Proceedings of Machine Learning Research vol 237:1–21, 2024 35th International Conference on Algorithmic Learning Theory

Corruption-Robust Lipschitz Contextual Search

Shiliang Zuo SZUO3@ILLINOIS.EDU
University of Illinois Urbana-Champaign

Editors: Claire Vernade and Daniel Hsu

Abstract

I study the problem of learning a Lipschitz function with corrupted binary signals. The learner tries to learn a L-Lipschitz function $f : [0,1]^d \to [0, L]$ that the adversary chooses. There is a total of T rounds. In each round t, the adversary selects a context vector x_t in the input space, and the learner makes a guess to the true function value $f(x_t)$ and receives a binary signal indicating whether the guess is high or low. In a total of C rounds, the signal may be corrupted, though the value of C is *unknown* to the learner. The learner's goal is to incur a small cumulative loss. This work introduces the new algorithmic technique *agnostic checking* as well as new analysis techniques. I design algorithms which: for the absolute loss, the learner achieves regret $L \cdot O(C \log T)$ when $d = 1$ and $L \cdot O_d(C \log T + T^{(d-1)/d})$ when $d > 1$; for the pricing loss, the learner achieves regret $L \cdot \widetilde{O}(T^{d/(d+1)} + C \cdot T^{1/(d+1)})$.

Keywords: Online Learning, Dynamic Pricing, Corruption-Robust Algorithms

1. Introduction

Consider the dynamic pricing problem, where a seller attempts to sell a product to a buyer each day without any prior knowledge of the maximum price u the buyer is willing to pay (i.e. the buyer's private valuation for the product). The seller must learn this price by observing the buyer's behavior, which takes the form of a binary signal: a purchased or an unpurchased item. An unpurchased item indicates an overpriced product, leading to a possible loss of the entire revenue u; a purchased item indicates an underpriced product, leading to a possible loss of surplus (the difference between u and the posted price). However, the seller does not directly observe the revenue loss and only observes the binary signal of the buyer. The seller adjusts the posted price according to the signal and gradually converges to the optimal price. An early important work by Kleinberg and Leighton (2003) studied dynamic pricing from a regret-minimization perspective and fully characterize the optimal regret of the seller for the most basic form of this problem.

The *contextual search* (or *contextual pricing*) problem (Liu et al. (2021); Leme and Schneider (2022); Mao et al. (2018)) is motivated by the fact that buyers have different valuations for differentiated products with different features. Henceforth the features of products are referred to as *context* vectors. A common assumption is that the buyer's valuation is a linear function on the context vector (Liu et al. (2021); Leme and Schneider (2022)). In this setting, Liu et al. (2021) obtains a nearly optimal regret bound. Another assumption is that the valuation function is Lipschitz with respect to the context vector. This setting was studied by Mao et al. (2018), in which they gave nearly optimal algorithms based on discretization of the input space.

In practice, it is unreasonable to assume the signals the seller receives are perfect. For example, buyers can act irrationally, or in extreme cases may even exhibit malicious behaviors, resulting in faulty signals. This motivates the study of designing contextual search algorithms robust to corruptions in the binary signals that the learner observes. Previous works (Krishnamurthy et al.

(2022); Leme et al. (2022)) studied *linear* contextual search with corruptions, in which an adversary has the power to corrupt the binary signals. This work studies *Lipschitz* contextual search with corruptions and proposes corruption-robust algorithms.

Loss Functions and Applications. The literature on contextual search usually considers two loss functions, the *pricing loss* and the *absolute loss* (e.g Mao et al. (2018), Krishnamurthy et al. (2022)). The pricing loss is defined by

$$\ell(q, f(x)) = f(x) - q \cdot \mathbb{1}\{q \leq f(x)\}.$$

The pricing loss captures the dynamic pricing setting described above. The optimal price for a product with context vector x is $f(x)$. If the seller overprices $q > f(x)$, she loses the entire revenue $f(x)$; if the seller underprices $q \leq f(x)$, she loses the potential surplus $f(x) - q$. The cumulative pricing loss then measures the regret of the seller had she known the valuation function f. It should be noted the actual loss of each round is never revealed to the learner, and only the binary signal is revealed.

The absolute loss is defined as

$$\ell(q, f(x)) = |q - f(x)|.$$

An application of absolute loss is personalized medicine. Consider a healthcare provider experimenting with a new drug. Assume the optimal dosage for a patient with context feature x is $f(x)$, and the given dosage is q. The absolute loss then measures the distance between the injected dosage and the optimal dosage. The learner (healthcare provider) is not able to directly observe this loss but can observe a binary signal informing whether she overdosed or underdosed.

For either the pricing loss or absolute loss ℓ, the cumulative loss (or regret, this work will use the two terms interchangebly) of the learner after T rounds is then defined as:

$$\sum_{t \in [T]} \ell_t = \sum_{t \in [T]} \ell(q_t, f(x_t)).$$

1.1. Contributions

In this work, I design corruption-robust algorithms for learning Lipschitz functions with binary signals. The learner tries to learn a Lipschitz function f chosen by an adversary. The domain of f is the d-dimensional cube $[0, 1]^d$. At each round, the adversary presents the learner with an adversarially chosen context vector $x_t \in [0, 1]^d$, and the learner submits a guess q_t to the value of $f(x_t)$ and observes a binary signal indicating whether the guess is low or high. Throughout the course of T rounds, a total of C signals may be corrupted, though the value of C is unknown to the learner. The goal of the learner will be to incur a small cumulative loss that degrades gracefully as C increases.

I design corruption-robust algorithms under two loss functions, absolute loss and pricing loss. The corruption-robust algorithm for absolute loss is given in Section 3, and the algorithm for pricing loss is given in Section 4. The main results and comparison with the most relevant works are summarized in Table 1. I also give lower bounds showing the upper bounds in this work are tight.

This work introduces a new algorithmic technique of *agnostic checking* and shows how *agnostic checking* can be incorporated into the adaptive discretization and uniform discretization procedure,

Setting	Adversary	Absolute Loss	Pricing Loss
Linear	No	$O(d \log d)$ Liu et al. (2021)	$O(d \log \log T + d \log d)$ Liu et al. (2021)
	Weak	$O(C + d \log T)$ Leme et al. (2022)	$O(Cd^3 \texttt{poly} \log(T))$ Krishnamurthy et al. (2022)
Lipschitz	No	$O(T^{(d-1)/d})$ $O(\log T)$ when $d = 1$ Mao et al. (2018)	$O(T^{d/(d+1)})$ Mao et al. (2018)
	Strong	$O(C \log T + T^{(d-1)/d})$ $O(C \log T)$ when $d = 1$ (This work)	$\widetilde{O}(C \cdot T^{1/(d+1)} + T^{d/(d+1)})$ (This work)

Table 1: Regret for contextual search under different settings. For the Lipschitz contextual search problem, the Lipschitz L is treated as constant and omitted in the regret. See Remark 2 for the definition of weak / strong adversary.

usually used in learning Lipschitz functions. At a very high level, the input space is discretized into "bins", and the learner maintains an associated range for each bin which serves as an estimate for the image of f in the bin. The associated range becomes more accurate as the learner submits more queries and observes more binary signals. However, since the binary signals may be corrupted, the associated range may not be accurate. The learner performs checking steps by querying the boundaries of the associated range and tries to ensure the associated range is accurate. A key challenge is that checking steps may also be corrupted, hence the new technique is termed "agnostic checking". New analysis techniques are introduced to bound the regret while the learner is agnostic as to whether the checking steps are corrupted or not.

