Dueling Optimization with a Monotone Adversary

Avrim Blum AVRIM@TTIC.EDU
Toyota Technological Institute Chicago

Meghal Gupta MEGHAL@BERKELEY.EDU
University of California Berkeley

Gene Li GENE@TTIC.EDU
Toyota Technological Institute Chicago

Naren Sarayu Manoj NSM@TTIC.EDU
Toyota Technological Institute Chicago

Aadirupa Saha AADIRUPA@TTIC.EDU
Toyota Technological Institute Chicago (currently at Apple Research)

Yuanyuan Yang YYANGH@CS.WASHINGTON.EDU
University of Washington

Editors: Claire Vernade and Daniel Hsu

Keywords: Online learning, Monotone adversary, Convex optimization

Abstract

We introduce and study the problem of *dueling optimization with a monotone adversary*, which is a generalization of (noiseless) dueling convex optimization. The goal is to design an online algorithm to find a minimizer $\boldsymbol{x}^\star$ for a function $f\colon \mathcal{X} \to \mathbb{R}$, where $\mathcal{X} \subseteq \mathbb{R}^d$. In each round, the algorithm submits a pair of guesses, i.e., $\boldsymbol{x}^{(1)}$ and $\boldsymbol{x}^{(2)}$, and the adversary responds with *any* point in the space that is at least as good as both guesses. The cost of each query is the suboptimality of the worse of the two guesses; i.e., $\max\left(f(\boldsymbol{x}^{(1)}), f(\boldsymbol{x}^{(2)})\right) - f(\boldsymbol{x}^\star)$. The goal is to minimize the number of iterations required to find an ε-optimal point and to minimize the total cost (regret) of the guesses over many rounds. Our main result is an efficient randomized algorithm for several natural choices of the function f and set $\mathcal{X}$ that incurs cost $O(d)$ and iteration complexity $O(d\log(1/\varepsilon)^2)$. Moreover, our dependence on d is asymptotically optimal, as we show examples in which any randomized algorithm for this problem must incur $\Omega(d)$ cost and iteration complexity.

1. Introduction

A growing body of literature studies learning with preference-based feedback [BV06; SJ11], with tremendous empirical success in recommendation systems, search engine optimization, information retrieval, and robotics. More recently, preference-based feedback has received a lot of attention as a mechanism to train large language models [OWJ+22]. Moreover, in recommender systems [BOHG13], a natural approach is to learn from users' preferences relations on a set of recommended items and update the system's belief for better future recommendations [JRTZ16] (e.g., given these items, which one do you prefer the most?).

Such preference-based feedback is not readily addressed by classical formulations for online decision making, such as bandits and reinforcement learning. In particular, algorithms for these problems rely on ordinal feedback per item (e.g., on a scale of 1 to 10, how much did the user like a particular item?). To address this, a long line of work studies the *dueling bandit framework* for online decision making under pairwise/preference-based feedback. There exist efficient algorithms with provable guarantees for the standard multi-armed bandit setup [YBKJ12; AKJ14; KHKN15], contextual bandits [DHSSZ15; SK22], as well as dueling convex optimization [JNR12; SKM21; SKM22], to name a few. The dueling bandit framework is especially applicable in settings where real-valued feedback is scarce or impossible to obtain, but preference-based feedback is readily available.

However, a key limitation of the dueling bandit framework is that the feedback that the learner receives is essentially "in-list". That is, the users are restricted to selecting items exclusively from the list of recommended items. This feedback model fails to capture the real-world scenarios where the users might select an out-of-list item they prefer. To illustrate, music streaming services like Spotify create personalized playlists for users. Concretely, each song can be encoded as a feature vector $\boldsymbol{x} \in \mathbb{R}^d$, and the goal is to recommend the songs with the highest utility for a hidden, well-structured utility function of $\boldsymbol{x}$. However, the users can also search for and play the songs they have a stronger preference (i.e., higher utility) than all recommendations.

This out-of-list feedback model falls into a monotone adversarial framework (see the chapter by Feige [Fei21]). In such models, an adversary is only allowed to make "helpful" changes. For example, in a graph clustering problem, the adversary is only allowed to add edges within communities and delete edges that cross communities (see, e.g., the chapter by Moitra [Moi21]). In our setting, the adversary is only allowed to respond with an item that is at least as good as any recommended item. A clear adaptation of the dueling bandit framework to this new feedback type is not evident.

1.1. Problem statement

As our main conceptual contribution, we introduce a theoretical formulation for this setting that we call *dueling optimization with a monotone adversary*. As we will see, our formulation supports "out-of-list" feedback.

Problem 1 (Dueling optimization with a monotone adversary)**.** *Let $\mathcal{X} \subseteq \mathbb{R}^d$ be a decision space, and let $f\colon \mathcal{X} \to \mathbb{R}$ be a cost function with an unknown global minimum $\boldsymbol{x}^\star$. A learner interacts with an adversary over rounds $t = 1, 2, \ldots$, where each round is of the following form.*

1. *The learner proposes two points $\boldsymbol{x}_t^{(1)}, \boldsymbol{x}_t^{(2)} \in \mathcal{X}$.*

2. *The adversary responds with a point* $\boldsymbol{r}_t$ *that satisfies* $f(\boldsymbol{r}_t) \leq \min\left\{f\left(\boldsymbol{x}_t^{(1)}\right), f\left(\boldsymbol{x}_t^{(2)}\right)\right\}$.

The goal is to design algorithms that:

1. *for some prespecified* $\varepsilon > 0$*, minimize the number of iterations to find a point* $\boldsymbol{x}$ *for which* $f(\boldsymbol{x}) - f(\boldsymbol{x}^\star) \leq \varepsilon$*;*
2. *minimize the total cost* $\sum_{t=1}^{\infty}\left(\max\left\{f\left(\boldsymbol{x}_t^{(1)}\right), f\left(\boldsymbol{x}_t^{(2)}\right)\right\} - f(\boldsymbol{x}^\star)\right)$.

Note that in Problem 1, we are interested in both the iteration complexity and the total cost. The first objective is a standard metric for measuring the performance of an iterative optimization algorithm. The second objective is motivated by online settings in which a practitioner may wish to minimize the total regret (cost) of its recommendations over an indefinitely long interaction with a user. In fact, the algorithms we propose in this paper simultaneously achieve both small iteration complexity (for any choice of ε) as well as total cost — see our technical overview in Section 1.3 for more details.

Problem 1 is a natural extension of (noiseless) dueling optimization [JNR12; SKM21; SKM22] to handle "out-of-list" responses, as in the Spotify recommendation example. The vanilla (noiseless) dueling optimization setup corresponds to the requirement that the user's response satisfies $\boldsymbol{r}_t \in \{\boldsymbol{x}_t^{(1)}, \boldsymbol{x}_t^{(2)}\}$. We allow the user to be potentially adversarial by allowing it to respond with any improvement to the learner's suggestions (in the sequel, we exclusively refer to the user as the adversary).

Even though the monotone adversary is only improving upon the learner's suggestions, existing algorithms for dueling optimization cannot be freely extended to handle the monotone feedback. At a high level, the difficulty arises from the fact that existing algorithms carefully select the queries $\boldsymbol{x}_t^{(1)}, \boldsymbol{x}_t^{(2)}$ so that learning whether $f(\boldsymbol{x}_t^{(1)}) > f(\boldsymbol{x}_t^{(2)})$ reveals information about the underlying f. However, a monotone adversary can return a point $\boldsymbol{r}_t$ that reveals no information about the relationship between $\boldsymbol{x}_t^{(1)}$ and $\boldsymbol{x}_t^{(2)}$.

To illustrate this point, consider a natural coordinate-wise binary search algorithm for the dueling optimization problem when $f(\boldsymbol{x}) = \|\boldsymbol{x} - \boldsymbol{x}^\star\|_2^2$ for some $\boldsymbol{x}^\star \in \mathcal{B}_2^d \coloneqq \{\boldsymbol{x} \ : \ \|\boldsymbol{x}\|_2 \leq 1\}$. For coordinates $i = 1, \cdots, d$, query points of the form $\boldsymbol{x}_t^{(1)} = c_1 \cdot \boldsymbol{e}_i$, $\boldsymbol{x}_t^{(2)} = c_2 \cdot \boldsymbol{e}_i$ and progressively refine the values $c_1, c_2 \in \mathbb{R}$ to search for the value of $\boldsymbol{x}^\star[i]$ (i.e., the i-th entry of $\boldsymbol{x}^\star$). It is easy to show that this approach has a query complexity of $O\left(d \log\left(1/\varepsilon\right)\right)$ in the vanilla dueling optimization setting. However, a monotone adversary can return orthogonal responses of the form $\boldsymbol{r}_t = C\boldsymbol{e}_j$ (where $j \neq i$ and C is a constant) that do not allow the learner to search along the intended coordinate i. Furthermore, Jamieson, Nowak, and Recht [JNR12] and Saha, Koren, and Mansour [SKM21] give more sophisticated algorithms for the dueling optimization problem that inherently depend upon the "in-list" feedback, which clearly cannot apply to our setting. We therefore need novel insights to solve Problem 1.

1.2. Our results

We study Problem 1 for various natural classes of functions f and provide tight upper and lower bounds on the number of queries required to find an ε-optimal point.

Upper bound for linear functions. First, we study dueling optimization with a monotone adversary when the function f is linear. This is a natural class to consider. In particular, an algorithm that solves Problem 1 can be adapted to achieve constant regret for (noiseless) linear contextual bandits [CLRS11], where the reward function is $r(\boldsymbol{x}) \coloneqq \langle \boldsymbol{x}, \boldsymbol{x}^\star \rangle$. Note that the key difference in the setup is that the learner does not get to observe the actual linear costs but instead only an improvement to the actions (points) that the learner selects.

Theorem 1. *Let $\mathcal{X} = \mathbb{S}_2^d$, let $\boldsymbol{x}^\star$ be such that $\|\boldsymbol{x}^\star\|_2 = 1$, and let $f \colon \mathcal{X} \to \mathbb{R}$ be $f(\boldsymbol{x}) = -\langle \boldsymbol{x}, \boldsymbol{x}^\star \rangle$. Fix any $\varepsilon > 0$. There exists an algorithm that, in the setting of Problem 1, with probability at least $1 - \exp(-O(d))$:*

- *outputs a point $\boldsymbol{x}$ satisfying $\langle \boldsymbol{x}^\star - \boldsymbol{x}, \boldsymbol{x}^\star \rangle \le \varepsilon$ within $O(d \log (1/\varepsilon)^2)$ iterations;*
- *incurs total cost $O(d)$.*

Each pair of guesses at time t can be computed in $O(d)$ time.

