{E}$ and consider two cases.

First, suppose that $T > t_{P,L}$, i.e., the last vector was not constructed at time T. As a result, this means that $\boldsymbol{x}_T$ corresponds either to a non-informative query—scenario (1)—in which case $F_{\boldsymbol{A},\boldsymbol{v},P,L}(\boldsymbol{x}_T) \geq F_{\boldsymbol{A}}(\boldsymbol{x}_T) \geq \eta$, or this means that $F_{\boldsymbol{A},\boldsymbol{v},P,L}(\boldsymbol{x}_t) \geq \eta(-P\gamma_1 - L\gamma_2 - \gamma/2)$—scenario (2a).

Second, we suppose that $T = t_{P,L}$, i.e., the last vector was constructed at time T. Then, by construction of $\boldsymbol{v}_{P,L}$ and $\boldsymbol{y}_{P,L}$, we have indices $i_{P,1}, \ldots, i_{P,r} \leq T$ such that with the Gram-Schmidt decomposition $\boldsymbol{b}_{P,1}, \ldots, \boldsymbol{b}_{P,r}$ of $\boldsymbol{x}_{i_{P,1}}, \ldots, \boldsymbol{x}_{i_{P,r}}$, we have $|\boldsymbol{b}_{p,r'}^\top \boldsymbol{y}_{P,L}| \leq d^{-3}$ for all $r' \leq r$. In particular, writing $\boldsymbol{x}_T = \alpha_1 \boldsymbol{b}_{P,1} + \ldots + \alpha_r \boldsymbol{b}_{P,r} + \tilde{\boldsymbol{x}}_T$, where $\tilde{\boldsymbol{x}}_T \in Span(\boldsymbol{x}_{i_{P,r'}}, r' \leq r)^\perp$, either we have $i_{P,r} = T$, in which case $\tilde{\boldsymbol{x}}_T = \boldsymbol{0}$, or $\boldsymbol{x}_T$ was not exploratory in which case we directly have $F_{\boldsymbol{A},\boldsymbol{v},P,L}(\boldsymbol{x}_T) \geq F_{\boldsymbol{A},\boldsymbol{v},P,L-1}(\boldsymbol{x}_T) > -\eta\gamma_1/2$, or we have $\|\tilde{\boldsymbol{x}}_T\| < \|\boldsymbol{x}_T\|\gamma_2/4 \leq \gamma_2/4$. For all remaining cases to consider, we obtain

$$|\boldsymbol{v}_{P,L}^\top \boldsymbol{x}_T| \leq |\boldsymbol{y}_{P,L}^\top \boldsymbol{x}_T| + \delta \leq \frac{\|\boldsymbol{\alpha}\|_1}{d^3} + \|\tilde{\boldsymbol{x}}_T\| + \delta \leq \frac{1}{d^3} + \frac{1}{d^2\sqrt{d}} + \frac{\gamma_2}{4} < \frac{\gamma_2}{2}.$$

In the last inequality, we used $d \geq 4$. This shows that $F_{\boldsymbol{A},\boldsymbol{v},P,L}(\boldsymbol{x}_T) \geq \eta(-P\gamma_1 - L\gamma_2 - \gamma_2/2)$. As a result, in all cases this shows that $F_{\boldsymbol{A},\boldsymbol{v},P,L}(\boldsymbol{x}^\star(alg)) \geq \eta(-P\gamma_1 - L\gamma_2 - \gamma_2/2) \geq -\eta(p_{max}+1)\gamma_1$. Now define the event

$$\mathcal{F} = \left\{ \min_{\boldsymbol{x} \in B_d(\boldsymbol{0},1)} F_{\boldsymbol{A},\boldsymbol{v}}(\boldsymbol{x}) \leq -\frac{\eta}{40\sqrt{(kp_{max}+1)\log d}} \right\}.$$

By Proposition 12 we have $\mathcal{P}(\mathcal{F}) \geq 1 - C\sqrt{\log d}/d$. From Eq (3),

$$(p_{max}+1)^{3/2} \leq \frac{1}{60\gamma_1\sqrt{k\log d}}.$$

Thus,

$$(p_{max}+1)\gamma_1 \leq \frac{1}{60\sqrt{k(p_{max}+1)\log d}} \leq \frac{1}{60\sqrt{(kp_{max}+1)\log d}}$$

Then, since $F_{\boldsymbol{A},\boldsymbol{v},P,L} \leq F_{\boldsymbol{A},\boldsymbol{v}}$, this shows that on $\mathcal{E}\cap\mathcal{F}$,

$$\begin{aligned} F_{\boldsymbol{A},\boldsymbol{v},P,L}(\boldsymbol{x}^\star(alg)) \geq -\eta(p_{max}+1)\gamma_1 &\geq \min_{\boldsymbol{x}\in B_d(\boldsymbol{0},1)} F_{\boldsymbol{A},\boldsymbol{v}}(\boldsymbol{x}) + \frac{\eta}{120\sqrt{(kp_{max}+1)\log d}} \\ &> \min_{\boldsymbol{x}\in B_d(\boldsymbol{0},1)} F_{\boldsymbol{A},\boldsymbol{v},P,L}(\boldsymbol{x}) + \frac{\eta}{2\sqrt{d}} \end{aligned}$$

where in the last inequality, we used $kp_{max} \leq c_{d,1}d - 1$. As a result, letting $\mathcal{G}$ be the event when alg succeeds for precision $\epsilon = \eta/(2\sqrt{d})$. By hypothesis, $\mathcal{P}(\mathcal{G}) \geq q$. By the above equations, one has $\mathcal{E}\cap\mathcal{F}\cap\mathcal{G} = \emptyset$. Therefore, $\mathbb{P}(\mathcal{G}\cap\mathcal{E}^c) \geq \mathcal{P}(\mathcal{G}) - \mathbb{P}(\mathcal{G}\cap\mathcal{E}\cap\mathcal{F}) - \mathbb{P}(\mathcal{F}^c) \geq q - C\sqrt{\log d}/d$. This ends the proof of the proposition. ■

A.3. Reduction of the optimization procedure to the Orthogonal Vector Game with Hints

Using the result from Proposition 7, we show that solving the optimization procedure implies solving the Orthogonal Game with Hints with high probability.

Proof of Proposition 8 Let alg be an M-bit algorithm solving the feasibility problem with mp_{max} queries with probability at least q. Below, we describe the strategy for Game 2.

In the first part of the strategy, the player observes $\boldsymbol{A}$. First, submit an empty query to the Oracle to obtain a vector $\boldsymbol{v}_0$, which as a result is uniformly distributed among $\mathcal{D}_\delta$. We then proceed to simulate the optimization procedure for alg using parameters $\boldsymbol{A}$ and $\boldsymbol{v}_0$ (lines 3-6 of Game 2). Precisely, whenever a new vector $\boldsymbol{v}_{p,l}$ needs to be defined according to the optimization procedure, the player submits the corresponding vectors $\boldsymbol{x}_{i_{p,1}}, \ldots, \boldsymbol{x}_{i_{p,r}}$ to the oracle and receives in return a vector which defines $\boldsymbol{v}_{p,l}$. In this manner, the player simulates exactly the optimization procedure. In all cases, the number of queries in this first phase is at most $1 + kp_{max} \leq d$. For the remaining queries to perform, the player can query whichever vectors, these will not be used in the rest of the strategy. If the simulation did not end period p_{max}, the complete procedure fails. We now describe the rest of the procedure when period p_{max} was ended. During the simulation, the algorithm records the time $i_{p,1}$ when period p started for all $p \leq p_{max}+1$. Recall that for $p_{max}+1$, we only define $i_{p_{max}+1,1}$, this is the time that ends period p_{max}. By hypothesis, $i_{p_{max}+1,1} \leq mp_{max}$. As a result, there must be a period $p \leq p_{max}$ which uses at most m queries: $i_{p+1,1} - i_{p,1} \leq m$. We define the memory $\mathsf{Message}$ to be the memory of alg just before starting iteration $i_{p,1}$, at the beginning of period p (line 7 of Game 2). Next, since the period p_{max} was ended, the vectors $\boldsymbol{v}_{p,l}$ for $p \leq p_{max}, l \leq l_p$ were all defined. The player can therefore submit the function $\boldsymbol{g}_{\boldsymbol{A},\boldsymbol{v}}$ to the

Oracle (line 8 of Game 2) as follows,

$$\boldsymbol{g}_{\boldsymbol{A},\boldsymbol{v}} : \boldsymbol{x} \mapsto \begin{cases} (\boldsymbol{g}_{\boldsymbol{A}}(\boldsymbol{x}), 1) & \text{if } F_{\boldsymbol{A},\boldsymbol{v}}(\boldsymbol{x}) = \|\boldsymbol{A}\boldsymbol{x}\|_\infty - \eta, \\ (\boldsymbol{v}_0, 2) & \text{otherwise and if } F_{\boldsymbol{A},\boldsymbol{v}}(\boldsymbol{x}) = \eta \boldsymbol{v}_0^\top \boldsymbol{x}, \\ (\boldsymbol{v}_{p,l}, 2 + (p-1)k + l) & \text{otherwise and if} \\ & (p, l) = \underset{(p',l') \leq_{lex} (p_{max}, l_{p_{max}})}{\arg\max} \boldsymbol{v}_{p',l'}^\top \boldsymbol{x} - p\gamma_1 - l\gamma_2. \end{cases} \tag{7}$$

Intuitively, the first component of $\boldsymbol{g}_{\boldsymbol{A},\boldsymbol{v}}$ gives the subgradient $\partial F_{\boldsymbol{A},\boldsymbol{v}}$ to the following two exceptions: we always return $\boldsymbol{a}_i$ instead of $\pm \boldsymbol{a}_i$ and we return $\boldsymbol{v}_0$ (resp. $\boldsymbol{v}_{p,l}$) instead of $\eta \boldsymbol{v}_0$ (resp. $\eta \boldsymbol{v}_{p,l}$). The second term of $\boldsymbol{g}_{\boldsymbol{A},\boldsymbol{v}}$ has values in $[2 + p_{max}k]$. Hence, since $2 + p_{max}k \leq d^2$, the function $\boldsymbol{g}_{\boldsymbol{A},\boldsymbol{v}}$ takes values in $(\{\boldsymbol{a}_j, j \leq n\} \cup \{\boldsymbol{v}_l, l \leq d\}) \times [d^2]$.