1.2. Related Work

The two most related threads to this work are contextual search and corruption-robust learning. The linear contextual search problem was studied in Liu et al. (2021); Leme and Schneider (2022). The recent work by Liu et al. (2021) achieved state-of-the-art regret bounds. For the Lipschitz contextual search problem, Mao et al. (2018) proposed algorithms based on adaptive discretization and binary search. They also showed their algorithm to be nearly optimal. The contextual search problem is also closely related to the dynamic pricing problem, for an overview see Den Boer (2015).

Recently, designing learning algorithms robust to corruption and data-poisoning attacks has received much attention. Leme et al. (2022) gave corruption-robust algorithms for the linear contextual problem based on density updates, improving over the work by Krishnamurthy et al. (2022). The study of adversarial attack and the design of corruption-robust algorithms has also appeared in bandit learning (Lykouris et al. (2018); Gupta et al. (2019); Jun et al. (2018); Zuo (2023); Garcelon et al. (2020); Bogunovic et al. (2021)) and reinforcement learning (Lykouris et al. (2021); Zhang et al. (2022); Chen et al. (2021)), to name a few.

This work is also related to the contextual bandits and bandits in metric spaces (Kleinberg et al. (2008); Kleinberg and Slivkins (2010); Cesa-Bianchi et al. (2017); Slivkins (2011)). The notable work by Slivkins (2011) studied a setting where the learner picks an arm after observing the contextual information each round, and subsequently the context-arm dependent reward (or equivalently loss) is revealed.

Another interesting direction peripheral to this work is learning with strategic agents. In this setting, the learner is not interacting with an adversary but with a strategic agent whose goal is to maximize his long-term utility. Amin et al. (2013) studied designing auctions with strategic buyers; Amin et al. (2014) and Drutsa (2020) incorporated contextual information and designed contextual auctions with strategic buyers.

1.3. Comparison with previous work

The problem of Lipschitz contextual search was studied in Mao et al. (2018), in which they proposed near-optimal algorithms when the binary signals are perfect each round (i.e. no corruptions). This work studies the Lipschitz contextual search with adversarial corruptions and designs corruption-robust algorithms while being agnostic to the corruption level C.

Krishnamurthy et al. (2022) and Leme et al. (2022) studied the linear contextual search problem with adversarial corruptions. In their work, the underlying unknown function f takes the parametric linear form $f(x) = \langle \theta, x \rangle$, where θ is the unknown parameter and x is the context vector. Krishnamurthy et al. (2022) proposed maintaining a knowledge set that contains the true parameter; Leme et al. (2022) proposed maintaining a density over the parameter space. This work does not make any parametric assumptions and works with the non-parametric Lipschitz assumption, i.e. f is only assumed to be Lipschitz. Moreover, the adversary model in this work is stronger than the adversary model studied in their work (see Remark 2 below).

Leme et al. (2021) studied an algorithmic solution to the non-stationary dynamic pricing problem, which is superficially similar to this work. Their work also used the notion of "checking steps". This work is different from theirs in the following aspects. The algorithm in this work is designed for an *adversarial corruption* setting with *context* information (under the Lipschitz assumption). Since checking steps can also be corrupted and the learner will not know whether the signal obtained from checking steps is accurate, this work terms these as *agnostic checking*. This work shows entirely new analysis techniques on how agnostic checking can be incorporated with the discretization technique used for learning Lipschitz functions. By contrast, Leme et al. (2021) studied a non-stationary dynamic pricing setting without context and when there is no corruption in the binary signals.

Some recent work also studied corruption-robust algorithms for contextual bandit problems (e.g. Bogunovic et al. (2021) and He et al. (2022)). Note that the feedback and reward model in this work is fundamentally different from contextual bandit problems. Specifically, for contextual bandits, the adversary first chooses a function (the reward function), and then the learner chooses some action (based on some context) each round and observes a (possibly corrupted) reward. In this work, the contexts are chosen by the adversary, and the learner guesses the value of the unknown function and only observes a (possibly corrupted) binary signal of whether the guess is low or high.

2. Problem Formulation

Definition 1 *Let $f : \mathbb{R}^d \to \mathbb{R}$. If f satisfies:*

$$|f(x) - f(y)| \leq L\|x - y\|_\infty$$

for any x, y, then f is L-Lipschitz.

An adversary selects an L-Lipschitz function $f : [0,1]^d \to [0,L]$. The function f is initially unknown to the learner. At each round t, the interaction protocol proceeds as follows:

1. The adversary selects a *context* x_t in the input space $\mathcal{X} = [0,1]^d$.
2. The learner observes x_t and submits a guess q_t to the value $f(x_t)$.
3. The adversary observes q_t and computes $\sigma(q_t - f(x_t))$. Here $\sigma(x)$ is the step function that takes value 1 if $x > 0$ and takes value 0 if $x \leq 0$.
4. The adversary decides whether to corrupt the signal and sends the (possibly corrupted) signal σ_t to the learner.
5. The learner observes the signal σ_t, and suffers an unobservable loss $\ell_t := \ell(q_t, f(x_t))$.

In step 4, it is assumed that throughout the course of T rounds, at most C rounds are corrupted. In uncorrupted rounds, the signal $\sigma_t = \sigma(q_t - f(x_t))$. In corrupted rounds, the signal $\sigma_t = 1 - \sigma(q_t - f(x_t))$. In other words, for uncorrupted rounds, a signal of $\sigma_t = 0$ indicates a guess too low and a signal of $\sigma_t = 1$ indicates a guess too high, and the adversary flips this signal in corrupted rounds. Critically, the value C is *unknown* to the learner. The algorithms presented in this work are agnostic to the quantity C, and the performance degrades gracefully as C increases.

In step 5, the learner suffers a loss $\ell_t = \ell(q_t, f(x_t))$. Note that the functional form of the loss function is known to the learner but the loss value is never revealed to the learner. The goal of the learner will be to incur as little cumulative loss as possible, where the cumulative loss is defined as

$$\sum_{t=1}^{T} \ell_t = \sum_{t=1}^{T} \ell(q_t, f(x_t)).$$

Two loss functions are considered in this work, the absolute loss

$$\ell(q, f(x)) = |f(x) - q|$$

and the pricing loss

$$\ell(q, f(x)) = f(x) - q \cdot \mathbb{1}\{q \leq f(x)\}.$$

Separate algorithms are presented for each loss function in the following sections.

Remark 2 *The adversary model in this work is much stronger compared with previous work on corrupion-robust linear contextual search (Krishnamurthy et al. (2022); Leme et al. (2022)) (apart from the obvious differences in the assumptions on f, i.e., this work assumes f to be Lipschitz whereas previous works assumed f to be linear). In previous works, the adversary has to commit to a 'corruption level' z_t* before *seeing the guess q_t submitted by the learner. In uncorrupted rounds $z_t = 0$, and in corrupted rounds z_t will be a bounded quantity (e.g. $z_t \in [-1, 1]$). Then the learner observes the signal $\sigma_t = \sigma(q_t - (f(x_t) + z_t))$. In other words, the binary signal is generated according to the corrupted function value $f(x_t) + z_t$ instead of the true function value $f(x_t)$.*

In this work, the adversary's power is significantly increased. The adversary has the power to directly corrupt the binary signal and does so after *the learner has submitted her guess. However, with this being said, Leme et al. (2022) also considered a stronger notion of regret (under the weak adversary), where C is defined as the cumulative corruption level instead of the number of rounds in which corruption occurred.*

Notations. For a one-dimensional interval I, the notation $\texttt{len}(I)$ denotes the length of I. For a hypercube $I \subset \mathbb{R}^d$, the notation $\texttt{len}(I)$ denotes the length of any side of I. In some cases, to highlight the dependence on T and C, the notation $O_d(\cdot)$ is used to hide dependence on d.