We prove Theorem 1 in Section 2.2, and the cost is near-optimal with respect to d.

Gollapudi, Guruganesh, Kollias, Manurangsi, Leme, and Schneider [GGKMLS21] study a closely related setup that they call *local contextual recommendation*. Their result (see their Theorem 6.4) can be interpreted as showing that if the action set $\mathcal{X}$ is a discrete set (namely a packing over the unit sphere), there exists a $2^{\Omega(d)}$ lower bound on the iteration complexity to find a point with constant suboptimality. In contrast, our Theorem 1 shows a much smaller upper bound when the domain is the entire unit sphere.

Upper bound for smooth and PŁ functions. Next, we study whether we can show guarantees for a large class of functions. We show a positive result for functions that are both β-smooth and α-Polyak-Łojasiewicz (abbreviated as PŁ). These assumptions are standard in optimization.

Definition 1 (β-smooth function [Bub15, Lemma 3.4])**.** *We say f is β-smooth if it satisfies (1.1).*

$$\text{For all } \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d : \quad |f(\boldsymbol{x}) - f(\boldsymbol{y}) - \langle \nabla f(\boldsymbol{y}), \boldsymbol{x} - \boldsymbol{y} \rangle| \le \frac{\beta}{2} \cdot \|\boldsymbol{x} - \boldsymbol{y}\|_2^2 \tag{1.1}$$

Definition 2 (α-PŁ function)**.** *We say f is α-PŁ if it satisfies (1.2).*

$$\text{For all } \boldsymbol{x} \in \mathbb{R}^d \text{ and minimizers } \boldsymbol{x}^\star : \quad f(\boldsymbol{x}) - f(\boldsymbol{x}^\star) \le \frac{1}{2\alpha} \|\nabla f(\boldsymbol{x})\|_2^2 \tag{1.2}$$

Our main result for this setting is Theorem 2.

Theorem 2. *Let $\mathcal{X} = \mathbb{R}^d$, and suppose f is β-smooth (Definition 1) and α-PŁ (Definition 2). Fix any $\varepsilon > 0$, as well as a known point $\boldsymbol{x}_1$ and a value B satisfying $B \ge f(\boldsymbol{x}_1) - f(\boldsymbol{x}^\star)$. There exists an algorithm that, in the setting of Problem 1, with probability at least $1 - \exp(-O(d))$:*

- *outputs a point $\boldsymbol{x}$ satisfying $f(\boldsymbol{x}) - f(\boldsymbol{x}^\star) \le \varepsilon$ within $O\left(\beta/\alpha \cdot d \cdot \log (B/\varepsilon)^2\right)$ iterations;*
- *incurs total cost $O(\beta/\alpha \cdot B \cdot d)$.*

Each pair of guesses at time t can be computed in $O(d)$ time.

We prove Theorem 2 in Section 2.3.

As an application, we show a positive result when the loss function is the Euclidean distance, and the decision space $\mathcal{X} = \mathcal{B}_2^d$ is a unit ball:

Theorem 3. *Let $\mathcal{X} = \mathcal{B}_2^d$, let $\boldsymbol{x}^\star$ be such that $\|\boldsymbol{x}^\star\|_2 \le 1$, and let $f\colon \mathcal{X} \to \mathbb{R}$ be $f(\boldsymbol{x}) = \|\boldsymbol{x} - \boldsymbol{x}^\star\|_2$. Fix any $\varepsilon > 0$. There exists an algorithm that, in the setting of Problem 1, with probability at least $1 - \exp(-O(d))$:*

- *outputs a point $\boldsymbol{x}$ satisfying $\|\boldsymbol{x} - \boldsymbol{x}^\star\|_2 \le \varepsilon$ within $O\left(d \cdot \log\left(B/\varepsilon\right)^2\right)$ iterations;*
- *incurs total cost $O(d)$.*

Each pair of guesses at time t can be computed in $O(d)$ time.

We prove Theorem 3 in Section 2.4.

Note that unlike in Theorem 2, Theorem 3 applies to the setting where the algorithm must guess points belonging to a given constraint set $\mathcal{X}$. Hence, in the proof of Theorem 3, we have to be careful to ensure that the convergence argument still holds when we apply the algorithm for Theorem 2 along with a projection step. It is not clear that this argument holds by default for all f satisfying the conditions requested by Theorem 2. Furthermore, as will become evident, we really only require that $\mathcal{X}$ be any convex body (though we state the result with $\mathcal{X} = \mathcal{B}_2^d$ to emphasize the consistency with our following lower bounds).

Lower bounds. We also prove that the dependence on d in our results is tight. In particular, when f is either a linear function or the distance to the target (as in Theorem 3), then $\Omega(d)$ queries are necessary to identify $\boldsymbol{x}^\star$. This will translate to a $\Omega(d)$ cost over an infinite number of rounds. In fact, our lower bound is valid when the adversary must return one of the two queried points, as in vanilla dueling optimization framework.

Our lower bound also covers a more general setting than that stated in Problem 1. Thus far, we have only discussed the setting where the algorithm can query only two points and is told the better of the two. In many practical instances, the algorithm can query m points and learn the point with the best objective value (we call this m-ary dueling optimization). Then, one may ask why we study only the $m = 2$ in this paper. In our construction, we prove that unless m is polynomial in d, we cannot decrease the total cost substantially below $\Omega(d)$. Thus, the most interesting case for constant m is when $m = 2$, and in this case, our result is optimal in d.

See Theorem 4 for a formal statement of our lower bound.

Theorem 4 (Lower bound, ℓ_2 distance)**.** *Let $\mathcal{X} = \mathcal{B}_2^d$. For any randomized algorithm for m-ary dueling optimization, there exists a choice of minimizer $\boldsymbol{x}^\star \in \mathcal{B}_2^d$ and function $f(\boldsymbol{x}) \coloneqq \|\boldsymbol{x} - \boldsymbol{x}^\star\|_2$ such that the algorithm must:*

- *perform $\Omega\left(d/\log m\right)$ iterations in expectation to find a point $\boldsymbol{x}$ for which $f(\boldsymbol{x}) - f(\boldsymbol{x}^\star) \le \varepsilon$.*
- *incur cost $\Omega\left(d/\log m\right)$ in expectation.*

Here, $\varepsilon > 0$ is an absolute numerical constant.

We prove Theorem 4 in Section 3. Using the same construction, we can also demonstrate that Theorem 1 is tight when $\mathcal{X}$ is the unit sphere.

Corollary 5 (Lower bound, linear f)**.** *Let $\mathcal{X} = \mathbb{S}_2^d$. For any randomized algorithm for m-ary dueling optimization there exists a choice of minimizer $\boldsymbol{x}^\star \in \mathbb{S}_2^d$ and function $f(\boldsymbol{x}) \coloneqq -\langle \boldsymbol{x}, \boldsymbol{x}^\star \rangle$ such that the same conclusions as in Theorem 4 hold.*

1.3. Technical overview

At a high level, our algorithms maintain a guess $\boldsymbol{x}_t$ for the optimal solution $\boldsymbol{x}^\star$. They will update this guess over many interactions with the adversary.

A general recipe. We first describe the primitives that our methods depend on. Our first technical innovation is the notion of *progress distributions*. Loosely speaking, these are distributions from which a learner is likely to sample a new guess $\boldsymbol{x}_{t+1}$ that decreases its suboptimality. See Definition 3.

Definition 3 (Progress Distribution)**.** *Let $f\colon \mathcal{X} \to \mathbb{R}$ for $\mathcal{X} \subseteq \mathbb{R}^d$. For $\boldsymbol{x} \in \mathcal{X}$ and $1 \le p < 2$, we say a distribution $\mathcal{D}(\boldsymbol{x})$ over vectors in $\mathbb{R}^d$ is a (p, γ, ρ)-progress distribution for $\boldsymbol{x}$ if we have the below.*

$$\Pr_{\boldsymbol{x}^+ \sim \mathcal{D}(\boldsymbol{x})} \left[\frac{f(\boldsymbol{x}) - f(\boldsymbol{x}^+)}{(f(\boldsymbol{x}) - f(\boldsymbol{x}^\star))^p} \ge \frac{\rho}{d} \right] \ge \gamma.$$

So, if for every $\boldsymbol{x}_t$ the learner had sample access to some progress distribution $\mathcal{D}(\boldsymbol{x}_t)$, the learner can significantly improve its solution (e.g. when $p = 1$, roughly $\sim d/\rho$ steps are sufficient for the learner to decrease its suboptimality by a constant factor). It is therefore natural that repeating such a sample-then-guess approach ad infinitum will yield an approximately optimal solution. In Theorem 8, we prove this whenever there exist families of progress distributions for every range of possible suboptimalities. Thus, assuming the learner can maintain a (possibly quite pessimistic) estimate of its suboptimality over all the rounds, we obtain a template for proving the iteration complexities of Theorems 1, 2, and 3. Note that ρ can be an arbitrarily small positive constant; even if there is a slim chance of decreasing the suboptimality, this is still sufficient because the monotone adversary ensures that the algorithm can never make negative progress.

Specifying progress distributions. We now discuss how we instantiate the above template for the β-smooth (Definition 1) and α-PŁ (Definition 2) case (Theorem 2). We focus on Theorem 2 for the sake of brevity; the proofs of Theorems 1 and 3 require some additional care but at a high level follow a similar structure. At step t, the algorithm maintains a guess $\boldsymbol{x}_t$ for the target $\boldsymbol{x}^\star$. It chooses some step size ε_t and a random vector $\boldsymbol{g}_t$ from $\varepsilon_t \cdot \mathsf{Unif}(\mathbb{S}_2^{d-1})$, where $\mathbb{S}_2^{d-1} \coloneqq \left\{ \boldsymbol{x} \in \mathbb{R}^d \ : \ \|\boldsymbol{x}\|_2 = 1 \right\}$. We then query $\boldsymbol{x}_t$ and $\boldsymbol{x}_t - \boldsymbol{g}_t$. The key observation is that with a constant probability, the angle between $\boldsymbol{g}_t$ and the gradient $\nabla f(\boldsymbol{x})$ is small. We will use this to show that the distribution $\boldsymbol{x}_t - \varepsilon_t \cdot \mathsf{Unif}(\mathbb{S}_2^{d-1})$ is a $(1, C_1, C_2)$-progress distribution (Definition 3) for constants C_1, C_2. Intuitively, this means that $\boldsymbol{x}_t - \boldsymbol{g}_t$ almost behaves like a step of gradient descent. To turn this observation into an algorithm, we need two main insights.

