# Contextual Search in the Presence of Irrational Agents

Akshay Krishnamurthy
Microsoft Research NYC
NYC, USA
akshaykr@microsoft.com

Thodoris Lykouris
Microsoft Research NYC
NYC, USA
thlykour@microsoft.com

Chara Podimata
Harvard University
Cambridge, USA
podimata@g.harvard.edu

Robert Schapire
Microsoft Research NYC
NYC, USA
schapire@microsoft.com

## ABSTRACT

We study contextual search, a generalization of binary search in higher dimensions, which captures settings such as feature-based dynamic pricing. Standard game-theoretic formulations of this problem assume that agents act in accordance with a specific behavioral model. In practice, some agents may not subscribe to the dominant behavioral model or may act in ways that seem to be *arbitrarily irrational*. Existing algorithms heavily depend on the behavioral model being (approximately) accurate for all agents and have poor performance even with a few arbitrarily irrational agents.

We initiate the study of contextual search when some of the agents can behave in ways inconsistent with the underlying behavioral model. In particular, we provide two algorithms, one based on multidimensional binary search methods and one based on gradient descent. Our techniques draw inspiration from learning theory, game theory, high-dimensional geometry, and convex analysis.

## CCS CONCEPTS

• **Theory of computation** → **Online learning algorithms**.

## KEYWORDS

dynamic pricing, binary search, adversarial corruptions

**ACM Reference Format:**
Akshay Krishnamurthy, Thodoris Lykouris, Chara Podimata, and Robert Schapire. 2021. Contextual Search in the Presence of Irrational Agents. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing (STOC '21), June 21–25, 2021, Virtual, Italy*. ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3406325.3451120

## 1 INTRODUCTION

We study *contextual search*, a fundamental algorithmic problem that extends classical binary search to higher dimensions and has direct applications to pricing and personalized medicine [4, 11, 27]. In the most standard, linear version of the problem, at every round $t$ within a horizon of $T$ rounds, some *context* $\mathbf{x}_t \in \mathbb{R}^d$ arrives. Associated with this context is an unknown *true value* $v_t \in \mathbb{R}$, which we assume to be a linear function of the context so that $v_t = \langle \boldsymbol{\theta}^\star, \mathbf{x}_t \rangle$ for some unknown vector $\boldsymbol{\theta}^\star \in \mathbb{R}^d$, called the *ground truth*. Based on the observed context $\mathbf{x}_t$, the decision-maker or *learner* selects a *query* $\omega_t \in \mathbb{R}$ with the goal of minimizing some *loss* that depends on the query as well as the true value; examples include the *absolute loss*, $|v_t - \omega_t|$, and the *ε-ball loss*, $\mathbb{1}\{|v_t - \omega_t| > \varepsilon\}$, both of which measure discrepancy between $\omega_t$ and $v_t$. Finally, the learner observes whether or not $v_t \geq \omega_t$, but importantly, the true value $v_t$ is never revealed, nor is the loss that was suffered.

For example, in feature-based dynamic pricing [11, 26, 27, 33], say, of Airbnb apartments, each context $\mathbf{x}_t$ describes a particular apartment with components, or features, providing the apartment's location, cleanliness, and so on. The true value $v_t$ is the price an incoming customer or *agent* is willing to pay, which is assumed to be a linear function (defined by $\boldsymbol{\theta}^\star$) of $\mathbf{x}_t$. Based on $\mathbf{x}_t$, the platform decides on a price $\omega_t$. If this price is less than the customer's value $v_t$, then the customer makes a reservation, yielding revenue $\omega_t$; otherwise, the customer passes, generating no revenue. The platform observes whether the reservation occurred (that is, if $v_t \geq \omega_t$). The natural loss in this setting is called the *pricing loss*, which captures how much revenue was lost relative to the maximum price that the customer was willing to pay.

A key challenge in contextual search is that the learner only observes *binary feedback*, i.e., whether or not $v_t \geq \omega_t$. This contrasts with classical machine learning where the learner observes either the entire loss function (full feedback) or only the loss itself for just the chosen query (bandit feedback).

The above model makes the strong assumption that the feedback is always consistent with the ground truth, typically called *full rationality* in behavioral economics. This is not always realistic. The assumption that the agent's value is described by a linear model may be violated to some degree. Moreover, some agents may behave in ways not prescribed by the dominant behavioral model, or may deviate from it for idiosyncratic reasons. Although some prior works allow for some benign, stochastic noise (see related work below), these techniques are generally not robust to any interference that is adversarial, and therefore cannot address settings where some agents may act in ways that are *irrational* with respect to the underlying ground truth.

In this paper, we present the first contextual search algorithms that can handle adversarial noise models. In particular, we allow some agents to behave in ways that are arbitrarily inconsistent with the ground truth and thereby seem *irrational*. Inspired by the recent

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
*STOC '21, June 21–25, 2021, Virtual, Italy*
© 2021 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-8053-9/21/06.
https://doi.org/10.1145/3406325.3451120

line of work on stochastic learning with adversarial corruptions [2, 7, 10, 19, 25, 28, 29, 41], we impose no assumption on the order in which irrational agents arrive and aim for guarantees that gracefully degrade according to the prevalence of such irrationality while attaining near-optimal guarantees when all agents are fully rational.

## 1.1 Our Contributions and Techniques

We first provide a unifying framework encompassing disparate behavioral models determining agent responses and various loss functions – specifically, the absolute, the $\varepsilon$-ball, and the pricing loss. In particular, we assume that the agent behaves according to a *perceived value* $\widetilde{v}$ and the precise behavioral model determines the transformation from true to perceived value. The loss functions can depend on either the true value (to capture parameter estimation objectives) or the perceived value (for pricing objectives). This formulation allows us to capture arbitrarily irrational agents, a setting not studied in prior work. Our model also captures stochastic noise settings studied in prior work such as *bounded rationality*.

As a starting point, we create a variant of gradient descent based on a proxy loss and show that it achieves a regret guarantee of $O(\sqrt{T} + C)$ for the absolute loss, and $O((\sqrt{T} + C)/\varepsilon)$ for the $\varepsilon$-ball loss, where $C$ is the unknown number of rounds on which the corresponding agent acts inconsistently with the ground truth. This algorithm is simple, based on known techniques, and runs in linear time; that said, it does not provide the state-of-the-art logarithmic regret guarantees for $C \approx 0$ and most importantly it does not extend to non-Lipschitz loss functions, like the pricing loss. We discuss this algorithm formally in the full version.

Our main result is an algorithm that works for all of the aforementioned loss functions ($\varepsilon$-ball, absolute, pricing) and applies to a variety of behavioral models. We prove that, with probability $1 - \delta$, this algorithm suffers a relative degradation in performance or *regret* of $O((1+C) \cdot d^3 \cdot \text{poly}\log(T/\delta))$. Our guarantee is logarithmic in the time-horizon, as is typical in binary search methods when $C \approx 0$, and degrades gracefully as $C$ becomes larger. Our algorithm builds on the ProjectedVolume algorithm of Lobel, Paes Leme, and Vladu [27] which is optimal for the $\varepsilon$-ball loss when $C = 0$.

Our main technical advance is a method for maintaining a set of candidates for $\boldsymbol{\theta}^\star$, successively removing candidates by hyperplane cuts while ensuring that $\boldsymbol{\theta}^\star$ is never removed. When $C = 0$, this is done via ProjectedVolume [27] which removes all parameters $\boldsymbol{\theta}$ that are inconsistent with the query response in a way that each *costly* query guarantees enough progress, measured via the volume of the set of remaining parameters. However, when some responses are corrupted, such an aggressive elimination method runs the risk of removing the ground truth $\boldsymbol{\theta}^\star$ from the parameter space.

To deal with this key challenge, we run the algorithm in epochs, each corresponding to one query of ProjectedVolume, and only proceed to the next epoch if we can find a halfspace that both makes volumetric progress and does not eliminate $\boldsymbol{\theta}^\star$. We start from an easier setting where we assume access to a known upper bound $\bar{c}$ on the number of corrupted responses ($C \le \bar{c}$) and only move to the next epoch and its respective knowledge set when we can find a halfspace making enough volumetric progress that includes all parameters that are misclassified by at most $\bar{c}$ queries. Note that $\boldsymbol{\theta}^\star$ is consistent with all non-corrupted responses, and hence, it is always included in the new knowledge set (as it can only suffer at most $\bar{c}$ misclassifications).