3. Algorithm for Absolute Loss

This section gives a corruption-robust algorithm for the absolute loss, defined by

$$\ell(q_t, f(x_t)) = |f(x_t) - q_t|.$$

I will first design an algorithm for $d = 1$ in this section (Algorithm 1), then extend the algorithm to $d \geq 2$.

Algorithm 1: Learning with corrupted binary signals under absolute loss for $d = 1$

```
Learner maintains a partition of intervals $I_j$ of the input space throughout the learning process;
For each interval $I_j$ in the partition, learner stores a checking interval $S_j$, and maintains an
  associated range $Y_j$;
The partition is initialized as 8 intervals $I_j$ with length $1/8$ each, and each $I_j$ has associated
  range $Y_j$ and checking interval $S_j$ set to $[0, L]$;
for $t = 1, 2, ..., T$ do
    Learner receives context $x_t$;
    Learner finds interval $I_j$ such that $x_t \in I_j$;
    Let $Y_j$ be the associated range of $I_j$;
    if Exists an endpoint of $S_j$ not yet queried then
        Learner selects an unqueried endpoint of $S_j$ as guess ;
        if Learner guessed $\min(S_j)$ and $\sigma_t = 1$, or guessed $\max(S_j)$ and $\sigma_t = 0$ then
            Mark $I_j$ as dubious;
        end
        if Both endpoints of checking interval $S_j$ have been queried then
            if $I_j$ marked dubious then
                Set range $Y_j := [0, L]$;
            end
            else
                Set range $Y_j = [\min(S_j) - L \cdot \texttt{len}(I_j)), \max(S_j) + L \cdot \texttt{len}(I_j))] \cap [0, L]$;
            end
        end
    end
    else
        $Y_j := \texttt{MidpointQuery}(I_j, Y_j)$;
        if $\texttt{len}(Y_j) < \max(4L \cdot \texttt{len}(I_j), 4L/T)$ then
            Bisect $I_j$ into $I_{j1}, I_{j2}$, set $S_{j1} = Y_j, S_{j2} = Y_j$;
        end
    end
end
```

Algorithm 2: Midpoint query procedure: $\texttt{MidpointQuery}(I_j, Y_j)$

```
Input: Interval $I_j$, associated range $Y_j$;
Learner queries $q$, the midpoint of $Y_j$;
if $\sigma_t$ = 1 then
    Set $Y_j' := [0, q_t + L \cdot \texttt{len}(I_j)] \cap Y_j$;
end
else
    Set $Y_j' := [q_t - L \cdot \texttt{len}(I_j), 1] \cap Y_j$;
end
Return $Y_j'$;
```

3.1. Algorithm for $C = 0$

It will be helpful to first give a brief description of an algorithm that appeared in Mao et al. (2018). This algorithm works for $d = 1$ and when there are no adversarial corruptions. At each point in time, the learner maintains a partition of the input space into intervals. For each interval I_j in the partition, the learner also maintains an associated range Y_j, which serves as an estimate of the image of I_j. When a context appears in I_j, the learner selects the midpoint of Y_j as the query point, and the associated range Y_j shrinks and gets refined over time. This corresponds roughly to the subprocedure summarized in Algorithm 2, which is termed the midpoint query procedure `MidpointQuery`. As shown in Mao et al. (2018), the `MidpointQuery` procedure preserves the relation $f(I_j) \subset Y_j$ (when there are no corruptions). Moreover, whenever $\texttt{len}(Y_j) \geq 4L \cdot \texttt{len}(I_j)$, the associated range Y_j shrinks by at least a constant factor after a single call to `MidpointQuery` (for more details see Appendix A). When Y_j reaches a point where significant refinement is no longer possible, the learner zooms in on I_j by bisecting it.

3.2. Corruption-Robust Search with Agnostic Checks

The main algorithm for absolute loss with $d = 1$ is summarized in Algorithm 1. This algorithm is corruption-robust and agnostic to the corruption level C. Below I give the key ideas in this algorithm.

When there are adversarial corruptions, it will generally be impossible to tell for certain whether the associated range Y_j contains the image of the interval I_j. That is, the learner will not know with complete certainty whether $f(I_j) \subset Y_j$ holds. The analysis divides intervals into three types: *safe intervals*, *correcting intervals*, and *corrupted intervals*.

Definition 3 (Corrupted, Correcting, and Safe Intervals) *Consider some interval I_j that occurs during the run of the algorithm.*

- *I_j is a* corrupted interval *if, there exists some t where $x_t \in I_j$ and the signal σ_t is corrupted.*
- *I_j is a* correcting interval *if its parent interval is a* corrupted interval *and for every round t where $x_t \in I_j$, the signal σ_t is uncorrupted.*
- *I_j is a* safe interval *if its parent interval is safe or correcting (or itself is a root interval), and for every round t where $x_t \in I_j$, the signal σ_t is uncorrupted.*

I introduce the *agnostic checking* measure to combat adversarial corruptions. Consider a new interval I_j that has been formed by bisecting its parent interval, and that a context x_t appears in this interval I_j. The learner will first query the two endpoints of Y_j and test whether $f(I_j) \subset Y_j$ holds (according to the possibly corrupted signals). The interval passes the agnostic check if according to the (possibly corrupted) signals $f(I_j) \subset Y_j$. If the interval does not pass the agnostic check, it is marked as *dubious*, and the learner resets Y_j to $[0, L]$ and effectively searches from scratch for the associated range.

The main theorem is stated below.

Theorem 4 *Algorithm 1 incurs $L \cdot O(C \cdot \log T)$ total absolute loss for $d = 1$.*

Proof [Proof Sketch] The proof consists of several steps.

Step 1. An interval not marked *dubious* bisects in $O(1)$ rounds. This is because the associated range is shrinking by at least a constant after each query and after $O(1)$ rounds the associated range will have shrunk enough and meet the criteria for bisecting the interval. By a similar logic, any interval marked *dubious* bisects in $O(\log T)$ rounds.

Step 2. Consider any *correcting* interval. After the agnostic checking steps, the associated range must contain the true range of the function on the interval. The associated range will then be accurate when the interval is bisected. Then consider any *safe* interval. Its parent must be safe or correcting, and by induction, a *safe* interval will not be marked *dubious*.

Step 3. There can be at most $O(C)$ *corrupted* intervals. Since any *correcting* interval has a *corrupted* interval as a parent, there can be at most $O(C)$ correcting intervals. A total of $O(\log T)$ queries can occur on corrupted and correcting intervals, contributing loss $L \cdot O(C \log T)$ (actually one can show correcting intervals contribute loss at most $L \cdot O(C)$). For *safe* intervals, a loss of magnitude $O(2^{-h})$ can occur at most $O(2^h)$ times (since there are $O(2^h)$ intervals at depth h), thus the total loss from safe intervals can be bounded by $L \cdot O(\log T)$.