Step size schedule. The principal difficulty of this approach is to choose the step size ε_t. It is not immediately obvious how to do so since the algorithm does not observe any actual gradients

or function values. Hence, if our step sizes are too large, the algorithm may overshoot the optimal solution $\boldsymbol{x}^\star$ and therefore not actually improve the quality of its current solution $\boldsymbol{x}_t$. On the other hand, if our step sizes are too small, the algorithm may not make enough progress in each step, which undesirably increases both the iteration complexity and the total cost.

To address this, we carefully construct a step size schedule that relies on a pessimistic upper bound on the suboptimality of the algorithm's current solution. With this schedule, we show that in every step, one of two things happens – either the step size ε_t is small enough such that there is the possibility of the algorithm decreasing the cost, or it is too large. For the first case, we use β-smoothness (Definition 1) to prove that there is a constant probability that the algorithm finds a descent direction, which decreases the cost of its current solution substantially. For the second case, we use the α-PŁ condition (Definition 2) to prove that the cost the algorithm incurs in such steps is low. After enough steps, we can show that either the second case always holds (i.e. that the suboptimality is already desirably small) or the maximum cost that the algorithm can pay per round is small. We then decrease the step size ε_t by a constant factor, update the suboptimality estimate accordingly, and infinitely recurse.

Bounding the failure probability over infinite rounds. It now remains to show that the probability that the algorithm fails to make enough progress over *infinitely many rounds* is small. This is where the distinction between the two goals of Problem 1 becomes apparent. Specifically, even if we have a subroutine that, with high probability, outputs an ε-approximate solution, this does not immediately convert to an algorithm that can achieve bounded cost over an infinite number of rounds – note that the failure probabilities may accumulate in a divergent manner. Hence, we will require a more careful probabilistic analysis.

To overcome this challenge, we design the algorithm to run in phases $i = 1, 2, \ldots$. In phase i, we use a step size ε_t proportional to $2^{-i/2}$ and run phase i for $\sim id$ steps. Using the fact that the family of distributions we are using for sampling next steps are $(1, C_1, C_2)$-progress distributions, it will be enough to prove that $\sim d \cdot \beta/\alpha$ steps yield enough improving steps to decrease the suboptimality by a constant factor. We can therefore apply a Chernoff bound to conclude that the probability that the algorithm fails to make enough progress in phase i is at most $\exp(-id \cdot \beta/\alpha)$. Finally, we apply a union bound that the total probability of failure by $\exp(-d \cdot \beta/\alpha) \leq \exp(-d)$.

To bound the total cost over all phases $i \in \mathbb{N}_{\geq 1}$, we note that the sum of the suboptimalities in each round is of the form $d \sum_{i \geq 1} i 2^{-i} = O(d)$. The guarantee on the iteration complexity follows by noting that to achieve a suboptimality of 2^{-i}, the algorithm runs $d \sum_{j \leq i} i = O\left(i^2 \cdot d\right)$ iterations.

1.4. Related works

Dueling convex optimization. As already mentioned, our formulation in Problem 1 is an extension of dueling convex optimization in the noiseless setting [JNR12; SKM21; SKM22]. Jamieson, Nowak, and Recht [JNR12] employ a coordinate-descent algorithm to show for α-smooth and β-strongly convex f, $\tilde{O}(d\beta/\alpha \log(1/\varepsilon))$ queries suffice to learn an ε-optimal point. As mentioned earlier, it is not clear how to adapt their algorithm to handle monotone feedback. In addition, the papers [SKM21; SKM22] show results for more general classes of f and in the presence of noise (where the adversary can return invalid response with nonzero probability); see Section 1.5 for a discussion on extending our results to noisy settings. However, their algorithms explicitly rely on

sign feedback $f(\boldsymbol{x}_t^{(1)}) \overset{?}{>} f(\boldsymbol{x}_t^{(2)})$ to construct gradient estimators, which are not possible in the monotone adversary setting.

Monotone adversaries. Our setting is an example of learning with a monotone adversary, where an adversary can choose to improve the feedback or information the algorithm gets. A common characteristic is that the improved information may paradoxically break or harm the performance of a given algorithm that works with non-improved information. Monotone adversaries are often studied in the semi-random model literature [BS95; Fei21; Moi21] for statistical estimation problems [CG18; Moi21; KLLST22] as well as learning problems, i.e., linear classification with Massart noise [MN06; DGT19].

Preference-based feedback. Our formulation in this paper falls within the growing body of literature that tackles learning with preference-based feedback, where the algorithm does not learn *how good* its options were in an absolute sense, just which one(s) were better than others. Other natural problems with preference-based feedback are contextual search [LS18; LLV18; LLS20], contextual recommendation (also called contextual inverse optimization) [BFL21; GGKMLS21], and 1-bit matrix completion [DPVW14].

1.5. Future work

We now discuss two interesting directions our work leaves open.

Dueling optimization with a monotone adversary under noise models. The most pressing next step is to determine noise models under which we can either obtain algorithmic results or hardness for solving Problem 1. There are several natural noise models that one could study. As a first step, one can consider the most analogous extension of the noise model studied by Jamieson, Nowak, and Recht [JNR12] and Saha, Koren, and Mansour [SKM21]. In the simplest form, they study a noise model where with probability $1/2 + \nu$ for some parameter $0 < \nu \leq 1/2$, the adversary returns $\operatorname{argmin}_{i\in\{1,2\}} f(\boldsymbol{x}_t^{(i)})$, and with probability $1/2 - \nu$, the adversary returns $\operatorname{argmax}_{i\in\{1,2\}} f(\boldsymbol{x}_t^{(i)})$. Note that this is a straightforward noise model to handle if the adversary must return one of $\boldsymbol{x}_t^{(1)}$ or $\boldsymbol{x}_t^{(2)}$. The algorithm simply queries the same pair of points roughly ν^{-2} times, which by Hoeffding's inequality is enough to determine the index i that corresponded to the better of the two guesses. Additionally, this strategy can be implemented even without knowledge of the noise parameter ν; see Section 6 of [SKM21] for more details.

The natural extension of this noise model to our monotone feedback setting is as follows. With probability $1/2+\nu$, the adversary returns a response $\boldsymbol{r}_t$ that satisfies $f(\boldsymbol{r}_t) \leq \min\{f(\boldsymbol{x}_t^{(1)}), f(\boldsymbol{x}_t^{(2)})\}$. On the other hand, with probability $1/2 - \nu$, the adversary returns an arbitrary point in $\mathcal{X}$. It is immediate that there is no analogue of the "majority vote" strategy in this setting. We therefore believe that entirely new algorithmic ideas will be needed to address this noise model. On the other hand, it would be very interesting to show an impossibility result for this noise model. An impossibility result would imply that the monotone feedback makes the problem provably harder than the vanilla dueling optimization setting.

Another noise model of interest is one where the adversary's valid monotone feedback is perturbed by a random variable $\triangle$ where $\mathbb{E}[\triangle] = 0$ and $\mathsf{Cov}[\triangle] \preceq \sigma^2 \mathbf{I}_d$. To our knowledge, this noise model has not been considered in past works dealing with optimization with preference-based feedback,

even in the vanilla dueling setting (particularly the works of Jamieson, Nowak, and Recht [JNR12] and Saha, Koren, and Mansour [SKM21]).

Dueling optimization for other function classes. An orthogonal thread would be to identify other function classes and feasible regions $\mathcal{X}$ for which we can build algorithms for Problem 1. For instance, Saha, Koren, and Mansour [SKM21] obtain results for functions that are just β-smooth (and not necessarily α-PŁ or α-strongly convex). Our Theorem 2 uses the α-PŁ condition to prove that steps that do not make much progress also do not incur much cost. Hence, to remove the α-PŁ assumption, one would need to either avoid this line of reasoning or find another way to argue that the costs in low-progress rounds are not too high.

2. Proofs of upper bound results

In this section, we prove Theorem 8 (in which we construct and analyze a meta-algorithm for Problem 1 when the algorithm can sample next steps from progress distributions (Definition 3)). We then show how to use this framework to prove Theorem 1 (results for $f(\boldsymbol{x}) = \langle -\boldsymbol{x}, \boldsymbol{x}^\star \rangle$), Theorem 2 (results for $f(\boldsymbol{x})$ being β-smooth and α-PŁ), and Theorem 3 (results for $f(\boldsymbol{x}) = \|\boldsymbol{x} - \boldsymbol{x}^\star\|_2$), in that order. It will be helpful to recall the overview from Section 1.3 throughout this section.

We prove Theorem 8 in Section 2.1, Theorem 1 in Section 2.2, Theorem 2 in Section 2.3, and Theorem 3 in Section 2.4.

Before we jump into the main proofs, we will need a couple straightforward numerical inequalities.

Lemma 6. *For $r \in (0,1)$ and $1 \le p < 2$, we have $\sum_{i \ge 0} i \cdot r^{(1-p/2)i} \le \frac{r^{p/2+1}}{(r - r^{p/2})^2}$.*

Proof of Lemma 6. Recall that

$$\sum_{i \ge 0} r^{(1-p/2)i} = \frac{1}{1 - r^{1-p/2}}.$$

Taking the derivative of both sides with respect to r yields

$$\sum_{i \ge 0} (1 - p/2)\, i \cdot r^{(1-p/2)i-1} = \frac{(2-p) r^{p/2}}{2(r - r^{p/2})^2}.$$

We multiply both sides by r and divide both sides by $1 - p/2$; we conclude that

$$\sum_{i \ge 0} i \cdot r^{(1-p/2)i} = \frac{r^{p/2+1}}{(r - r^{p/2})^2}$$

which recovers the statement of Lemma 6. □

Lemma 7 (Inner product with a random vector)**.** *Let $\boldsymbol{g} \sim \mathsf{Unif}(\mathbb{S}_2^{d-1})$ and let $\boldsymbol{y} \in \mathbb{S}_2^{d-1}$ be fixed. Then*

$$\Pr_{\boldsymbol{g}} \left[\langle \boldsymbol{g}, \boldsymbol{y} \rangle \ge \frac{1}{2\sqrt{d}} \right] \ge \frac{1}{8}.$$

Proof. By rotational invariance, without loss of generality, we can let $\boldsymbol{y} = \boldsymbol{e}_1$. We apply Lemma 2.2 (a) due to Dasgupta and Gupta [DG03] with $\beta = 1/4$ to conclude that

$$\Pr_{\boldsymbol{g}}\left[\boldsymbol{g}_1^2 \leq \frac{1}{4d}\right] \leq \exp\left(\frac{1}{2}\left(1 - \frac{1}{4} + \ln\left(\frac{1}{4}\right)\right)\right) < \frac{3}{4}$$

which means that $\Pr_{\boldsymbol{g}}\left[|\langle \boldsymbol{g}, \boldsymbol{y}\rangle| \geq \frac{1}{2\sqrt{d}}\right] \geq \frac{1}{4}$. The result of Lemma 7 follows by symmetry. □

2.1. A general algorithm for Problem 1 with progress distributions

The goal of this subsection is to develop the general tools we need to prove our main results.