The strategy then proceeds to play the Orthogonal Vector Game in a second part (lines 9-12 of Game 2) and use the responses of the Oracle to simulate the run of alg for the optimization procedure in period p. To do so, we set the memory state of the algorithm alg to be Message. Then, for the next m iterations we proceed as follows. At iteration i of the process, we run alg with its current state to obtain a new query $\boldsymbol{z}_i$ which is then submitted to the oracle of the Orthogonal Vector Game, to get a response $(\boldsymbol{g}_i, s_i)$. We then use this response to simulate the response that was given by the optimization procedure in the first phase, computing $(v_i, \tilde{\boldsymbol{g}}_i)$ as follows

$$(v_i, \tilde{\boldsymbol{g}}_i) = \begin{cases} (|\boldsymbol{g}_i^\top \boldsymbol{z}_i| - \eta, sign(\boldsymbol{g}_i^\top \boldsymbol{z}_i)\boldsymbol{g}_i) & s_i = 1, \\ (\eta \boldsymbol{g}_i^\top \boldsymbol{z}_i, \eta \boldsymbol{g}_i) & s_i = 2, \\ (\eta(\boldsymbol{g}_i^\top \boldsymbol{z}_i - p\gamma_1 - l\gamma_2), \eta \boldsymbol{g}_i) & s_i = 2 + (p-1)k + l, p \leq p_{max}, 1 \leq l \leq k. \end{cases} \tag{8}$$

We can easily check that in all cases, $v_i = F_{\boldsymbol{A},\boldsymbol{v}}(\boldsymbol{z}_i)$ and that $\tilde{\boldsymbol{g}}_i = \partial F_{\boldsymbol{A},\boldsymbol{v}}(\boldsymbol{z}_i)$. We then pass $(\boldsymbol{v}_i, \tilde{\boldsymbol{g}}_i)$ as response to alg for the query $\boldsymbol{z}_i$ so it can update its state. Further, having defined $i_1 = 1$, the player can keep track of exploratory queries by checking whether

$$v_i \leq -\frac{\eta\gamma_1}{2} \quad \text{and} \quad \frac{\|P_{Span(\boldsymbol{z}_{i_{r'}}, r' \leq r)^\perp}(\boldsymbol{z}_i)\|}{\|\boldsymbol{z}_i\|} \geq \frac{\gamma_2}{4},$$

where $i_1, \ldots, i_r$ are the indices defined so far. We perform m such iterations unless alg stops and use the last remaining queries arbitrarily. Next, we check if the last index i_k was defined. If not, we pose $i_k = m + 1$ and let $\boldsymbol{z}_{m+1}$ be the next query of alg. The final returned vectors are $\frac{\boldsymbol{z}_{i_1}}{\|\boldsymbol{z}_{i_1}\|}, \ldots, \frac{\boldsymbol{z}_{i_k}}{\|\boldsymbol{z}_{i_k}\|}$. This ends the description of the player's strategy.

We now show that the player wins with good probability. First, since alg makes at most $mp_{max} \leq d^2$ queries, by Proposition 7, on an event $\mathcal{E}$ of probability at least $q - C\sqrt{\log d}/d$, alg succeeds and ends the period p_{max}. On $\mathcal{E}$, by construction, the first phase of the strategy does not fail. Next, we show that in the second phase (lines 9-12 of Game 2), the queried vectors coincide exactly with the queried vectors from the corresponding period p in the first phase (lines 3-6 of Game 2). To do so, we only need to check that the responses provided to alg coincide with the response given by the optimization procedure. First, recall that on $\mathcal{E}$, all periods are completed, hence $F_{\boldsymbol{A},\boldsymbol{v},P,L} = F_{\boldsymbol{A},\boldsymbol{v}}$. Next, by Proposition 6, the responses of the procedure are consistent with optimizing $F_{\boldsymbol{A},\boldsymbol{v},P,L}$ and subgradients $\partial F_{\boldsymbol{A},\boldsymbol{v},P,L}$ on an event $\mathcal{F}$ of probability at least $1 - C'\sqrt{\log d}/d^2$. Therefore, on $\mathcal{E} \cap \mathcal{F}$, it suffices to check that the responses provided to alg are consistent with $F_{\boldsymbol{A},\boldsymbol{v}}$,

Algorithm 3: Strategy of the Player for the Orthogonal Vector Game with Hints

Input: d, k, p_{max}, m, algorithm alg

Part 1: Strategy to store Message knowing $\boldsymbol{A}$

1 Initialize the memory of alg to be $\mathbf{0}$.
2 Submit $\emptyset$ to the Oracle and use the response as $\boldsymbol{v}_0$.
3 Run alg with the optimization procedure knowing $\boldsymbol{A}$ and $\boldsymbol{v}_0$ until the first exploratory query $\boldsymbol{x}_{i_{1,1}}$.
4 **for** $p \in [p_{max}]$ **do**
5   Let Memory_p be the current memory state of alg and $i_{p,1}$ the current iteration step.
6   Run alg with the feasibility procedure until period p ends at iteration step $i_{p+1,1}$. If alg stopped before, **return** the strategy fails. When needed to sample a unit vector $\boldsymbol{v}_{p',l'}$, submit vectors $\boldsymbol{x}_{i_{p',1}}, \ldots \boldsymbol{x}_{i_{p',r'}}$ to the Oracle where $i_{p',1}, \ldots, i_{p',r'}$ are the exploratory queries defined at that stage. We use the corresponding response of the Oracle as $\boldsymbol{v}_{p',l'}$.
7   **if** $i_{p+1,1} - i_{p,1} \le m$ **then**
8   Set $\mathsf{Message} = \mathsf{Memory}_p$
9 **end**
10 **for** *Remaining queries to perform to Oracle* **do** Submit arbitrary query, e.g. $\emptyset$;
11 **if** $\mathsf{Message}$ *has not been defined yet* **then return** The strategy fails;
12 Submit $\boldsymbol{g}_{\boldsymbol{A},\boldsymbol{v}}$ to the Oracle as defined in Eq (7).

Part 2: Strategy to make queries

13 Set the memory state of alg to be $\mathsf{Message}$ and define $i_1 = 1$, $r = 1$.
14 **for** $i \in [m]$ **do**
15   Run alg with current memory to obtain a query $\boldsymbol{z}_i$.
16   Submit $\boldsymbol{z}_i$ to the Oracle from Game 2, to get response $(\boldsymbol{g}_i, s_i)$.
17   Compute $(v_i, \tilde{\boldsymbol{g}}_i)$ using $\boldsymbol{z}_i$, $\boldsymbol{g}_i$ and s_i as defined in Eq (8) and pass $(v_i, \tilde{\boldsymbol{g}}_i)$ as response to alg.
18   **if** $v_i \le -\eta\gamma_1/2$ *and* $\|P_{Span(\boldsymbol{z}_{i_{r'}}, r' \le r)^\perp}(\boldsymbol{z}_i)\|/\|\boldsymbol{z}_i\| \ge \frac{\gamma_2}{4}$ **then**
19   Set $i_{r+1} = i$ and increment $r \leftarrow r + 1$.
20 **end**

Part 3: Strategy to return vectors

21 **if** *index* i_k *has not been defined yet* **then**
22   With the current memory of alg find a new query $\boldsymbol{z}_{m+1}$ and set $i_k = m + 1$.
23 **return** $\left\{ \frac{\boldsymbol{z}_{i_1}}{\|\boldsymbol{z}_{i_1}\|}, \ldots, \frac{\boldsymbol{z}_{i_k}}{\|\boldsymbol{z}_{i_k}\|} \right\}$ to the Oracle.

which we already noted: at every step i, $(v_i, \tilde{\boldsymbol{g}}_i) = (F_{\boldsymbol{A},\boldsymbol{v}}(\boldsymbol{z}_i), \partial F_{\boldsymbol{A},\boldsymbol{v}}(\boldsymbol{z}_i))$. This proves that the responses and queries coincide exactly with those given by the optimization procedure on $\mathcal{E} \cap \mathcal{F}$.

Next, by construction, the chosen phase p had at most m iterations. Thus, on $\mathcal{E} \cap \mathcal{F}$, among $\boldsymbol{z}_1, \ldots, \boldsymbol{z}_{m+1}$, we have the vectors $\boldsymbol{x}_{i_{p,1}}, \ldots, \boldsymbol{x}_{i_{p,k}}$. Further, if i_k was not defined during part 2 of the strategy, this means that $i_k = m+1$, as defined in the player's strategy (line 21-22 of Algorithm 3). As a result, for all $u \le k$, we have $\boldsymbol{z}_{i_u} = \boldsymbol{x}_{i_{p,u}}$. We now show that the returned vectors $\frac{\boldsymbol{x}_{i_{p,1}}}{\|\boldsymbol{x}_{i_{p,1}}\|}, \ldots, \frac{\boldsymbol{x}_{i_{p,k}}}{\|\boldsymbol{x}_{i_{p,k}}\|}$ are successful for Game 2. First, because $i_{p,1}, \ldots, i_{p,k}$ are exploratory queries, we have directly for $u \leq k$,

$$\frac{\|P_{Span(\boldsymbol{x}_{i_{p,v}}, v<u)^\perp}(\boldsymbol{x}_{i_{p,u}})\|}{\|\boldsymbol{x}_{i_{p,u}}\|} \geq \frac{\gamma_2}{4}.$$

Next, if l is the index of the last constructed vector $\boldsymbol{v}_{p,l}$ before $i_{p,u}$ in the optimization procedure, one has $F_{\boldsymbol{A},\boldsymbol{v},p,l}(\boldsymbol{x}_{i_{p,u}}) \leq -\eta\gamma_1/2$. Therefore, $\|\boldsymbol{A}\boldsymbol{x}_{i_{p,u}}\|_\infty \leq F_{\boldsymbol{A},\boldsymbol{v},p,l}(\boldsymbol{x}_{i_{p,u}}) + \eta \leq \eta$. Further, $\eta \boldsymbol{v}_0^\top \boldsymbol{x}_{i_{p,u}} \leq F_{\boldsymbol{A},\boldsymbol{v},p,l}(\boldsymbol{x}_{i_{p,u}}) \leq -\eta\gamma_1/2$. This proves that $\|\boldsymbol{x}_{i_{p,u}}\| \geq \gamma_1/2$. Putting the previous two inequalities together yields

$$\frac{\|\boldsymbol{A}\boldsymbol{x}_{i_{p,u}}\|_\infty}{\|\boldsymbol{x}_{i_{p,u}}\|} \leq \frac{2\eta}{\gamma_1}.$$

As a result, this shows that the returned vectors are successful for Game 2 for the desired parameters $\alpha = 2\eta/\gamma_1$ and $\beta = \gamma_2/4$. Thus, the player wins on $\mathcal{E} \cap \mathcal{F}$, which has probability at least $q - (C + C')\sqrt{\log d}/d^2$ by the union bound. This ends the proof of the proposition. ■

A.4. Query lower bound for the Orthogonal Vector Game with Hints

Before proving a lower bound on the necessary number of queries for Game 2, we need to introduce two results. The first one is a known concentration result for vectors in the hypercube. It shows that for a uniform vector in the hypercube, being approximately orthogonal to k orthonormal vectors has exponentially small probability in k.