To avoid being fooled by corrupted responses, we face several challenges. First, even after arbitrarily many queries, we cannot guarantee that one of them induces such a halfspace. Interestingly, when we do not restrict ourselves to the halfspaces associated with one particular query but allow for *improper* cuts, we can use ideas from convex analysis (specifically, the Carathéodory theorem) to show that one such query exists after collecting $O(d^2\bar{c})$ queries. In particular, we identify a point in the parameter space that is outside of a convex body including all the *protected parameters* (the ones with misclassification at most $\bar{c}$) and the separating hyperplane theorem implies the existence of the desired cut.

A second challenge is that the above argument is only existential and does not suggest a direct way to compute the halfspace. To deal with this, we use geometric techniques (in particular, volume cap arguments) to provide a sampling process that, with significant probability, identifies a point $\mathbf{q}$ that is sufficiently far from the aforementioned separating hyperplane. We can then compute this hyperplane by running the classic learning-theoretic Perceptron algorithm repeatedly using points sampled from this process.

There are two remaining, intertwined challenges. On the one hand, the running time of Perceptron depends on the number of subregions created by removing all possible combinations of $\bar{c}$ queries which is exponential in $\bar{c}$. On the other hand, our algorithm needs to be agnostic to $C$. We deal with both of these via a multi-layering approach introduced in [28] that runs multiple parallel versions of the aforementioned algorithm with only $\bar{c} \approx \log T$. This results in a final algorithm that is quasipolynomial in the time horizon and does not assume knowledge of $C$.

## 1.2 Related Work

Our work is closely related to the problem of *dynamic pricing* when facing an agent with *unknown* demand curve (see [13] for a survey). The single-dimensional case has been studied both from a parametric model standpoint ([8, 12, 14, 22, 24]), and a non-parametric one with the goal of regret minimization. Our work falls in the second category, where Kleinberg and Leighton [23] first formulated the problem. Besbes and Zeevi considered a similar dynamic pricing setting with inventory constraints [5]. Mohri and Munoz-Medina [31] and Feldman, Koren, Livni, Mansour, and Zohar [16] consider extensions where the agents are long-living and strategic over time. Recently, Cesa-Bianchi, Cesari, and Perchet [9] studied a variant, where there is more than one demand curves.

Moving from the single- to the higher-dimensional case, there have been mainly two families of algorithms. The more classical approach involves statistical methods based, for example, on linear regression and the central limit theorem [1, 3, 4, 17, 18, 20, 32, 35, 38]. This line of work does not deal with adversarial contexts and tends to obtain performance loss of the order of $\sqrt{T}$, but allows for some stochastic noise in the behavior of the agents (*bounded rationality*). Closer to our work are approaches based on multidimensional binary search. This line of work was introduced by Cohen, Lobel, and Paes Leme [11] who studied contextual pricing with adversarial contexts. Their results were improved for the $\varepsilon$-ball loss [27] and the pricing loss [26, 33]. Mao, Paes Leme, and Schneider [30] studied a variant of the standard contextual pricing problem, where the buyers' utilities are Lipschitz, rather than linear in the contexts. With respect to stochastic noise models, [11] extend their results to a stochastic noise model which we also discuss in Section 2.2. [1] A concurrent work by Liu, Paes Leme, and Schneider [26] also extends results for the absolute loss to a stochastic noise model where the feedback is flipped. However, all of the aforementioned works, both the statistical and the binary search approaches, do not extend to adversarial noise models such as the one we consider.

The single-dimensional version of our problem bears similarities to *Ulam's game* [40], where one wants to make the least number of queries to an opponent in order to identify a target number among $n$ ordered numbers. The opponent can only inform the player that the queried point is larger/smaller than the target, and may lie at most $C$ times over the course of the game, where $C$ is known to the learner ([39], see also [34] for a survey). Rivest, Meyer, Kleitman, Winklmann, and Spencer [36] provide the optimal solution for this problem which is $\Theta(\log n + C \log \log n + C \log C)$. This question is also related with works in noisy binary search [21]. Although Ulam's game can be interpreted as the single-dimensional version of the contextual search problem that we are solving, the techniques presented in [36] require exponential computation time for higher dimensions. To the best of our knowledge, adversarial noise models have not been studied for binary search in higher dimensions. Our work can be viewed as a contextual multidimensional version of Ulam's game that also does not assume knowledge of $C$.

Our work also incorporates ideas from the recent literature on bandit learning with adversarial corruptions; indeed, we use "corruptions" to model our adversarially irrational agents. Learning in the presence of adversarial corruptions was introduced by Lykouris, Mirrokni, and Paes Leme [28] for stochastic multi-armed bandits and their results have been subsequently strengthened by Gupta, Koren, and Talwar [19] and Zimmert and Seldin [41]. This line of work has been extended to several other settings [2, 7, 10, 25, 29]. Our work differs from these in that we use adversarial corruptions as a *modeling tool* to capture arbitrarily irrational agent behavior in game-theoretic settings.[2] On a technical level, our setting involves a continuous action space which requires new analytical tools, while all prior results involve discrete (potentially large) action spaces.

## 2 MODEL

In this section, we provide a general framework (Section 2.1) that allows us to study contextual search under different behavioral models (Section 2.2) and different loss functions (Section 2.3).

### 2.1 Protocol

We consider the following repeated interaction between the learner and nature. Following classical works in contextual search [11, 27, 33] we assume that the learner has access to a parameter space $\mathcal{K} = \{\mathbf{u} \in \mathbb{R}^d : \|\mathbf{u}\|_2 \leq 1\}$ and a context space $\mathcal{X} = \{\mathbf{x} \in \mathbb{R}^d : \|\mathbf{x}\|_2 = 1\}$. We denote by $\Omega = \mathbb{R}$ the decision space of the learner and by $\mathcal{V} = \mathbb{R}$ a value space; in the pricing setting, $\Omega$ can be thought of as the set of possible prices available to the learner and $\mathcal{V}$ as a set of values associated with incoming agents. Domain $\mathcal{V}$ helps express both the true value of the agents and the perceived value driving their decisions. Finally, we consider a behavioral model determining the transformation from the agent's *true value* to a *perceived value* that drives the decision at each round. All of the above are known to the learner throughout the learning process.

The setting proceeds for $T$ rounds. Before the first round, nature chooses a ground truth $\boldsymbol{\theta}^\star \in \mathcal{K}$; this is fixed across rounds and is *not* known to the learner. This ground truth determines both the agent's *true* value function $v : \mathcal{X} \to \mathcal{V}$ and the learner's loss function $\ell : \Omega \times \mathcal{V} \times \mathcal{V} \to [0, 1]$. We note that both value and loss functions are also functions of the ground truth $\boldsymbol{\theta}^\star$; given that $\boldsymbol{\theta}^\star$ is fixed throughout this process, we drop the dependence on $\boldsymbol{\theta}^\star$ to ease notation. The functional form of both $v(\cdot)$ and $\ell(\cdot)$ as a function of the ground truth $\boldsymbol{\theta}^\star$ is known to the learner but the learner does not know $\boldsymbol{\theta}^\star$. For each round $t = 1, \ldots, T$:

(1) Nature chooses (potentially adaptively and adversarially) and reveals context $\mathbf{x}_t \in \mathcal{X}$.
(2) Nature chooses but *does not* reveal a perceived value $\widetilde{v}_t \in \mathcal{V}$ based on the behavioral model.
(3) Learner selects query point $\omega_t \in \Omega$ (in a randomized manner) and observes $y_t = \text{sgn}(\widetilde{v}_t - \omega_t)$. [3]
(4) Learner incurs (but does *not* observe) loss: $\ell(\omega_t, v(\mathbf{x}_t), \widetilde{v}_t) \in [0, 1]$.

Nature is an adaptive adversary (subject to the behavioral model), i.e., it knows the learner's algorithm along with the realization of all randomness up to and including round $t - 1$ (i.e, it knows all $\omega_\tau, \forall \tau \leq t - 1$), but does not know the learner's randomness at the current round $t$. Moreover, the learner only observes the context $\mathbf{x}_t$ and the *binary* variable $y_t$ (Steps 1 and 3 of the protocol), and has access to neither the perceived value $\widetilde{v}_t$ nor the loss $\ell(\omega_t, v(\mathbf{x}_t), \widetilde{v}_t)$. Finally, in the pricing setting, $y_t$ corresponds to whether the agent associated with round $t$ made a purchase or not.

