Contextual Pricing for Lipschitz Buyers

Jieming Mao
Princeton University

Renato Paes Leme
Google Research

Jon Schneider
Princeton University

Abstract

We investigate the problem of learning a Lipschitz function from binary feedback. In this problem, a learner is trying to learn a Lipschitz function $f : [0, 1]^d \to [0, 1]$ over the course of T rounds. On round t, an adversary provides the learner with an input x_t, the learner submits a guess y_t for $f(x_t)$, and learns whether $y_t > f(x_t)$ or $y_t \leq f(x_t)$. The learner's goal is to minimize their total loss $\sum_t \ell(f(x_t), y_t)$ (for some loss function ℓ). The problem is motivated by *contextual dynamic pricing*, where a firm must sell a stream of differentiated products to a collection of buyers with non-linear valuations for the items and observes only whether the item was sold or not at the posted price.

For the symmetric loss $\ell(f(x_t), y_t) = |f(x_t) - y_t|$, we provide an algorithm for this problem achieving total loss $O(\log T)$ when $d = 1$ and $O(T^{(d-1)/d})$ when $d > 1$, and show that both bounds are tight (up to a factor of $\sqrt{\log T}$). For the pricing loss function $\ell(f(x_t), y_t) = f(x_t) - y_t \mathbf{1}\{y_t \leq f(x_t)\}$ we show a regret bound of $O(T^{d/(d+1)})$ and show that this bound is tight. We present improved bounds in the special case of a population of linear buyers.

1 Introduction

A major problem in revenue management is designing pricing strategies for highly differentiated products. Besides the usual tension between exploration and exploitation (often call learning and earning in revenue management) the problem poses the following additional challenges: (i) the feedback in pricing problems is very limited: for each item the seller only learns whether the item was sold or not; (ii) the loss function is discontinuous and asymmetric: pricing slightly under the buyer's valuation causes a small loss while pricing slightly above causes the item not to be sold and therefore a large loss.

The study of learning in pricing settings was pioneered by Kleinberg and Leighton [15] who designed optimal pricing policies in a variety of settings when the products are undifferentiated. Motivated by applications to online commerce and internet advertisement, there has been a lot of interest in extending such results to contextual settings, where the seller is able to observe characteristics of each product, typically encoded by a high-dimensional feature vector $x_t \in \mathbb{R}^d$. The typical approach in those problems has been to assume that the valuation of the buyer is linear (Amin et al [2], Cohen et al [10], Lobel et al [20], Nazerzadeh and Javanmard [14], Javanmard [13] and Paes Leme and Schneider [19]) or that the demand function of a population of buyers is linear (Qiang and Bayati [21]).

Here we focus on the cases where the buyer's valuation is non-linear in the feature vectors, or where there are multiple buyers all with linear valuation functions. These cases can be cast as special cases of the semi-Lipschitz bandits model of Cesa-Bianchi et al [8]. Our goal is to exploit the special structure of the pricing problem and obtain improved bounds compared to those achieved for semi-Lipschitz bandits.

The model is as follows: our seller receives a new item for each of T rounds. The item at time t is described by a feature vector $x_t \in \mathbb{R}^d$. The seller is selling these items to a population of b buyers,

Preprint. Work in progress.

where buyer i is willing to pay up to $V_i(x_t)$ for item x_t (for some valuation function V_i unknown to the seller). Every round the seller gets to choose a price p_t for the current item. If $p_t \leq V_i(x_t)$ for some i, then some buyer purchases the item and the seller receives revenue p_t. Otherwise, no buyer purchases the item and the seller receives revenue 0. The goal of the seller is to maximize their revenue, and in particular minimize the difference between their revenue and the revenue of a seller who knows the V_i's ahead of time (their *regret*).

For the special case where there is a single buyer ($b = 1$) and his valuation is linear in x_t, the tight bound of $O(\mathsf{poly}(d) \log \log T)$ was recently given in [19]. In this paper, we consider the setting where the number of buyers b is very large (potentially infinite), and we want regret bounds independent of b. We show:

- If all the V_i are L-Lipschitz, then there is an algorithm for this contextual pricing problem that achieves regret $\Theta((LT)^{d/(d+1)})$, which is tight (Theorems 4, 7). This improves over the $O(T^{(d+1)/(d+2)})$ bound that we obtain by applying the algorithms for semi-Lipschitz bandits [8].
- If all the V_i are linear (i.e. of the form $V_i(x) = \langle v_i, x \rangle$ for some $v_i \in [0,1]^d$), then there is an algorithm for this contextual pricing problem that achieves regret $O_d(T^{(d-1)/d})$ (Corollary 11). We exploit the special structure by casting this pricing problem as learning the extreme points of a convex set from binary feedback. We also show that any algorithm for this problem must incur regret at least $\Omega_d(T^{(d-1)/(d+1)})$ (Theorem 12). The lower bound is obtained through a connection to spherical codes.

To prove these results, we investigate a more general problem, which we term *learning a Lipschitz function with binary feedback*, and which may be of independent interest. In this problem, a learner is trying to learn an L-Lipschitz function $f : [0,1]^d \to [0,1]$ over the course of T rounds. On round t, the learner is (adversarially) provided with a *context* x_t; the learner must then submit a guess y_t for $f(x_t)$, upon which they learn whether $y_t > f(x_t)$ or $y_t \leq f(x_t)$ (and notably, not the value of $f(x_t)$). The learner's goal is to minimize their total loss $\sum_t \ell(f(x_t), y_t)$, for some loss function $\ell(\cdot, \cdot)$.

For the symmetric loss function $\ell(\theta, y) = |\theta - y|$ we provide the following regret bounds:

- when $d = 1$, there is an algorithm which achieves regret $O(L \log T)$ (Theorem 2). Any algorithm for this problem must incur regret $\tilde{\Omega}(L\sqrt{\log T})$ (Theorem 8).
- for $d > 1$, there is an algorithm which achieves regret $\Theta(LT^{(d-1)/d})$, which is tight (Theorems 3, 6).

We note that our problem for the symmetric loss function is no longer an instance of Lipschitz or semi-Lipschitz bandits, since the feedback is very restricted: the algorithm doesn't learn the actual loss – it only receives binary feedback as to whether its guess was above or below the true value.

We present two types of algorithms for this problem. The first set of algorithms are based around the divide-and-conquer strategy of *iterative partition refinement* which is the main workhorse for dealing with Lipschitz assumptions in learning [18, 17, 23, 12]. Here the algorithm starts with a partition of the domain of f (perhaps just the domain itself), and tries to approximate f on each element of this partition. When the algorithm approximates f on a given element of the partition accurately enough, it further divides that element.

The second set of algorithms does not keep track of a partition of the domain but instead maintains lower and upper estimates of the function we are trying to learn. For example, we show that the natural algorithm which simply chooses the point halfway between the smallest possible value of $f(x_t)$ and the largest possible value of $f(x_t)$ consistent with the information known so far (the "midpoint algorithm") also achieves our optimal regret bounds. Such algorithms have the advantage that information learned about $f(x_t)$ is not necessarily confined to points in the vicinity of x_t, and thus may perform better in practice. See Section 2.3 for details.

Our lower bounds largely follow directly from the analysis of our algorithms, with the notable exception of the $\Omega(\sqrt{\log T})$ lower bound for the symmetric loss when $d = 1$. To prove this lower bound, we demonstrate how to construct a family of Lipschitz functions which encode random walks

of length $\approx \log T$ in the slopes between queried points. Understanding how to close the gap between $\Omega(\sqrt{\log T})$ and $O(\log T)$ for this case is an interesting open question.

The remainder of this paper is organized as follows. In the rest of this section, we discuss related work and formally define the problem of learning a Lipschitz function with binary feedback. In Section 2, we present our algorithms for learning a Lipschitz function with binary feedback, and in Section 3, we provide corresponding lowerbounds. Finally, in Section 4, we discuss how to apply these results to the contextual pricing problem (with emphasis on the setting with multiple linear buyers). For conciseness, the majority of proofs are omitted from the main body and appear in Appendix B.