The key primitive of our analysis is a general algorithm (Algorithm 1) that solves Problem 1 when we are given certain convenient distributions from which we sample new guesses. We call these *progress distributions*; recall Definition 3.

Let us describe Algorithm 1. In each step, Algorithm 1 maintains a current guess $\boldsymbol{x}_t$ and chooses a slight perturbation of that guess $\boldsymbol{x}_t^+ \sim \mathcal{D}(\boldsymbol{x}_t)$, where $\mathcal{D}(\boldsymbol{x}_t)$ is a (p, γ, ρ)-progress distribution (Definition 3). Algorithm 1 then submits the pair of guesses $\{\boldsymbol{x}_t, \boldsymbol{x}_t^+\}$. To analyze Algorithm 1, the main observation is that with probability $\geq \gamma$, the point $\boldsymbol{x}_t^+$ substantially improves over the cost of $\boldsymbol{x}_t$ – this follows directly from Definition 3. We exploit this intuition to give our most general result (Theorem 8) and to prove the correctness of Algorithm 1.

Theorem 8. *Let $f\colon \mathcal{X} \to \mathbb{R}$. Let B and $\boldsymbol{x}_1 \in \mathcal{X}$ be such that $f(\boldsymbol{x}_1) - f(\boldsymbol{x}^\star) \leq B$. For $C > 0$, constant $r \in (0, 0.99)$, and for all $i \in \mathbb{N}_{\geq 1}$, suppose there exists intervals of the form $C \cdot \left[r^{i+1}, r^i\right]$ such that their union covers the interval $[0, B]$.*

If there exists a (p, γ, ρ)-progress distribution $\mathcal{D}_i(\boldsymbol{x})$ whenever $f(\boldsymbol{x}) - f(\boldsymbol{x}^\star) \in C \cdot \left[r^{i+1}, r^i\right]$ for all $i \geq 1$ and where p, γ, ρ do not depend on $\boldsymbol{x}$ and i, then there is an algorithm (Algorithm 1) for Problem 1 that, with probability at least $1 - \exp\left(-O\left(\frac{d}{\rho B^{p-1}}\right)\right)$, incurs total cost

$$O\left(\frac{B\log(1/r)}{B^{p-1}\gamma\rho\min\left\{r^{p(p-1/2-p)}, \left(r - r^{p/2}\right)^2\right\}} \cdot d\right).$$

Additionally, Algorithm 1 finds a point $\boldsymbol{x}$ satisfying $f(\boldsymbol{x}) - f(\boldsymbol{x}^\star) \leq \varepsilon$ in

$$O\left(\frac{1}{B^{p-1}\gamma\rho} \cdot d \cdot \log\left(\frac{B}{\varepsilon}\right)^2\right)$$

iterations with at least the aforementioned probability.

The proof of Theorem 8 has two main parts. In the first part, we will prove that for each value of i (call the set of timesteps belonging to a particular value of i "phase i"), the number of steps $T(i)$ is sufficient to ensure that the cost of the algorithm's solution decays gracefully with sufficiently large probability. In the second part, we will prove that the total cost the algorithm pays over all phases $i \geq 1$ is $\sim B \cdot d/\gamma\rho$ as promised. Theorem 8 will easily follow by combining these facts.

We start with stating and proving Lemma 9.

Algorithm 1 General recipe algorithm for dueling convex optimization

1: **Input**: Interaction with a monotone adversary $\mathcal{M}$ as defined in Problem 1; initial point $\boldsymbol{x}_1$ and bound B satisfying $f(\boldsymbol{x}_1) - f(\boldsymbol{x}^\star) \le B$; values C and r for which there exist corresponding intervals and (p, γ, ρ)-progress distribution families $\mathcal{D}_i$ (see the statement of Theorem 8).
2: Initialize $\boldsymbol{x}_1 = 0$, $t = 1$.
3: **for** $i = 1, \dots$ **do**
4: **for** $T(i) \coloneqq {}^{2i}/_{(\gamma \min(1,\rho))} \cdot (Cr^{i+1})^{-(p-1)} \cdot \log({}^1/_r) \cdot d$ iterations **do**
5: Sample $\boldsymbol{x}_t^+$ from $\mathcal{D}_i(\boldsymbol{x}_t)$.
6: **Submit guesses** $\{\boldsymbol{x}_t, \boldsymbol{x}_t^+\}$ and **receive response** $\boldsymbol{r}_t$.
7: Let $\boldsymbol{x}_{t+1} = \boldsymbol{r}_t$.
8: Update $t \leftarrow t + 1$.
9: **end for**
10: **end for**

Lemma 9. *Let $i \ge 1$ and let $T(i)$ be defined below (or see Line 4 of Algorithm 1).*

$$T(i) \coloneqq \frac{2i}{\gamma \min(1, \rho) \cdot (Cr^{i+1})^{p-1}} \cdot \log({}^1/_r) \cdot d.$$

Let t_i be the first iteration of phase i. If $f(\boldsymbol{x}_{t_i}) - f(\boldsymbol{x}^\star) \le Cr^i$, then, with probability $\ge 1 - r^{\frac{dB^{-(p-1)}}{4\rho} \cdot i}$, we have $f(\boldsymbol{x}_{t_i+T(i)+1}) - f(\boldsymbol{x}^\star) \le Cr^{i+1}$.

Proof of Lemma 9. Assume that we have $f(\boldsymbol{x}_{t_i}) - f(\boldsymbol{x}^\star) \in C \cdot \left[r^{i+1}, r^i\right]$ (otherwise, we are done immediately).

Define the indicator random variable Y_t as follows.

$$Y_t \coloneqq \mathbb{1}\left\{\frac{f(\boldsymbol{x}_t) - f(\boldsymbol{x}_t^+)}{(f(\boldsymbol{x}_t) - f(\boldsymbol{x}^\star))^p} \ge \frac{\rho}{d}\right\}.$$

Consider the distribution of guesses $\mathcal{D}_i$ (let us omit the argument $\boldsymbol{x}_t$ for the sake of brevity). Since $\mathcal{D}_i$ is a (p, γ, ρ)-progress distribution, we have

$$\Pr_{\boldsymbol{x}^+ \sim \mathcal{D}_i}\left[\frac{f(\boldsymbol{x}_t) - f(\boldsymbol{x}_t^+)}{f(\boldsymbol{x}_t) - f(\boldsymbol{x}^\star)} \ge \frac{\rho}{d} \cdot \left(Cr^{i+1}\right)^{p-1}\right] \ge \Pr_{\boldsymbol{x}^+ \sim \mathcal{D}_i}[Y_t = 1] \ge \gamma.$$

Call every step t for which $Y_t = 1$ a "successful step." Let us give a high-probability count on the number of successful steps. Recall that a form of the Chernoff bound states that, for $\delta \in [0, 1]$ and independent indicator random variables Y_j,

$$\Pr\left[\sum_{j=t_i}^{t_i+T(i)} Y_j \le (1-\delta)\mathbb{E}\left[\sum_{j=t_i}^{t_i+T(i)} Y_j\right]\right] \le \exp\left(-\frac{\delta^2 \cdot \mathbb{E}\left[\sum_{j=t_i}^{t_i+T(i)} Y_j\right]}{2}\right).$$

Applying the Chernoff bound with $\delta = {}^1/_2$ yields

$$\Pr\left[\sum_{j=t_i}^{t_i+T(i)} Y_j \le \frac{T(i)\gamma}{2}\right] \le \exp\left(-\frac{i \cdot \frac{d}{\rho(Cr^{i+1})^{p-1}} \cdot \log({}^1/_r)}{4}\right) \le r^{\frac{dB^{-(p-1)}}{4\rho} \cdot i}$$

where we use $Cr^{i+1} \le Cr^2 \le B$.

It remains to show that after at least $T(i)\gamma/2$ successful steps, we have $f(\boldsymbol{x}_{t_i+T(i)+1}) - f(\boldsymbol{x}^\star) \le Cr^{i+1}$. Recall that we assume that $f(\boldsymbol{x}_{t_i}) - f(\boldsymbol{x}^\star) \ge Cr^{i+1}$ and note that for every successful step, we have

$$\frac{f(\boldsymbol{x}_t) - f(\boldsymbol{x}_t^+)}{f(\boldsymbol{x}_t) - f(\boldsymbol{x}^\star)} \ge \frac{\rho}{d} \cdot \left(Cr^{i+1}\right)^{p-1}$$

which implies

$$\frac{f(\boldsymbol{x}_t^+) - f(\boldsymbol{x}^\star)}{f(\boldsymbol{x}_t) - f(\boldsymbol{x}^\star)} \le 1 - \frac{\rho}{d} \cdot \left(Cr^{i+1}\right)^{p-1}.$$

We multiply over all steps in phase i, giving

$$\begin{aligned}\frac{f(\boldsymbol{x}_{t_i+T(i)+1}) - f(\boldsymbol{x}^\star)}{f(\boldsymbol{x}_{t_i}) - f(\boldsymbol{x}^\star)} &= \prod_{t=t_i}^{t_i+T(i)} \frac{f(\boldsymbol{x}_{t+1}) - f(\boldsymbol{x}^\star)}{f(\boldsymbol{x}_t) - f(\boldsymbol{x}^\star)} \le \left(1 - \frac{\rho}{d} \cdot \left(Cr^{i+1}\right)^{p-1}\right)^{T(i)\gamma/2} \\ &\le \left(1 - \frac{2i}{\gamma} \cdot \frac{\log\left(1/r\right)}{T(i)}\right)^{T(i)\gamma/2} \le \exp\left(\frac{2i}{\gamma} \cdot \frac{\log\left(1/r\right)}{T(i)} \cdot \frac{T(i)\gamma}{2}\right) = r^i \le r.\end{aligned}$$

Finally, recall that $f(\boldsymbol{x}_{t_i}) - f(\boldsymbol{x}^\star) \le Cr^i$. Combining this with the above gives $f(\boldsymbol{x}_{t_i+T(i)+1}) - f(\boldsymbol{x}^\star) \le Cr^{i+1}$, concluding the proof of Lemma 9. □

Next, we have Lemma 10, which controls the total cost that Algorithm 1 incurs assuming that the cost is sufficiently low in each phase.