The above protocol unifies contextual search under general behavioral models and loss functions. In Section 2.2, we discuss the different behavioral models we consider that determine the agents' perceived values (Step 2 in the protocol). In Section 2.3, we discuss the main loss functions for this setting (Step 4 in the protocol) as well as the corresponding performance metrics.

### 2.2 Behavioral Models

We assume that the agents' true value function is $v(\mathbf{x}) = \langle \mathbf{x}, \boldsymbol{\theta}^\star \rangle$ for any $\mathbf{x} \in \mathcal{X}$ (i.e., independent of the agent's behavioral model). The behavioral model affects the perceived value $\widetilde{v}$ at round $t$, which then affects both the loss incurred and the feedback observed by the learner. The behavioral model that is mostly studied in contextual search works is *full rationality*. This assumes that agents always behave according to their true value, i.e., $\widetilde{v}_t = v(\mathbf{x}_t) = \langle \mathbf{x}_t, \boldsymbol{\theta}^\star \rangle$. In learning-theoretic terms, this consistency with respect to a ground truth is typically referred to as *realizability*.

Our main focus in this work is the study of a behavioral model that we call *adversarial irrationality*. There, nature selects the rounds where the irrational agents arrive ($c_t = 1$ if irrational agents arrive, else $c_t = 0$), together with an upper bound $C$ on this number of rounds (i.e., $\sum_{t \in [T]} c_t \leq C$). Neither the sequence $\{c_t\}_{t \in [T]}$ nor

[1] Models of stochastic noise have also been considered for the problem of binary search in graphs in [15].

[2] [10] also consider a behavioral setting, but do not make a connection to irrationality.

[3] $\text{sgn}(\cdot)$ is the sign function, i.e., $\text{sgn}(x) = 1$ if $x \geq 0$ and $-1$ otherwise

the number $C$ are ever revealed to the learner. If $c_t = 0$, then nature is constrained to $\widetilde{v}_t = v(\mathbf{x}_t)$, but can select adaptively and adversarially $\widetilde{v}_t$ if $c_t = 1$. This model is inspired by the model of adversarial corruptions in stochastic bandit learning [28].

Our results extend to *bounded rationality* which posits that the perceived value is the true value plus some noise parameter. The noise parameter is drawn from a $\sigma$-subgaussian distribution $\texttt{subG}(\sigma)$, *fixed* across rounds and *known* to the learner, i.e., nature selects it before the first round and reveals it. At every round $t$ a realized $\xi_t \sim \texttt{subG}(\sigma)$ is drawn, but $\xi_t$ is never revealed to the learner. The agent's perceived value is then $\widetilde{v}_t = v(\mathbf{x}_t) + \xi_t$. This stochastic noise model has been studied in the contextual search literature as a way to incorporate idiosyncratic market shocks [11]. Our results for this behavioral model can be found in the full version.

## 2.3 Loss Functions and Objective

We study three variants for the learner's loss function: the $\varepsilon$-ball, the absolute, and the pricing loss. Abstracting away from $t$ subscripts and dependencies on $\mathbf{x}$, the loss $\ell(\omega, v, \widetilde{v})$ evaluates the loss of a query $\omega$ when the true value is $v$ and the perceived value is $\widetilde{v}$.

The first class of loss functions includes parameter estimation objectives that estimate the value of $\boldsymbol{\theta}^\star$. One such function is the *$\varepsilon$-ball loss* which is defined with respect to an accuracy parameter $\varepsilon > 0$. The $\varepsilon$-ball is 1 if the difference between the query point $\omega$ and the true value $v$ is larger than $\varepsilon$ and 0 otherwise. Formally, $\ell(\omega, v, \widetilde{v}) = \mathbb{1}\{|v - \omega| \geq \varepsilon\}$. Another parameter estimation loss function is the *absolute* or *symmetric* loss that captures the absolute difference between the query point and the true value, i.e., $\ell(\omega, v, \widetilde{v}) = |v - \omega|$. The aforementioned loss functions are unobservable to the learner as the true value $v$ is latent; this demonstrates that binary feedback does not offer strictly more information than the bandit feedback as the latter reveals the loss of the selected query.

Another important objective in pricing is the revenue collected which is the price $\omega$ in the event that the purchase occurred, i.e., $\widetilde{v} \geq \omega$. This can be expressed based on observable information by setting a reward equal to $\omega$ when $\widetilde{v} \geq \omega$ and 0 otherwise, i.e., $\omega \cdot \mathbb{1}\{\widetilde{v} \geq \omega\}$. However, this expression does not convey that querying $\omega = \widetilde{v}$ is optimal and does not enable logarithmic performance guarantees that are typical in binary search. A loss function exploiting this structure is the *pricing loss* which is defined as the difference between the highest revenue that the learner could have achieved at this round (the agent's *perceived* value $\widetilde{v}$) and the revenue that the learner currently receives, i.e., $\omega$ if a purchase happens, and 0 otherwise. The outcome of whether a purchase happens or not is tied to whether $\omega$ is higher or smaller than the perceived value $\widetilde{v}$. Putting everything together: $\ell(\omega, v, \widetilde{v}) = \widetilde{v} - \omega \cdot \mathbb{1}\{\widetilde{v} \geq \omega\}$.

We remark that the $\varepsilon$-ball and the absolute loss depend only on the true value $v$ (and not the perceived value $\widetilde{v}$); indeed, when these losses are considered $\widetilde{v}$ affects only the feedback that the learner receives. That said, we define $\ell(\cdot, \cdot, \cdot)$ with three arguments for unification purposes, since the pricing loss does depend on the feedback that the learner receives (and hence, on $\widetilde{v}$).

The learner's goal is to minimize a notion of regret. For adversarially irrational agents, the loss of the best-fixed action in hindsight is *at least* 0 and *at most* $C$. Hence, to simplify exposition for the case of adversarially irrational agents we slightly abuse notation and conflate the loss and the regret:

$$R(T) = \sum_{t \in [T]} \ell(\omega_t, v(\mathbf{x}_t), \widetilde{v}_t). \tag{1}$$

It is no longer possible to provide sublinear guarantees for the quantity defined in Equation (1) when facing boundedly rational agents. For these agents we slightly relax the benchmark that we are comparing against. We defer the more careful definition of the regret in these cases to the full version.

## 3 MAIN IDEAS OF ALGORITHM & ANALYSIS

In this section, we provide an algorithmic scheme that handles all the aforementioned behavioral models and loss functions. Our main result is an algorithm, which we term CorPV.AI for the adversarial irrationality behavioral model when there is an *unknown* upper bound $C$ on the number of irrational agents. The regret of this algorithm is upper bounded by the following theorem.

**Theorem 3.1.** For an unknown corruption level $C$, and any $\beta > 0$, CorPV.AI incurs regret $O(d^3 \cdot \log(T/\beta) \cdot \log(d/\varepsilon) \cdot \log(1/\beta) \cdot (\log T + C))$ for the $\varepsilon$-ball loss, and, setting $\varepsilon = 1/T$, $O(d^3 \log(dT) \log(T/\beta) \cdot (\log T + C) \log(1/\beta))$ for the absolute and pricing loss with probability at least $1 - \beta$. The expected runtime of the algorithm is quasi-polynomial; in particular, it is $O((d^2 \log T)^{\text{poly} \log T} \cdot \text{poly}(d, \log T))$.

We now introduce the main algorithmic and analytical ideas behind the above theorem and defer its complete proof to the full version.

A useful intermediate setting is the case where we know an upper bound $\bar{c}$ on the number of adversarial agents, i.e., $C \leq \bar{c}$; we refer to this case as the *$\bar{c}$-known-corruption* setting. This allows us to introduce our key ideas and serves as a building block towards CorPV.AI (Section 3.2). In Section 3.1, we highlight the novel insights in our work, by focusing on a simplified version of our CorPV.Known algorithm for the *$\bar{c}$-known corruption* setting. We refer to this simplified algorithm as CorPV.Known.Simple and discuss additional complexities in moving from it to CorPV.Known in Section 3.3.