Lemma 13 ([23]) *Let $\boldsymbol{h} \sim \mathcal{U}(\{\pm 1\}^d)$. Then, for any $t \in (0, 1/2]$ and any matrix $\boldsymbol{Z} = [\boldsymbol{z}_1, \ldots, \boldsymbol{z}_k] \in \mathbb{R}^{d\times k}$ with orthonormal columns,*

$$\mathbb{P}(\|\boldsymbol{Z}^\top \boldsymbol{h}\|_\infty \leq t) \leq 2^{-c_H k}.$$

We will also need an anti-concentration bound for random vectors, which intuitively provides a lower bound for the previous concentration result. The following lemma shows that for a uniformly random unit vector, being orthogonal to k orthonormal vectors is still achievable with exponentially small probability in k.

Lemma 14 *Let $k < d$ and $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k$ be k orthonormal vectors. Then,*

$$\mathbb{P}_{\boldsymbol{y}\sim\mathcal{U}(S^{d-1})}\left(|\boldsymbol{x}_i^\top \boldsymbol{y}| \leq \frac{1}{d^3}, \forall i \leq k\right) \geq \frac{1}{e^{d^{-4}} d^{3k}}.$$

Proof Let $\boldsymbol{y} \sim \mathcal{U}(S^{d-1})$ be a uniformly random unit vector. Then, for $i < k$ and any $y_1, \ldots, y_{i-1}$ such that $|y_1|, \ldots, |y_{i-1}| \leq \frac{1}{d^3}$, we have

$$\begin{aligned}
\mathbb{P}\left(|y_i| \leq \frac{1}{d^3} \mid y_1, \ldots, y_{i-1}\right) &= \mathbb{P}_{\boldsymbol{u}\sim\mathcal{U}(S^{d-i})}\left(|u_1| \leq \frac{1}{d^3\sqrt{1-(y_1^2+\ldots+y_{i-1}^2)}}\right) \\
&\geq \frac{\int_0^{1/d^3}(1-y^2)^{(d-i-1)/2}dy}{\int_0^1 (1-y^2)^{(d-i-1)/2}dy} \\
&\geq \frac{(1-d^{-6})^{d/2}}{d^3} \geq \frac{e^{-d^{-5}}}{d^3},
\end{aligned}$$

where in the last equation we used $d \geq 2$. Therefore, we can show by induction that $\mathbb{P}(|y_i| \leq 1/d^3, \forall i \leq k) \geq \frac{e^{-kd^{-5}}}{d^{3k}}$. Thus, by isometry this shows that

$$\mathbb{P}\left(|\boldsymbol{x}_i^\top \boldsymbol{y}| \leq \frac{1}{d^3}, \forall i \leq k\right) \geq \frac{1}{e^{d^{-4}} d^{3k}}.$$

This ends the proof of the lemma. ■

We are now ready to prove the query lower bound for Game 2 given in Proposition 9. Precisely, we show that for an appropriate choice of parameters, one needs $m = \tilde{\Omega}(d)$ queries. The proof is closely inspired from the arguments given in [23]. The main added difficulty arises from bounding the information leakage of the provided hints. As such, our goal is to show that these do not provide more information than the message itself.

Proof of Proposition 9 We first define some notations. Let $\boldsymbol{Y} = [\boldsymbol{y}_1, \ldots, \boldsymbol{y}_k]$ be the matrix storing the final outputs from the algorithm. Next, for the responses of the oracle $(\boldsymbol{g}_1, s_1), \ldots, (\boldsymbol{g}_m, s_m)$, we first store all the scalar responses in a vector $\boldsymbol{c} = [s_1, \ldots, s_m]$. We then focus on the responses $\boldsymbol{g}_1, \ldots, \boldsymbol{g}_m$. Let $\tilde{\boldsymbol{G}}$ denote the matrix containing these responses of the oracle which are lines of $\boldsymbol{A}$. Let $\boldsymbol{G}$ be the matrix containing unique columns from $\tilde{\boldsymbol{G}}$, augmented with rows of $\boldsymbol{A}$ so that it has exactly m columns which are all different rows of $\boldsymbol{A}$. Last, let $\boldsymbol{A}'$ be the matrix $\boldsymbol{A}$ once the rows from $\boldsymbol{G}$ are removed. Next, let $\tilde{\boldsymbol{V}}$ be a matrix containing the responses of the oracle which are vectors $\boldsymbol{v}_l$, ordered by increasing index l. As before, let $\boldsymbol{V}$ be the matrix $\tilde{\boldsymbol{V}}$ where we only conserve unique columns and append it with additional vectors $\boldsymbol{v}_l$ so that $\boldsymbol{V}$ has exactly m columns. We denote by $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_m$ these vectors, and recall that they are vectors $\boldsymbol{v}_l$ ordered by increasing order of index l. Last, we define a vector $\boldsymbol{j}$ of indices such that $j(i)$ contains the information of which column of the matrices $\boldsymbol{G}$ or $\boldsymbol{V}$ corresponds $\boldsymbol{g}_i$. Precisely, if $\boldsymbol{g}_i$ is a line $\boldsymbol{a}$ from $\boldsymbol{A}$, we set $j(i) = j$ where j is the index of the column from $\boldsymbol{G}$ corresponding to $\boldsymbol{a}$. Otherwise, if j is the index of the column from $\boldsymbol{V}$ corresponding to $\boldsymbol{g}_i$, we set $j(i) = m + j$.

Next, we argue that $\boldsymbol{Y}$ is a deterministic function of $\mathsf{Message}$, the matrices $\boldsymbol{G}, \boldsymbol{V}$ and the vector of indices $\boldsymbol{j}$ and $\boldsymbol{c}$. First, $\boldsymbol{c}$ provides the scalar responses directly. For the d-dimensional component of the responses, first, note that from $\boldsymbol{G}$, $\boldsymbol{V}$ and $\boldsymbol{j}$ one can easily recover the vectors $\boldsymbol{g}_1, \ldots, \boldsymbol{g}_m$. Next, using the algorithm for the second section of the Orthogonal Vector Game with Hints set with initial memory $\mathsf{Message}$ and the vectors $\boldsymbol{g}_1, \ldots, \boldsymbol{g}_m$ as responses of the oracle, one can inductively compute the queries $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m$. Last, $\boldsymbol{Y}$ is a deterministic function of $\boldsymbol{x}_i, \boldsymbol{g}_i, i \in [m]$ and $\mathsf{Message}$. This ends the claim that there is a function ϕ such that $\boldsymbol{Y} = \phi(\mathsf{Message}, \boldsymbol{G}, \boldsymbol{V}, \boldsymbol{j}, \boldsymbol{c})$. By the data processing inequality,

$$I(\boldsymbol{A}'; \boldsymbol{Y} \mid \boldsymbol{G}, \boldsymbol{V}, \boldsymbol{j}, \boldsymbol{c}) \leq I(\boldsymbol{A}'; \mathsf{Message} \mid \boldsymbol{G}, \boldsymbol{V}, \boldsymbol{j}, \boldsymbol{c}) \leq H(\mathsf{Message} \mid \boldsymbol{G}, \boldsymbol{V}, \boldsymbol{j}, \boldsymbol{c}) \leq M. \quad (9)$$

In the last inequality we used the fact that $\mathsf{Message}$ uses at most M bits. We have that

$$I(\boldsymbol{A}'; \boldsymbol{Y} \mid \boldsymbol{G}, \boldsymbol{V}, \boldsymbol{j}, \boldsymbol{c}) = H(\boldsymbol{A}' \mid \boldsymbol{G}, \boldsymbol{V}, \boldsymbol{j}, \boldsymbol{c}) - H(\boldsymbol{A}' \mid \boldsymbol{Y}, \boldsymbol{G}, \boldsymbol{V}, \boldsymbol{j}, \boldsymbol{c}). \quad (10)$$

In the next steps we bound the two terms. We start with the second term of the right hand side of Eq (10) using similar arguments to the proof given in [23]. Let $\mathcal{E}$ be the event when the Player succeeds at Game 2. Consider the case when $\boldsymbol{Y}$ is a winning matrix. Then we have $\|\boldsymbol{A}\boldsymbol{y}_i\|_\infty \leq \alpha$

for all $i \leq k$. As a result, any line $\boldsymbol{a}$ of $\boldsymbol{A}'$ satisfies $\|\boldsymbol{Y}^\top \boldsymbol{a}\|_\infty \leq \alpha$. Further, we have that $\|P_{Span(\boldsymbol{y}_j, j<i)^\perp}(\boldsymbol{y}_i)\| \leq \beta$ for all $i \leq k$. By Lemma 22, there exist $\lceil k/5 \rceil$ orthonormal vectors $\boldsymbol{Z} = [\boldsymbol{z}_1, \ldots, \boldsymbol{z}_{\lceil k/5 \rceil}]$ such that for any $\boldsymbol{x} \in \mathbb{R}^d$ one has $\|\boldsymbol{Z}^\top \boldsymbol{x}\|_\infty \leq \left(\frac{\sqrt{d}}{\beta}\right)^{5/4} \|\boldsymbol{Y}^\top \boldsymbol{x}\|_\infty$. In particular, all lines $\boldsymbol{a}$ of $\boldsymbol{A}'$ satisfy

$$\|\boldsymbol{Z}^\top \boldsymbol{a}\|_\infty \leq \left(\frac{\sqrt{d}}{\beta}\right)^{5/4} \alpha \leq \frac{1}{2},$$

where we used the hypothesis in the parameters α and β. By Lemma 13, one has

$$\left|\left\{\boldsymbol{a} \in \{\pm 1\}^d : \|\boldsymbol{Z}^\top \boldsymbol{a}\|_\infty \leq \frac{1}{2}\right\}\right| \leq 2^d \mathbb{P}_{\boldsymbol{h} \sim \mathcal{U}(\{\pm 1\}^d)} \left(\|\boldsymbol{Z}^\top \boldsymbol{h}\|_\infty \leq \frac{1}{2}\right) \leq 2^{d - c_H \lceil k/5 \rceil}.$$

Therefore, we proved that if $\boldsymbol{Y}'$ is a winning vector, $H(\boldsymbol{A}' \mid \boldsymbol{Y} = \boldsymbol{Y}') \leq (n-m)(d - c_H k/5)$. Otherwise, if $\boldsymbol{Y}'$ loses, we can directly use $H(\boldsymbol{A}' \mid \boldsymbol{Y} = \boldsymbol{Y}') \leq (n-m)d$. Combining these equations gives

$$\begin{aligned} H(\boldsymbol{A}' \mid \boldsymbol{Y}, \boldsymbol{G}, \boldsymbol{V}, \boldsymbol{j}, \boldsymbol{c}) &\leq H(\boldsymbol{A}' \mid \boldsymbol{Y}) \\ &\leq \mathbb{P}(\mathcal{E}^c)(n-m)d + \mathbb{P}(\mathcal{E})(n-m)(d - c_H k/5) \\ &\leq (n-m)(d - \mathbb{P}(\mathcal{E}) c_H k/5). \end{aligned}$$

Next, we turn to the first term of the right-hand side of Eq (10).