Lemma 10. *For a timestep t, let $i(t)$ be the phase that t belongs to.*

If for all t we have $f(\boldsymbol{x}_t) - f(\boldsymbol{x}^\star) \le Cr^{i(t)}$, then Algorithm 1 incurs total cost

$$O\left(\frac{B\log\left(1/r\right)}{C^{p-1}\gamma\rho\min\left\{r^{p(p-1/2-p)}, \left(r - r^{p/2}\right)^2\right\}} \cdot d\right).$$

Proof of Lemma 10. Recall throughout this proof that $r \le 0.99$ and p is a constant such that $p < 2$.

Observe that in phase i, the algorithm incurs cost at most

$$T(i) \cdot Cr^i = \frac{2i}{\gamma} \cdot \frac{dCr^i}{\rho\left(Cr^{i+1}\right)^{p-1}} \cdot \log\left(1/r\right) = \frac{2i}{\gamma} \cdot \frac{d\log\left(1/r\right)}{\rho C^{p-2}} \cdot r^{i-(i+1)(p-1)}.$$

We will find a threshold i_p for which for all $i \ge i_p$, the above cost is exponentially decaying. This will allow us to control the sum of the costs over infinitely many rounds. We choose $i_p = 2 \cdot \lceil (p-1)/(2-p) \rceil$. Notice that for all $i \ge i_p$, the exponent on r can be bounded as

$$i - (i+1)(p-1) = i(2-p) - (p-1) \ge \left(1 - \frac{p}{2}\right) i.$$

Note that this also implies that $(i_p + 1)(p-1) \le p/2 \cdot i_p = p\lceil (p-1)/(2-p) \rceil$.

To total the cost, we consider two cases. First, suppose $1 \le i \le i_p - 1$. Observe that in each of these phases, we pay cost at most B, so we have

$$\sum_{i=1}^{i_p-1} B \cdot T(i) \le B\left(T_{i_p} \cdot i_p\right) = 2B\left(2 \cdot \frac{p-1}{2-p}\right)^2 \cdot \frac{\log\left(1/r\right)}{\gamma} \cdot \frac{d}{\rho C^{p-1} r^{(i_p+1)(p-1)}}$$
$$\le 2B\left(2 \cdot \frac{p-1}{2-p}\right)^2 \cdot \frac{\log\left(1/r\right)}{\gamma} \cdot \frac{d}{\rho C^{p-1} r^{p(p-1/2-p)}}. \tag{2.1}$$

Next, we sum over all phases $i \ge i_p$. We obtain a cost that is at most

$$\sum_{i \ge i_p} \frac{2i}{\gamma} \cdot \frac{d \log\left(1/r\right)}{\rho C^{p-2}} \cdot r^{i-(i+1)(p-1)} \le \frac{2d \log\left(1/r\right)}{\gamma\rho \cdot C^{p-2}} \sum_{i \ge i_p} i \cdot r^{(1-p/2)i} \le \frac{2d \log\left(1/r\right)}{\gamma\rho \cdot C^{p-2}} \cdot \frac{r^{p/2+1}}{(r - r^{p/2})^2} \tag{2.2}$$

where the last inequality follows from Lemma 6. Combining (2.1) and (2.2) yields

$$2B\left(2 \cdot \frac{p-1}{2-p}\right)^2 \cdot \frac{\log\left(1/r\right)}{\gamma} \cdot \frac{d}{\rho C^{p-1} r^{p(p-1/2-p)}} + \frac{2d \log\left(1/r\right)}{\gamma\rho \cdot C^{p-2}} \cdot \frac{r^{p/2+1}}{(r - r^{p/2})^2}$$
$$\le 2B\left(2 \cdot \frac{p-1}{2-p}\right)^2 \cdot \frac{\log\left(1/r\right)}{\gamma} \cdot \frac{d}{\rho C^{p-1} r^{p(p-1/2-p)}} + \frac{2d \log\left(1/r\right)}{\gamma\rho \cdot C^{p-1}} \cdot \frac{B}{(r - r^{p/2})^2}$$
$$= O\left(\frac{B \log\left(1/r\right)}{C^{p-1}\gamma\rho \min\left\{r^{p(p-1/2-p)}, \left(r - r^{p/2}\right)^2\right\}} \cdot d\right)$$

This concludes the proof of Lemma 10. □

We are now ready to prove Theorem 8.

Proof of Theorem 8. It is sufficient to prove that with probability $\ge 1 - \exp\left(-O\left(\frac{d}{\rho B^{p-1}}\right)\right)$, at the end of phase i, we have $f(\boldsymbol{x}_t) - f(\boldsymbol{x}^\star) \le Cr^{i+1}$. Recall the conclusion of Lemma 9 and that $f(\boldsymbol{x}_1) - f(\boldsymbol{x}^\star) \le B \le Cr$; by a union bound, we have for all phases i that $f(\boldsymbol{x}_t) - f(\boldsymbol{x}^\star) \le Cr^{i+1}$ with probability

$$1 - \sum_{i \ge 1} r^{\frac{dB^{-(p-1)}}{4\rho} \cdot i} \ge 1 - \exp\left(-O\left(\frac{d}{\rho B^{p-1}}\right)\right)$$

where we use $0 < r < 0.99$. The first part of Theorem 8 now follows directly from applying Lemma 10. The rest of the statement of Theorem 8 follows by noting that

$$\sum_{j \le i} T(j) = \frac{2C^{-(p-1)} \log\left(1/r\right)}{\gamma \min(1, \rho)} \cdot d \cdot \sum_{j \le i} j\left(r^{-(j+1)(p-1)}\right) \lesssim \frac{2C^{-(p-1)}}{\gamma \min(1, \rho)} \cdot d \cdot i^2.$$

where we again use $r < 0.99$. We set $\varepsilon = Cr^i$ and conclude. □

2.2. Proof of Theorem 1

The goal of this subsection is to prove Theorem 1.

Our plan will be to use the general guarantee of Theorem 8. Thus, the main task is to prove that there is an appropriate interval cover and corresponding sequence $\mathcal{D}_i(\boldsymbol{x})$ of progress distributions for all $\boldsymbol{x}$ belonging to phase i that satisfy the conditions of Theorem 8.

We prove this fact in Lemma 11. We remark that we made no effort to optimize the numerical constants; we choose the constants that appear in the Lemma statement to simplify calculations, as these will not impact our asymptotic results.

Lemma 11. *Let $f : \mathbb{R}^d \to \mathbb{R}$ be the negative inner product function defined on $\mathbb{S}_2^{d-1}$ with respect to some unknown target $\boldsymbol{x}^\star$. Then for any $\boldsymbol{x}$ for which $f(\boldsymbol{x}) - f(\boldsymbol{x}^\star) \in \left[10^{-(i+1)}, 10^{-i}\right]$ and for which $\langle \boldsymbol{x}^\star, \boldsymbol{x}\rangle > 0$, there is a $(1.5, 10^{-1}, 10^{-4})$-progress distribution (Definition 3) that can be computed in time $O(d)$.*

Proof of Lemma 11. We explain the construction of the distribution $\mathcal{D}(\boldsymbol{x})$.

Impose the following coordinates on $\mathbb{R}^d$. Let the first coordinate x_1 be the direction of $\boldsymbol{x}$, and the remaining $d-1$ coordinates be an arbitrary coordinate system for the perpendicular directions. Then, $\boldsymbol{x}$ has coordinates $(1, 0 \ldots)$. Next, let $z \coloneqq 10^{-(i+1)}$ and $s \coloneqq \frac{z}{10\sqrt{d-1}}$. Let $\boldsymbol{s}$ be a point randomly drawn from a $d-1$ dimensional sphere of radius s whose coordinates are denoted $s_1 \ldots s_{d-1}$. Then, the distribution $\mathcal{D}(\boldsymbol{x})$ is the distribution of $(\sqrt{1-s^2}, s_1 \ldots s_{d-1})$. It is easy to verify that these points lie on $\mathbb{S}_2^{d-1}$.

This distribution can be computed in time $O(d)$. We will now show that it is a $(1.5, 10^{-1}, 10^{-4})$ progress distribution.

Let $z_0 \coloneqq f(\boldsymbol{x}) - f(\boldsymbol{x}^\star)$; recall that $z_0 \in [0, 1]$ and $z \le z_0 \le 10z$. We write the target vector $\boldsymbol{x}^\star = (1-z_0)\boldsymbol{x} + \sqrt{1-(1-z_0)^2}\boldsymbol{y} = (1-z_0)\boldsymbol{x} + \sqrt{2z_0 - z_0^2}\boldsymbol{y}$ where $\boldsymbol{y}$ is a unit vector and $\langle \boldsymbol{y}, \boldsymbol{x}\rangle = 0$. Note that this expression holds because $\langle \boldsymbol{x}^\star, \boldsymbol{x}\rangle = 1 - z_0$.