1.1 Related Work

Our work belongs to the intersection of two major streams of literature: (i) learning for revenue optimization and (ii) continuum-armed and Lipschitz bandits. For revenue optimization, besides the work on contextual learning cited earlier, there are interesting other interesting directions such a learning with limited inventory. See for example Besbes and Zeevi [5], Babaioff et al [3], Badanidiyuru et al [4], Wang et al [24] and den Boer and Zwart [11]. Also relevant is the work on learning parametric models: Broder and Rusmevichientong [7], Chen and Farias [9] and Besbes and Zeevi [6].

Another relevant line of work is research on continuum-armed and Lipschitz bandits. The problem was introduced by Agrawal [1] and nearly tight bounds were obtained by Kleinberg [18]. Later the model was been extended to general metric spaces by Slivkins [22], Kleinberg and Slivkins [16] and Kleinberg, Slivkins and Upfal [17]. The problem with similarity information on contexts is studied by Hazan and Megiddo [12]. Slivkins [23] extends the Lipschitz bandits to contextual settings, i.e., when there is similarity information on both contexts and arms. Cesa-Bianchi et al [8] study the problem under partial feedback and weaken the Lipschitz assumption in previous work to semi-Lipschitz.

1.2 Learning a Lipschitz function from binary feedback

Definition 1. *A function $f : \mathbb{R} \to \mathbb{R}$ is L-Lipschitz if, for all $x, y \in \mathbb{R}$, $|f(x) - f(y)| \leq L|x - y|$.*

In this paper we study the problem of *learning a Lipschitz function from binary feedback*. This problem can be thought of as the following game between an adversary and a learner. At the beginning, the adversary chooses an L-Lipschitz function $f : [0, 1] \to [0, 1]$. Then, on round t (for T rounds), the adversary begins by providing the learner with a point $x_t \in [0, 1]$. The learner must then submit a guess y_t for $f(x_t)$. The learner then learns whether $y_t > f(x_t)$ or not. The goal of the learner is to minimize their *total loss* (alternatively, *regret*) over T rounds, $\mathsf{Reg} = \sum_{t=1}^{T} \ell(f(x_t), y_t)$, where $\ell(\cdot, \cdot)$ is some loss function.

In this paper, we consider the following two loss functions:

Symmetric loss. The symmetric loss is given by the function $\ell(f(x_t), y_t) = |f(x_t) - y_t|$. This is simply the distance between the learner's guess and the true value.

Pricing loss. The pricing loss is given by the function $\ell(f(x_t), y_t) = f(x_t) - y_t \mathbf{1}\{y_t \leq f(x_t)\}$. In other words, the pricing loss equals the symmetric loss when the guess y_t is less than $f(x_t)$ (and goes to 0 as $y_t \to f(x_t)^-$), but equals $f(x_t)$ when the guess y_t is larger than $f(x_t)$. This loss often arises in pricing applications (where setting a price slightly larger than optimal leads to no sale and much higher regret than a price slightly lower than optimal).

We also consider a variant of this problem for higher-dimensional Lipschitz functions. For functions $f : \mathbb{R}^d \to \mathbb{R}$, we define L-Lipschitz with respect to the L_∞-norm on $\mathbb{R}^d$: $|f(x) - f(y)| \leq L\|x - y\|_\infty$ for all $x, y \in \mathbb{R}^d$. Our results hold for other L_p norms on $\mathbb{R}^d$, up to polynomial factors in d. We can then define the problem of learning a (higher-dimensional) Lipschitz function $f : [0, 1]^d \to [0, 1]$ analogously as to above.

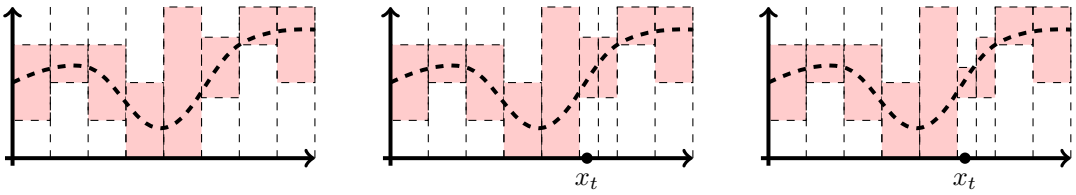

Figure 1: Illustration of Algorithm 1: the dashed curve corresponds to the (unknown) Lipschitz function, the rectangles correspond to feasible regions for the function. When an update results in a part of the partition with small relative height, we bisect this part of the partition.

Oftentimes, we will want to think of d as fixed, and consider only the asymptotic dependence on T of some quantity (e.g. the regret of some algorithm). We will use the notation $O_d(\cdot)$ and $\Omega_d(\cdot)$ to hide the dependency on d.

2 Algorithms for learning a Lipschitz function

2.1 Symmetric Loss

In this subsection we present algorithms for learning Lipschitz functions under the symmetric loss that incur sublinear total regret. Without loss of generality, we will assume in this section that $L \geq 1$ (the results of Appendix A allow us to extend these algorithms to $L \leq 1$ with slight modifications to the regret bounds).

We begin by examining the case where $d = 1$ (the functions are from $\mathbb{R} \to \mathbb{R}$). The following algorithm (Algorithm 1) achieves total loss $O(L \log T)$. Algorithm 1 maintains a partition of the domain of f ($[0, 1]$) into a collection of intervals X_j. For each interval X_j, the algorithm maintains an associated interval Y_j that satisfies $f(X_j) \subseteq Y_j$.

When a point x in X_j is queried, the learner submits as their guess the midpoint y of the interval Y_j. The binary feedback of whether $y > f(x)$ or not allows the learner to update the interval Y_j, shrinking it. Once Y_j grows small enough with respect to X_j, we bisect X_j into two smaller intervals. This procedure is illustrated in Figure 1.

Algorithm 1 Algorithm for learning a L-Lipschitz function from $\mathbb{R}$ to $\mathbb{R}$ under symmetric loss with regret $O(L \log T)$.

Learner maintains a partition of $[0, 1]$ into intervals X_j.
Along with each interval X_j, learner maintains an associated range $Y_j \subseteq [0, 1]$ such that if $x \in X_j$, $f(x) \in Y_j$.
Initially, learner partitions $[0, 1]$ into $\lceil 8L \rceil$ intervals X_j of equal length $\leq 1/8L$ and sets all $Y_j = [0, 1]$.
for $t = 1$ to T **do**
    Learner receives an $x_t \in [0, 1]$ from the adversary.
    Learner finds j s.t. $x_t \in X_j$. Let $\ell_j = \mathsf{length}(X_j)$.
    Learner guesses $y_t = (\max(Y_j) + \min(Y_j))/2$.
    if $y_t > f(x_t)$ **then**
        $Y_j \leftarrow Y_j \cap [0, y_t + L\ell_j]$
    else
        $Y_j \leftarrow Y_j \cap [y_t - L\ell_j, 1]$.
    end if
    Let $h_j = \mathsf{length}(Y_j)$.
    if $h_j < 4L\ell_j$ **then**
        Bisect X_j to form two new intervals X_{j_1} and X_{j_2}. Set $Y_{j_1} = Y_{j_2} = Y_j$.
    end if
end for

Theorem 2. *Algorithm 1 achieves regret $O(L \log T)$ for learning a L-Lipschitz function with symmetric loss.*

Roughly, the proof of Theorem 2 follows from the following two properties: i) after a constant number of queries belonging to any interval X_j, the interval Y_j will shrink enough to trigger a bisection, and ii) the regret of a query in an interval X_j is at most $\mathsf{length}(Y_j)$ which itself is $O(\mathsf{length}(X_j))$.