Putting everything together gives the total $L \cdot O(C \cdot \log T)$ regret bound. ■

Remark 5 *Note that in the algorithm, I_j gets bisected when $\texttt{len}(Y_j)$ reaches below $\max(4L \cdot \texttt{len}(I_j), 4L/T)$. The latter quantity $4L/T$ ensures that any interval must be bisected in $O(\log T)$ rounds. This is also enough to ensure that all* safe *intervals with depth $> \log T$ contribute total loss at most $O(1)$.*

Remark 6 *The regret is tight up to $\log T$ factors. To see this, consider the first C rounds, where the adversary corrupts the signal with probability $1/2$ each round. The learner essentially receives no information during this period, and each round incurs regret $\Omega(L)$.*

Remark 7 *An interesting aspect of this algorithm is that the learner remains agnostic to the type of each interval. In other words, during the run of the algorithm, the learner will in general not know for certain which type an interval belongs to. The analysis makes use of the three types of the interval, not the algorithm itself.*

3.3. Extending to $d > 1$

Algorithm 1 can be extended in a straightforward way to accommodate the case $d > 1$. The full algorithm and analysis are given in Appendix A.2. The only difference is that the learner maintains d-dimensional hypercubes instead of intervals. The main result is as follows.

Theorem 8 *Algorithm 4 incurs $L \cdot O_d(C \log T + T^{(d-1)/d})$ total absolute loss for $d \geq 2$.*

Remark 9 *In Mao et al. (2018), it was shown the optimal regret when $C = 0$ is $\Omega(T^{(d-1)/(d)})$. Hence, the dependence on C and T are both optimal in the above theorem (up to $\log T$ factors).*

4. Algorithm for Pricing Loss

Algorithm 3: Learning with corrupted binary signal under pricing loss

```
Set parameter $\eta_0 = T^{-1/(d+1)}$, agnostic check schedule $\tau_0$;
Learner uniformly discretizes input space into hypercubes with lengths $\Theta(\eta_0)$ each;
For each hypercube $I_j$, learner maintains an associated range $Y_j$ (initialized as $[0, L]$), query
 count $c_j$ (initialized as 0);
for $t = 1, 2, ..., T$ do
    Learner receives context $x_t$;
    Learner finds hypercube $I_j$ such that $x_t \in I_j$;
    Let $Y_j$ be the associated range of $I_j$;
    if len$(Y_j) < 10L \cdot \eta_0$ then
        $c_j := c_j + 1$;
        if $c_j > \tau_0$ then
            Query $\max(Y_j)$;
            Set $c_j := 0$;
        end
        else
            Query $\min(Y_j)$;
        end
        if Learner is surprised (see Definition 12) then
            Set $Y_j := [0, L]$;
            Set $c_j := 0$;
        end
    end
    else
        $Y_j :=$ MidpointQuery$(I_j, Y_j)$;
    end
end
```

This section discusses the new ideas needed to design corruption-robust algorithm for the pricing loss. The description shall be given in the context of dynamic pricing, and the learner shall be referred to as the seller in this section. The main algorithm is summarized in Algorithm 3. Note that for the pricing loss, the case with $d = 1$ and $d > 1$ are treated together.

Extending Algorithm 1 for the absolute loss to pricing loss is not straightforward, since agnostic checks will overprice and the seller necessarily incurs a large loss whenever she overprices. The learner did not have this problem with absolute loss, since the absolute loss is continuous.

4.1. Algorithm for $C = 0$

I first give the description of an algorithm for $C = 0$. The algorithm is adapted from Mao et al. (2018) but uses uniform discretization. The input space is discretized into hypercubes with length $\eta_0 = T^{-1/(d+1)}$. When a context appears in some hypercube I_j, one of two things can happen:

1. If the associated range is larger than $10L \cdot \eta_0$, the learner performs `MidpointQuery` and updates the associated range. There can be at most $O(\eta_0^d \log T)$ rounds of this type, since for each hypercube the associated range will shrink below $10L \cdot \eta_0$ after $O(\log T)$ queries.

2. Otherwise, the associated range is less than $10L \cdot \eta_0$, and the learner can directly set the lower end of the interval as the price. The total loss from these rounds can be bounded as $10L \cdot T\eta_0$.

The total loss is then $L \cdot O(\eta_0^d \log T) + L \cdot O(T\eta_0) = L \cdot O(T^{d/(d+1)} \log T)$.

4.2. Corruption-Robust Pricing with Scheduled Agnostic Checks

The seller could potentially run into issues when there are adversarial corruptions. The adversary can manipulate the seller into underpricing by a large margin by only corrupting a small number of signals. Consider the following example. The buyer is willing to pay 0.5 for an item with context x. The adversary can manipulate the signals in the `MidpointQuery` procedure so that the seller's associated range for x is $[0, \eta]$, where $\eta < 10L \cdot \eta_0$. Hence, the associated range does not contain and is well below the optimal price 0.5 for this item. The seller then posts a price of 0 for item x. Even though the buyer purchases the item at a price of 0, the seller is losing 0.5 revenue per round, and this happens with the adversary corrupting only $O(\log T)$ rounds.

To combat adversarial corruptions, the learner performs agnostic check queries. These queries differ from agnostic checks for the absolute loss in the following two aspects. First, whereas for the absolute loss, the learner performs agnostic check queries on both ends of the associated range Y_j, for the pricing loss the seller only performs agnostic checks on the upper end of Y_j. This ensures the seller is not underpricing the product by a large margin. Second, the seller only performs agnostic check queries when the associated range of the hypercube is sufficiently small (below $L \cdot O(\eta_0)$). The seller expects the buyer not the purchase the item during agnostic check queries.

The learner will set a schedule for performing agnostic checks. At a high level, performing agnostic checks too often incurs large regret from overpricing, while performing agnostic checks too few may not detect corruptions effectively. The schedule for agnostic checks serves as a mean to balance the two. This schedule informs the learner how often to perform agnostic check so as not to incur large regret when overpricing, and at the same time control the loss incurred from corrupted signals. Specifically, the learner keeps track of how many rounds the context vectors arrived in each hypercube. The learner then performs an agnostic check every τ_0 rounds. Here, τ_0 will be a paramter to be chosen later.

Some notions are introduced.

Definition 10 (Pricing-ready hypercubes) *For hypercube I_j, if the length of the associated range is below $10L \cdot \eta_0$, then the hypercube is* pricing-ready.

Definition 11 (Pricing round, checking round, searching round) *If a hypercube is* pricing-ready*, then setting the price as the lower end of the associated range is termed* pricing round*, and setting the price as the upper end of the associated range is termed* checking round*. If a hypercube is not* pricing-ready*, then performing a* `MidpointQuery` *is termed* searching round*.*

Definition 12 (Surprises) *The seller is said to become* surprised *when the signal she receives is inconsistent with her current knowledge. Specifically, the seller becomes surprised when either: she performs a checking round but observes an underprice signal ($\sigma_t = 0$); or she performs a pricing round but observes an overprice signal ($\sigma_t = 1$).*

Definition 13 (Runs) *Fix some hypercube I_j, the set of queries that occurs on I_j before the learner becomes* surprised *(or before the algorithm terminates) called a* run*.*

When a run ends, the learner resets the associated range and a new run begins. Note there can be multiple runs on the same hypercube.