Let $\boldsymbol{x}^+$ be a random point chosen from $\mathcal{D}(\boldsymbol{x})$. Let $y_1 \ldots y_{d-1}$ be the coordinates of $\boldsymbol{y}$ in the $d-1$ dimensional coordinate plane perpendicular to $\boldsymbol{x}$ defined above. We compute

$$\langle \boldsymbol{x}^\star, \boldsymbol{x}^+\rangle = \left(1 - z_0, \sqrt{2z_0 - z_0^2}y_1 \ldots \sqrt{2z_0 - z_0^2}y_{d-1}\right) \cdot \left(\sqrt{1-s^2}, s_1 \ldots s_{d-1}\right) \tag{2.3}$$

$$= (1-z_0)\sqrt{1-s^2} + \left(\sqrt{2z_0 - z_0^2}y_1 \ldots \sqrt{2z_0 - z_0^2}y_{d-1}\right) \cdot (s_1 \ldots s_{d-1}). \tag{2.4}$$

By Lemma 7, we have (note that we weaken the constants from Lemma 7 for numerical convenience later in the proof)

$$\Pr_{\boldsymbol{s}}\left[\left(\sqrt{2z_0 - z_0^2}y_1 \ldots \sqrt{2z_0 - z_0^2}y_{d-1}\right) \cdot (s_1 \ldots s_{d-1}) \ge \frac{0.1}{\sqrt{d-1}} \cdot \sqrt{2z_0 - z_0^2} \cdot s\right] \ge 0.1.$$

Because $2z_0 - z_0^2 \ge z_0 \ge z$, we have

$$\frac{0.1}{\sqrt{d-1}} \cdot \sqrt{2z_0 - z_0^2} \cdot s \ge \frac{s\sqrt{z}}{10\sqrt{d-1}}.$$

In turn, this shows

$$\Pr_s\left[\left(\sqrt{2z_0 - z_0^2}y_1 \dots \sqrt{2z_0 - z_0^2}y_{d-1}\right)\cdot(s_1 \dots s_{d-1}) \geq \frac{s\sqrt{z}}{10\sqrt{d-1}}\right] \geq 0.1.$$

Combining this with Equation 2.3, we obtain

$$\Pr_{\boldsymbol{x}^+\sim\mathcal{D}_{\boldsymbol{x}}}\left[\langle \boldsymbol{x}^\star, \boldsymbol{x}^+\rangle \geq (1-z_0)\sqrt{1-s^2} + \frac{s\sqrt{z}}{10\sqrt{d-1}}\right] \geq 0.1. \tag{2.5}$$

Now, we will find a lower bound for $(1-z_0)\sqrt{1-s^2} + \frac{s\sqrt{z}}{10\sqrt{d-1}}$. We have

$$\begin{aligned}(1-z_0)\sqrt{1-s^2} + \frac{s\sqrt{z}}{10\sqrt{d-1}} &\geq (1-z_0)(1-s^2) + \frac{s\sqrt{z}}{10\sqrt{d-1}}\\ &= (1-z_0)(-s^2) + \frac{s\sqrt{z}}{10\sqrt{d-1}} + (1-z_0)\\ &\geq (1-z)(-s^2) + \frac{s\sqrt{z}}{10\sqrt{d-1}} + (1-z_0).\end{aligned}$$

Now, using that $s = \frac{z}{10\sqrt{d-1}}$, we get

$$\begin{aligned}(1-z)(-s^2) + \frac{s\sqrt{z}}{10\sqrt{d-1}} + (1-z_0) &= (1-10s\sqrt{d-1})(-s^2) + \frac{s^{3/2}}{\sqrt{10}(d-1)^{1/4}} + (1-z_0)\\ &= -s^2 + 10s^3\sqrt{d-1} + \frac{s^{3/2}}{\sqrt{10}(d-1)^{1/4}} + (1-z_0)\\ &\geq \left(10s^3\sqrt{d-1} + \frac{s^{3/2}}{8(d-1)^{1/4}} - s^2\right)\\ &\quad + \frac{s^{3/2}}{6(d-1)^{1/4}} + (1-z_0)\end{aligned}$$

where the last line follows from $1/\sqrt{10} > 1/8 + 1/6$. Finally, applying weighted AM-GM lets us see

$$10s^3\sqrt{d-1} + \frac{s^{3/2}}{8(d-1)^{1/4}} \geq \frac{3}{2^{2/3}}\left(10s^3\sqrt{d-1}\right)^{1/3}\left(\frac{s^{3/2}}{8(d-1)^{1/4}}\right)^{2/3} = \frac{3\cdot 10^{1/3}}{2^{2/3}2^2}s^2 > s^2$$

where we use a weight of $1/3$ on the first term and a weight of $2/3$ on the second term. We now write

$$(1-z_0)\sqrt{1-s^2} + \frac{s\sqrt{z}}{10\sqrt{d-1}} \geq \frac{s^{3/2}}{6(d-1)^{1/4}} + (1-z_0).$$

Substituting s once again and recalling that $\langle \boldsymbol{x}^\star, \boldsymbol{x}\rangle = (1-z_0)$ and $z \geq \frac{z_0}{10} = \frac{\langle \boldsymbol{x}^\star, \boldsymbol{x}^\star - \boldsymbol{x}\rangle}{10}$, we get

$$(1-z_0)\sqrt{1-s^2} + \frac{s\sqrt{z}}{10\sqrt{d-1}} \geq \frac{z^{3/2}}{6\cdot 10^{3/2}\cdot d} + \langle \boldsymbol{x}^\star, \boldsymbol{x}\rangle > \frac{10^{-4}\langle \boldsymbol{x}^\star, \boldsymbol{x}^\star - \boldsymbol{x}\rangle^{3/2}}{d} + \langle \boldsymbol{x}^\star, \boldsymbol{x}\rangle.$$

Combining this with (2.5), we now have

$$\Pr_{\boldsymbol{x}^+\sim\mathcal{D}(\boldsymbol{x})}\left[\langle\boldsymbol{x}^\star,\boldsymbol{x}^+\rangle \geq \frac{10^{-4}\langle\boldsymbol{x}^\star,\boldsymbol{x}^\star-\boldsymbol{x}\rangle^{3/2}}{d}+\langle\boldsymbol{x}^\star,\boldsymbol{x}\rangle\right]\geq 0.1$$

which means that

$$\Pr_{\boldsymbol{x}^+\sim\mathcal{D}(\boldsymbol{x})}\left[\langle\boldsymbol{x}^\star,\boldsymbol{x}^+-\boldsymbol{x}\rangle \geq \frac{10^{-4}\langle\boldsymbol{x}^\star,\boldsymbol{x}^\star-\boldsymbol{x}\rangle^{3/2}}{d}\right]\geq 0.1.$$

This exactly aligns with the definition of a $(1.5, 10^{-1}, 10^{-4})$ progress distribution, completing the proof of Lemma 11. □

We will now conclude Theorem 1 using Theorem 8.

Proof of Theorem 1. To apply Theorem 8, we need to present C, r, and a sequence of parameterizations $\mathcal{D}_i$ that satisfy the premises.

Set $r = 0.1$ and $C = 10$. By Lemma 11, we can find progress distributions for each interval $[Cr^i, Cr^{i-1}]$ of suboptimality of the current function value, since we can find such progress distributions as long as the suboptimality of the function is at most 1.

Note that the algorithm can begin with a point $\boldsymbol{x}$ where $f(\boldsymbol{x}) - f(\boldsymbol{x}^\star) < 1$ by first querying two opposite points on a sphere; one can easily see that at least one of the two points queried satisfies $f(\boldsymbol{x}) - f(\boldsymbol{x}^\star) < 1$.

We therefore conclude the proof of Theorem 1. □

2.3. Proof of Theorem 2

The goal of this subsection is to prove Theorem 2.

As before, we use the general guarantee of Theorem 8 via proving that there is an appropriate interval cover and corresponding sequence $\mathcal{D}_i(\boldsymbol{x})$ of progress distributions for all $\boldsymbol{x}$ for which $f(\boldsymbol{x}) - f(\boldsymbol{x}^\star) \in \left[Cr^{i+1}, Cr^i\right]$.

We prove this fact in Lemma 12.

Lemma 12. *Fix $i \in \mathbb{N}_{\geq 1}$. Let $\varepsilon_i = \sqrt{2B\alpha/\beta^2}\cdot 1/\sqrt{d}\cdot 2^{-i/2-1}$. If f is β-smooth and α-PŁ, and if we have $f(\boldsymbol{x}) - f(\boldsymbol{x}^\star) \in B\cdot\left[2^{-i}, 2^{-i+1}\right]$, then the distribution $\mathcal{D}_i(\boldsymbol{x}) = \boldsymbol{x} + \varepsilon_i\cdot\mathsf{Unif}(\mathbb{S}_2^{d-1})$ is a $(1,\gamma,\rho)$-progress distribution for $(\gamma,\rho) = (1/8, \alpha/8\beta)$.*

Proof of Lemma 12. Let $\boldsymbol{g} \coloneqq \boldsymbol{x} - \boldsymbol{x}^+$.

It is sufficient to consider the case where we have $\|\boldsymbol{g}\|_2 \leq \frac{1}{2\beta}\cdot\frac{\|\nabla f(\boldsymbol{x})\|_2}{\sqrt{d}}$. To see this, suppose this is not the case. We apply the PŁ inequality and write

$$f(\boldsymbol{x}) - f(\boldsymbol{x}^\star) \leq \frac{1}{2\alpha}\|\nabla f(\boldsymbol{x})\|_2^2 \leq d\cdot\frac{2\beta^2}{\alpha}\|\boldsymbol{g}\|_2^2 = d\cdot\frac{2\beta^2}{\alpha}\left(\sqrt{\frac{2B\alpha}{\beta^2 d}}\cdot\frac{1}{2^{i/2+1}}\right)^2 = \frac{B}{2^i}$$

which implies that the suboptimality $f(\boldsymbol{x}) - f(\boldsymbol{x}^\star)$ does not belong to the range we are considering.

Next, we use Lemma 7 to write the below.

$$\Pr_{\boldsymbol{g}}\left[\left\langle \frac{\nabla f(\boldsymbol{x})}{\|\nabla f(\boldsymbol{x})\|_2}, \frac{\boldsymbol{g}}{\|\boldsymbol{g}\|_2} \right\rangle \geq \frac{1}{2\sqrt{d}}\right] \geq \frac{1}{8}.$$

By Definition 1, we have for a β-smooth function and for any $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$ that

$$|f(\boldsymbol{x}) - f(\boldsymbol{y}) - \langle \nabla f(\boldsymbol{y}), \boldsymbol{x} - \boldsymbol{y}\rangle| \leq \frac{\beta}{2} \cdot \|\boldsymbol{x} - \boldsymbol{y}\|_2^2,$$

from which it easily follows that

$$|f(\boldsymbol{x} - \boldsymbol{g}) - f(\boldsymbol{x}) + \langle \nabla f(\boldsymbol{x}), \boldsymbol{g}\rangle| \leq \frac{\beta}{2} \cdot \|\boldsymbol{g}\|_2^2.$$

The above rearranges to

$$\begin{aligned} f(\boldsymbol{x}) - f(\boldsymbol{x} - \boldsymbol{g}) &\geq \langle \nabla f(\boldsymbol{x}), \boldsymbol{g}\rangle - \frac{\beta}{2} \cdot \|\boldsymbol{g}\|_2^2 \\ &= \|\nabla f(\boldsymbol{x})\|_2 \cdot \|\boldsymbol{g}\|_2 \cdot \left(\left\langle \frac{\nabla f(\boldsymbol{x})}{\|\nabla f(\boldsymbol{x})\|_2}, \frac{\boldsymbol{g}}{\|\boldsymbol{g}\|_2} \right\rangle - \frac{\beta/2 \cdot \|\boldsymbol{g}\|_2}{\|\nabla f(\boldsymbol{x})\|_2}\right) \\ &\geq \frac{\beta}{2} \cdot \|\boldsymbol{g}\|_2^2 \qquad \text{with probability } > 1/8 \end{aligned}$$

We therefore conclude that with probability $> \gamma \coloneqq 1/8$,

$$f(\boldsymbol{x}) - f(\boldsymbol{x} - \boldsymbol{g}) \geq \frac{\beta}{2} \cdot \|\boldsymbol{g}\|_2^2 = \frac{\beta}{2} \cdot \left(\sqrt{\frac{2B\alpha}{\beta^2 d}} \cdot \frac{1}{2^{i/2+1}}\right)^2 = \frac{\alpha}{\beta} \cdot \frac{B}{d} \cdot \frac{1}{2^{i+2}}.$$

This means that

$$\frac{f(\boldsymbol{x}) - f(\boldsymbol{x} - \boldsymbol{g})}{f(\boldsymbol{x}) - f(\boldsymbol{x}^\star)} \geq \frac{\alpha/\beta \cdot B/d \cdot 1/2^{i+2}}{B/2^{i-1}} = \frac{\alpha}{\beta} \cdot \frac{1}{8d}$$

which means we can take $\rho = \alpha/8\beta$. This concludes the proof of Lemma 12. □

The proof of Theorem 2 follows very easily from Lemma 12.