Now, if we start with $\Theta(1)$ intervals of length $\Theta(1)$, throughout the process there will be at most $O(2^r)$ intervals of length $\Theta(2^{-r})$ (those intervals bisected r times). Since each query in an interval of length ℓ contributes $O(\ell)$ to the overall regret, this means that the total regret from T queries is at most $O(1 + 2 \cdot 2^{-1} + 2^2 \cdot 2^{-2} + \cdots + 2^{\log T} \cdot 2^{-\log T}) = O(\log T)$. The full proof can be found in Appendix B.

It is possible to extend Algorithm 1 (in a straightforward way) to Lipschitz functions from $\mathbb{R}^d$ to $\mathbb{R}$. Pseudocode for this algorithm is provided in Algorithm 3 in Appendix B. Here, for $d > 1$, we no longer get logarithmic regret; instead, Algorithm 3 achieves regret $O(LT^{(d-1)/d})$.

Theorem 3. *Algorithm 3 achieves regret $O(LT^{(d-1)/d})$ for learning a L-Lipschitz function from $\mathbb{R}^d$ to $\mathbb{R}$ with symmetric loss.*

The main difference between Theorem 3 and Theorem 2 is that there are now $O(2^{dr})$ "intervals" (d-dimensional boxes) of diameter $\Theta(2^{-r})$, so the total regret from T queries is now $O(1 + 2^d \cdot 2^{-1} + 2^{2d} \cdot 2^{-2} + \cdots + 2^{\log T} \cdot 2^{-(\log T)/d}) = O(T^{(d-1)/d})$.

2.2 Pricing Loss

We now explore algorithms that achieve low regret with respect to the pricing loss function. Our main approach will be to adapt Algorithm 3 (which achieves low regret with respect to the symmetric loss function for Lipschitz functions from $\mathbb{R}^d$ to $\mathbb{R}$) but stop subdividing once the length of a range Y_j drops below some threshold. The details are summarized in Algorithm 4 in Appendix B.

We show that Algorithm 4 achieves regret $O((LT)^{d/(d+1)})$. Note that for $d = 1$, this is $O(L\sqrt{T})$; unlike in the symmetric loss case, it is impossible to achieve logarithmic regret for the pricing loss (see Theorem 7).

Theorem 4. *Algorithm 4 achieves regret $O((LT)^{d/(d+1)})$ for learning a L-Lipschitz function from $\mathbb{R}^d$ to $\mathbb{R}$ with pricing loss.*

As with Theorem 3, a similar analysis to that of Theorem 2 holds, with the exception that the regret of a query in an interval is $O(1)$ (until the length of the interval shrinks below some threshold, in which case we play $\min Y_j$ and are guaranteed regret at most $\mathsf{length}(Y_j)$). Choosing this threshold optimally results in the above regret bound.

2.3 Midpoint algorithms

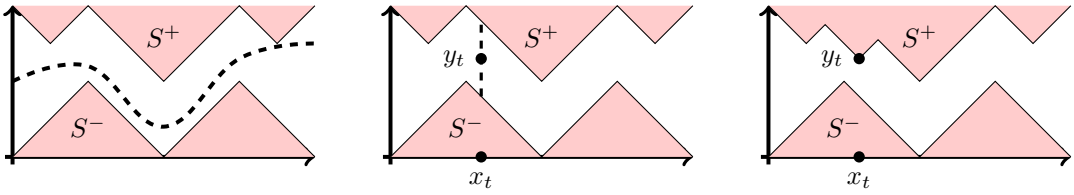

Figure 2: Illustration of the Midpoint Algorithm (Algorithm 2).

Let us return to considering the one-dimensional instance of learning an L-Lipschitz function under the symmetric loss. One very natural algorithm for this problem is the following. Throughout the algorithm, maintain two subregions of $[0, 1]^2$; S^+, a set of points $\{(x, y)\}$ that we know are guaranteed to satisfy $y \geq f(x)$, and S^-, a set of points $\{(x, y)\}$ that we know are guaranteed to satisfy $y \leq f(x)$.

Initially, S^+ and S^- start empty (or more accurately, containing the two lines $[0, 1] \times \{1\}$ and $[0, 1] \times \{0\}$, respectively). Each time we receive feedback of the form $y_t > f(x_t)$, we can add all points (x, y) which satisfy $y \geq y_t + L|x_t - x|$ to S^+; by the L-Lipschitz condition, all such points

satisfy $y \geq f(x)$. Similarly, each time we receive feedback of the form $y_t < f(x_t)$, we can add all points (x, y) which satisfy $y \leq y_t - L|x_t - x|$ to S^-.

Finally, to choose y_t given x_t, we should choose some y_t between $y^- = \max\{y|(x_t, y) \in S^-\}$ and $y^+ = \min\{y|(x_t, y) \in S^+\}$. A natural choice is their midpoint $(y^- + y^+)/2$. We call this algorithm the *midpoint algorithm*; its details are summarized in Algorithm 2. This process is depicted in Figure 2.

Algorithm 2 Midpoint algorithm for learning a L-Lipschitz function from $\mathbb{R}$ to $\mathbb{R}$ under symmetric loss with regret $O(L \log T)$.

Learner maintains two polygonal subsets S^+ and S^- of $[0, 1]^2$.
Initially, $S^+ = \{(x, 1)|x \in [0, 1]\}$ and $S^- = \{(x, 0)|x \in [0, 1]\}$.
for $t = 1$ to T **do**
  Learner receives an $x_t \in [0, 1]$ from the adversary.
  Learner computes $y^- = \max\{y|(x_t, y) \in S^-\}$ and $y^+ = \min\{y|(x_t, y) \in S^+\}$.
  Learner guesses $y_t = (y^- + y^+)/2$.
  if $y_t > f(x_t)$ **then**
    $S^+ \leftarrow S^+ \cup \{(x, y)|y \geq y_t + L|x_t - x|\}$.
  else
    $S^- \leftarrow S^- \cup \{(x, y)|y \leq y_t - L|x_t - x|\}$.
  end if
end for

Note that while Algorithms 1, 3, and 4 are low-regret (with essentially tight matching lower-bounds) and efficiently implementable, they don't share information between different intervals X_i. One advantage of the midpoint algorithm over these algorithms is that information provided from a query at a point x is not necessarily localized to the immediate neighborhood around x.

We show that, like Algorithm 1, the midpoint algorithm is also low regret.

Theorem 5. *Algorithm 2 achieves regret $O(\log T)$ for learning a L-Lipschitz function from $\mathbb{R}$ to $\mathbb{R}$ with symmetric loss.*

It is likewise possible to adapt the midpoint algorithm to multiple dimensions and to the pricing loss function (by choosing y^- whenever $y^+ - y^-$ is below some threshold) and prove analogues of Theorems 3 and 4. We omit the details for conciseness.

3 Lower bounds for learning a Lipschitz function

In this section, we state lower bounds for our results in Section 2. Interestingly all our lower bounds also hold for a slightly easier problem in which the algorithm learns the value of $f(x_t)$ after round t (instead of just whether $y < f(x_t)$).

Generally, all of our lower bounds work in the following way. We construct a collection $\mathcal{C}$ of L-Lipschitz functions and a sequence of queries $x_1, x_2, \ldots, x_T$ for the adversary such that for a random function f in $\mathcal{C}$, $f(x_t)$ is equally likely to be $\frac{1}{2} + \delta_t$ or $\frac{1}{2} - \delta_t$ for some δ_t, even conditioned on the values of $f(x_1)$ through $f(x_{t-1})$.

For both the symmetric loss when $d > 1$, and the pricing loss (for all d), constructing such a collection is easy; we simply divide the domain into small cubes, let x_1 through x_T run over the centers of such cubes, and let $f(x_t)$ be either $\frac{1}{2} + \delta$ or $\frac{1}{2} - \delta$ with equal probability. Optimizing δ leads to the following tight lower bounds.