The algorithm gives the following cumulative loss bound. A detailed analysis can be found in Appendix B.

Theorem 14 *Using Algorithm 3, the learner incurs a total pricing loss*

$$L \cdot O(C \log T + T^{d/(d+1)} \log T + C\tau_0 + T/\tau_0)$$

for any unknown corruption budget C. Specifically, setting $\tau_0 = T^{1/(d+1)}$, the learner incurs total pricing loss

$$L \cdot \widetilde{O}(T^{d/(d+1)} + C \cdot T^{1/(d+1)}).$$

Proof [Proof Sketch] The proof consists of several steps.

Step 1. The seller can only become surprised when at least one corruption occurs on the current 'run' of some hypercube. Thus the seller can become surprised at most C times.

Step 2. For searching rounds, the total loss can be bounded as $L \cdot O((C + \eta_0^{-d}) \log T)$. There can be at most $O(T/\tau_0)$ checking rounds, contributing loss $L \cdot O(T/\tau_0)$.

Step 3. For pricing rounds, the total loss can be bounded as $L \cdot O(C\tau_0 + T\eta_0)$. At a very high level, if the pricing interval is accurate (meaning the lower endpoint is just below the true price by a margin of $L \cdot O(\eta_0)$), then these rounds contribute loss at most $L \cdot O(T\eta_0)$. Otherwise, the seller can detect if the pricing interval is inaccurate by performing agnostic checks, and the loss from these rounds can be bound by $L \cdot O(C\tau_0)$.

Putting everything together completes the proof. ■

If the corruption budget C is known, the learner can set τ_0 to balance the terms and achieve a sharper regret bound.

Corollary 15 *Assume the corruption budget C is known to the learner. Using Algorithm 3 with $\tau_0 = \sqrt{T/C}$, the learner incurs total loss*

$$L \cdot \widetilde{O}(T^{d/(d+1)} + \sqrt{TC}).$$

4.3. Lower Bounds

This subsection shows two lower bounds for either known or unknown C, showing the upper bounds in the previous subsection are essentially tight. Define an environment as the tuple $(f, C, \mathcal{S})$, representing the function, corruption budget, and corruption strategy of the adversary respectively. In the following assume $L = 1$, the scaling to other L is straightforward.

Theorem 16 *Let A be any algorithm to which the corruption budget C is unknown. Suppose A achieves a cumulative pricing loss $R(T)$ when $C = 0$, and that $R(T) = o(T)$. Then, there exists some corruption strategy with $C = 2R(T)$, such that the algorithm suffers $\Omega(T)$ regret.*

Proof Let us consider two environments. In the first environment, the adversary chooses $f(x) \equiv 0.5$ and never corrupts the signal. In the second environment, the adversary chooses $f(x) \equiv 1$ and corrupts the signal whenever the query is above 0.5.

Now, since the algorithm achieves regret $R(T)$ when $C = 0$, then when facing the first environment, the seller can only query values above 0.5 for at most $2R(T)$ times. Then, by choosing $C = 2R(T)$, the adversary can make the two environments indistinguishable from the seller. Hence the seller necessarily incurs $\Omega(T)$ regret in the second environment. ■

Remark 17 *The above lower bound shows one cannot hope to achieve a regret bound of the form $O(T^{(d/(d+1))}) + C \cdot o(T^{1/(d+1)})$ when C is unknown. First note any algorithm must achieve $\Omega(T^{d/(d+1)})$ in the worst case when $C = 0$ by the lower bound in Mao et al. (2018). Then, suppose the algorithm could achieve $O(T^{d/(d+1)})$ when $C = 0$, the algorithm must achieve $\Omega(T)$ regret for some $C = \Theta(T^{d/(d+1)})$.*

Next, a lower bound is given when the corruption budget C is known. The lower bound will be against a randomized adversary, who draws an environment from some probability distribution $\mathcal{D}$, and the corruption budget is defined as the expected value of C under distribution $\mathcal{D}$. Note that the upper bounds (Theorem 14, Corollary 15) also hold for randomized adversaries.

Theorem 18 *There exists some $\mathcal{D}$ where any algorithm incurs expected regret $\Omega(\sqrt{CT})$, even with complete knowledge of $\mathcal{D}$.*

Proof The adversary uses two environments. In the first environment, the function $f(x) \equiv 0.5$ and the corruption budget is 0. In the second environment, the function $f(x) \equiv 1$, the corruption budget is C_0, and the adversary corrupts any query above 0.5 until the budget is depleted. The adversary chooses the first environment with probability $1 - p$ and the second environment with probability p.

If the learner's algorithm queried more than C_0 rounds above 0.5, then the learner incurs regret $0.5C_0$ in the first environment, giving expected regret $\Omega((1-p)C_0)$. If the algorithm did not query more than C_0 rounds above 0.5, then the algorithm incurs regret $0.5(T - C_0)$ in the second environment, giving expected regret $\Omega(p(T - C_0))$. Choosing $p = \sqrt{C/T}, C_0 = \sqrt{CT}$ completes the proof. ■

5. Conclusion and Future Work

In this work, I design corruption-robust algorithms for the Lipschitz contextual search problem. I present the *agnostic checking* technique and demonstrate its effectiveness in designing corruption-robust algorithms. An open problem is closing the gap between upper bounds and lower bounds, in particular for the absolute loss when $d = 1$. Specifically, can one actually achieve $O(C + \log T)$ regret in this setting? Another interesting future direction is to relax the Lipschitzness assumption. For example, this work assumes the learner knows the Lipschitz constant L. Can the learner design efficient no-regret algorithms without knowledge of L?

Acknowledgments

The author would like to thank Jialu Li for her help in preparation of this manuscript. The anonymous reviewers at ALT gave helpful suggestions on improving the presentation of this paper.

References

Kareem Amin, Afshin Rostamizadeh, and Umar Syed. Learning prices for repeated auctions with strategic buyers. *Advances in Neural Information Processing Systems*, 26, 2013.

Kareem Amin, Afshin Rostamizadeh, and Umar Syed. Repeated contextual auctions with strategic buyers. *Advances in Neural Information Processing Systems*, 27, 2014.

Ilija Bogunovic, Arpan Losalka, Andreas Krause, and Jonathan Scarlett. Stochastic linear bandits robust to adversarial attacks. In *International Conference on Artificial Intelligence and Statistics*, pages 991–999. PMLR, 2021.

Nicolò Cesa-Bianchi, Pierre Gaillard, Claudio Gentile, and Sébastien Gerchinovitz. Algorithmic chaining and the role of partial feedback in online nonparametric learning. In *Conference on Learning Theory*, pages 465–481. PMLR, 2017.

Yifang Chen, Simon Du, and Kevin Jamieson. Improved corruption robust algorithms for episodic reinforcement learning. In *International Conference on Machine Learning*, pages 1561–1570. PMLR, 2021.

Arnoud V Den Boer. Dynamic pricing and learning: historical origins, current research, and new directions. *Surveys in operations research and management science*, 20(1):1–18, 2015.

Alexey Drutsa. Optimal non-parametric learning in repeated contextual auctions with strategic buyer. In *International Conference on Machine Learning*, pages 2668–2677. PMLR, 2020.

Evrard Garcelon, Baptiste Roziere, Laurent Meunier, Jean Tarbouriech, Olivier Teytaud, Alessandro Lazaric, and Matteo Pirotta. Adversarial attacks on linear contextual bandits. *Advances in Neural Information Processing Systems*, 33:14362–14373, 2020.