Proof of Theorem 2. Our plan is to apply Theorem 8. To do so, we need to present C, r, and a sequence of $\mathcal{D}_i$ that satisfy the premise of Theorem 8. We will use the settings of these objects guaranteed by Lemma 12.

Let $C = 2B$ and $r = 1/2$. It is clear that the intervals given by Lemma 12 cover $[0, B]$, and so for every $i \geq 1$, there exists a corresponding $(1, 1/8, \alpha/8\beta)$-progress distribution family $\mathcal{D}_i$. We now apply Lemma 12 along with Theorem 8 to conclude the proof of Theorem 2. □

2.4. Proof of Theorem 3

In this subsection, we prove Theorem 3.

Again, we present an appropriate interval cover and corresponding sequence of progress distributions $\mathcal{D}_i(\boldsymbol{x})$ that satisfy the conditions of Theorem 8. See Lemma 13.

Lemma 13. *Fix $i \in \mathbb{N}_{\geq 1}$. Let $\varepsilon = 1/\sqrt{d} \cdot 2^{-i/2}$. If $f \colon \mathcal{B}_2^d \to \mathbb{R}$ is $f(\boldsymbol{x}) = \|\boldsymbol{x} - \boldsymbol{x}^\star\|_2$ for $\boldsymbol{x}^\star \in \mathcal{B}_2^d$ and if $\|\boldsymbol{x} - \boldsymbol{x}^\star\|_2 \leq \sqrt{2} \cdot \left[2^{-(i+1)/2}, 2^{-i/2}\right]$, then there exists a distribution $\mathcal{D}(\boldsymbol{x})$ that can be efficiently sampled from and is a $(1, \gamma, \rho)$-progress distribution for $(\gamma, \rho) = (1/8, 1/8)$.*

Proof of Lemma 13. Let $\boldsymbol{x}^+$ have distribution

$$\frac{\boldsymbol{x} - \boldsymbol{g}}{\max\{1, \|\boldsymbol{x} - \boldsymbol{g}\|_2\}}, \qquad \text{where } \boldsymbol{g} \sim \varepsilon_t \cdot \mathsf{Unif}(\mathbb{S}_2^{d-1}). \tag{2.6}$$

Note that this distribution can be described as, "add a uniformly random direction of length ε_t to $\boldsymbol{x}$ and project the result back onto $\mathcal{X} = \mathcal{B}_2^d$."

It is easy to see that $\|\boldsymbol{x}^+\|_2 \leq 1$, so the iterates of Algorithm 1 will always remain inside $\mathcal{B}_2^d$. We now prove that $\mathcal{D}$ as described above in fact is a $(1, \gamma, \rho)$-progress distribution for the promised parameters.

First, use the fact that $\|\boldsymbol{x} - \boldsymbol{x}^\star\|_2^2$ is 2-smooth and 2-PŁ along with Lemma 12 to conclude that

$$\Pr_{\boldsymbol{g}}\left[\frac{\|\boldsymbol{x} - \boldsymbol{x}^\star\|_2^2 - \|\boldsymbol{x} - \boldsymbol{g} - \boldsymbol{x}^\star\|_2^2}{\|\boldsymbol{x} - \boldsymbol{x}^\star\|_2^2} \geq \frac{2\rho}{d}\right] \geq \gamma.$$

Condition on this event. A basic property of the Euclidean projection onto a convex set implies that

$$\left\|\boldsymbol{x}^+ - \boldsymbol{x}^\star\right\|_2^2 \leq \|(\boldsymbol{x} - \boldsymbol{g}) - \boldsymbol{x}^\star\|_2^2$$

which yields

$$\Pr_{\boldsymbol{g}}\left[\frac{\|\boldsymbol{x} - \boldsymbol{x}^\star\|_2^2 - \|\boldsymbol{x}^+ - \boldsymbol{x}^\star\|_2^2}{\|\boldsymbol{x} - \boldsymbol{x}^\star\|_2^2} \geq \frac{2\rho}{d}\right] \geq \gamma.$$

Finally, observe that the above event implies

$$\left(\frac{\|\boldsymbol{x}^+ - \boldsymbol{x}^\star\|_2}{\|\boldsymbol{x} - \boldsymbol{x}^\star\|_2}\right)^2 \leq \left(\sqrt{1 - \frac{2\rho}{d}}\right)^2 \leq \left(1 - \frac{\rho}{d}\right)^2.$$

Taking the square root of both sides and rearranging concludes the proof of Lemma 13. □

We remark that the above proof goes through if $\mathcal{X}$ is an arbitrary convex set; we simply replace (2.6) with $\Pi_{\mathcal{X}}(\boldsymbol{x} - \boldsymbol{g})$, where $\Pi_{\mathcal{X}}(\boldsymbol{z})$ is the Euclidean projection of $\boldsymbol{z}$ onto $\mathcal{X}$.

Now, the proof of Theorem 3 will follow in a very similar manner to that of Theorem 2.

Proof of Theorem 3. To apply Theorem 8, we need to present C, r, and a sequence of distribution parameterizations $\mathcal{D}_i$ that satisfy the premises.

Let $\boldsymbol{x}_1 = 0$. It is clear that $\|\boldsymbol{x}^\star\|_2 \le 1 = B$, which means that the intervals of the form $\sqrt{2} \cdot \left[2^{-(i+1)/2}, 2^{-i/2}\right]$ for $i \ge 1$ cover the interval $[0, 1]$. Hence, for every $i \ge 1$, there exists a corresponding $(1, 1/8, 1/8)$-progress distribution. Theorem 3 follows immediately. □

3. Proofs of lower bound results

In this section, we will prove Theorem 4 and Corollary 5. We first state the following well-known fact (see, e.g., [Ver18]) that there exist $2^{\Omega(d)}$ points inside the unit ℓ_2 ball which are sufficiently far apart from one another.

Fact 14. *There exists a subset $S \subset \mathcal{B}_2^d$ such that $|S| = 2^{\Omega(d)}$, and for all $\boldsymbol{x}, \boldsymbol{y} \in S$ such that $\boldsymbol{x} \neq \boldsymbol{y}$, we have $\|\boldsymbol{x} - \boldsymbol{y}\|_2 \ge 0.1$.*

We are now ready to prove Theorem 4.

Proof of Theorem 4. We actually prove the lower bound even when the adversary must return the item *in the list* with smallest function value (breaking ties consistently, e.g., according to lexicographic order). Since the adversary is only weaker in this case, this implies the lower bound for the monotone adversary.

By Yao's Lemma [Yao77], it suffices to give a distribution over instances such that every deterministic algorithm satisfies the conclusions of the theorem. Hence, choose S from Fact 14 and let $\boldsymbol{x}^\star$ be sampled uniformly from S.

Fix any deterministic algorithm. The deterministic algorithm branches into at most m states every round, depending on the response the adversary gives. Therefore after $r := \lfloor \log_m |S| \rfloor - 1$ rounds, the algorithm has at most $m^r < \frac{1}{2}|S|$ distinct states. Each of these states Q can be represented as a tuple of the form $\left\{(\boldsymbol{x}_t^{(1)}, \cdots, \boldsymbol{x}_t^{(m)}, i_t)\right\}_{t \in [r]}$, where the $\boldsymbol{x}_t^{(i)} \in \mathcal{X}$ and the $i_t \in [m]$, which represents a set of the algorithm's guesses as well as the closest-point responses for the first r rounds.

Cost lower bound. Let us denote $c_r(\boldsymbol{x}^\star, Q)$ to be the total cost incurred for the state Q if the target is $\boldsymbol{x}^\star \in S$. We claim that all but at most one $\boldsymbol{x}^\star \in S$ have $c_r(\boldsymbol{x}^\star, Q) > 0.05r$. Suppose there were two points $\boldsymbol{x}^\star$ and $\boldsymbol{x}^{\star\prime}$ which had $c_r(\boldsymbol{x}^\star, Q) < 0.05r$. Then $c_r(\boldsymbol{x}^\star, Q) + c_r(\boldsymbol{x}^{\star\prime}, Q) < 0.1r$, so there exists some round $t \in [r]$ for which

$$\max\left\{\left\|\boldsymbol{x}_t^{(1)} - \boldsymbol{x}^\star\right\|, \cdots, \left\|\boldsymbol{x}_t^{(m)} - \boldsymbol{x}^\star\right\|\right\} + \max\left\{\left\|\boldsymbol{x}_t^{(1)} - \boldsymbol{x}^{\star\prime}\right\|, \cdots, \left\|\boldsymbol{x}_t^{(m)} - \boldsymbol{x}^{\star\prime}\right\|\right\} < 0.1.$$

However, this cannot hold by triangle inequality since $\boldsymbol{x}^\star$ and $\boldsymbol{x}^{\star\prime}$ are well-separated.

For any target $\boldsymbol{x}^\star$, the cost paid in the first r steps is at least $c_r(\boldsymbol{x}^\star, Q(\boldsymbol{x}^\star))$, where $Q(\boldsymbol{x}^\star)$ is the state of the algorithm after r rounds when the target is $\boldsymbol{x}^\star$. In particular, it is of the form $c_r(\boldsymbol{x}^\star, Q)$ for some Q. Since there are only $\frac{1}{2}|S|$ possible algorithm states, at most $\frac{1}{2}|S|$ values of $\boldsymbol{x}^\star$ can have total cost less than $0.05r$. Therefore, the average cost over instances uniformly drawn from S must be at least

$$\frac{1}{|S|}\left(|S| - \frac{1}{2}|S|\right) \cdot 0.05r \ge 0.025\left(\frac{\log S}{\log m} - 2\right) = \Omega(d/\log m).$$

Iteration lower bound. We will use the cost lower bound to prove the iteration lower bound. Recall that we proved that for any algorithm, there existed an instance $\boldsymbol{x}^\star$ for which the algorithm incurs $\Omega(d/\log m)$ cost over the first r rounds.