## 3.1 The Known-Corruption Algorithm

CorPV.Known.Simple (Algorithm 1) builds on the algorithm of Lobel, Paes Leme, and Vladu [27] called ProjectedVolume, which is optimal in terms of regret for the $\varepsilon$-ball loss when $\bar{c} = 0$. The main idea in ProjectedVolume is to maintain a *knowledge set* $\mathcal{K}_t$ which includes all candidate parameters $\boldsymbol{\theta}$ that are *consistent* with what has been observed so far. The true parameter $\boldsymbol{\theta}^\star$ is always consistent (and therefore is never eliminated from the knowledge set), and the volume of the knowledge set is intuitively a measure of progress for the algorithm.

Given a context $\mathbf{x}_t$, there are two scenarios. Before we describe them, we define the *width* of a body $\mathcal{K}$ on direction $\mathbf{x}$ as $w(\mathcal{K}, \mathbf{x}) = \sup_{\boldsymbol{\theta}, \boldsymbol{\theta}' \in \mathcal{K}} \langle \boldsymbol{\theta} - \boldsymbol{\theta}', \mathbf{x} \rangle$. If the width of the knowledge set in the direction of $\mathbf{x}_t$ is $w(\mathcal{K}_t, \mathbf{x}_t) \leq \varepsilon$, then the algorithm can make an *exploit* query $\omega_t = \langle \mathbf{x}_t, \boldsymbol{\theta}_t \rangle$ for *any* point $\boldsymbol{\theta}_t \in \mathcal{K}_t$ thereby guaranteeing an $\varepsilon$-ball loss equal to 0. Otherwise, if $w(\mathcal{K}_t, \mathbf{x}_t) > \varepsilon$, the algorithm queries the point $\omega_t = \langle \mathbf{x}_t, \boldsymbol{\kappa}_t^\star \rangle$, where $\boldsymbol{\kappa}_t^\star$ is the centroid of $\mathcal{K}_t$ defined as $\boldsymbol{\kappa}_t^\star = \frac{1}{\texttt{vol}(\mathcal{K}_t)} \int_{\mathcal{K}_t} \mathbf{u} d\mathbf{u}$, where $\texttt{vol}(\cdot)$ denotes the volume of a set. This is called an *explore* query. By querying this point, the algorithm learns that $\boldsymbol{\theta}^\star$ lies in one of the two halfspaces passing through $\boldsymbol{\kappa}_t^\star$ with normal vector $\mathbf{x}_t$, so it can update the knowledge set by taking intersection with this halfspace. We use $(\mathbf{h}, \omega)$, $\mathbf{H}^+(\mathbf{h}, \omega)$, and $\mathbf{H}^-(\mathbf{h}, \omega)$ to denote the hyperplane with normal vector $\mathbf{h} \in \mathbb{R}^d$ and intercept $\omega$, and the positive and negative halfspaces it creates with intercept $\omega$, i.e., $\{\mathbf{x} \in \mathbb{R}^d : \langle \mathbf{h}, \mathbf{x} \rangle \geq \omega\}$ and $\{\mathbf{x} \in \mathbb{R}^d : \langle \mathbf{h}, \mathbf{x} \rangle \leq \omega\}$, respectively. By properties of $\boldsymbol{\kappa}_t^\star$, the volume of the updated knowledge set is a constant factor of the initial volume, leading to geometric volume progress.

Having described ProjectedVolume, which works when there are no corruptions, we turn to our algorithm. The problem is that, even when there are few corruptions, ProjectedVolume may quickly eliminate $\boldsymbol{\theta}^\star$ from $\mathcal{K}_t$. To deal with this, we run the algorithm in epochs consisting of multiple queries. The goal of the epochs is to ensure that $\boldsymbol{\theta}^\star$ is never eliminated from the knowledge set and that, at the end of the epoch, we make enough volumetric progress on the latter's size. We face three important design decisions, discussed separately below: what occurs inside an epoch, when to stop an epoch, and how to initialize the next one.

**What occurs within an epoch?** CorPV.Known.Simple (Algorithm 1) outlines what happens within an epoch $\phi$. The knowledge set is updated only at its end; this means that all rounds $t$ in epoch $\phi$ have the same knowledge set $\mathcal{K}_\phi$ (and hence, the same centroid $\boldsymbol{\kappa}_\phi^\star$). If the width of the knowledge set in the direction of $\mathbf{x}_t$ is smaller than $\varepsilon$, then, as in ProjectedVolume, we make an *exploit* query precisely described in Section 3.3. Otherwise, we make an *explore* query $\omega_t = \langle \mathbf{x}_t, \boldsymbol{\kappa}_\phi^\star \rangle$. The epoch keeps track of all explore queries that occur within its duration in a set $\mathcal{A}_\phi$. When it ends ($\phi' = \phi+1$), the knowledge set $\mathcal{K}_{\phi+1}$ of the new epoch is initialized. Sets $L_\phi$ and $S_\phi$ contain what we call the *large* and the *small* dimensions of epoch $\phi$ respectively. Loosely speaking, these are the dimensions where the width of $\mathcal{K}_\phi$ is larger (and smaller, respectively) than some scalar $\delta > 0$, specified in Section 3.3. [4]

**ALGORITHM 1:** CorPV.Known.Simple

```
Global parameters: Budget $\bar{c}$, accuracy $\varepsilon$
Initialize $\phi = 1$, $\mathcal{K}_\phi \leftarrow \mathcal{K}$, $S_\phi \leftarrow \emptyset$, $\boldsymbol{\kappa}_\phi^\star \leftarrow$ centroid($\mathcal{K}_\phi$),
  $L_\phi \leftarrow$ orthonorm-basis($\mathbb{R}^d$), $\mathcal{A}_\phi \leftarrow \emptyset$
for $t \in [T]$ do
    Observe context $\mathbf{x}_t$.
    if $w(\mathcal{K}_\phi, \mathbf{x}_t) \leq \varepsilon$ or $L_\phi = \emptyset$ then
        Select query point $\omega_t$ = CorPV.Exploit($\mathbf{x}_t, \mathcal{K}_\phi$)
    else
        $(\phi', \mathcal{A}_\phi) \leftarrow$ CorPV.Explore($\mathbf{x}_t, \phi, \boldsymbol{\kappa}_\phi^\star, L_\phi, \mathcal{A}_\phi$)
        if $\phi' = \phi + 1$ then                              ▷ epoch changed
            Compute separating cut:
              $(\widetilde{\mathbf{h}}, \widetilde{\omega}) \leftarrow$ CorPV.SeparatingCut($\boldsymbol{\kappa}_\phi^\star, S_\phi, L_\phi, \mathcal{A}_\phi$)
            Make updates $(\mathcal{K}_{\phi+1}, S_{\phi+1}, L_{\phi+1}) \leftarrow$
              CorPV.EpochUpdates($\mathcal{K}_\phi, S_\phi, L_\phi, \widetilde{\mathbf{h}}, \widetilde{\omega}$)
            Initialize next epoch: $\phi \leftarrow \phi'$, $\boldsymbol{\kappa}_\phi^\star \leftarrow$ centroid($\mathcal{K}_\phi$), and
              and $\mathcal{A}_\phi \leftarrow \emptyset$.
```

[4]The reason why $L_\phi$ and $S_\phi$ are needed is that, in reality we do not work directly with $\mathcal{K}_\phi$ but rather with a *cylindrified* version that allows us to deal with contexts along directions where the knowledge set is *thin*. Since this introduces complications not essential in understanding our techniques, we defer this distinction to Section 3.3.

CorPV.Explore (Algorithm 2) describes how we handle an explore query. When $\bar{c} = 0$, we can eliminate the halfspace that lies in the opposite direction of the feedback $y_t$ after each explore query. However, when $\bar{c} > 0$, this may eliminate $\boldsymbol{\theta}^\star$. Instead, we keep all explore queries that occurred in epoch $\phi$ as well as the halfspace consistent with the observed feedback in $\mathcal{A}_\phi$ and wait until we have enough data to identify a halfspace of the knowledge set that includes $\boldsymbol{\theta}^\star$ and makes sufficient volumetric progress; we refer to this as a *separating cut*. We then move to epoch $\phi' = \phi + 1$. In terms of notation, we clarify that $\Pi_{L_\phi} \mathbf{x}_t$ corresponds to the *projection* of feature vector $\mathbf{x}_t$ onto the set of large dimensions $L_\phi$.