Anupam Gupta, Tomer Koren, and Kunal Talwar. Better algorithms for stochastic bandits with adversarial corruptions. In *Conference on Learning Theory*, pages 1562–1578. PMLR, 2019.

Jiafan He, Dongruo Zhou, Tong Zhang, and Quanquan Gu. Nearly optimal algorithms for linear contextual bandits with adversarial corruptions. *Advances in Neural Information Processing Systems*, 35:34614–34625, 2022.

Kwang-Sung Jun, Lihong Li, Yuzhe Ma, and Jerry Zhu. Adversarial attacks on stochastic bandits. *Advances in Neural Information Processing Systems*, 31, 2018.

Robert Kleinberg and Tom Leighton. The value of knowing a demand curve: Bounds on regret for online posted-price auctions. In *44th Annual IEEE Symposium on Foundations of Computer Science, 2003. Proceedings.*, pages 594–605. IEEE, 2003.

Robert Kleinberg and Aleksandrs Slivkins. Sharp dichotomies for regret minimization in metric spaces. In *Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms*, pages 827–846. SIAM, 2010.

Robert Kleinberg, Aleksandrs Slivkins, and Eli Upfal. Multi-armed bandits in metric spaces. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 681–690, 2008.

Akshay Krishnamurthy, Thodoris Lykouris, Chara Podimata, and Robert Schapire. Contextual search in the presence of adversarial corruptions. *Operations Research*, 2022.

Renato Paes Leme and Jon Schneider. Contextual search via intrinsic volumes. *SIAM Journal on Computing*, 51(4):1096–1125, 2022.

Renato Paes Leme, Balasubramanian Sivan, Yifeng Teng, and Pratik Worah. Learning to price against a moving target. In *International Conference on Machine Learning*, pages 6223–6232. PMLR, 2021.

Renato Paes Leme, Chara Podimata, and Jon Schneider. Corruption-robust contextual search through density updates. In *Conference on Learning Theory*, pages 3504–3505. PMLR, 2022.

Allen Liu, Renato Paes Leme, and Jon Schneider. Optimal contextual pricing and extensions. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1059–1078. SIAM, 2021.

Thodoris Lykouris, Vahab Mirrokni, and Renato Paes Leme. Stochastic bandits robust to adversarial corruptions. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 114–122, 2018.

Thodoris Lykouris, Max Simchowitz, Alex Slivkins, and Wen Sun. Corruption-robust exploration in episodic reinforcement learning. In *Conference on Learning Theory*, pages 3242–3245. PMLR, 2021.

Jieming Mao, Renato Leme, and Jon Schneider. Contextual pricing for lipschitz buyers. *Advances in Neural Information Processing Systems*, 31, 2018.

Aleksandrs Slivkins. Contextual bandits with similarity information. In *Proceedings of the 24th annual Conference On Learning Theory*, pages 679–702. JMLR Workshop and Conference Proceedings, 2011.

Xuezhou Zhang, Yiding Chen, Xiaojin Zhu, and Wen Sun. Corruption-robust offline reinforcement learning. In *International Conference on Artificial Intelligence and Statistics*, pages 5757–5773. PMLR, 2022.

Shiliang Zuo. Near optimal adversarial attack on UCB bandits. In *The Second Workshop on New Frontiers in Adversarial Machine Learning*, 2023. URL https://openreview.net/forum?id=Slt9waUI6F.

Appendix A. Proof Details for Absolute Loss

A.1. Absolute Loss with $d = 1$

An interval has depth r if it was bisected from an interval at depth $r-1$. Initial root intervals have depth 0. Hence, an interval I_j at depth r has length $\texttt{len}(I_j) = O(2^{-r})$.

The below two lemmas discuss properties of `MidpointQuery`, and are adapted from Mao et al. (2018). The below lemma shows that each call to `MidpointQuery` preserves the property $f(I_j) \subset Y_j$, assuming the signal is uncorrupted.

Lemma 19 *Consider a call to the* $\texttt{MidpointQuery}(I_j, Y_j)$ *procedure. Let* Y_j *be the associated range before the query, and let* $Y_j' = \texttt{MidpointQuery}(I_j, Y_j)$ *be the range after the query. If the signal was uncorrupted and* $f(I_j) \subset Y_j$*, then* $f(I_j) \subset Y_j'$*.*

Proof Let q_t be the query point (i.e. midpoint of Y_j). Assume the signal $\sigma_t = 0$, so that $f(x_t) \geq q_t$. The case where $\sigma_t = 1$ will be similar.

Since f is L-Lipschitz, it must be that

$$f(I_j) \geq f(x_t) - L \cdot \texttt{len}(I_j) \geq q_t - L \cdot \texttt{len}(I_j),$$

hence the update

$$Y_j' = Y_j \cap [q_t - L \cdot \texttt{len}(I_j), L]$$

still guarantees that $f(I_j) \subset Y_j'$. ■

The next lemma shows the length of Y_j shrinks by a constant factor after each call to `MidpointQuery`, regardless of whether the signal was corrupted or not.

Lemma 20 *Consider a call to the* $\texttt{MidpointQuery}(I_j, Y_j)$ *procedure. Let* Y_j *be the associated range before the query, and let* $Y_j' = \texttt{MidpointQuery}(I_j, Y_j)$ *be the range after the query. Suppose* $\texttt{len}(Y_j) \geq 4L \cdot \texttt{len}(I_j)$*, then* $\frac{1}{2}\texttt{len}(Y_j) \leq \texttt{len}(Y_j') \leq \frac{3}{4}\texttt{len}(Y_j)$ *regardless of whether the signal is corrupted or not.*

Proof Let q_t be the query point (i.e. midpoint of Y_j). Assume the signal $\sigma_t = 0$, the case with $\sigma_t = 1$ will be similar.

Before the call to $\texttt{MidpointQuery}(I_j, Y_j)$, the learner has $\texttt{len}(Y_j) \geq 4L \cdot \texttt{len}(I_j)$. This implies

$$q_t - \min(Y_j) = \frac{\texttt{len}(Y_j)}{2} \geq 2L \cdot \texttt{len}(I_j)$$

so that

$$q_t - L \cdot \texttt{len}(I_j) > \min(Y_j).$$

The update can be written as

$$\begin{aligned} Y_j' &= [q_t - L \cdot \texttt{len}(I_j), L] \cap Y_j \\ &= [q_t - L \cdot \texttt{len}(I_j), q_t] \cup [q_t, \max(Y_j)] \end{aligned}$$

The lemma then follows from $\texttt{len}([q_t, \max(Y_j)]) = \frac{1}{2}\texttt{len}(Y_j)$, $\texttt{len}([q_t - L \cdot \texttt{len}(I_j), q_t]) = L \cdot \texttt{len}(I_j) \leq \frac{1}{4}\texttt{len}(Y_j)$. ■

I now give several lemmas that discuss various properties of the three types of intervals (*safe*, *correcting*, or *corrupted*).

Lemma 21 *If an interval is marked dubious, then the interval is bisected after $O(\log T)$ queries. If an interval is not marked dubious, then the interval is bisected after $O(1)$ queries.*

Proof If the interval is marked dubious, then the range of the interval is reset to $[0, L]$. By Lemma 20, the range Y_j shrinks by at least a factor of $\frac{3}{4}$ each query, hence after $O(\log T)$ queries, the learner has $\texttt{len}(Y_j) < \max(4L \cdot \texttt{len}(I_j), 4L/T)$.