Now suppose we had an algorithm $\mathcal{A}$ which achieved an expected iteration complexity of finding an ε-optimal point of $C \cdot d/\log m$ for any $\boldsymbol{x}^\star \in S$, where $\varepsilon, C > 0$ are sufficiently small numerical constants. We can convert this into a low-cost algorithm $\mathcal{A}'$ for the first r rounds that (1) runs $\mathcal{A}$ to find an ε-optimal point $\boldsymbol{x}$; then (2) until round r repeatedly suggests $\boldsymbol{x}_t^{(1)} = \cdots = \boldsymbol{x}_t^{(m)} = \boldsymbol{x}$. The expected cost of algorithm $\mathcal{A}'$ for the first r rounds is at most

$$2 \cdot \frac{Cd}{\log m} + \varepsilon \cdot r \leq (2C + \varepsilon) \frac{d}{\log m}.$$

For sufficiently small ε and C, we have a contradiction with the previous cost lower bound; thus we can conclude that any algorithm must perform $\Omega\left(d/\log m\right)$ iterations in expectation to find an ε-optimal point $\boldsymbol{x}$.

This concludes the proof of Theorem 4. □

Proof of Corollary 5. The argument for linear f is a reprise of the lower bound for ℓ_2 distance. Observe that Fact 14 implies that the points in S also satisfy $\langle \boldsymbol{x}, \boldsymbol{y} \rangle \leq 0.995$ for any $\boldsymbol{x} \neq \boldsymbol{y}$.

Therefore, we again use Yao's Lemma and consider deterministic algorithms that branch into m states in every round. Letting $c_r(\boldsymbol{x}^\star, Q)$ denote the total cost incurred for state Q if the target is $\boldsymbol{x}^\star$, we again have the claim that all but at most one $\boldsymbol{x}^\star \in S$ have $c_r(\boldsymbol{x}^\star, Q) > C \cdot r$ for some constant $C > 0$, from which it follows that at most $\frac{1}{2}|S|$ values of $\boldsymbol{x}^\star$ can have total cost less than $C \cdot r$. We conclude that the average cost over instances drawn uniformly from S must be $\Omega\left(d/\log m\right)$.

The argument for the iteration lower bound also proceeds similarly, so we omit the details.

This concludes the proof of Corollary 5. □

Acknowledgements

AB is supported by NSF Awards CCF-2212968 and ECCS-2216899 and by the Defense Advanced Research Projects Agency under cooperative agreement HR00112020003. MG is supported by NSF Graduate Research Fellowship. GL is supported by the Institute for Data, Econometrics, Algorithms, and Learning (IDEAL). NSM is supported by NSF Graduate Research Fellowship and NSF Award ECCS-2216899. YY is supported by NSF Award CCF-2045402 and NSF Award CCF-2019844. YY thanks her advisor Jamie Morgenstern for her continued support and encouragement. We thank Aditya Bhaskara for suggesting that our algorithmic guarantees hold for smooth functions.

References

[AKJ14] Nir Ailon, Zohar Karnin, and Thorsten Joachims. Reducing dueling bandits to cardinal bandits. In *International Conference on Machine Learning*, pages 856–864. PMLR, 2014 (cited on page 2).

[BV06] Eyal Beigman and Rakesh Vohra. Learning from revealed preference. In *Proceedings of the 7th ACM Conference on Electronic Commerce*, pages 36–42, 2006 (cited on page 2).

[BFL21] Omar Besbes, Yuri Fonseca, and Ilan Lobel. Contextual inverse optimization: offline and online learning, 2021. DOI: 10.48550/ARXIV.2106.14015. URL: https://arxiv.org/abs/2106.14015 (cited on page 8).

[BS95] Avrim Blum and Joel Spencer. Coloring random and semi-random k-colorable graphs. *Journal of Algorithms*, 19(2):204–234, 1995 (cited on page 8).

[BOHG13] Jesús Bobadilla, Fernando Ortega, Antonio Hernando, and Abraham Gutiérrez. Recommender systems survey. *Knowledge-based systems*, 46:109–132, 2013 (cited on page 2).

[Bub15] Sébastien Bubeck. Convex optimization: algorithms and complexity, 2015. arXiv: 1405.4980 [math.OC] (cited on page 4).

[CG18] Yu Cheng and Rong Ge. Non-convex matrix completion against a semi-random adversary. In *Conference On Learning Theory*, pages 1362–1394. PMLR, 2018 (cited on page 8).

[CLRS11] Wei Chu, Lihong Li, Lev Reyzin, and Robert Schapire. Contextual bandits with linear payoff functions. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 208–214. JMLR Workshop and Conference Proceedings, 2011 (cited on page 4).

[DG03] Sanjoy Dasgupta and Anupam Gupta. An elementary proof of a theorem of johnson and lindenstrauss. *Random Structures & Algorithms*, 22(1):60–65, 2003. DOI: https://doi.org/10.1002/rsa.10073 (cited on page 10).

[DPVW14] Mark A Davenport, Yaniv Plan, Ewout Van Den Berg, and Mary Wootters. 1-bit matrix completion. *Information and Inference: A Journal of the IMA*, 3(3):189–223, 2014 (cited on page 8).

[DGT19] Ilias Diakonikolas, Themis Gouleakis, and Christos Tzamos. Distribution-independent pac learning of halfspaces with massart noise. *Advances in Neural Information Processing Systems*, 32, 2019 (cited on page 8).

[DHSSZ15] Miroslav Dudík, Katja Hofmann, Robert E Schapire, Aleksandrs Slivkins, and Masrour Zoghi. Contextual dueling bandits. In *Conference on Learning Theory*, pages 563–587. PMLR, 2015 (cited on page 2).

[Fei21] Uriel Feige. *Introduction to semirandom models*. In *Beyond the Worst-Case Analysis of Algorithms*. TimEditor Roughgarden, editor. Cambridge University Press, 2021, pages 189–211. DOI: 10.1017/9781108637435.013 (cited on pages 2, 8).

[GGKMLS21] Sreenivas Gollapudi, Guru Guruganesh, Kostas Kollias, Pasin Manurangsi, Renato Paes Leme, and Jon Schneider. Contextual recommendations and low-regret cutting-plane algorithms, 2021. DOI: 10.48550/ARXIV.2106.04819. URL: https://arxiv.org/abs/2106.04819 (cited on pages 4, 8).

[JNR12] Kevin G Jamieson, Robert Nowak, and Ben Recht. Query complexity of derivative-free optimization. *Advances in Neural Information Processing Systems*, 25, 2012 (cited on pages 2, 3, 7–9).

[JRTZ16] Dietmar Jannach, Paul Resnick, Alexander Tuzhilin, and Markus Zanker. Recommender systems—beyond matrix completion. *Communications of the ACM*, 59(11):94–102, 2016 (cited on page 2).

[KLLST22] Jonathan A Kelner, Jerry Li, Allen Liu, Aaron Sidford, and Kevin Tian. Semi-random sparse recovery in nearly-linear time. *arXiv preprint arXiv:2203.04002*, 2022 (cited on page 8).

[KHKN15] Junpei Komiyama, Junya Honda, Hisashi Kashima, and Hiroshi Nakagawa. Regret lower bound and optimal algorithm in dueling bandit problem. In *Conference on learning theory*, pages 1141–1154. PMLR, 2015 (cited on page 2).

[LS18] Renato Paes Leme and Jon Schneider. Contextual search via intrinsic volumes, 2018. DOI: 10.48550/ARXIV.1804.03195. URL: https://arxiv.org/abs/1804.03195 (cited on page 8).

[LLS20] Allen Liu, Renato Paes Leme, and Jon Schneider. Optimal contextual pricing and extensions, 2020. DOI: 10.48550/ARXIV.2003.01703. URL: https://arxiv.org/abs/2003.01703 (cited on page 8).

[LLV18] Ilan Lobel, Renato Paes Leme, and Adrian Vladu. Multidimensional binary search for contextual decision-making. *Operations Research*, 66(5):1346–1361, 2018 (cited on page 8).

[MN06] Pascal Massart and Élodie Nédélec. Risk bounds for statistical learning, 2006 (cited on page 8).

[Moi21] Ankur Moitra. *Semirandom stochastic block models*. In *Beyond the Worst-Case Analysis of Algorithms*. TimEditor Roughgarden, editor. Cambridge University Press, 2021, pages 212–233. DOI: 10.1017/9781108637435.014 (cited on pages 2, 8).

[OWJ+22] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744, 2022 (cited on page 2).

[SKM21] Aadirupa Saha, Tomer Koren, and Yishay Mansour. Dueling convex optimization. In *International Conference on Machine Learning*, pages 9245–9254. PMLR, 2021 (cited on pages 2, 3, 7–9).

[SKM22] Aadirupa Saha, Tomer Koren, and Yishay Mansour. Dueling convex optimization with general preferences. *arXiv preprint arXiv:2210.02562*, 2022 (cited on pages 2, 3, 7).

[SK22] Aadirupa Saha and Akshay Krishnamurthy. Efficient and optimal algorithms for contextual dueling bandits under realizability. In *International Conference on Algorithmic Learning Theory*, pages 968–994. PMLR, 2022 (cited on page 2).

[SJ11] Pannagadatta K Shivaswamy and Thorsten Joachims. Online learning with preference feedback. *arXiv preprint arXiv:1111.0712*, 2011 (cited on page 2).

[Ver18] R. Vershynin. *High-Dimensional Probability: An Introduction with Applications in Data Science*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 2018. ISBN: 9781108415194. URL: https://books.google.com/books?id=NDdqDwAAQBAJ (cited on page 19).

[Yao77] Andrew Chi-Chin Yao. Probabilistic computations: toward a unified measure of complexity. In *18th Annual Symposium on Foundations of Computer Science (sfcs 1977)*, pages 222–227. IEEE Computer Society, 1977 (cited on page 19).

[YBKJ12] Yisong Yue, Josef Broder, Robert Kleinberg, and Thorsten Joachims. The k-armed dueling bandits problem. *Journal of Computer and System Sciences*, 78(5):1538–1556, 2012 (cited on page 2).