$$\begin{aligned} H(\boldsymbol{A}' \mid \boldsymbol{G}, \boldsymbol{V}, \boldsymbol{j}, \boldsymbol{c}) = H(\boldsymbol{A} \mid \boldsymbol{G}, \boldsymbol{V}, \boldsymbol{j}, \boldsymbol{c}) &= H(\boldsymbol{A} \mid \boldsymbol{V}) - I(\boldsymbol{A}; \boldsymbol{G}, \boldsymbol{j}, \boldsymbol{c} \mid \boldsymbol{V}) \\ &\geq H(\boldsymbol{A} \mid \boldsymbol{V}) - H(\boldsymbol{G}, \boldsymbol{j}, \boldsymbol{c}) \\ &\geq H(\boldsymbol{A} \mid \boldsymbol{V}) - md - m \log(2m) - m \log(d^2) \\ &= H(\boldsymbol{A}) - I(\boldsymbol{A}; \boldsymbol{V}) - md - 3m \log(2d) \\ &= (n-m)d - 3m \log(2d) - I(\boldsymbol{A}; \boldsymbol{V}). \end{aligned}$$

In the second inequality, we use the fact that $\boldsymbol{G}$ uses md bits and $\boldsymbol{j}$ can be stored with $m \log(2m)$ bits. By the chain rule,

$$I(\boldsymbol{A}; \boldsymbol{V}) = \sum_{i \leq m} I(\boldsymbol{A}; \boldsymbol{w}_i \mid \boldsymbol{w}_1, \ldots, \boldsymbol{w}_{i-1}).$$

Next, if $\boldsymbol{w}_i = \boldsymbol{v}_l$, recalling that the vectors $\boldsymbol{w}_{i'} = \boldsymbol{v}_{l'}$ are ordered by increasing index of l', we have

$$\begin{aligned} I(\boldsymbol{A}; \boldsymbol{w}_i \mid \boldsymbol{w}_1, \ldots, \boldsymbol{w}_{i-1}) &= H(\boldsymbol{w}_i \mid \boldsymbol{w}_1, \ldots, \boldsymbol{w}_{i-1}) - H(\boldsymbol{w}_i \mid \boldsymbol{A}, \boldsymbol{w}_1, \ldots, \boldsymbol{w}_i) \\ &\leq H(\boldsymbol{w}_i) - H(\boldsymbol{w}_i \mid \boldsymbol{A}, \boldsymbol{w}_1, \ldots, \boldsymbol{w}_i, \boldsymbol{x}_{l,1}, \ldots, \boldsymbol{x}_{l,r_l}) \\ &= \log |\mathcal{D}_\delta| - H(\boldsymbol{w}_i \mid \boldsymbol{x}_{l,1}, \ldots, \boldsymbol{x}_{l,r_l}). \end{aligned}$$

In the last equality, we used the fact that if $\boldsymbol{b}_{l,1}, \ldots, \boldsymbol{b}_{l,r_l}$ are the resulting vectors from the Gram-Schmidt decomposition of $\boldsymbol{x}_{l,1}, \ldots, \boldsymbol{x}_{l,r_l}$, $\boldsymbol{y}_l$ is generated uniformly in $S^{d-1} \cap \{\boldsymbol{y} : \forall r \leq r_l, |\boldsymbol{b}_{l,r}^\top \boldsymbol{y}| \leq d^{-3}\}$ independently from the past history, and $\boldsymbol{v}_l = \phi_\delta(\boldsymbol{y}_l)$. By Lemma 14, we know that

$$\mathbb{P}_{\boldsymbol{z} \sim \mathcal{U}(S^{d-1})} \left(\forall r \leq r_l, |\boldsymbol{b}_{l,r}^\top \boldsymbol{z}| \leq d^{-3}\right) \geq \frac{1}{e^{d^{-4}} d^{3k}}.$$

As a result, for any $\boldsymbol{b}_j(\delta) \in \mathcal{D}_\delta$, one has

$$\mathbb{P}(\boldsymbol{w}_i = \boldsymbol{b}_j(\delta) \mid \boldsymbol{x}_{l,1}, \ldots, \boldsymbol{x}_{l,r_l}) \leq \frac{\mathbb{P}_{\boldsymbol{z} \sim \mathcal{U}(S^{d-1})}(\boldsymbol{z} \in V_j(\delta))}{\mathbb{P}_{\boldsymbol{z} \sim \mathcal{U}(S^{d-1})}\left(\forall r \leq r_l, |\boldsymbol{b}_{l,r}^\top \boldsymbol{z}| \leq d^{-3}\right)} \leq \frac{e^{d^{-4}} d^{3k}}{|\mathcal{D}_\delta|},$$

where we used the fact that each cell has the same area. In particular, this shows that

$$H(\boldsymbol{w}_i \mid \boldsymbol{x}_{l,1}, \ldots, \boldsymbol{x}_{l,r_l}) = \mathbb{E}_{\boldsymbol{b} \sim \boldsymbol{w}_i | \boldsymbol{x}_{l,1}, \ldots, \boldsymbol{x}_{l,r_l}}[-\log p_{\boldsymbol{w}_i | \boldsymbol{x}_{l,1}, \ldots, \boldsymbol{x}_{l,r_l}}(\boldsymbol{b})] \geq \log\left(\frac{|\mathcal{D}_\delta|}{e^{d^{-4}} d^{3k}}\right).$$

Hence,

$$I(\boldsymbol{A}; \boldsymbol{w}_i \mid \boldsymbol{w}_1, \ldots, \boldsymbol{w}_{i-1}) \leq 3k \log d + d^{-4} \log e.$$

Putting everything together gives

$$\begin{aligned} I(\boldsymbol{A}'; \boldsymbol{Y} \mid \boldsymbol{G}, \boldsymbol{V}, \boldsymbol{j}) &\geq (n-m)d - 3m\log(2d) - 3km\log d - 2md^{-4} - (n-m)(d - \mathbb{P}(\mathcal{E})c_H k/5) \\ &\geq \frac{c_H}{10} k(n-m) - 3km\log d - 1 - 3d\log(2d), \end{aligned}$$

where in the last equation we used $d \geq 2$. Together with Eq (9), this implies

$$m \geq \frac{c_H kn/10 - M - 1 - 3d\log(2d)}{k(3\log d + c_H/10)}.$$

As a result, since $k \geq 20\frac{M + 3d\log(2d) + 1}{c_H n}$ and $n \geq d/4$, we obtain

$$m \geq \frac{c_H n}{60\log d + 2c_H} \geq \frac{c_H}{8(30\log d + c_H)} d.$$

This ends the proof of the proposition. ■

Appendix B. Memory-constrained feasibility problem

In this section, we prove the lower bound from Theorem 2 for the feasibility problem.

B.1. Defining the feasibility procedure

Similarly to Section 3, we pose $n = \lceil d/4 \rceil$. Also, for any matrix $\boldsymbol{A} \in \{\pm 1\}^{n \times d}$, we use the same functions $\boldsymbol{g}_{\boldsymbol{A}}$ and $\tilde{\boldsymbol{g}}_{\boldsymbol{A}}$. We use similar techniques as those we introduced for the optimization problem. However, since in this case, the separation oracle only returns a separating hyperplane, without any value considerations of an underlying function, Procedure 1 can be drastically simplified, which leads to improved lower bounds.

Let $\eta_0 = 1/(24d^2)$, $\eta_1 = \frac{1}{2\sqrt{d}}$, $\delta = 1/d^3$, and $k \leq d/3 - n$ be a parameter. Last, let $p_{max} = \lfloor (c_{d,1}d - 1)/(k-1) \rfloor$, where $c_{d,1}$ is the same quantity as in Eq (3). The feasibility procedure is defined in Procedure 4. The oracle first randomly samples $\boldsymbol{A} \sim \mathcal{U}(\{\pm 1\}^{n \times d})$ and $\boldsymbol{v}_0 \sim \mathcal{U}(\mathcal{D}_\delta)$. This matrix and vector are then fixed in the rest of the procedure. Whenever the player queries a point $\boldsymbol{x}$ such that $\|\boldsymbol{A}\boldsymbol{x}\|_\infty > \eta_0$ (resp. $\boldsymbol{v}_0^\top \boldsymbol{x} > -\eta_1$), the oracle returns $\tilde{\boldsymbol{g}}_{\boldsymbol{A}}(\boldsymbol{x})$ (resp. $\boldsymbol{v}_0$). All other queries are called *informative* queries. With this definition, it remains to define the separation oracle on informative queries. The oracle proceeds by periods in which the behavior is different. In each period p, the oracle constructs vectors $\boldsymbol{v}_{p,1}, \ldots, \boldsymbol{v}_{p,k-1}$ inductively and keeps in memory some queries $i_{p,1}, \ldots, i_{p,k}$ that will be called *exploratory*. The first informative query t will be the first exploratory query and starts period 1.

Given a new query $\boldsymbol{x}_t$,

1. If $\|\boldsymbol{A}\boldsymbol{x}\|_\infty > \eta_0$, the oracle returns $\tilde{\boldsymbol{g}}_{\boldsymbol{A}}(\boldsymbol{x}_t)$.
2. If $\boldsymbol{v}_0^\top \boldsymbol{x}_t > -\eta_1$, the oracle returns $\boldsymbol{v}_0$.
3. If $\boldsymbol{x}_t$ was queried in the past sequence, the oracle returns the same vector that was returned previously.
4. Otherwise, let p be the index of the current period and let $\boldsymbol{v}_{p,1}, \ldots, \boldsymbol{v}_{p,l}$ be the vectors from the current period constructed so far, together with their corresponding exploratory queries $i_{p,1} \ldots, i_{p,l} < t$. Potentially, if $p = 1$ one may not have defined any such vectors at the beginning of time t. In this case, let $l = 0$.
   (a) If $\max_{1 \leq l' \leq l} \boldsymbol{v}_{p,l'}^\top \boldsymbol{x}_t > -\eta_1$ (with the convention $\max_\emptyset = -\infty$), the oracle returns $\boldsymbol{v}_{p,l'}$ where $l' = \arg\max_{l \leq r} \boldsymbol{v}_{p,l}^\top \boldsymbol{x}_t$. Ties are broken alphabetically.
   (b) Otherwise, if $l < k-1$, we first define $i_{p,l+1} = t$. Then, let $\boldsymbol{b}_{p,1}, \ldots, \boldsymbol{b}_{p,l+1}$ be the result from the Gram-Schmidt decomposition of $\boldsymbol{x}_{i_{p,1}}, \ldots, \boldsymbol{x}_{i_{p,l+1}}$ and let $\boldsymbol{y}_{p,l+1}$ be a sample of the distribution obtained by the uniform distribution $\boldsymbol{y}_{p,l+1} \sim \mathcal{U}(S^{d-1} \cap \{\boldsymbol{z} \in \mathbb{R}^d : |\boldsymbol{b}_{p,r}^\top \boldsymbol{z}| \leq \frac{1}{d^3}, \forall r \leq l+1\})$. We then pose $\boldsymbol{v}_{p,l+1} = \phi_\delta(\boldsymbol{y}_{p,l+1})$. Having defined this new vector, the oracle returns $\boldsymbol{v}_{p,l}$. We then increment l.
   (c) Otherwise, if $r = k$, we define $i_{p,k} = i_{p+1,1} = t$. If $p + 1 \leq p_{max}$, this starts the next period $p+1$. As above, let $\boldsymbol{b}_{p+1,1}$ be the result of the Gram-Schmidt decomposition of $\boldsymbol{x}_{i_{p+1,1}}$ and sample $\boldsymbol{y}_{p+1,1}$ according to a uniform $\boldsymbol{y}_{p+1,1} \sim \mathcal{U}(S^{d-1} \cap \{\boldsymbol{z} \in \mathbb{R}^d : |\boldsymbol{b}_{p+1,1}^\top \boldsymbol{z}| \leq \frac{1}{d^3}\})$. We then pose $\boldsymbol{v}_{p+1,1} = \phi_\delta(\boldsymbol{y}_{p+1,1})$ and the oracle returns $\boldsymbol{v}_{p+1,1}$. We can then increment p and reset $l = 1$.