Theorem 6. *For $d > 1$ and $L \leq T^{1/d}$, any algorithm for learning an L-Lipschitz function with symmetric loss must incur $\Omega\left(LT^{\frac{d-1}{d}}\right)$ regret for the d-dimensional case.*

Theorem 7. *For $L \leq T^{1/d}$, any algorithm for learning an L-Lipschitz function with the pricing loss must incur $\Omega\left((LT)^{\frac{d}{d+1}}\right)$ regret for the d-dimensional case.*

More interesting is the case of the symmetric loss when $d = 1$. Here we obtain an $\tilde{\Omega}(\sqrt{\log T})$ lower bound.

Theorem 8. *Any algorithm for learning an L-Lipschitz function with symmetric loss must incur $\Omega\left(L\sqrt{\frac{\log T}{\log\log T}}\right)$ regret.*

The proof of Theorem 8 proceeds roughly as follows. Our queries x_t will range over all the dyadic rationals, in order of increasing denominator (e.g. in the order $1/2, 1/4, 3/4, 1/8, 3/8, 5/8, 7/8$). We now use this sequence of x_t's to adaptively construct a Lipschitz function $f(x)$ in the following way. We start by setting $f(0) = f(1) = 1/2$. To set the value of $f(x_t)$ for some $x_t = \frac{2i+1}{2^r}$, let $L = i/2^{r-1}$ and $R = (i+1)/2^{r-1}$ (note that x_t is the midpoint of $[L, R]$, and $f(L)$ and $f(R)$ have already been chosen inductively). Let m be the slope between $(L, f(L))$ and $(R, f(R))$. Now, we choose $f(x_t)$ so that the slope between $(L, f(L))$ and $(x_t, f(x_t))$ is $m+\delta$ with probability $1/2$, and $m-\delta$ with probability $1/2$. If this causes the Lipschitz condition to be violated (because $m+\delta > L$ or $m-\delta < -L$), we instead just set $f(x_t) = (f(L) + f(R))/2$.

This process has the interesting property that the slope of a segment of length 2^{-r} of this function f is δ times a random walk of length r. If we choose $\delta = \Theta(1/\sqrt{\log T})$, then we can run this random walk for $\approx L \log T$ steps without running into this Lipschitz constraint (since the expected maximum value of a random walk of length n is $\widetilde{\Theta}(\sqrt{n})$). Analyzing the regret for this choice of δ leads to the regret bound in Theorem 8. For more details, see Appendix B.

4 Contextual Pricing for Linear Buyers

We now show how to apply our solutions to the problem of learning a Lipschitz function (in particular, with respect to the pricing loss function) to the problem of contextual dynamic pricing (with a particular emphasis on when all the buyers have linear valuation functions).

We begin by examining the case where each buyer i (for $1 \le i \le b$) has an L-Lipschitz valuation function $V_i : [0,1]^d \to [0,1]$, with $V_i(x)$ representing how much they would be willing to pay for an item with features $x \in \mathbb{R}^d$. Let $f(x) = \max_i V_i(x)$. Note that the seller successfully makes a sale at round t if $p_t \le f(x_t)$, in which case the seller receives revenue p_t; otherwise, the seller receives revenue 0. But now, note that since f is the maximum of several L-Lipschitz functions, f is also L-Lipschitz. This problem is therefore exactly the problem of learning a Lipschitz function with respect to the pricing loss function. Since f can be any L-Lipschitz function from $[0,1]^d \to [0,1]$, lower bounds for learning such functions carry over to this dynamic pricing problem. Theorems 4 and 7 thus imply the following corollary.

Corollary 9. *There exists an algorithm for solving the contextual dynamic pricing problem for L-Lipschitz buyers in d dimensions with total regret $O((LT)^{d/(d+1)})$. Any algorithm for solving the contextual dynamic pricing problem for L-Lipschitz buyers in d dimensions must incur total regret at least $\Omega((LT)^{d/(d+1)})$.*

An interesting special case is the one where all buyers have linear valuations, i.e., $V_i(x) = \langle v_i, u\rangle$ for some vector $v_i \in [0,1]^d$. The case with $b = 1$ buyer is very well-studied and a regret bound of $O(\mathsf{poly}(d)\log\log T)$ is possible [19]. For $b > 1$, we exploit the special structure of the problem to improve over the $O(T^{d/(d+1)})$ guarantee of Corollary 9.

We begin by reinterpreting this problem geometrically as follows. Define S to be the convex hull $\mathsf{conv}(0, v_1, v_2, \ldots, v_b)$. Note that there exists a buyer willing to buy an item $x_t \in [0,1]^d$ at price p_t iff the hyperplane $\{u \in \mathbb{R}^d; \langle x_t, u\rangle = p_t\}$ intersects the set S. For this reason, we will abuse notation and refer to this convex hull S as the "set of buyers" (indeed, adding a buyer with a v corresponding to any point within S does not change the outcome any sale). One can then alternatively view the dynamic pricing problem for linear buyers as the problem of learning the extreme points of a convex set $S \subseteq [0,1]^d$ from binary feedback.

In this problem, the context provided by the adversary is the feature vector x_t of the item at time t. Since without loss of generality, this context x_t is a unit vector in $\mathbb{R}^d$ (if it is not one, it can be scaled to become one along with the price, at the cost of at most a $\sqrt{d}$ factor in regret), and is therefore a $(d-1)$-dimensional object. We will parametrize these unit vectors via *generalized spherical coordinates*; that is, the $(d-1)$-tuple $(\theta_1, \theta_2, \ldots, \theta_{d-1}) \in [0, \pi/2]^{d-1}$ corresponds to the unit vector defined by

$$(\cos\theta_1, \sin\theta_1\cos\theta_2, \sin\theta_1\sin\theta_2\cos\theta_3, \ldots, \sin\theta_1\sin\theta_2\cdots\sin\theta_{d-2}\cos\theta_{d-1})\,.$$

Let $x(\theta)$ (for $\theta \in [0, \pi/2]^{d-1}$) be the above unit vector in $\mathbb{R}^d$. We make the following observation.

Lemma 10. *Let $F(\theta) = \max_{v\in S}\langle x(\theta), v\rangle$. Then F is L-Lipschitz for $L = O(d^2)$.*

Now, note that the dynamic pricing problem for linear buyers is exactly the problem of learning the function F with respect to the pricing loss; every round, the adversary supplies a context θ, the seller submits a price p, and the seller receives revenue p if $F(\theta) \geq p$ and revenue 0 otherwise. Theorem 4 immediately implies the following corollary.

Corollary 11. *There exists an algorithm for solving the contextual dynamic pricing problem in $d > 1$ dimensions with total regret $O(d^{2(d-1)/d}T^{(d-1)/d}) = O_d(T^{(d-1)/d})$.*

Unfortunately, not every Lipschitz function can occur as a valid $F(\theta)$, so the lower bounds from Section 3 do not immediately hold. Nonetheless, we can adapt the ideas from Theorem 7 to prove that any algorithm for solving the contextual dynamic pricing problem in d dimensions must incur regret $\Omega_d(T^{(d-1)/(d+1)})$.

Theorem 12. *Any algorithm for solving the contextual dynamic pricing problem in $d > 1$ dimensions must incur total regret at least $\Omega_d(T^{(d-1)/(d+1)})$.*

To prove Theorem 12, we will need the following lemma regarding the maximum size of spherical codes.

Lemma 13. *Let $\alpha > 0$. Then there exists a set U_α of $\Theta_d(\alpha^{-(d-1)})$ unit vectors in $(\mathbb{R}^+)^d$ such that for any two distinct elements u, u' of U_α, $\langle u, u'\rangle \leq \cos\alpha$ (i.e. any two distinct unit vectors are separated by angle at least α).*

We now proceed to prove Theorem 12.

Proof of Theorem 12. Choose $\alpha = \Theta_d(T^{-1/(d+1)})$. The adversary will choose the set B of buyers as follows. For every element v of the set U_α (defined in Lemma 13), the adversary with probability half adds v to B, and otherwise adds $(\cos\alpha)v$ to B. The adversary will then choose the contexts as follows: for each element u in U_t, the adversary will set $u_t = u$ for $T/|U_\alpha|$ rounds.