**ALGORITHM 2:** CorPV.Explore

```
Parameters: $\mathbf{x}_t, \phi, \boldsymbol{\kappa}_\phi^\star, L_\phi, \mathcal{A}_\phi$
Select query point $\omega_t = \langle \mathbf{x}_t, \boldsymbol{\kappa}_\phi^\star \rangle$ and observe feedback $y_t$.
Update explore queries: if $y_t = +1$: $\mathcal{A}_\phi \leftarrow \mathcal{A}_\phi \bigcup \mathbf{H}^+\left(\Pi_{L_\phi}\mathbf{x}_t, \omega_t\right)$
  else $\mathcal{A}_\phi \leftarrow \mathcal{A}_\phi \bigcup \mathbf{H}^-\left(\Pi_{L_\phi}\mathbf{x}_t, \omega_t\right)$.
if $|\mathcal{A}_\phi| \geq \tau$ then                  ▷ $\tau := 2d \cdot \bar{c} \cdot (d+1) + 1$
    Move to next epoch $\phi' \leftarrow \phi + 1$.
else Stay in the same epoch $\phi' \leftarrow \phi$.
return $(\phi', \mathcal{A}_\phi)$
```

**When does the epoch end?** It turns out that $\tau = 2d \cdot \bar{c} \cdot (d+1) + 1$ explore queries suffice to identify such a separating cut. In particular, it suffices to ensure that all candidate parameters $\boldsymbol{\theta}$ on its negative halfspace are misclassified by at least $\bar{c} + 1$ explore queries; since there are at most $\bar{c}$ corruptions, this guarantees that $\boldsymbol{\theta}^\star$ is not incorrectly eliminated. In other words, if the set of all candidate parameters $\boldsymbol{\theta}$ that are misclassified by at most $\bar{c}$ explore queries are on the non-eliminated halfspace of the hyperplane, then this hyperplane can serve as separating cut. We refer to the set of these parameters as the *$\bar{c}$-protected region* $\mathcal{P}(\bar{c})$ as we ensure that they are not eliminated. We note that this region may be non-convex as shown in Figure 1 where $\mathcal{P}(\bar{c})$ is the blue area.

One key technical component in our analysis is to show that, after $\tau = 2d \cdot \bar{c} \cdot (d+1) + 1$ explore queries, there exists a point $\mathbf{p}^\star$ close to $\boldsymbol{\kappa}_\phi^\star$ that is outside the convex hull of the protected region. A cut that separates the point $\mathbf{p}^\star$ from the convex hull of the protected region then exists because of the separating hyperplane theorem and implies enough volumetric progress.

To prove this technical component, we introduce the notion of *undesirability level* which counts the number of misclassifications for points in $\mathcal{K}$ and we show that there exists one point $\mathbf{p}^\star$ with undesirability level larger than $\bar{c} + 1$. For the latter, we use the Carathéodory's theorem, i.e., that each point in the convex hull of $\mathcal{P}(\bar{c})$ ($\mathbf{p} \in \mathsf{conv}(\mathcal{P}(\bar{c}))$), denoted by the horizontal-line patterned area in Figure 1, can be written as a convex combination of $d+1$ points from $\mathcal{P}(\bar{c})$. Since the undesirability level of any point $\mathbf{p} \in \mathsf{conv}(\mathcal{P}(\bar{c}))$ is at most $\bar{c}$ and any undesirability of a point in the convex hull can be tracked to undesirability in one of these $d+1$ points in $\mathcal{P}(\bar{c})$, once a point has undesirability of $(d+1) \cdot \bar{c} + 1$, it is no longer in the $\mathsf{conv}(\mathcal{P}(\bar{c}))$. Hence, the number of rounds needed to certify the existence of the separating hyperplane is contingent on the existence of a point with undesirability more than $(d+1)\bar{c}$. To do so, we consider $2d$ auxiliary points (called "landmarks") and we guarantee that at each round *at least one of them* gets one misclassification. By the pigeonhole principle, this means that after $2d\bar{c}(d+1)$ rounds, at least one of the landmarks has undesirability at least $\bar{c}(d+1)+1$. Moving forward we call this point $\mathbf{p}^\star$ (see Figure 2). This outline explains the reason why we pick up the additional $d^2$ factor in the regret compared to the ProjectedVolume algorithm in [27].

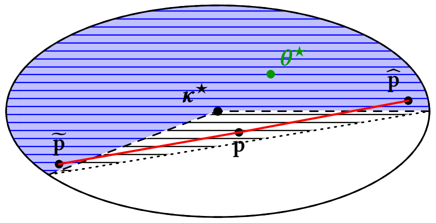

**Figure 1: Pictorial representation of Carathéodory's theorem in two dimensions. The ellipsoid corresponds to the current knowledge set $\mathcal{K}$. The blue area corresponds to the $\bar{c}$-protected region, $\mathcal{P}(\bar{c})$. The patterned area corresponds to its convex hull, $\mathsf{conv}(\mathcal{P}(\bar{c}))$. Point $\mathbf{p} \in \mathsf{conv}(\mathcal{P}(\bar{c}))$ can be written as a convex combination of points $\widehat{\mathbf{p}}, \widetilde{\mathbf{p}} \in \mathcal{P}(\bar{c})$.**

**How do we initialize the next epoch?** Since the existence of the separating cut is established, if we were able to compute this cut, we would be able to compute the knowledge set of the next epoch by taking its intersection with the positive halfspace of the cut. However, the separating hyperplane theorem provides only an existential argument and no direct way to compute the separating cut. To deal with this, recall that the separating cut should have $\mathbf{p}^\star$ on its negative halfspace and the whole protected region in its positive halfspace. To compute it, we use the Perceptron algorithm [37], which is typically used to provide a linear classifier for a set of (positive and negative) points in the *realizable* setting (i.e., when there exists a hyperplane that correctly classifies these points). Perceptron proceeds by iterating across the points and suggesting a classifier. Every time that a point is misclassified, Perceptron makes an update. If the entire protected region is classified as positive and $\mathbf{p}^\star$ as negative by Perceptron, then we return its hyperplane as the separating cut; otherwise we feed one point that violates the intended labeling to Perceptron, which makes a mistake and updates its classifier. The main guarantee of Perceptron is that, if there exists a classifier with margin of $\gamma > 0$ (i.e., smallest distance to any data point is $\gamma$), the number of mistakes that Perceptron makes is at most of the order of $1/\gamma^2$.

The problem is that we do not know $\mathbf{p}^\star$ and, even if we deal with this, $\mathbf{p}^\star$ does not necessarily have a large enough margin from the protected region. To overcome this, we provide a sampling process that with big enough probability identifies a different point $\mathbf{q}$, *in the vicinity* of $\mathbf{p}^\star$, whose margin to the protected region is lower bounded by $\gamma$. If $\mathbf{q}$ does have the desired margin, the mistake bound of Perceptron controls the running time needed to identify the separating hyperplane. Otherwise, we proceed with a new random point. This takes care of the margin-problem of $\mathbf{p}^\star$.

To pin down $\mathbf{p}^\star$, we construct a set of points $\Lambda_\phi$, which we call *landmarks*, such that at least one of them is outside of the convex hull of the protected region. We run multiple versions of Perceptron, each with a random $\mathbf{p}^\star \in \Lambda_\phi$ and a point $\mathbf{q}$ randomly selected in a ball around $\mathbf{p}^\star$ of an appropriately defined radius $\zeta$, which we denote by $\mathcal{B}_{L_\phi}(\mathbf{p}^\star, \zeta)$. [5] If $\mathbf{q}$ has a big enough margin $\gamma$ then the mistake bound of Perceptron ensures that CorPV.SeparatingCut (Algorithm 3) returns the separating cut.