The above construction ends when the period p_{max} is finished. At this point, the oracle has defined the vectors $\boldsymbol{v}_{p,l}$ for all $p \leq p_{max}$ and $l \leq k$. We then define the successful set as

$$Q_{\boldsymbol{A},\boldsymbol{v}} = \left\{ \boldsymbol{x} \in B_d(\boldsymbol{0}, 1) : \|\boldsymbol{A}\boldsymbol{x}\|_\infty \leq \eta_0, \boldsymbol{v}_0^\top \boldsymbol{x} \leq -\eta_1, \max_{p \leq p_{max}, l \leq k-1} \boldsymbol{v}_{p,l}^\top \boldsymbol{x} \leq -\eta_1 \right\}.$$

From now on, the procedure uses any separation oracle for $Q_{\boldsymbol{A},\boldsymbol{v}}$ as responses to the algorithm, while making sure to be consistent with previous oracle reponses if a query is exactly duplicated. We next define what we mean by solving the above feasibility procedure.

Definition 15 *Let alg be an algorithm for the feasibility problem. When running alg with the responses of the feasibility procedure, we denote by $\boldsymbol{v}$ the set of constructed vectors and $\boldsymbol{x}^\star(alg)$ the final answer returned by alg. We say that an algorithm alg is successful for the feasibility procedure with probability $q \in [0,1]$, if taking $\boldsymbol{A} \sim \mathcal{U}(\{\pm 1\}^{n \times d})$, with probability at least q over the randomness of $\boldsymbol{A}$ and of the procedure, $\boldsymbol{x}^\star(alg) \in Q_{\boldsymbol{A},\boldsymbol{v}}$.*

In the rest of this section, we first relate this feasibility procedure to the standard feasibility problem, then prove query lower bounds to solve the feasibility procedure.

Procedure 4: The feasibility procedure for algorithm alg

Input: d, k, p_{max}, algorithm alg

Sample $\boldsymbol{A} \sim \mathcal{U}(\{\pm 1\}^{n\times d})$ and $\boldsymbol{v}_0 \sim \mathcal{U}(\mathcal{D}_\delta)$.
Initialize the memory of alg to **0** and let $p = 1$, $l = 0$.
for $t \geq 1$ **do**
  Run alg with current memory to obtain a query $\boldsymbol{x}_t$
  if $\|\boldsymbol{A}\boldsymbol{x}_t\| > \eta_0$ **then return** $\tilde{\boldsymbol{g}}_{\boldsymbol{A}}(\boldsymbol{x}_t)$ as response to alg ;
  else if $\boldsymbol{v}_0^\top \boldsymbol{x}_t > -\eta_1$ **then return** $\boldsymbol{v}_0$ as response to alg ;
  else if *Query $\boldsymbol{x}_t$ was made in the past* **then return** the same vector that was returned for $\boldsymbol{x}_t$;
  else
    if $\max_{1\leq l'\leq l} \boldsymbol{v}_{p,l'}^\top \boldsymbol{x}_t > -\eta_1$ **then**
      return $\boldsymbol{v}_{p,l'}$ where $l' = \arg\max_{l\leq r} \boldsymbol{v}_{p,l}^\top \boldsymbol{x}_t$.
    else if $l < k-1$ **then**
      Let $i_{p,l+1} = t$ and compute Gram-Schmidt decomposition $\boldsymbol{b}_{p,1}, \ldots, \boldsymbol{b}_{p,l+1}$ of $\boldsymbol{x}_{i_{p,1}}, \ldots, \boldsymbol{x}_{i_{p,l+1}}$.
      Sample $\boldsymbol{y}_{p,l+1}$ uniformly on $\mathcal{S}^{d-1} \cap \{\boldsymbol{z} \in \mathbb{R}^d : |\boldsymbol{b}_{p,l'}^\top \boldsymbol{z}| \leq d^{-3}, \forall l' \leq l+1\}$ and define $\boldsymbol{v}_{p,l+1} = \phi_\delta(\boldsymbol{y}_{p,l+1})$.
      return $\boldsymbol{v}_{p,l+1}$ as response to alg and increment $l \leftarrow l+1$.
    else if $p + 1 \leq p_{max}$ **then**
      Set $i_{p,k} =, i_{p+1,1} = t$ and compute the Gram-Schmidt decomposition $\boldsymbol{b}_{p+1,1}$ of $\boldsymbol{x}_{i_{p+1,1}}$.
      Sample $\boldsymbol{y}_{p+1,1}$ uniformly on $\mathcal{S}^{d-1} \cap \{\boldsymbol{z} \in \mathbb{R}^d : |\boldsymbol{b}_{p+1,1}^\top \boldsymbol{z}| \leq d^{-3}\}$ and define $\boldsymbol{v}_{p+1,1} = \phi_\delta(\boldsymbol{y}_{p+1,1})$.
      return $\boldsymbol{v}_{p+1,1}$ as response to alg, increment $p \leftarrow p+1$ and reset $l = 1$.
    else Set $i_{p_{max},k} = t$ and break the **for** loop;
end
for $t' \geq t$ **do** Use any separation oracle for $Q_{\boldsymbol{A},\boldsymbol{v}}$ consistent with previous responses ;

B.2. Reduction from the feasibility problem to the feasibility procedure

In the next proposition, we check that the above procedure indeed corresponds to a valid feasibility problem.

Proposition 16 *On an event of probability at least $1 - C\sqrt{\log d}/d$, the procedure described above is a valid feasibility problem. More precisely, the following hold.*

- *There exists $\bar{\boldsymbol{x}} \in B_d(\mathbf{0}, 1)$ such that $\|\boldsymbol{A}\bar{\boldsymbol{x}}\|_\infty = 0$, $\boldsymbol{v}_0^\top \bar{\boldsymbol{x}} \leq -4\eta_1$, and*

$$\max_{p\leq p_{max}, l\leq k-1} \boldsymbol{v}_{p,l}^\top \bar{\boldsymbol{x}} \leq -4\eta_1.$$

- *Let $\epsilon = \min\{\eta_0/\sqrt{d}, \eta_1\}/2$. Then, $B_d\left(\bar{\boldsymbol{x}} - \epsilon\frac{\bar{\boldsymbol{x}}}{\|\bar{\boldsymbol{x}}\|}, \epsilon\right) \subseteq B_d(\mathbf{0}, 1) \cap B_d(\bar{\boldsymbol{x}}, 2\epsilon) \subseteq Q_{\boldsymbol{A},\boldsymbol{v}}$.*

- *Throughout the run of the feasibility problem, the separation oracle always returned a valid cut, i.e., for any iteration t, if $\boldsymbol{x}_t$ denotes the query and $\boldsymbol{g}_t$ is the returned vector from the oracle, one has*

$$\forall \boldsymbol{x} \in Q_{\boldsymbol{A},\boldsymbol{v}}, \quad \langle \boldsymbol{g}_t, \boldsymbol{x}_t - \boldsymbol{x}\rangle > 0.$$

Further, responses are consistent: if $\boldsymbol{x}_t = \boldsymbol{x}_{t'}$, the responses of the procedure at times t and t' coincide.

We use a similar proof to that of Proposition 12.

Proof For convenience, we rename $\boldsymbol{v}_{p,l} = \boldsymbol{v}_{(p-1)(k-1)+l}$. Also, let $l_{max} = p_{max}(k-1) \leq c_{d,1}d-1$. Next, let $C_d = \sqrt{40 l_{max} \log d}$. We define the vector

$$\bar{\boldsymbol{x}} = -\frac{1}{C_d}\sum_{l=0}^{l_{max}} P_{Span(\boldsymbol{a}_i, i\leq n)^\perp}(\boldsymbol{v}_l).$$

Since $l_{max} \leq p_{max}(k-1) \leq c_{d,1}d - 1$, the same arguments as in the proof of Proposition 12 show that on an event $\mathcal{E}$ of probability at least $1 - C\sqrt{\log d}/d$, we have $\|\bar{\boldsymbol{x}}\| \leq 1$ and

$$\max_{0\leq l\leq l_{max}} \boldsymbol{v}_l^\top \bar{\boldsymbol{x}} \leq -\frac{1}{40\sqrt{(l_{max}+1)\log d}} \leq -\frac{2}{\sqrt{d}} = -4\eta_1,$$

where in the second inequality we used $l_{max} \leq c_{d,1}d - 1$. By construction, one has $\|\boldsymbol{A}\bar{\boldsymbol{x}}\|_\infty = 0$. This ends the proof of the first claim of the proposition. We then turn to the second claim, which is immediate from the fact that $\boldsymbol{x} \mapsto \|\boldsymbol{A}\boldsymbol{x}\|_\infty$ is $\sqrt{d}$-Lipschitz and both $\boldsymbol{x} \mapsto \boldsymbol{v}_0^\top \boldsymbol{x}$ and $\boldsymbol{x} \mapsto \max_{p\leq p_{max}, l\leq k} \boldsymbol{v}_{p,l}^\top \boldsymbol{x}$ are 1-Lipschitz. Therefore, $B_d(\bar{\boldsymbol{x}} - \epsilon\bar{\boldsymbol{x}}/\|\bar{\boldsymbol{x}}\|, \epsilon) \subseteq B_d(\boldsymbol{0}, 1) \cap B_d(\bar{\boldsymbol{x}}, 2\epsilon) \subset Q_{\boldsymbol{A},\boldsymbol{v}}$. It remains to check that the third claim is satisfied. It suffices to check that this is the case during the construction phase of the feasibility procedure. By construction of $Q_{\boldsymbol{A},\boldsymbol{v}} \subset \{\boldsymbol{x} : \|\boldsymbol{A}\boldsymbol{x}\|_\infty \leq \eta_0\}$.