We claim no learning algorithm achieves $O_d(T^{(d-1)/(d+1)})$ regret against this adversary. Consider each element u of U_α, and consider the rounds where $x_t = u$. Either one of two cases must occur:

- **Case 1**: the algorithm never sets a price larger than $\cos\alpha$. Then, with probability $1/2$ (if $u \in B$), the maximal price the algorithm could have set was 1, so the algorithm incurs expected regret at least $\frac{1}{2}(1-\cos\alpha)(T/|U_\alpha|) = \Omega_d\left(\alpha^2\frac{T}{\alpha^{-(d-1)}}\right) = \Omega_d(T\alpha^{(d+1)}) = \Omega_d(1)$.
- **Case 2**: the algorithm at some point sets a price larger than $\cos\alpha$. Then, with probability $1/2$ (if $u \notin B$) the largest price the algorithm could have set was $\cos\alpha$ (since $\langle u', u\rangle \leq \cos\alpha$ for all other $u' \in u_t$, and we know $(\cos\alpha)u \in B$), so the algorithm overprices and incurs expected regret at least $\frac{1}{2}\cos\alpha = \Omega(1)$.

In either case, the algorithm incurs at least $\Omega_d(1)$ regret. Over all $|U_t|$ different contexts, this is at least $\Omega_d(T^{(d-1)/(d+1)})$ regret. □

Closing the gap between the upper bound of $O_d(T^{(d-1)/d})$ and the lower bound of $\Omega_d(T^{(d-1)/(d+1)})$ is an interesting open problem.

References

[1] Rajeev Agrawal. The continuum-armed bandit problem. *SIAM journal on control and optimization*, 33(6):1926–1951, 1995.

[2] Kareem Amin, Afshin Rostamizadeh, and Umar Syed. Repeated contextual auctions with strategic buyers. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 622–630, 2014.

[3] Moshe Babaioff, Shaddin Dughmi, Robert D. Kleinberg, and Aleksandrs Slivkins. Dynamic pricing with limited supply. *ACM Trans. Economics and Comput.*, 3(1):4:1–4:26, 2015.

[4] Ashwinkumar Badanidiyuru, Robert Kleinberg, and Aleksandrs Slivkins. Bandits with knapsacks. *J. ACM*, 65(3):13:1–13:55, 2018.

[5] Omar Besbes and Assaf Zeevi. Dynamic pricing without knowing the demand function: Risk bounds and near-optimal algorithms. *Operations Research*, 57(6):1407–1420, 2009.

[6] Omar Besbes and Assaf Zeevi. On the (surprising) sufficiency of linear models for dynamic pricing with demand learning. *Management Science*, 61(4):723–739, 2015.

[7] Josef Broder and Paat Rusmevichientong. Dynamic pricing under a general parametric choice model. *Operations Research*, 60(4):965–980, 2012.

[8] 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, 2017.

[9] Yiwei Chen and Vivek F Farias. Simple policies for dynamic pricing with imperfect forecasts. *Operations Research*, 61(3):612–624, 2013.

[10] Maxime C Cohen, Ilan Lobel, and Renato Paes Leme. Feature-based dynamic pricing. In *Proceedings of the 2016 ACM Conference on Economics and Computation*, pages 817–817. ACM, 2016.

[11] Arnoud V den Boer and Bert Zwart. Dynamic pricing and learning with finite inventories. *Operations research*, 63(4):965–978, 2015.

[12] Elad Hazan and Nimrod Megiddo. Online learning with prior knowledge. In *International Conference on Computational Learning Theory*, pages 499–513. Springer, 2007.

[13] Adel Javanmard. Perishability of data: dynamic pricing under varying-coefficient models. *The Journal of Machine Learning Research*, 18(1):1714–1744, 2017.

[14] Adel Javanmard and Hamid Nazerzadeh. Dynamic pricing in high-dimensions. *Working paper, University of Southern California*, 2016.

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

[16] 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, SODA 2010, Austin, Texas, USA, January 17-19, 2010*, pages 827–846, 2010.

[17] 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. ACM, 2008.

[18] Robert D Kleinberg. Nearly tight bounds for the continuum-armed bandit problem. In *Advances in Neural Information Processing Systems*, pages 697–704, 2005.

[19] Renato Paes Leme and Jon Schneider. Contextual search via intrinsic volumes. *CoRR*, abs/1804.03195, 2018.

[20] Ilan Lobel, Renato Paes Leme, and Adrian Vladu. Multidimensional binary search for contextual decision-making. *Operations Research*, 2017.

[21] Sheng Qiang and Mohsen Bayati. Dynamic pricing with demand covariates. *Available at SSRN 2765257*, 2016.

[22] Aleksandrs Slivkins. Multi-armed bandits on implicit metric spaces. In *Advances in Neural Information Processing Systems 24: 25th Annual Conference on Neural Information Processing Systems 2011. Proceedings of a meeting held 12-14 December 2011, Granada, Spain.*, pages 1602–1610, 2011.

[23] Aleksandrs Slivkins. Contextual bandits with similarity information. *Journal of Machine Learning Research*, 15(1):2533–2568, 2014.

[24] Zizhuo Wang, Shiming Deng, and Yinyu Ye. Close the gaps: A learning-while-doing algorithm for single-product revenue management problems. *Operations Research*, 62(2):318–331, 2014.

A Reductions Between Different Lipschitz Constants for Symmetric Loss

In this section, we provide black-box reductions that allow us to convert low-regret algorithms for learning L-Lipschitz functions into L'-Lipschitz functions. These reductions prove important for obtaining the correct regret bounds in L in Section 2, along with allowing us to argue only about $L \geq 1$ (for the symmetric loss).

Lemma 14. *Suppose for some specific L, an algorithm A achieves regret $r(T)$ for learning an L-Lipschitz function with symmetric loss in the d-dimensional case, where $r(T)$ is concave in T. For any $L' > L$, let $\alpha = \lceil \frac{L'}{L} \rceil$. Then there exists an algorithm A' that achieves regret $\alpha^d \cdot r\left(\frac{T}{\alpha^d}\right)$ for learning an L'-Lipschitz function with symmetric loss in the d-dimensional case.*

Proof. Partition $[0,1]^d$ into α^d small cubes of edge length $1/\alpha$. For each such cube, if we expand the edge length by a multiple of α, then any function that is L'-Lipschitz in the cube will become L-Lipschitz. So our new algorithm will just expand the edge length of all cubes by a multiple of α and run algorithm A on each small cube separately.

Label the α^d small cubes 1 through α^d, and assume cube i receives T_i queries. We have $T = T_1 + \cdots + T_{\alpha^d}$. By Jensen's inequality, the new algorithm will have regret

$$\sum_{i=1}^{\alpha^d} r(T_i) \leq \alpha^d \cdot r\left(\frac{T}{\alpha^d}\right).$$

□

Lemma 15. *Suppose an algorithm A achieves regret $r(T)$ for learning an 1-Lipschitz function with symmetric loss in the d-dimensional case. For any $L < 1$, let $\alpha = \lfloor \frac{1}{L'} \rfloor$, we have an algorithm that achieves regret at most $\frac{4}{\alpha} \cdot r(T) + O(1)$ for learning an L-Lipschitz function with symmetric loss in the d-dimensional case.*

Proof. When $\alpha \leq 4$, the bound follows trivially from just applying A. Now we assume $\alpha \geq 4$ and therefore $L \leq 1/4$. When the function is $\frac{1}{4}$-Lipschitz, we know the gap between the maximum value and the minimum value of the function in $[0,1]^d$ is at most $\frac{1}{4}$.

Now, similarly to the process in Algorithm 1, we do a binary search to find a small interval Y such that $f([0,1]^d) \subset Y$, stopping when $\mathsf{length}(Y)$ is no more than $\frac{4}{\alpha}$. This step results in at most $O(1)$ regret (since the length of Y decreases by a constant factor every query). We can then multiply the function range by a factor $\frac{\alpha}{4}$ and run algorithm A for the remainder of the rounds. This part has regret at most $\frac{4}{\alpha} \cdot r(T)$. This new algorithm has regret at most $\frac{4}{\alpha} \cdot r(T) + O(1)$ for learning an L-Lipschitz function with symmetric loss.