**ALGORITHM 3:** CorPV.SeparatingCut

```
Parameters: $\kappa^\star_\phi, S_\phi, L_\phi, \mathcal{A}_\phi$ ▷ size of small dimensions $\delta := \frac{\varepsilon}{4(d+\sqrt{d})}$
Fix landmarks $\Lambda_\phi = \left\{\kappa^\star_\phi \pm \bar{\nu} \cdot e_i, \forall e_i \in E_\phi\right\}$ where $E_\phi$ is an
  orthonormal basis on $L_\phi$ and $\bar{\nu} = \frac{\varepsilon - 2\sqrt{d}\cdot\delta}{4\sqrt{d}}$
while true do
    Initialize Perceptron to $(\widetilde{\mathbf{h}}, \widetilde{\omega})$ and mistake counter to $M = 0$.
    Sample a random point $\mathbf{q}$ from ball $\mathcal{B}_{L_\phi}(\mathbf{p}^\star, \zeta)$ with radius $\zeta = \bar{\nu}$
      around random $\mathbf{p}^\star \in \Lambda_\phi$.
    while $M < \frac{d-1}{\zeta^2 \cdot \ln^2(3/2)}$ do                ▷ Perceptron mistake bound
        Set $m \leftarrow 0$.
        if $\mathbf{q} \in \mathrm{H}^-(\widetilde{\mathbf{h}}, \widetilde{\omega})$ then make Perceptron update of $(\widetilde{\mathbf{h}}, \widetilde{\omega})$ on $\mathbf{q}$
          with label $-$; set $m \leftarrow m+1$.
        if $\kappa_\phi \in \mathrm{H}^+(\widetilde{\mathbf{h}}, \widetilde{\omega})$ then make Perceptron update of $(\widetilde{\mathbf{h}}, \widetilde{\omega})$ on
          $\kappa_\phi$ with label $+$; set $m \leftarrow m+1$.
        for subsets $D_\phi \subseteq \mathcal{A}_\phi$ such that $|D_\phi| = \bar{c}$ do
            Let $P$ be the polytope created by halfspaces of $\mathcal{A}_\phi \setminus D_\phi$
              and $\mathrm{H}^-(\widetilde{\mathbf{h}}, \widetilde{\omega})$.
            if $P \neq \emptyset$ then make Perceptron update of $(\widetilde{\mathbf{h}}, \widetilde{\omega})$ on
              $\mathbf{z} \in P$ with label $+$; set $m \leftarrow m+1$.
        if $m \neq 0$ then increase mistake counter $M \leftarrow M + m$.
        else return $(\widetilde{\mathbf{h}}, \widetilde{\omega})$
```

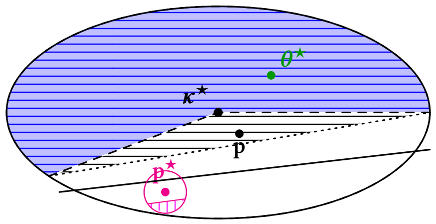

**Figure 2: Pictorial representation of the computation of a valid, separating cut. All points inside $\mathsf{conv}(\mathcal{P}(\bar{c}))$ (horizontal-line pattern) have undesirability *at most* $(d+1)\bar{c}$. Point q has undesirability *at least* $2d(d+1)\bar{c}+1$. Sampling points from the spherical cap of the magenta ball (vertical-line pattern) gives a big enough margin for Perceptron. The black, solid line corresponds to the valid separating cut.**

Indeed, we can lower bound the probability that $\mathbf{q}$ belongs to the *spherical cap* that is illustrated by the vertical-line-patterned region in Figure 2, which is appropriately close to $\mathbf{p}^\star$ (i.e., far enough to guarantee a margin but close enough to guarantee proximity to $\kappa^\star$).

[5]Such a point can be computed efficiently by normalizing $\mathcal{B}_{L_\phi}(\mathbf{p}^\star, \zeta)$ to a unit ball and then using the techniques presented in [6, Section 2.5].

If the sampled $\mathbf{q}$ belongs to this spherical cap, then the Perceptron algorithm terminates after $O(1/\gamma^2)$ mistakes; otherwise the whole process repeats. This bounds the number of iterations of the outer *while* loop and thereby also the running time. A technical novelty is that we use the mistake bound of the Perceptron[6] algorithm *as a witness* in order to guarantee that we have sampled an appropriate $\mathbf{q}$ rather than as a way to directly bound regret.

## 3.2 Adapting to an Unknown Corruption Level

We now outline the algorithm when the corruption level $C$ is unknown, called CorPV.AI. Due to its cumbersome notation, we defer its formal statement to the full version. This section extends ideas from [28] for multi-armed bandits to contextual search which poses an additional difficulty as the search space is continuous. This is not as straightforward as a doubling trick for the unknown $C$, as both the loss and the corruption $c_t$ are unobservable; doubling tricks require identifying a proxy for the quantity under question and doubling once a threshold is reached.

The basic idea is to maintain multiple copies of CorPV.Known, which we refer to as *layers*. At every round, we decide which copy to play probabilistically. Each copy $j$ keeps its own environment with its corresponding epoch $\phi(j)$ and knowledge set $\mathcal{K}_{j,\phi(j)}$. Smaller values $j$ for the copies are *less robust* to corruption and we impose a monotonicity property among them by ensuring that the knowledge sets are nested, i.e., $\mathcal{K}_{j,\phi(j)} \subseteq \mathcal{K}_{j',\phi(j')}$ for $j \leq j'$. This allows more robust layers to correct mistakes of less robust layers that may inadvertently eliminate $\boldsymbol{\theta}^\star$ from their knowledge set.

More formally, we run $\log T$ parallel versions of the $\bar{c}$-known-corruption algorithm with a corruption level of $\bar{c} \approx \log(T)$. At the beginning of each round $t$, the algorithm randomly selects layer $j$ with probability $2^{-j}$ and executes the layer's algorithm for this round. Since the adversary does not know the randomness in the algorithm, this makes layers $j$ with $C \leq 2^j$ robust to corruption level of $C$. The reason is that the expected number of corruptions occurring at layer $j$ is at most 1 and, with high probability, less than $\log T$ which is accounted by the $\bar{c} = \log T$ upper bound on corruption based on which we run CorPV.Known on this layer.

However, there is a problem: all layers with $C > 2^j$ are not robust to corruption of $C$ so they may eliminate $\boldsymbol{\theta}^\star$ and, to make things worse, the algorithm follows the recommendation of these layers with large probability. As a result, we need a way to supervise their decisions by more robust layers. To achieve that, we use nested active sets; when the layer $j_t$ selected at round $t$ proceeds with a separating cut on its knowledge set, we also make the same cut on all less robust layers $j' < j_t$. This allows non-robust layers that have eliminated $\boldsymbol{\theta}^\star$ from their knowledge set to correct their mistakes by removing the incorrect parameters of their version space that they had converged to from their knowledge sets.

There are two additional points that arise in the contextual search setting. First, the aforementioned cut may not make enough volumetric progress in the knowledge sets of layers $j' < j_t$. As a result, we only move to the next epoch for layer $j'$ if its centroid is removed from the knowledge set or another change discussed in Section 3.3 is triggered. Second, with respect to exploit queries, we want to make sure that we do not keep confidence on non-robust layers, so we follow the exploit recommendation of the largest layer $j \geq j_t$ that has converged to exploit recommendation in this direction, i.e., $w(\mathcal{K}_{j,\phi(j)}) \leq \varepsilon$. This eventually allows us to bound the regret from all non-robust layers by the smallest robust layer $\lceil \log C \rceil$.

[6]The use of Perceptron is not mandated to find a separating hyperplane between the landmark and $\mathsf{conv}(\mathcal{P}(\bar{c}))$. In fact, one of the STOC'21 reviewers proposed an approach that builds on our construction but, instead of using Perceptron and spherical-cap arguments, creates $2d$ appropriate convex programs in order to identify the separating hyperplane. We discuss the proposed algorithm in the full version.

## 3.3 Remaining Components of the Algorithm

The presentation of the algorithm until this point has disregarded some technical parts, which we highlight in this section.

**Centroid versus approximate centroid.** Computing the centroid $\boldsymbol{\kappa}^\star$ of a convex body $\mathcal{K}$ is #P-hard, so instead, algorithm CorPV.Known (and all the subroutines it calls) uses *approximate* centroids which can be efficiently approximated [27]. This approximate centroid is close enough to the actual one so that the regret computations are not affected.