Hence, it suffices to check that for informative queries $\boldsymbol{x}_t$, the returned vectors $\boldsymbol{g}_t$ are valid separation hyperplanes. By construction, these can only be either $\boldsymbol{v}_0$ or $\boldsymbol{v}_{p,l}$ for $p \leq p_{max}, l \leq k-1$. We denote by $\boldsymbol{w}$ this vector. Let t' be the first time $\boldsymbol{x}_t$ was queried. There are two cases. Either $\boldsymbol{w}$ was not constructed at time t', in which case, by construction this means that we are in scenario (2) or (4a). Both cases imply $\boldsymbol{w}^\top \boldsymbol{x}_t > -\eta_1$. Hence, $\boldsymbol{w}$ which is returned by the procedure is a valid separation hyperplane. Now suppose that $\boldsymbol{w} = \boldsymbol{v}_{p,l}$ was constructed at time t'—scenarios (4b) or (4c). By construction, one has $|\boldsymbol{b}_{p,r}^\top \boldsymbol{y}_{p,l}| \leq d^{-3}$ for all $r \leq l$. Decomposing $\boldsymbol{x}_t = \boldsymbol{x}_{i_{p,l}} = \alpha \boldsymbol{b}_{p,1} + \ldots + \alpha_l \boldsymbol{b}_{p,l}$, we obtain

$$|\boldsymbol{x}_t^\top \boldsymbol{y}_{p,l}| \leq \frac{\|\boldsymbol{\alpha}\|_1}{d^3} \leq \frac{1}{d^2\sqrt{d}}.$$

As a result, $\boldsymbol{y}_{p,l}^\top \boldsymbol{x}_t \geq -1/(d^2\sqrt{d})$. Because $\boldsymbol{v}_{p,l} = \phi_\delta(\boldsymbol{y}_{p,l})$, we have $\|\boldsymbol{v}_{p,l} - \boldsymbol{y}_{p,l}\| \leq \delta$. Hence, for any $d \geq 2$,

$$\boldsymbol{w}^\top \boldsymbol{x}_t \geq -1/(d^2\sqrt{d}) - \delta > -\eta_1.$$

Hence, $\boldsymbol{w}$ was a valid separation hyperplane. The last claim that the responses of the procedure are consistent over time is a direct consequence from its construction. This ends the proof of the proposition. ■

As a simple consequence of this result, solving the feasibility problem is harder than solving the feasibility procedure with high probability.

Proposition 17 *Let alg be an algorithm that solves the feasibility problem with accuracy $\epsilon = 1/(48d^2\sqrt{d})$. Then, it solves the feasibility procedure with probability at least $1 - C\sqrt{\log d}/d$.*

Proof Let $\mathcal{E}$ be the event of probability at least $1 - C\sqrt{\log d}/d$ defined in Proposition 16. We show that on $\mathcal{E}$, alg solves the feasibility procedure. On $\mathcal{E}$, the feasibility procedure emulates is a valid feasibility oracle. Further, on $\mathcal{E}$, the successful set contains a closed ball of radius ϵ. As a result, on $\mathcal{E}$, alg finds a solution to the feasibility problem emulated by the procedure. ■

Next, we show that it is necessary to finish the p_{max} periods to solve the feasibility procedure.

Proposition 18 *Fix an algorithm alg. Then, if $\mathcal{A}$ denotes the event when alg succeeds and $\mathcal{B}$ denotes the event when the procedure ends period p_{max} with alg, then $\mathcal{E} \subseteq \mathcal{B}$.*

Proof Consider the case when the period p_{max} was not ended. Let $\boldsymbol{x}^\star$ denote the last query performed by alg. We consider the scenario in which $\boldsymbol{x}^\star$ fell. Let t be the first time when alg submitted query $\boldsymbol{x}^\star$. For any of the scenarios (1), (2), or (4a), by construction of $Q_{\boldsymbol{A},\boldsymbol{v}}$, we already have $\boldsymbol{x}_t \notin Q_{\boldsymbol{A},\boldsymbol{v}}$. It remains to check scenarios (4b) and (4c) for which the procedure constructs a new vector $\boldsymbol{v}_{p,l}$, where p is the index of the period of t and $i_{p,1}, \ldots, i_{p,l} = t$ are the previous exploratory queries in period p. We decompose $\boldsymbol{x}_t = \boldsymbol{x}_{i_{p,l}} = \alpha_1 \boldsymbol{b}_{p,1} + \alpha_l \boldsymbol{b}_{p,l}$. By construction,

$$|\boldsymbol{x}_t^\top \boldsymbol{y}_{p,l}| = |\boldsymbol{x}_{i_{p,l}}^\top \boldsymbol{y}_{p,l}| \le \frac{\|\boldsymbol{\alpha}\|_1}{d^3} \le \frac{1}{d^2\sqrt{d}}.$$

As a result, $\boldsymbol{x}_t^\top \boldsymbol{v}_{p,l} \ge -|\boldsymbol{x}_t^\top \boldsymbol{y}_{p,l}| - \delta \ge -d^{-2.5} - d^{-3} > -\eta_1$, for any $d \ge 2$. Thus, $\boldsymbol{x}_t = \boldsymbol{x}^\star \notin Q_{\boldsymbol{A},\boldsymbol{v}}$. This shows that in order to succeed at the feasibility procedure, an algorithm needs to end all p_{max} periods. ■

B.3. Reduction to the Orthogonal Vector Game with Hints.

The remaining piece of our argument is to show that solving the feasibility procedure is harder than solving the Orthogonal Vector Game with Hints, Game 2.

Proposition 19 *Let $\boldsymbol{A} \sim \mathcal{U}(\{\pm 1\}^{n\times d})$. If there exists an M-bit algorithm that solves the feasibility problem described above using mp_{max} queries with probability at least q over the randomness of the algorithm, choice of $\boldsymbol{A}$ and the randomness of the separation oracle, then there is an algorithm for Game 2 for parameters $(d, k, m, M, \alpha = \frac{\eta_0}{\eta_1}, \beta = \frac{\eta_1}{2})$, for which the Player wins with probability at least q over the randomness of the player's strategy and $\boldsymbol{A}$.*

Proof Let alg be an M-bit algorithm solving the feasibility problem with mp_{max} queries with probability at least q. In Algorithm 5, we describe the strategy of the player in Game 2.

In the first part of the strategy, the player observes $\boldsymbol{A}$. Then they proceed to simulate the feasibility problem with alg using parameters $\boldsymbol{A}$. When needed to sample a vector $\boldsymbol{v}_{p,l}$ (resp. $\boldsymbol{v}_0$), the player submits the corresponding queries $\boldsymbol{x}_{i_{p,1}}, \ldots, \boldsymbol{x}_{i_{p,l}}$ (resp. $\emptyset$) useful to define $\boldsymbol{v}_{p,l}$. The player then takes the response given by the Oracle as that vector $\boldsymbol{v}_{p,l}$ (resp. $\boldsymbol{v}_0$), which simulates exactly a run of the feasibility procedure. Further, since $1 + p_{max}(k-1) \le d$, the player does not run out of queries. Importantly, during the run, the player keeps track of the length $i_{p,k} - i_{p,1}$ of period p. The first time we encounter a period p with length at most m, we set $\mathsf{Message} = \mathsf{Memory}_p$, the memory state of alg at the beginning of period p. If there is no such period, the strategy fails. Also, if alg

Algorithm 5: Strategy of the Player for the Orthogonal Vector Game with Hints

Input: d, k, p_{max}, m, algorithm alg

Part 1: Strategy to store Message knowing $\boldsymbol{A}$

1 Initialize the memory of alg to be $\mathbf{0}$.
2 Submit $\emptyset$ to the Oracle and use the response as $\boldsymbol{v}_0$.
3 Run alg with the optimization procedure knowing $\boldsymbol{A}$ and $\boldsymbol{v}_0$ until the first exploratory query $\boldsymbol{x}_{i_{1,1}}$.
4 **for** $p \in [p_{max}]$ **do**
5   Let Memory$_p$ be the current memory state of alg and $i_{p,1}$ the current iteration step.
6   Run alg with the feasibility procedure until period p ends at iteration step $i_{p+1,1}$. If alg stopped before, **return** the strategy fails. When needed to sample a unit vector $\boldsymbol{v}_{p',l'}$, submit vectors $\boldsymbol{x}_{i_{p',1}}, \ldots \boldsymbol{x}_{i_{p',l'}}$ to the Oracle. We use the corresponding response of the Oracle as $\boldsymbol{v}_{p',l'}$.
7   **if** $i_{p+1,1} - i_{p,1} \leq m$ **then**
8   Set Message $=$ Memory$_p$
9 **end**
10 **for** *Remaining queries to perform to Oracle* **do** Submit arbitrary query, e.g. $\emptyset$;
11 **if** Message *has not been defined yet* **then return** The strategy fails;
12 Submit $\tilde{\boldsymbol{g}}_{\boldsymbol{A},\boldsymbol{v}}$ to the Oracle as defined in Eq (11).

Part 2: Strategy to make queries

13 Set the memory state of alg to be Message.
14 **for** $i \in [m]$ **do**
15   Run alg with current memory to obtain a query $\boldsymbol{z}_i$.
16   Submit $\boldsymbol{z}_i$ to the Oracle from Game 2, to get response $(\boldsymbol{g}_i, s_i)$.
17   Compute $\tilde{\boldsymbol{g}}_i$ using $\boldsymbol{z}_i$, $\boldsymbol{g}_i$ and s_i as defined in Eq (12) and pass $\tilde{\boldsymbol{g}}_i$ as response to alg.
18 **end**

Part 3: Strategy to return vectors

19 **for** $l \in [k]$ **do** Set i_l to be the index i of the first query $\boldsymbol{z}_i$ for which $s_i = l$, if it exists ;
20 **if** *index* i_k *has not been defined yet* **then**
21   With the current memory of alg find a new query $\boldsymbol{z}_{m+1}$ and set $i_k = m+1$.
22 **return** $\left\{\frac{\boldsymbol{z}_{i_1}}{\|\boldsymbol{z}_{i_1}\|}, \ldots, \frac{\boldsymbol{z}_{i_k}}{\|\boldsymbol{z}_{i_k}\|}\right\}$ to the Oracle.

stopped before ending period p_{max}, the strategy fails. Next, the algorithm submits the following function $\tilde{\boldsymbol{g}}_{\boldsymbol{A},\boldsymbol{v}}$ to the Oracle. Since the responses of the feasibility procedure are consistent over time, we adopt the following notation. For a previously queried vector $\boldsymbol{x}$ of alg, we denote $\boldsymbol{g}(\boldsymbol{x})$ the vector which was returned to alg during the first part (lines 3-9 of Algorithm 5).

$$\tilde{\boldsymbol{g}}_{\boldsymbol{A},\boldsymbol{v}} : \boldsymbol{x} \mapsto \begin{cases} (\mathbf{0}, 1) & \text{if } \boldsymbol{x} \text{ was never queried in the first part,} \\ (\boldsymbol{a}_i, 1) & \text{ow. and if } \boldsymbol{g}(\boldsymbol{x}) \in \{\pm \boldsymbol{a}_i\}, i \leq n, \\ (\boldsymbol{v}_0, 2) & \text{ow. and if } \boldsymbol{g}(\boldsymbol{x}) = \boldsymbol{v}_0, \\ (\boldsymbol{v}_{p',l'}, 2 + l'\mathbb{1}_{p'=p} + k\mathbb{1}_{p'=p+1,l'=1}) & \text{ow. and if } \boldsymbol{g}(\boldsymbol{x}) = \boldsymbol{v}_{p',l'}, p' \leq p_{max}, l \leq k-1. \end{cases} \tag{11}$$

Intuitively, the first component of $\tilde{\boldsymbol{g}}$ gives the returned vector in the first period, at the exception that we always return $\boldsymbol{a}_i$ instead of $\{\pm \boldsymbol{a}_i\}$. The second term has values in $[2+k \leq d^2]$. Hence, the submitted function is valid.