If the interval were not marked dubious, then the range Y_j is updated as

$$Y_j = [\min(S_j) - L \cdot \texttt{len}(I_j), \max(S_j) + L \cdot \texttt{len}(I_j)] \cap [0, L].$$

From the above, it can be seen that

$$\texttt{len}(Y_j) \leq \texttt{len}(S_j) + 2L \cdot \texttt{len}(I_j).$$

By the splitting condition,

$$\texttt{len}(S_j) \leq \max(8L \cdot \texttt{len}(I_j), 4L/T).$$

If the interval I_j has length larger than $1/(2T)$, then $\texttt{len}(S_j) \leq 8L \cdot \texttt{len}(I_j)$, and after the update the learner has $\texttt{len}(Y_j) \leq 10L \cdot \texttt{len}(I_j)$. If the interval I_j has length no larger than $1/(2T)$, then $\texttt{len}(S_j) \leq 4L/T$, and after the update $\texttt{len}(Y_j) \leq 5L/T$. In either case, Invoking Lemma 20 again, the range shrinks by constant factor each query and it takes $O(1)$ rounds for the range to shrink below $\max(4L \cdot \texttt{len}(I_j), 4L/T)$. ■

Lemma 22 *Let I_j be a correcting interval. When I_j is bisected, it holds that $f(I_j) \subset Y_j$.*

Proof It is first shown that $f(I_j) \subset Y_j$ after the update following the agnostic check procedure. There are two cases to consider, whether I_j has been marked dubious or not after the agnostic check.

If I_j has been marked dubious, then the update resets Y_j to $[0, L]$, hence the learner has $f(I_j) \subset Y_j$ trivally.

If I_j has not been marked dubious, the update

$$Y_j = [\min(S_j) - L \cdot \texttt{len}(I_j), \max(S_j) + L \cdot \texttt{len}(I_j)]$$

takes place. Since I_j is a correcting interval, the signals are uncorrupted, and by Lipschitzness of f, the learner has

$$f(I_j) \geq \min(S_j) - L \cdot \texttt{len}(I_j)$$
$$f(I_j) \leq \max(S_j) + L \cdot \texttt{len}(I_j).$$

Hence the learner has $f(I_j) \subset Y_j$ after the update.

Putting these two cases together, after the update on the associated range following the agnostic checking steps, the learner has $f(I_j) \subset Y_j$. By inducting on Lemma 19, repeated calls to `MidpointQuery` guarantees $f(I_j) \subset Y_j$ when I_j is bisected. ■

Lemma 23 *Safe intervals are not marked dubious.*

Proof Induction is used to show the following for any safe interval I_j:

1. $f(I_j) \subset S_j$
2. $f(I_j) \subset Y_j$ when I_j is bisected

Let I_j be a safe interval. If I_j is a root interval, then trivially $f(I_j) \subset S_j$. Further by Lemma 19, $f(I_j) \subset Y_j$ when I_j is split.

If I_j has a correcting interval as its parent interval, then by Lemma 22, $f(I_j) \subset S_j$, and consequently by Lemma 19 $f(I_j) \subset Y_j$ when I_j is split.

If I_j has a safe interval as its parent interval, then a simple induction shows the desired result. ■

The following corollary follows directly from Lemma 21 and Lemma 23.

Corollary 24 *Safe intervals bisect in $O(1)$ rounds.*

Lemma 25 *Corrupted intervals contribute $L \cdot O(C \log T)$ loss.*

Proof There are at most C corrupted intervals, and each interval splits in $O(\log T)$ rounds by Lemma 21, with each round incurring $O(L)$ regret (trivially). Hence corrupted intervals contribute total $L \cdot O(C \log T)$ regret. ■

Lemma 26 *Correcting intervals contribute $L \cdot O(C)$ loss.*

Proof There are at most $2C$ correcting intervals. For each correcting interval I_j, querying the two endpoints of S_j incur $O(L)$ regret. Each call to `MidpointQuery`(I_j, Y_j) incur $O(\texttt{len}(Y_j))$ regret. By Lemma 20, $\texttt{len}(Y_j)$ shrinks by a factor of $\frac{3}{4}$, hence the loss is geometrically decreasing with each query, and the total loss incurred within I_j is $O(L)$. ■

The below lemma bounds the regret from safe intervals, and is adapted from Mao et al. (2018).

Lemma 27 *Safe intervals contribute $L \cdot O(\log T)$ loss.*

Proof Consider a safe interval I_j at depth $h < \log T$. The two queries on the endpoints incur $O(2^{-h})$ loss, since

$$\texttt{len}(S_j) < 8L \cdot \texttt{len}(I_j) = O(2^{-h}).$$

Each call to $\texttt{MidpointQuery}(I_j, Y_j)$ also incurs $L \cdot O(2^{-h})$ loss since $\texttt{len}(Y_j) = O(L \cdot \texttt{len}(I_j)) = L \cdot O(2^{-h})$. By Corollary 24, safe intervals splits in $O(1)$ rounds, hence the loss incurred at interval I_j is $L \cdot O(2^{-h})$. Note that there can be at most $O(2^h)$ intervals at depth h, therefore a loss of magnitude $L \cdot O(2^{-h})$ can be charged at most $O(2^h)$ times. In a total of T rounds, the loss can be at most $L \cdot O(\log T)$.

■

Theorem 4 *Algorithm 1 incurs $L \cdot O(C \cdot \log T)$ total absolute loss for $d = 1$.*

Proof Summing over the loss contributed by corrupted intervals (Lemma 25), correcting intervals (Lemma 26), and safe intervals (Lemma 27) completes the proof. ■

A.2. Absolute Loss with $d > 1$

The corruption-robust algorithm for learning a Lipschitz function from $\mathbb{R}^d \to \mathbb{R}(d > 1)$ under absolute loss is summarized in Algorithm 4. The proposed algorithm initializes the input space as a single hypercube with S_j initialized to $[0, 8L]$ (instead of 8^d hypercubes with S_j initialized as $[0, L]$). The two initializations are essentially equivalent but the former slightly simplifies analysis somewhat: there is now only a single root hypercube.