**Cylindrification, small, and large dimensions.** To facilitate relating the volume progress to a bound on the explore queries, similar to [27], we keep two sets of vectors/dimensions $S_\phi$ and $L_\phi$ whose union creates an orthonormal basis. The set $S_\phi$ has *small dimensions* $\mathbf{s} \in S_\phi$ with width $w(\mathcal{K}_\phi, \mathbf{s}) \leq \delta$ for $\delta := \frac{\varepsilon}{4(d+\sqrt{d})}$. The set $L_\phi$ is any basis for the subspace orthogonal to $S_\phi$, with the property that $\forall \mathbf{l} \in L_\phi : w(\mathcal{K}_\phi, \mathbf{l}) > \delta$. When an epoch ends, sets $S_\phi$ and $L_\phi$ are updated together with the knowledge set $\mathcal{K}_\phi$ according to CorPV.EpochUpdates. If the new direction $\widetilde{\mathbf{h}}$ of the separating cut projected to the large dimensions has width $w(\Pi_{L_\phi}\mathcal{K}_{\phi+1}, \widetilde{\mathbf{h}}) \leq \delta$, we add it to $S_{\phi+1}$ and we update $L_{\phi+1}$ to keep the invariant that no large dimension has width larger than $\delta$; see the full version for the formal description of CorPV.EpochUpdates.

Overall, the potential function we use to make sure that we make progress depends on the projected volume of the knowledge set on the large dimensions $L_\phi$, as well as the number of small dimensions $S_\phi$. Sets $S_\phi$ and $L_\phi$ serve in explaining which dimensions are identified well enough so that we can focus our attention on making progress in the remaining dimensions.

In algorithm CorPV.Known.Simple we have assumed that all operations and conditions are done/taken directly on the knowledge set $\mathcal{K}_\phi$. This is not precisely true. In reality, all operations/conditions are done/taken on a *variant* of set $\mathcal{K}_\phi$, called the *cylindrification* of the set, which creates a box covering the knowledge set and removes the significance of the small dimensions. This is also done in the algorithm ProjectedVolume of Lobel, Paes Leme, and Vladu [27]. Formally, the Cylindrification of a set is defined below.

**Definition 3.2** (Cylindrification, Definition 4.1 of [27]). *Given a set of orthonormal vectors $S = \{\mathbf{s}_1, \dots, \mathbf{s}_n\}$, let $L = \{\mathbf{u} | \langle \mathbf{u}, \mathbf{s} \rangle = 0; \forall \mathbf{s} \in S\}$ be a subspace orthogonal to $\mathsf{span}(S)$ and $\Pi_L\mathcal{K}$ be the projection of convex set $\mathcal{K} \subseteq \mathbb{R}^d$ onto L. We define:*

$$\mathtt{Cyl}(\mathcal{K}, S) = \left\{ \mathbf{z} + \sum_{i=1}^{n} b_i \mathbf{s}_i \middle| \mathbf{z} \in \Pi_L \mathcal{K}, \min_{\boldsymbol{\theta} \in \mathcal{K}} \langle \boldsymbol{\theta}, \mathbf{s}_i \rangle \leq b_i \leq \max_{\boldsymbol{\theta} \in \mathcal{K}} \langle \boldsymbol{\theta}, \mathbf{s}_i \rangle \right\}.$$

By working with the Cylindrification $\mathtt{Cyl}(\mathcal{K}_\phi, S_\phi)$ (Definition 3.2) rather than the original set of small dimensions $S_\phi$, we can ensure that we make queries that make volumetric progress with respect to the large dimensions, that have been less well understood. This is also the reason why the landmark $\mathbf{p}^\star$ that we identify lives in the large dimensions while also being close to the centroid $\boldsymbol{\kappa}_\phi$.

**Exploit queries for different loss functions.** When the width of the knowledge set on the direction of the incoming context is small, i.e., $w(\mathcal{K}_\phi, \mathbf{x}_t) \leq \varepsilon$, we proceed with an exploit query. This module evaluates the loss of each query $\omega$ with respect to any parameter that is consistent with the knowledge set, i.e., $\boldsymbol{\theta}^\star \in \mathcal{K}_\phi$. It then employs a min-max approach by selecting the query $\omega_t$ that has the minimum loss for the worst-case selection of $\boldsymbol{\theta} \in \mathcal{K}_\phi$.

**ALGORITHM 4:** CorPV.Exploit

```
Parameters: $\mathbf{x}_t$, $\mathcal{K}_\phi$
Compute query point $\omega_t = \min_{\omega\in\Omega} \max_{\boldsymbol{\theta}\in\mathcal{K}_\phi} \ell(\omega, \langle \boldsymbol{\theta}, \mathbf{x}_t \rangle, \langle \boldsymbol{\theta}, \mathbf{x}_t \rangle)$
return $\omega_t$
```

For the $\varepsilon$-ball loss, any query point $\omega_t = \langle \mathbf{x}_t, \boldsymbol{\theta}' \rangle$ with $\boldsymbol{\theta}' \in \mathcal{K}_\phi$ results in loss equal to 0; this is what ProjectedVolume also does to achieve optimal regret for the $\varepsilon$-ball loss function.

Moving to the pricing loss and assuming that the query point is $\omega_t = \langle \mathbf{x}_t, \boldsymbol{\theta} \rangle$ for some $\boldsymbol{\theta} \in \mathcal{K}_\phi$, although the distance of $\boldsymbol{\theta}^\star$ to hyperplane $(\mathbf{x}_t, \omega_t$ is less than $\varepsilon$, there is a big difference based on which side of the hyperplane $\boldsymbol{\theta}^\star$ lies in (i.e., whether $\boldsymbol{\theta}^\star \in \mathbf{H}^+(\mathbf{x}_t, \omega_t)$ or not). Specifically, if $\omega_t > \langle \mathbf{x}_t, \boldsymbol{\theta}_t^\star \rangle$ then a fully rational agent does not buy and we get zero revenue, thereby incurring a loss of $\langle \mathbf{x}_t, \boldsymbol{\theta}^\star \rangle$. However, querying $\omega^\star = \langle \mathbf{x}_t, \boldsymbol{\theta}^\star \rangle$ would lead to a purchase from a fully rational agent, and hence, to a pricing loss of 0. [7]

One way to deal with this discontinuity is by querying point $\omega_t$ with $\omega_t = \langle \mathbf{x}_t, \boldsymbol{\theta}^\star \rangle - \varepsilon$, as the value of the fully rational agent is certainly above this price. For the behavioral model of bounded rationality, even if we know $\boldsymbol{\theta}^\star$, for the pricing loss, we should select a slightly smaller price to account for the noise and the definition of $\omega_t$ takes into account the distributional information about the noise of the behavioral model.

**Computational complexity.** We finally discuss the computational complexity of our algorithm. From Algorithm 3, checking whether the protected region is contained in the positive halfspace of the Perceptron hyperplane requires going over all $\binom{|\mathcal{A}_\phi|}{\bar{c}}$ ways to remove $\bar{c}$ hyperplanes and checking whether the resulting region intersects the negative halfspace (if this happens, then points with misclassification of at most $\bar{c}$ may be misclassified). This suggests a running time that is exponential in $\bar{c}$. Fortunately, to handle the unknown corruption or the other intricacies in our actual behavioral model beyond the $\bar{c}$-known-corruption, we only run this algorithm with $\bar{c} \approx \log(T)$. As a result, the final running time of our algorithms is quasi-polynomial in $T$.

## 4 CONCLUSION

In this paper, we initiated the study of contextual search under adversarial noise models, motivated by pricing settings where some agents may act *irrationally*, i.e., in ways that are inconsistent with respect to the underlying ground truth. Although classical algorithms may be prone to even a few such agents, we attained the logarithmic (uncorrupted) regret guarantee, while degrading gracefully with the number $C$ of irrational agents.

Our work opens up two fruitful avenues for future research. First, the regret in both of our algorithms is sublinear when $C = o(T)$ but becomes linear when $C = \Theta(T)$. Designing algorithms that can provide sublinear regret against the ex-post best linear model, in the latter regime, is an exciting direction of future research and our model offers a concrete formulation of this problem. Second, our algorithm which attains the logarithmic guarantee has a regret of the order of $Cd^3 \text{poly} \log T$. Improving the dependence on $d$ is an interesting open direction. After a sequence of papers, this dependence is now optimized when all agents are fully rational [11, 26, 27, 33].

[7] This discontinuity in pricing loss poses further complications in extending our gradient descent algorithm to this loss as we discuss in the full version.