□

B Omitted algorithms and proofs

Proof of Theorem 2

Proof. We begin by proving some preliminary lemmas. We begin by showing that it is always the case that $f(X_j) \subseteq Y_j$.

Lemma 16. *Let X_j be an interval in the partition maintained by Algorithm 1 (at some time t). Then for any $x \in X_j$, $f(x) \in Y_j$.*

Proof. Note that in the initial partition, all the Y_j's start off equal to $[0,1]$, where this holds trivially. In addition, whenever we create new intervals (via bisection of old intervals), we initially set the range for the each of the new intervals equal to the range for the old itnervals. It therefore suffices to show that this property continues to hold whenever we update Y_j in response to learning whether $y_t > f(x_t)$.

We update Y_j in response to learning that $y > f(x)$ (or $\leq f(x)$) for some $x \in X_j$ and some $y \in Y_j$. Note that if $f(x) < y$, then by the Lipschitz property, for any $x' \in X_j$, $f(x') < f(x) + L|x' - x| <$

$y + L \cdot \mathsf{length}(X_j)$. Likewise, if $f(x) > y$, then by the Lipschitz property, for any $x' \in X_j$, $f(x') > f(x) - L|x' - x| > y - L \cdot \mathsf{length}(X_j)$. Letting $\ell_j = \mathsf{length}(X_j)$, it follows that in the first case $f(X_j) \subseteq [0, y + L\ell_j]$ and in the second case, $f(X_j) \subseteq [y - L\ell_j, 1]$, and therefore the update rules preserve this property. □

We now show that if $h_j \geq 4L\ell_j$, then performing the update in lines 8-12 of the algorithm decreases h_j by a factor of at least $3/4$ and at most $1/2$.

Lemma 17. *Let h_j equal the length of Y_j before the update in lines 8-12, and let h'_j equal the length of Y_j after the update in lines 8-12. Then if $h_j \geq 4L\ell_j$, $h'_j/h_j \in [1/2, 3/4]$.*

Proof. Recall that the guess y_t is the midpoint of the interval Y_j. Regardless of whether $y_t > f(x_t)$ or $y_t \leq f(x_t)$, $h'_j = h_j/2 + L\ell_j$. It immediately follows that $h'_j/h_j \geq 1/2$. On the other hand, since $h_j \geq 4L\ell_j$, $h'_j/h_j = 1/2 + L\ell_j/h_j \leq \frac{3}{4}$. □

Finally, we show that it is always the case that (at the beginning of a turn) $h_j \geq 4L\ell_j$.

Lemma 18. *Let X_j be an interval in the partition (at the beginning of some turn t). Then $\mathsf{length}(Y_j) \geq 4L \cdot \mathsf{length}(X_j)$, and $\mathsf{length}(Y_j) \leq 8L \cdot \mathsf{length}(X_j)$.*

Proof. Let $h_j = \mathsf{length}(Y_j)$ and $\ell_j = \mathsf{length}(X_j)$.

By the construction of the initial partition, this is true at the beginning of turn 1. We must show that whenever h_j drops below $4L\ell_j$, bisecting the interval X_j (as in lines 14-16) restores this property. But this is true, since by Lemma 17, we must still have that $h_j > 2L\ell_j$ after the update to Y_j. When X_j is bisected to form intervals X_{j_1} and X_{j_2}, $\ell_{j_1} = \ell_{j_2} = \ell_j/2$, but $h_{j_1} = h_{j_2} = h_j$, so it is true that $h_{j_1} > 4L\ell_{j_1}$. Similarly, since $h_j < 4L\ell_j$ right before a bisection, $h_{j_1} = h_j < 4L\ell_j = 8L\ell_{j_1}$, as desired. □

We now analyze the regret incurred by this algorithm. We say an interval X_j in our partition has depth r if it is the result of the bisection of an interval at depth $r - 1$ (where the initial intervals all have depth 0). Note that if X_j is at depth r, then $\mathsf{length}(X_j) \leq 2^{-r}/8L$. By Lemma 18, it follows that if X_j is at depth r, then $\mathsf{length}(Y_j) \leq 2^{-r}$.

We claim that an adversary can choose a point x_t belonging to an interval X_j of depth r at most $O(1)$ times before our algorithm splits X_j into two smaller intervals. To see why this is true, note that by Lemma 18, Y_j starts off satisfying $\mathsf{length}(Y_j) \leq 8L \cdot \mathsf{length}(X_j)$, and we must bisect X_j as soon as $\mathsf{length}(Y_j) \leq 4L \cdot \mathsf{length}(X_j)$. But by Lemma 17, $\mathsf{length}(Y_j)$ shrinks by a factor of at least $3/4$ each time x_t belongs to X_j. Since $(3/4)^3 < 1/2$, after at most 3 iterations, we must divide X_j into two smaller intervals. Note that the total amount of regret incurred in turns where x_t belongs to an interval X_j of depth r is at therefore at most $\mathsf{length}(Y_j) \cdot O(1) = O(2^{-r})$.

Now, there are at most $O(L2^r)$ intervals of depth r, so the total regret contributed from intervals of depth r is $O(L)$. Over the course of T rounds (where, by this analysis you can be charged regret $O(2^{-r})$ at most $O(L2^r)$ times), this leads to a total regret of at most $O(L \log T)$. □

Proof of Theorem 3 (Algorithm 3)

We define the diameter $\mathsf{diam}(X)$ of a subset $X \subseteq \mathbb{R}^d$ to be the maximum of $||x - y||_\infty$ over all pairs $x, y \in X$. Note that $\mathsf{diam}([0,1]^d) = 1$.

Proof. We proceed similarly to the proof of Theorem 2. Note that, without loss of generality we can assume that $L = 1/8$ (so the initial partition just contains the box $[0,1]^d$). If we show a regret bound of $O(T^{(d-1)/d})$ for the case where $L = 1/8$, then Lemma 14 implies a regret bound of $O(L^d(T/L^d)^{(d-1)/d}) = O(LT^{(d-1)/d})$ in general.

As before, the following analogues of Lemmas 16, 17, and 18 hold (with the proofs carrying over essentially verbatim).

Lemma 19. *Let X_j be an interval in the partition maintained by Algorithm 1 (at some time t). Then for any $x \in X_j$, $f(x) \in Y_j$.*

Algorithm 3 Algorithm for learning a L-Lipschitz function from $\mathbb{R}^d$ to $\mathbb{R}$ ($d > 1$) under symmetric loss with regret $O(LT^{(d-1)/d})$.

1: Learner maintains a partition of $[0,1]^d$ into boxes (cartesian products of intervals) X_j.
2: Along with each interval X_j, learner maintains an associated range $Y_j \subseteq [0,1]$ such that if $x \in X_j$, $f(x) \in Y_j$.
3: Initially, learner partitions $[0,1]^d$ into $\lceil (8L)^d \rceil$ boxes X_j with side lengths $\leq 1/8L$ and sets all $Y_j = [0,1]$.
4: **for** $t = 1$ to T **do**
5: Learner receives an $x_t \in [0,1]$ from the adversary.
6: Learner finds j s.t. $x_t \in X_j$. Let $\ell_j = \mathsf{diam}(X_j)$.
7: Learner guesses $y_t = (\max(Y_j) + \min(Y_j))/2$.
8: **if** $y_t > f(x_t)$ **then**
9: $Y_j \leftarrow Y_j \cap [0, y_t + L\ell_j]$
10: **else**
11: $Y_j \leftarrow Y_j \cap [y_t - L\ell_j, 1]$.
12: **end if**
13: Let $h_j = \mathsf{length}(Y_j)$.
14: **if** $h_j < 4L\ell_j$ **then**
15: Bisect each side of X_j to form 2^d new boxes X_{j_1} through $X_{j_{2^d}}$. Set $Y_{j_k} = Y_j$ (for $1 \leq k \leq 2^d$).
16: **end if**
17: **end for**