Theorem 8 *Algorithm 4 incurs $L \cdot O_d(C \log T + T^{(d-1)/d})$ total absolute loss for $d \geq 2$.*

The analysis will be similar to that of Algorithm 1 and Theorem 4.
Proof There are $O(C)$ corrupted intervals, contributing $L \cdot O(C \log T)$ loss. There are $O(C \cdot 2^d)$ correcting intervals, contributing $L \cdot O(C \cdot 2^d)$ loss. For safe intervals, there are at most $O(2^{hd})$ intervals at depth h. Throughout T rounds, a loss with magnitude $L \cdot O(2^{-h})$ can be charged at most $O(2^{hd})$ times, and the total loss of all safe intervals is then at most $L \cdot O(T^{(d-1)/d})$, since the loss coming from safe intervals can be upper bounded by

$$L \cdot \sum_{h=0}^{\frac{\log T}{d}} O(2^{(d-1)h}) = L \cdot O(T^{(d-1)/d}).$$

Putting the loss of all three types of intervals, the total loss is

$$L \cdot O(C \log T + C \cdot 2^d + T^{(d-1)/d}) = L \cdot O_d(C \log T + T^{(d-1)/d}),$$

which completes the proof. ■

Algorithm 4: Learning with corrupted binary signal under absolute loss for $d > 1$

```
Learner maintains a partition of hypercubes $I_j$ of the input space $[0,1]^d$ throughout learning process;
For each hypercube $I_j$, learner stores a checking interval $S_j$ and maintains an associated range $Y_j$;
The partition is initialized as a single hypercube with checking interval $S_j$ set to $[0, 8L]$;
for $t = 1, 2, ..., T$ do
    Learner receives context $x_t$;
    Learner finds hypercube $I_j$ such that $x_t \in I_j$;
    Let $Y_j$ be the feasible interval of $I_j$;
    if Exists an endpoint of $S_j$ not queried then
        Query an unqueried endpoint of $S_j$;
        if Queried $\min(S_j)$ and $\sigma_t = 1$, or queried $\max(S_j)$ and $\sigma_t = 0$ then
            Mark $I_j$ as dubious;
        end
        if Both endpoints of $S_j$ have been queried then
            if $I_j$ marked dubious then
                Set range $Y_j := [0, 8L]$;
            end
            else
                Set range $Y_j = [\min(S_j) - L \cdot \texttt{len}(I_j)), \max(S_j) + L \cdot \texttt{len}(I_j))] \cap [0,1]$;
            end
        end
    end
    else
        $Y_j := \texttt{MidpointQuery}(I_j, Y_j)$;
        if $\texttt{len}(Y_j) < \max(4L \cdot \texttt{len}(I_j), 4L/T)$ then
            Bisect each side of $I_j$ to form $2^d$ new hypercubes each with length $\texttt{len}(I_j)/2$;
            For each new hypercube $I_{ji}$ $(1 \le i \le 2^d)$ set $S_{ji} = Y_j$;
        end
    end
end
```

Appendix B. Proof Details for Pricing Loss

Recall that on a pricing-ready hypercube I_j, the learner is said to become *surprised* when the signal is inconsistent with Y_j. In other words, the learner becomes surprised when she queried $\min(Y_j)$ and receives an overprice signal ($\sigma_t = 1$), or queried $\max(Y_j)$ and receives an underprice signal ($\sigma_t = 0$).

Lemma 28 *The learner becomes surprised at most C times.*

Proof The learner can become surprised for the following two reasons: the signal itself is corrupted, or the signal is uncorrupted, but the associated range Y_j is inaccurate ($f(I_j) \not\subset Y_j$). In the latter case, a corruption must have occurred in the search queries before the hypercube became pricing ready. Since there are at most C corruptions, the learner can become surprised at most C times. ■

Lemma 29 *Consider a run of length ζ on some hypercube I_j. Further, assume a total of ξ signals were corrupted during this period. Then pricing rounds pick up a loss of $L \cdot O(\zeta\eta_0 + \xi\tau_0)$ during this run.*

Proof First note that once the hypercube becomes pricing-ready, the associated range Y_j does not change until this run terminates. If $f(I_j) \subset Y_j$, then each pricing round incurs loss $L \cdot O(\eta_0)$, hence a run with length ζ incurs loss $L \cdot O(\zeta\eta_0)$.

Otherwise, there are four cases.

1. $f(I_j) \geq \max(Y_j)$
2. $f(I_j)$ has some overlap with Y_j and $f(I_j) \geq \min(Y_j)$, $\max(Y_j) \in f(I_j)$
3. $f(I_j) \leq \min(Y_j)$
4. $f(I_j)$ has some overlap with Y_j and $f(I_j) \leq \max(Y_j)$, $\min(Y_j) \in f(I_j)$

For case 1, any uncorrupted signal in agnostic check queries makes the learner surprised. Hence assuming the adversary spent corruption budge ξ, then the run must terminate in $O(\xi\tau_0)$ rounds since the adversary must be corrupting every agnostic check query. The learner then trivially incurs loss $L \cdot O(\xi\tau_0)$.

For case 2, the loss collected is at most $L \cdot O(\eta_0)$ per pricing round, hence the total loss is $L \cdot O(\zeta\eta_0)$.

For case 3, any uncorrupted signal in pricing rounds makes the learner surprised. Hence assuming the adversary spent a corruption budget ξ, the run terminates in $O(\xi)$ rounds (since the adversary must be corrupting all pricing rounds), and the learner incurs regret $L \cdot O(\xi)$.

For case 4, the adversary does not corrupt agnostic check rounds, or the learner will become immediately surprised. If the learner does not become surprised during pricing rounds, this could be due to either: 1. the learner did not overprice, or 2. the learner overpriced but the adversary corrupted the signal. Consequently in pricing rounds, uncorrupted queries accumulate $L \cdot O(\zeta\eta_0)$ loss, and corrupted queries accumulate $L \cdot O(\xi)$ loss.

Putting all cases together completes the proof. ■

In the following, let $\mathcal{T}$ be a multi-set with elements denoting the length of runs of pricing-ready hypercubes. Let $\mathcal{C}$ be a multi-set with elements denoting the number of corruptions that occurred in runs of pricing-ready hypercubes.

Theorem 14 *Using Algorithm 3, the learner incurs a total pricing loss*

$$L \cdot O(C \log T + T^{d/(d+1)} \log T + C\tau_0 + T/\tau_0)$$

for any unknown corruption budget C. Specifically, setting $\tau_0 = T^{1/(d+1)}$, the learner incurs total pricing loss

$$L \cdot \widetilde{O}(T^{d/(d+1)} + C \cdot T^{1/(d+1)}).$$

Proof First, an upper bound is obtained on the total loss incurred from searching rounds (i.e., when $\texttt{len}(Y_j) > 10L \cdot \eta_0$). The learner searches from scratch when starting from initialization (Y_j is set to $[0, L]$ at initialization) or when she becomes surprised (Y_j is reset to $[0, L]$). This can happen at most $C + \eta_0^{-d}$ times, since the learner becomes surprised at most C times (Lemma 28) and there are η_0^{-d} hypercubes at initialization. Whenever the learner searches from scratch, it takes $O(\log T)$ queries before the interval becomes pricing-ready. Thus the total loss incurred from search queries is

$$L \cdot ((C + \eta_0^{-d}) \log T). \tag{1}$$

Next, the loss incurred when range becomes pricing-ready can be separated into two parts, loss from agnostic check queries and loss from pricing rounds. The total loss from agnostic check queries can be bounded by

$$L \cdot O(T/\tau_0), \tag{2}$$

since there can be $O(T/\tau_0)$ agnostic check queries, and each query contribute loss at most L.

From Lemma 29, the total loss from pricing rounds can be bounded by

$$L \cdot O\left(\sum_{\xi \in \mathcal{C}} \xi \cdot \tau_0 + \sum_{\zeta \in \mathcal{T}} \zeta \cdot \eta_0\right) \leq L \cdot O(C\tau_0 + T\eta_0). \tag{3}$$

Putting everything together, the total loss can be bounded by the sum of loss incurred during agnostic checking rounds, pricing rounds, and searching rounds:

$$L \cdot O\left(T/\tau_0 + C\tau_0 + T\eta_0 + \eta_0^{-d} \log T + C \log T\right).$$

Plugging the choice of parameter $\eta_0 = T^{-1/(d+1)}$ finishes the proof. ■