## FULL VERSION

The full version of the paper can be found on the following ArXiv link: https://arxiv.org/abs/2002.11650. The STOC'21 peer-reviewed version is: https://arxiv.org/abs/2002.11650v3.

## ACKNOWLEDGMENTS

We thank the anonymous reviewers for valuable suggestions and ideas that helped us improve our paper. A large part of the work was done while CP was an intern at Microsoft Research NYC. CP is supported in part under grant No. CCF-1718549 of the National Science Foundation and the Harvard Data Science Initiative.

## REFERENCES

[1] Kareem Amin, Afshin Rostamizadeh, and Umar Syed. 2014. Repeated contextual auctions with strategic buyers. In *Advances in Neural Information Processing Systems.*
[2] Idan Amir, Idan Attias, Tomer Koren, Roi Livni, and Yishay Mansour. 2020. Prediction with Corrupted Expert Advice. *Proceedings of 32nd Advances in Neural Processing Systems (NeurIPS)* (2020).
[3] Gah-Yi Ban and N Bora Keskin. 2017. Personalized dynamic pricing with machine learning. *Available at SSRN 2972985* (2017).
[4] Hamsa Bastani and Mohsen Bayati. 2020. Online Decision Making with High-Dimensional Covariates. *Oper. Res.* 68, 1 (2020), 276–294. https://doi.org/10.1287/opre.2019.1902
[5] Omar Besbes and Assaf Zeevi. 2009. Dynamic Pricing Without Knowing the Demand Function: Risk Bounds and Near-Optimal Algorithms. *Oper. Res.* 57, 6 (Nov. 2009), 1407–1420. https://doi.org/10.1287/opre.1080.0640
[6] Avrim Blum, John Hopcroft, and Ravindran Kannan. 2016. *Foundations of data science.*
[7] Ilija Bogunovic, Andreas Krause, and Jonathan Scarlett. 2020. Corruption-Tolerant Gaussian Process Bandit Optimization. *International Conference on Artificial Intelligence and Statistics (AISTATS)* (2020).
[8] Josef Broder and Paat Rusmevichientong. 2012. Dynamic pricing under a general parametric choice model. *Operations Research* 60, 4 (2012), 965–980.
[9] Nicolo Cesa-Bianchi, Tommaso Cesari, and Vianney Perchet. 2019. Dynamic pricing with finitely many unknown valuations. In *Algorithmic Learning Theory.* PMLR, 247–273.
[10] Xi Chen, Akshay Krishnamurthy, and Yining Wang. 2019. Robust Dynamic Assortment Optimization in the Presence of Outlier Customers. *arXiv:1910.04183* (2019).
[11] Maxime Cohen, Ilan Lobel, and Renato Paes Leme. 2019. Feature-based dynamic pricing. *Management Science* (2019).
[12] Arnoud V den Boer. 2014. Dynamic pricing with multiple products and partially specified demand distribution. *Mathematics of operations research* 39, 3 (2014), 863–888.
[13] Arnoud V den Boer. 2015. Dynamic pricing and learning: historical origins, current research, and new directions. *Surveys in operations research and management science* 20, 1 (2015), 1–18.
[14] Arnoud V den Boer and Bert Zwart. 2014. Simultaneously learning and optimizing using controlled variance pricing. *Management science* 60, 3 (2014), 770–783.
[15] Ehsan Emamjomeh-Zadeh, David Kempe, and Vikrant Singhal. 2016. Deterministic and probabilistic binary search in graphs. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing.* 519–532.

[16] Michal Feldman, Tomer Koren, Roi Livni, Yishay Mansour, and Aviv Zohar. 2016. Online Pricing with Strategic and Patient Buyers. In *Advances in Neural Information Processing Systems*, D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett (Eds.), Vol. 29. Curran Associates, Inc.

[17] Negin Golrezaei, Patrick Jaillet, and Jason Cheuk Nam Liang. 2019. Incentive-aware Contextual Pricing with Non-parametric Market Noise. *arXiv preprint arXiv:1911.03508* (2019).

[18] Negin Golrezaei, Adel Javanmard, and Vahab Mirrokni. 2019. Dynamic incentive-aware learning: Robust pricing in contextual auctions. In *Advances in Neural Information Processing Systems*. 9759–9769.

[19] Anupam Gupta, Tomer Koren, and Kunal Talwar. 2019. Better algorithms for stochastic bandits with adversarial corruptions. In *Conference on Learning Theory*.

[20] Adel Javanmard and Hamid Nazerzadeh. 2019. Dynamic pricing in high-dimensions. *The Journal of Machine Learning Research* 20, 1 (2019), 315–363.

[21] Richard M Karp and Robert Kleinberg. 2007. Noisy binary search and its applications. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*. 881–890.

[22] N Bora Keskin and Assaf Zeevi. 2014. Dynamic pricing with an unknown demand model: Asymptotically optimal semi-myopic policies. *Operations Research* 62, 5 (2014), 1142–1167.

[23] Robert Kleinberg and Tom Leighton. 2003. The value of knowing a demand curve: Bounds on regret for online posted-price auctions. In *Symposium on Foundations of Computer Science*. IEEE.

[24] Thibault Le Guen. 2008. *Data-driven pricing*. Ph.D. Dissertation. Massachusetts Institute of Technology.

[25] Yingkai Li, Edmund Y Lou, and Liren Shan. 2019. Stochastic Linear Optimization with Adversarial Corruption. *arXiv:1909.02109* (2019).

[26] Allen Liu, Renato Paes Leme, and Jon Schneider. 2021. Optimal Contextual Pricing and Extensions. In *Symposium on Discrete Algorithms*.

[27] Ilan Lobel, Renato Paes Leme, and Adrian Vladu. 2018. Multidimensional binary search for contextual decision-making. *Operations Research* (2018).

[28] Thodoris Lykouris, Vahab S. Mirrokni, and Renato Paes Leme. 2018. Stochastic bandits robust to adversarial corruptions. In *Symposium on Theory of Computing*.

[29] Thodoris Lykouris, Max Simchowitz, Aleksandrs Slivkins, and Wen Sun. 2019. Corruption Robust Exploration in Episodic Reinforcement Learning. *ArXiv* abs/1911.08689 (2019).

[30] Jieming Mao, Renato Paes Leme, and Jon Schneider. 2018. Contextual pricing for Lipschitz buyers. In *Advances in Neural Information Processing Systems*.

[31] Mehryar Mohri and Andres Munoz Medina. 2014. Optimal Regret Minimization in Posted-Price Auctions with Strategic Buyers. In *Advances in Neural Information Processing Systems*, Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger (Eds.), Vol. 27. Curran Associates, Inc.

[32] Mila Nambiar, David Simchi-Levi, and He Wang. 2019. Dynamic learning and pricing with model misspecification. *Management Science* 65, 11 (2019), 4980–5000.

[33] Renato Paes Leme and Jon Schneider. 2018. Contextual search via intrinsic volumes. In *Symposium on Foundations of Computer Science*. IEEE.

[34] Andrzej Pelc. 2002. Searching games with errors—fifty years of coping with liars. *Theoretical Computer Science* 270, 1-2 (2002), 71–109.

[35] Sheng Qiang and Mohsen Bayati. 2016. Dynamic pricing with demand covariates. *Available at SSRN 2765257* (2016).

[36] Ronald L. Rivest, Albert R. Meyer, Daniel J. Kleitman, Karl Winklmann, and Joel Spencer. 1980. Coping with errors in binary search procedures. *J. Comput. System Sci.* 20, 3 (1980), 396–404.

[37] Frank Rosenblatt. 1958. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review* (1958).

[38] Virag Shah, Ramesh Johari, and Jose Blanchet. 2019. Semi-parametric dynamic contextual pricing. In *Advances in Neural Information Processing Systems*. 2360–2370.

[39] Joel Spencer. 1992. Ulam's searching game with a fixed number of lies. *Theoretical Computer Science* 95, 2 (1992), 307–321.

[40] Stanisław M. Ulam. 1976. *Adventures of a mathematician*. Charles Scribner's Sons, New York, NY, USA.

[41] Julian Zimmert and Yevgeny Seldin. 2021. Tsallis-INF: An optimal algorithm for stochastic and adversarial bandits. *Journal of Machine Learning Research (JMLR)* (2021).