Lemma 20. *Let h_j equal the length of Y_j before the update in lines 8-12, and let h'_j equal the length of Y_j after the update in lines 8-12. Then if $h_j \geq 4L\ell_j$, $h'_j/h_j \in [1/2, 3/4]$.*

Lemma 21. *Let X_j be an interval in the partition (at the beginning of some turn t). Then $\mathsf{length}(Y_j) \geq 4L \cdot \mathsf{diam}(X_j)$, and $\mathsf{length}(Y_j) \leq 8L \cdot \mathsf{diam}(X_j)$.*

We again define the notion of *depth* by saying that a box X_j in our partition has depth r if it is the result of the bisection (into 2^d parts) of a box at depth $r-1$ (where the initial boxes all have depth 0). Since the diameter of each box in a bisection is exactly half of that of the original box, if X_j is at depth r, then $\mathsf{diam}(X_j) \leq 2^{-r}/8L$. By Lemma 18, it follows that if X_j is at depth r, then $\mathsf{diam}(Y_j) \leq 2^{-r}$.

Similarly as to in the proof of Theorem 3, we can show that an adversary can choose a point x_t belonging to a box X_j of depth r at most $O(1)$ times before our algorithm splits X_j into 2^d smaller boxes. Note that the total amount of regret incurred in turns where x_t belongs to a box X_j of depth r is at therefore at most $\mathsf{length}(Y_j) \cdot O(1) = O(2^{-r})$.

Now, there are at most $O(2^{rd})$ intervals of depth r (since $L = 1/8$, so there is only 1 interval of depth 0), so the total regret contributed from boxes of depth r is $O(2^{r(d-1)})$. Over the course of T rounds (where, by this analysis you can be charged regret $O(2^{-r})$ at most $O(2^{dr})$ times), this leads to a total regret of at most $O(T^{(d-1)/d})$, as desired. □

Proof of Theorem 4 (Algorithm 4)

Proof. We adapt the analysis from Theorem 3 (of Algorithm 3) to Algorithm 4. There are two primary differences between Algorithm 3 and Algorithm 4:

1. We stop following Algorithm 3 once the size of a range Y_j corresponding to some box drops below $\tau = ((8L)^d/T)^{1/d+1}$. At this point, we always choose $\min(Y_j)$, so we are guaranteed not to overguess the value of $f(x_t)$, and the total loss from queries in such boxes is therefore at most $O(T\tau) = O((8LT)^{d/(d+1)}) = O((LT)^{d/d+1})$.
2. Unlike in Theorem 3, the total amount of regret incurred in turns where x_t belongs to a box of depth r is now $O(1)$ (instead of $O(\mathsf{length}(Y_j)) = O(2^{-r})$), since overguessing can

Algorithm 4 Algorithm for learning a L-Lipschitz function from $\mathbb{R}^d$ to $\mathbb{R}$ ($d > 1$) under pricing loss with regret $O_d((LT)^{d/(d+1)})$.

1: Learner maintains a partition of $[0,1]^d$ into boxes (cartesian products of intervals) X_j.
2: Along with each interval X_j, learner maintains an associated range $Y_j \subseteq [0,1]$ such that if $x \in X_j$, $f(x) \in Y_j$.
3: Initially, learner partitions $[0,1]^d$ into $\lceil (8L)^d \rceil$ boxes X_j with side lengths $\leq 1/(8L)$ and sets all $Y_j = [0,1]$.
4: **for** $t = 1$ to T **do**
5: Learner receives an $x_t \in [0,1]$ from the adversary.
6: Learner finds j s.t. $x_t \in X_j$. Let $\ell_j = \mathsf{diam}(X_j)$, and let $h_j = \mathsf{length}(Y_j)$.
7: **if** $h_j \leq ((8L)^d/T)^{1/d+1}$ **then**
8: Learner guesses $y_t = \min(Y_j)$.
9: **else**
10: Learner guesses $y_t = (\max(Y_j) + \min(Y_j))/2$.
11: **if** $y_t > f(x_t)$ **then**
12: $Y_j \leftarrow Y_j \cap [0, y_t + L\ell_j]$
13: **else**
14: $Y_j \leftarrow Y_j \cap [y_t - L\ell_j, 1]$.
15: **end if**
16: **if** $h_j < 4L\ell_j$ **then**
17: Bisect each side of X_j to form 2^d new boxes X_{j_1} through $X_{j_{2^d}}$. Set $Y_{j_k} = Y_j$ (for $1 \leq k \leq 2^d$).
18: **end if**
19: **end if**
20: **end for**

lead to $\Theta(1)$ regret. This means that the total regret contributed from boxes of depth r is $O((8L)^d 2^{rd})$, i.e. the number of intervals of depth r.

Now, since the length of Y_j for a box X_j of depth r is at most 2^{-r}, any box of depth at least $\log 1/\tau$ satisfies $\mathsf{length}(Y_j) \leq \tau$. The total number of boxes at depth at most $\log 1/\tau$ is $O((8L)^d 2^{d \log 1/\tau}) = O((8L/\tau)^d) = O((8LT)^{d/(d+1)}) = O((LT)^{d/d+1})$. Each of these boxes contributes $O(1)$ loss, so our total loss is $O((LT)^{d/d+1})$, as desired. □

Proof of Theorem 5

Proof. We will mirror the analysis of Theorem 2.

Augment Algorithm 2 to keep track of a partition of $[0,1]$ into intervals X_j along with associated ranges Y_j for each interval X_j. Similarly as in Algorithm 1, we start with the partition into $\lceil 16L \rceil$ intervals X_j of equal length $\leq 1/16L$. At any point in time, we define Y_j via

$$Y_j = \left[\min_{x \in X_j} \max\{y \,|\, (x,y) \in S^-\}, \max_{x \in X_j} \min\{y \,|\, (x,y) \in S^+\} \right].$$

Similarly as in Algorithm 1, once $\mathsf{length}(Y_j) < 6L\mathsf{length}(X_j)$, we will divide X_j to form four new intervals of equal lengths.

By the definition of Y_j (and S^+ and S^-), it is true that if $x \in X_j$, then $f(x) \in Y_j$. We will now prove the analogues of Lemmas 17 and 18.

Lemma 22. *Assume $x_t \in X_j$, and let $\ell_j = \mathsf{length}(X_j)$. Let h_j equal the length of Y_j before the update in lines 7-11, and let h'_j equal the length of Y_j after the update in lines 7-11. Then if $h_j \geq 6L\ell_j$, $h'_j/h_j \in [1/3, 5/6]$.*

Proof. Define $y_{mid} = (\min(Y_j) + \max(Y_j))/2$ (i.e., the guess Algorithm 1 would have made in this situation). We first claim that $y_t = (y^- + y^+)/2$ is close to y_{mid}, namely that $|y_{mid} - y_t| \leq L\ell_j$.

To do this, note that the boundary of S^- and S^+ is composed of line segments of slope L, $-L$, and 0, so it is L-Lipschitz, and therefore $|y^- - \min(Y_j)| \le \ell_j$, and $|y^+ - \max(Y_j)| \le \ell_j$.

Now, if $y_t \le f(x_t)$, then $\min(Y_j)$ increases (and h_j decreases) by at least $|y_t - \min(Y_j)| - L\ell_j \ge |y_{mid} - \min(Y_j)| - |y_t - y_{mid}| - L\ell_j \ge h_j/2 - 2L\ell_j$. Since $h_j \ge 6L\ell_j$, this is at least $L\ell_j$, so $h'_j/h_j \le \frac{5}{6}$. Similarly, $\min(Y_j)$ increases by at most $|y_t - \min(Y_j)| \le |y_{mid} - \min(Y_j)| + |y_t - y_{mid}| \le h_j/2 + L\ell_j$, so $h'_j/h_j \ge \frac{1}{3}$.