Next, in the second part of the algorithm, the player proceeds to simulate a run the feasibility procedure with alg on period p. To do so, we first set the memory state of alg to Message. Each new query $\boldsymbol{z}_i$ is submitted to the Oracle of Game 2 to get a response $(\boldsymbol{g}_i, s_i)$. Then, we compute $\tilde{\boldsymbol{g}}_i$ as follows

$$\tilde{\boldsymbol{g}}_i = \begin{cases} \boldsymbol{g}_i & \text{if } s_i \geq 2, \\ sign(\boldsymbol{g}_i^\top \boldsymbol{z}_i)\boldsymbol{g}_i & \text{if } s_i = 1. \end{cases} \tag{12}$$

One can easily check that $\tilde{\boldsymbol{g}}_i$ corresponds exactly to the response that was passed to alg in the first part of the strategy. The player then passes $\tilde{\boldsymbol{g}}_i$ to alg so that it can update its state. We repeat this process for m steps. Further, the player can also keep track of the exploratory queries: the index i_l of the first response satisfying $s_i = 2+l$ for $l \leq k-1$ (resp. $s_i = 2+k$)is the exploratory query which led to the construction of $\boldsymbol{v}_{p,l}$ (resp. $\boldsymbol{v}_{p+1,1}$) in the first part. Last, we check if the last index i_k was defined. If not, we pose $i_k = m+1$ and let $\boldsymbol{z}_{m+1}$ be the next query of alg with the current memory. The player then returns the vectors $\frac{\boldsymbol{z}_{i_1}}{\|\boldsymbol{z}_{i_1}\|}, \ldots, \frac{\boldsymbol{z}_{i_k}}{\|\boldsymbol{z}_{i_k}\|}$. This ends the description of the player's strategy.

By Proposition 18, on an event $\mathcal{E}$ of probability at least q, the algorithm alg succeeds and ends period p_{max}. As a result, similarly as in the proof of Proposition 8, since alg makes at most mp_{max} queries, and there are p_{max} periods, there must be a period of length at most m. Hence the strategy never fails at this phase of the player's strategy on the event $\mathcal{E}$. Further, we already checked that in the second phase, the vectors $\tilde{\boldsymbol{g}}_i$ passed to alg coincide exactly with the responses passed to alg in the first part. Thus, this shows that during the second part, the player simulates exactly the run of the feasibility problem on period p. More precisely, the queries coincide with the queries in the feasibility problem at times $i_{p,1}, \ldots, \min\{i_{p,k}, i_{p,1}+m-1\}$. Because the first part succeeded on $\mathcal{E}$, we have $i_{p,k} \leq i_{p,0}+m$. Therefore, if i_k has not yet been defined, this means that we had $i_{p,k} = i_{p,1}+m$. Hence, the next query with the current memory $\boldsymbol{z}_{m+1}$ is exactly the query $\boldsymbol{x}_{i_{p,k}}$ for the feasibility problem. This shows that the vectors $\boldsymbol{z}_{i_1}, \ldots, \boldsymbol{z}_{i_k}$ coincide exactly with the vectors $\boldsymbol{x}_{i_{p,1}}, \ldots, \boldsymbol{x}_{i_{p,k}}$ when running alg on the feasibility problem in the first part.

We now show that the returned vectors are successful for Game 2. By construction, $\boldsymbol{x}_{i_{p,1}}, \ldots, \boldsymbol{x}_{i_{p,k}}$ are all informative. In particular, $\|\boldsymbol{A}\boldsymbol{x}_{i_{p,l}}\|_\infty \leq \eta_0$ for all $1 \leq l \leq k$. Further, these queries did not fall in scenario (2), hence $\boldsymbol{v}_0^\top \boldsymbol{x}_{i_{p,l}} < -\eta_1$, which implies $\|\boldsymbol{x}_{i_{p,l}}\| > \eta_1$ for all $l \leq k$. As a result,

$$\frac{\|\boldsymbol{A}\boldsymbol{x}_{i_{p,l}}\|_\infty}{\|\boldsymbol{x}_{i_{p,l}}\|} \leq \frac{\eta_0}{\eta_1}.$$

Next fix $l \leq k-1$. By construction of $\boldsymbol{y}_{p,l}$,

$$\|P_{Span(\boldsymbol{x}_{i_{p,l'}}, l' \leq l)}(\boldsymbol{y}_{p,l})\|^2 = \sum_{l' \leq l} |\boldsymbol{b}_{p,l'}^\top \boldsymbol{y}_{p,l}|^2 \leq \frac{k}{d^6} \leq \frac{1}{d^5}.$$

Hence,

$$\|\boldsymbol{v}_{p,l} - P_{Span(\boldsymbol{x}_{i_{p,l'}}, l' \leq l)^\perp}(\boldsymbol{y}_{p,l})\| \leq \|P_{Span(\boldsymbol{x}_{i_{p,l'}}, l' \leq l)}(\boldsymbol{y}_{p,l})\| + \delta \leq \frac{1}{d^5} + \delta.$$

As a result, since $\boldsymbol{x}_{p,l+1}^\top \boldsymbol{v}_{p,l} < -\eta_1$, we have

$$\|P_{Span(\boldsymbol{x}_{i_{p,l'}}, l' \leq l)^\perp}(\boldsymbol{x}_{p,l+1})\| \geq |\boldsymbol{x}_{p,l+1}^\top P_{Span(\boldsymbol{x}_{i_{p,l'}}, l' \leq l)^\perp}(\boldsymbol{y}_{p,l})\| > \eta_1 - \frac{1}{d^5} - \delta \geq \frac{\eta_1}{2}.$$

This shows that the returned vectors $\frac{\boldsymbol{x}_{i_{p,1}}}{\|\boldsymbol{x}_{i_{p,1}}\|}, \ldots, \frac{\boldsymbol{x}_{i_{p,k}}}{\|\boldsymbol{x}_{i_{p,k}}\|}$ are successful for Game 2 with parameters $\alpha = \frac{\eta_0}{\eta_1}$ and $\beta = \frac{\eta_1}{2}$. This ends the proof that strategy succeeds on $\mathcal{E}$ for these parameters, which ends the proof of the proposition. ■

We are now ready to prove the main result.

Proof of Theorem 2 Suppose that there is an algorithm alg for solving the feasibility problem to optimality $\epsilon = 1/(48d^2\sqrt{d})$ with memory M and at most Q queries. Let $k = \lceil 20 \frac{M+3d\log(2d)+1}{c_H n} \rceil$. By Proposition 17, it solves the feasibility procedure with parameter k with probability at least $1 - C\sqrt{\log d}/d$. By Proposition 19 there is an algorithm for Game 2 that wins with probability $1/3$ with $m = \lceil Q/p_{max} \rceil$ and paraeters $\alpha = \eta_0/\eta_1$ and $\beta = \eta_1/2$. We check that

$$\alpha \left(\frac{\sqrt{d}}{\beta} \right)^{5/4} \leq 12 d^2 \eta_0 = \frac{1}{2}.$$

Hence, by Proposition 9, we have

$$m \geq \frac{c_H}{8(30 \log d + c_H)} d.$$

This shows that

$$Q \geq \Omega \left(p_{max} \frac{d}{\log d} \right) = \Omega \left(\frac{d^2}{k \log^3 d} \right) = \Omega \left(\frac{d^3}{(M + \log d) \log^3 d} \right).$$

This implies that for a memory $M = d^{2-\delta}$ with $0 \leq \delta \leq 1$ the number of queries is $Q = \tilde{\Omega}(d^{1+\delta})$. ■

Appendix C. Concentration bounds

The following result gives concentration bounds for the norm of the projection of a random unit vector onto linear subspaces.

Proposition 20 *Let P be a projection in $\mathbb{R}^d$ of rank r and let $\boldsymbol{x} \in \mathbb{R}^d$ be a random vector sampled uniformly on the unit sphere $\boldsymbol{x} \sim \mathcal{U}(S^{d-1})$. Then, for every $t > 0$,*

$$\max \left\{ \mathbb{P}\left(\|P(\boldsymbol{x})\|^2 - \frac{r}{d} \geq t \right), \mathbb{P}\left(\|P(\boldsymbol{x})\|^2 - \frac{r}{d} \leq -t \right) \right\} \leq e^{-dt^2}.$$

Further, if $r = 1$ and $d \geq 2$,

$$\mathbb{P}\left(\|P(\boldsymbol{x})\| \geq \sqrt{\frac{t}{d-1}} \right) \leq 2\sqrt{t} e^{-t/2}.$$

Proof First, by isometry, we can assume that P is the projection onto the coordinate vectors $\boldsymbol{e}_1, \dots \boldsymbol{e}_r$. Then, let $\boldsymbol{y} \sim \mathcal{N}(0,1)$ be a normal vector. Note that $\boldsymbol{x} = \frac{\boldsymbol{y}}{\|\boldsymbol{y}\|} \sim \mathcal{U}(S^{d-1})$. Further,

$$\|\boldsymbol{x}\|^2 \geq \frac{r}{d} + t \iff \left(1 - \frac{r}{d} - t\right) \sum_{i=1}^r y_i^2 \geq \left(\frac{r}{d} + t\right) \sum_{i=r+1}^d y_i^2.$$

Note that $Z_1 = \sum_{i=1}^r y_i^2$ and $Z_2 = \sum_{i=r+1}^d y_i^2$ are two independent random chi squared variables of parameters r and $d-r$ respectively. Recalling that the moment generating function of $Z \sim \chi^2(k)$ is $\mathbb{E}[e^{sZ}] = (1-2s)^{-k/2}$ for $s < 1/2$. Therefore, for any

$$-\frac{1}{2(r/d+t)} < s < \frac{1}{2(1-r/d-t)}, \tag{13}$$

one has

$$\begin{aligned}\mathbb{P}\left(\|P(\boldsymbol{x})\|^2 - \frac{r}{d} \geq t\right) &\leq \mathbb{E}\left[\exp\left(s\left(1-\frac{r}{d}-t\right)Z_1 - s\left(\frac{r}{d}+t\right)Z_2\right)\right] \\ &= \frac{\left[1-2s\left(1-\frac{r}{d}-t\right)\right]^{-r/2}}{\left[1-2s\left(\frac{r}{d}+t\right)\right]^{-(d-r)/2}}.\end{aligned}$$

Let $s = \frac{1}{2}\left(\frac{1-r/d}{1-r/d-t} - \frac{r/d}{r/d+t}\right)$, which satisfies Eq (13). The previous equation readily yields

$$\mathbb{P}\left(\left|\|P(\boldsymbol{x})\|^2 - \frac{r}{d}\right| \geq t\right) \leq \exp\left(-\frac{d}{2}d_{KL}\left(\frac{r}{d};\frac{r}{d}+t\right)\right) \leq e^{-dt^2}.$$

In the last inequality we used Pinsker's inequality $d_{KL}(r/d; r/d+t) \geq 2\delta(\mathcal{B}(r/d), \mathcal{B}(d/r+t))^2 = 2t^2$, where $\mathcal{B}(q)$ is the Bernouilli distribution of parameter q. Replacing P with $Id - P$ and r with $d-r$ gives the other inequality

$$\mathbb{P}\left(\|P(\boldsymbol{x})\|^2 - \frac{r}{d} \leq -t\right) \leq e^{-dt^2}.$$

This gives first claim. For the second claim, supposing that $r = 1 < d$, from the above equation, we have

$$\mathbb{P}\left(\|P(\boldsymbol{x})\|^2 \geq \frac{t}{d}\right) \leq \exp\left(-\frac{d}{2}d_{KL}\left(\frac{1}{d};\frac{t}{d}\right)\right) = \sqrt{t}\left(\frac{1-\frac{t}{d}}{1-\frac{1}{d}}\right)^{(d-1)/2} \leq \sqrt{2t}e^{-t(d-1)/(2d)}.$$

Thus,

$$\mathbb{P}\left(\|P(\boldsymbol{x})\|^2 \geq \frac{t}{d-1}\right) \leq \sqrt{\frac{2(d-1)}{d}}\sqrt{t}e^{-t/2},$$

which ends the proof of the proposition. ■

Next, we need the following lemma which gives a concentration inequality for discretized samples in $\mathcal{D}_d$ and approximately perpendicular to $k \leq d/3 - 1$ vectors.

Lemma 21 *Let $0 \leq k \leq d/3 - 1$ and $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k \in B_d(\mathbf{0}, 1)$ be k orthonormal vectors in the unit ball, and $\boldsymbol{x} \in B_d(\mathbf{0}, 1)$. Denote by μ the distribution on the unit sphere corresponding to the uniform distribution $\boldsymbol{y} \sim \mathcal{U}(S^{d-1} \cap \{\boldsymbol{w} \in \mathbb{R}^d : |\boldsymbol{x}_i^\top \boldsymbol{w}| \leq d^{-3}, \forall i \leq k\})$. Let $\boldsymbol{y} \sim \mu$. Then, for $t \geq 2$,*

$$\mathbb{P}\left(|\boldsymbol{x}^\top \boldsymbol{y}| \geq \sqrt{\frac{t}{d}} + \frac{1}{d^2}\right) \leq 2\sqrt{t}e^{-t/3}.$$

Further, let $\delta \leq 1$ and $\boldsymbol{z} = \phi_\delta(\boldsymbol{y})$. Then for $t \geq 4$,

$$\mathbb{P}\left(|\boldsymbol{x}^\top \boldsymbol{z}| \geq \sqrt{\frac{t}{d}} + \frac{1}{d^2} + \delta\right) \leq 2\sqrt{t}e^{-t/3}.$$

Proof We use the same notations as above and denote by $\mathcal{E} = \{|\boldsymbol{x}_i^\top \boldsymbol{y}| \leq d^{-3}, \forall i \leq k\}$ the event considered and $\boldsymbol{y} \sim \mu$. We decompose $\boldsymbol{y} = \alpha_1 \boldsymbol{x}_1 + \ldots + \alpha_k \boldsymbol{x}_k + \boldsymbol{y}'$, where $\boldsymbol{y}' \in Span(\boldsymbol{x}_i, i \leq k)^\perp := E$. Note that $\frac{\boldsymbol{y}'}{\|\boldsymbol{y}'\|}$ is a uniformly random unit vector in E. As a result, using Proposition 20, we obtain for any $t \geq 2$,

$$\begin{aligned}\mathbb{P}\left(|\boldsymbol{x}^\top \boldsymbol{y}'| \geq \sqrt{\frac{t}{d-k-1}}\right) &= \mathbb{P}\left(|P_E(\boldsymbol{x})^\top \boldsymbol{y}'| \geq \sqrt{\frac{t}{d-k-1}}\right) \\ &\leq 2\sqrt{t}e^{-t/2}.\end{aligned}$$

Also, because by definition of μ, we have $|\alpha_i| \leq d^{-3}$ for all $i \leq k$, we obtain $|\boldsymbol{x}^\top \boldsymbol{y}| \leq \frac{k}{d^3} + |\boldsymbol{x}^\top \boldsymbol{y}'| \leq \frac{1}{d^2} + |\boldsymbol{x}^\top \boldsymbol{y}'|$. As a result, using the fact that $d - k - 1 \geq 2d/3$, the previous equation shows that

$$\mathbb{P}\left(|\boldsymbol{x}^\top \boldsymbol{y}| \geq \sqrt{\frac{3t}{2d}} + \frac{1}{d^2}\right) \leq \mathbb{P}\left(|\boldsymbol{x}^\top \boldsymbol{y}'| \geq \sqrt{\frac{t}{d-k-1}}\right) \leq 2\sqrt{t}e^{-t/2}.$$

Next, we use the fact that $\|\boldsymbol{z} - \boldsymbol{y}\| = \|\phi_\delta(\boldsymbol{y}) - \boldsymbol{y}\| \leq \delta$ to obtain

$$\mathbb{P}\left(|\boldsymbol{x}^\top \boldsymbol{z}| \geq \sqrt{\frac{t}{d}} + \frac{1}{d^2} + \delta\right) \leq \mathbb{P}\left(|\boldsymbol{x}^\top \boldsymbol{y}| \geq \sqrt{\frac{t}{d}} + \frac{1}{d^2}\right) \leq 2\sqrt{t}e^{-t/3}.$$

This ends the proof of the lemma. ■

Appendix D. An improved result on robustly-independent vectors

The following lemma serves the same purpose as [23, Lemma 34]. Namely, from successful vectors of the Game 2, it allows to recover an orthonormal basis that is still approximately in the nullspace of $\boldsymbol{A}$. The following version gives a stronger version that improves the dependence in d of our chosen parameters.

Lemma 22 *Let $\delta \in (0, 1]$ and suppose that we have $r \leq d$ unit norm vectors $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_r \in \mathbb{R}^d$. Suppose that for any $i \leq k$,*

$$\|P_{Span(\boldsymbol{y}_j, j<i)^\perp}(\boldsymbol{y}_i)\| \geq \delta.$$

Let $\boldsymbol{Y} = [\boldsymbol{y}_1, \ldots, \boldsymbol{y}_r]$ and $s \geq 2$. There exists $\lceil r/s \rceil$ orthonormal vectors $\boldsymbol{Z} = [\boldsymbol{z}_1, \ldots, \boldsymbol{z}_{\lceil r/s \rceil}]$ such that for any $\boldsymbol{a} \in \mathbb{R}^d$,

$$\|\boldsymbol{Z}^\top \boldsymbol{a}\|_\infty \leq \left(\frac{\sqrt{d}}{\delta}\right)^{s/(s-1)} \|\boldsymbol{Y}^\top \boldsymbol{a}\|_\infty.$$

Proof Let $\boldsymbol{B} = (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_r)$ be the orthonormal basis given by the Gram-Schmidt decomposition of $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_r$. By definition of the Gram-Schmidt decomposition, we can write $\boldsymbol{Y} = \boldsymbol{BC}$ where $\boldsymbol{C}$ is an upper-triangular matrix. Further, its diagonal is exactly $diag(\|P_{Span(\boldsymbol{y}_{l'}, l'<l)^\perp}(\boldsymbol{y}_l)\|, l \leq r)$. Hence,

$$\det(\boldsymbol{Y}) = \det(\boldsymbol{C}) = \prod_{l \leq r} \|P_{Span(\boldsymbol{y}_{l'}, l'<l)^\perp}(\boldsymbol{y}_l)\| \geq \delta^r.$$

We then introduce the singular value decomposition $\boldsymbol{Y} = \boldsymbol{U} diag(\sigma_1, \ldots, \sigma_r) \boldsymbol{V}^\top$, where $\boldsymbol{U} \in \mathbb{R}^{d \times r}$ and $\boldsymbol{V} \in \mathbb{R}^{r \times r}$ have orthonormal columns, and $\sigma_1 \geq \ldots \geq \sigma_r$. Next, for any vector $\boldsymbol{z} \in \mathbb{R}^d$, since the columns of $\boldsymbol{Y}$ have unit norm,

$$\|\boldsymbol{Y}\boldsymbol{z}\|_2 \leq \sum_{l \leq r} |z_l| \|\boldsymbol{y}_l\|_2 \leq \|\boldsymbol{z}\|_1 \leq \sqrt{d} \|\boldsymbol{z}\|_2.$$

In the last inequality we used Cauchy-Schwartz. Therefore, all singular values of $\boldsymbol{Y}$ are upper bounded by $\sigma_1 \leq \sqrt{d}$. Thus, with $r' = \lceil r/s \rceil$

$$\delta^r \leq \det(\boldsymbol{Y}) = \prod_{l=1}^{r} \sigma_l \leq d^{(r'-1)/2} \sigma_{r'}^{r-r'+1} \leq d^{r/2s} \sigma_{r'}^{(s-1)r/s},$$

so that $\sigma_{r'} \geq \delta^{s/(s-1)}/d^{1/(2s)}$. We are ready to define the new vectors. We pose for all $i \leq r'$, $\boldsymbol{z}_i = \boldsymbol{u}_i$ the i-th column of $\boldsymbol{U}$. These correspond to the r' largest singular values of $\boldsymbol{Y}$ and are orthonormal by construction. Then, for any $i \leq r'$, we also have $\boldsymbol{z}_i = \boldsymbol{u}_i = \frac{1}{\sigma_i} \boldsymbol{Y} \boldsymbol{v}_i$ where $\boldsymbol{v}_i$ is the i-th column of $\boldsymbol{V}$. Hence, for any $\boldsymbol{a} \in \mathbb{R}^d$,

$$|\boldsymbol{z}_i^\top \boldsymbol{a}| = \frac{1}{\sigma_i} |\boldsymbol{v}_i^\top \boldsymbol{Y}^\top \boldsymbol{a}| \leq \frac{\|\boldsymbol{v}_i\|_1}{\sigma_i} \|\boldsymbol{Y}^\top \boldsymbol{a}\|_\infty \leq \frac{d^{1/2+1/(2s)}}{\delta^{s/(s-1)}} \|\boldsymbol{Y}^\top \boldsymbol{a}\|_\infty.$$

This ends the proof of the lemma. ■