By symmetry, when $y_t > f(x_t)$, it is also true that $h'_j/h_j \in [1/3, 5/6]$. □

Lemma 23. *Let X_j be an interval in the partition (at the beginning of some turn t). Then* $\mathsf{length}(Y_j) \ge 6L \cdot \mathsf{length}(X_j)$, *and* $\mathsf{length}(Y_j) \le 16L \cdot \mathsf{length}(X_j)$.

Proof. Let $h_j = \mathsf{length}(Y_j)$ and $\ell_j = \mathsf{length}(X_j)$.

By the construction of the initial partition, this is true at the beginning of turn 1. We must show that whenever h_j drops below $6L\ell_j$, trisecting the interval X_j (as in lines 14-16) restores this property. But this is true, since by Lemma 17, we must still have that $h_j > 2L\ell_j$ after the update to Y_j. When X_j is divided into four equal intervals X_{j_1}, X_{j_2}, X_{j_3}, and X_4, $\mathsf{length}(X_{j_k}) = \ell_j/4$. On the other hand, for each Y_{j_k}, $h_j - 2L\ell_j \le \mathsf{length}(Y_{j_k}) \le h_j$ (since $\max(Y_j) - \max(Y_{j_k}) \in [0, L\ell_j]$ by the Lipschitz condition, and likewise $\min(Y_{j_k}) - \min(Y_j) \in [0, L\ell_j]$). It follows that $\mathsf{length}(Y_{j_k}) \ge 6L\mathsf{length}(X_{j_k})$. Similarly, since $h_j < 4L\ell_j$ right before a division, $\mathsf{length}(Y_{j_k}) \le 16L\mathsf{length}(X_{j_k})$, as desired. □

With these two lemmas, the proof of Theorem 2 applies to show that the midpoint algorithm achieves regret $O(L \log T)$.

□

Proof of Theorem 6

Proof. Without loss of generality, we assume that $T = \alpha^d$ where α is an integer. Partition $[0,1]^d$ into α^d small cubes of edge length $1/\alpha$. Pick $x_1, \ldots, x_T$ to be the center of all these cubes. Note that for any $i \ne j$, $\|x_i - x_j\|_\infty \ge 1/\alpha$. For each $i = 1, ..., T$, we pick $f(x_i)$ uniformly random to be $1/2 + L/(2\alpha)$ or $1/2 - L/(2\alpha)$ (independent of other $f(x_j)$'s). For any realized $f(x_1), ..., f(x_T)$, the function is L-Lipschitz as $\forall i \ne j$,

$$|f(x_i) - f(x_j)| \le L/\alpha \le L\|x_i - x_j\|_\infty.$$

The function values are also bounded in $[0, 1]$ since $L \le T^{1/d} = \alpha$.

Since each $f(x_i)$ is independent from other $f(x_j)$'s, it's easy to see that any algorithm will have $L/(2\alpha)$ expected loss in each round. Therefore any algorithm will have expected regret at least

$$\frac{L}{2\alpha} \cdot T = \frac{LT}{2T^{1/d}} = \Omega(LT^{\frac{d-1}{d}}).$$

□

Proof of Theorem 7

Proof. When $LT < 1$, the theorem statement becomes trivial. We now assume $LT \ge 1$.

Without loss of generality, we assume that $LT = \alpha^{d+1}$ where α is an integer. Partition $[0,1]^d$ into α^d small cubes of edge length $1/\alpha$. Pick $x_1, \ldots, x_T$ to be the center of all these cubes such that each center is queried for $T^{1/(d+1)}/L^{d/(d+1)}$ rounds. We then have for any $i \ne j$ that $\|x_i - x_j\|_\infty \ge 1/\alpha$. For each center, we pick its function value uniformly random to be $3/4 + L/(4\alpha)$ or $3/4 - L/(4\alpha)$. For any realized $f(x_1), ..., f(x_T)$, the function is L-Lipschitz as $\forall x_i \ne x_j$,

$$|f(x_i) - f(x_j)| \le L/(2\alpha) \le L\|x_i - x_j\|_\infty.$$

The function values are also bounded in $[1/2, 1]$ since $L \le \alpha$.

Consider the center of each cube. Each center is queried $T^{1/(d+1)}/L^{d/(d+1)}$ times and has value either $3/4 + L/(4\alpha)$ or $3/4 - L/(4\alpha)$. Since we are concerned with the expected regret of our algorithm over some distribution of functions, we can wlog assume the algorithm is deterministic. We divide the deterministic algorithms for this specific center into 2 cases:

- Case 1: the algorithm never query some value $> 3/4+L/(4\alpha)$. Then with probability half, when the function value is actually $3/4 + L/(4\alpha)$, the algorithm has pricing loss $L/(2\alpha)$. In this case, the algorithm has expected regret at least $\frac{T^{1/(d+1)}}{L^{d/(d+1)}} \cdot \frac{L}{2\alpha} = \Omega(1)$ on one center.
- Case 2: the algorithm queries some value $> 3/4 - L/(4\alpha)$. Then with probability half, when the function value is actually $3/4 - L/(4\alpha)$, the algorithm has pricing loss $3/4 - L/(4\alpha)$ just in the round when it queries some value $> 3/4 - L/(4\alpha)$. In this case, the algorithm also has expected regret at least $\Omega(1)$ on one center.

To sum up, any algorithm will have expected regret at least $\Omega(1)$ on each center. Since there are $\alpha^d = (LT)^{\frac{d}{d+1}}$ centers, any algorithm will have expected regret at least $\Omega\left((LT)^{\frac{d}{d+1}}\right)$. □

Proof of Theorem 8

Proof. We will sample a function randomly and we will show that any algorithm will have $\Omega\left(L\sqrt{\frac{\log T}{\log\log T}}\right)$ regret in expectation.

We first prove the case when $L \leq 1$. Let $m = \lfloor \log_2(T+1) \rfloor$. We will focus on the first $2^m - 1$ rounds. For the random function f, we fix $f(0) = f(1) = 1/2$. For function values on other values of x_t, we will sample $f(x_t)$ in round t and make sure that there exists an L-lipschitz function that are consistent with the known $(x, f(x))$ pairs.

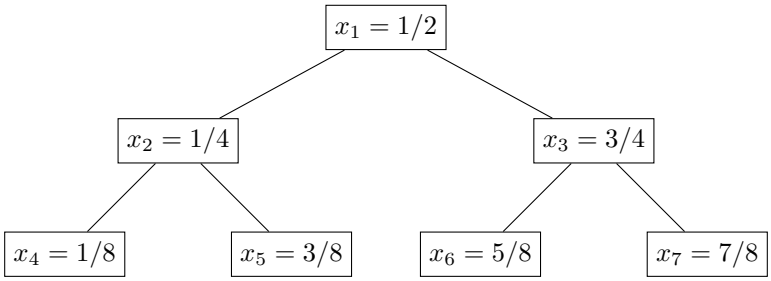

Figure 3: Construction of x_t in the lower bound

Let's first fix the values of $x_1, ..., x_{2^m-1}$ (as in Figure 3). We determine them in m levels. In level i, we determine the value of $x_{2^{i-1}}, ..., x_{2^i-1}$. For $j = 0, ..., 2^{i-1} - 1$, $x_{2^{i-1}+j}$ is set to $\frac{2j+1}{2^i}$.

Let's now define how to choose $f(x_t)$. We will proceed level by level. Set $\delta = 1/\sqrt{\log T \log\log T}$. For each t in level i (i.e. $t = 2^{i-1} + j$ for some $j \in \{0, ..., 2^{i-1} - 1\}$ and $x_t = \frac{2j+1}{2^i}$), we will define $f(x_t)$ based on its nearest defined locations: $f\left(\frac{j}{2^{i-1}}\right)$ and $f\left(\frac{j+1}{2^{i-1}}\right)$. Let the slope

$$d_t = \frac{f\left(\frac{j+1}{2^{i-1}}\right) - f\left(\frac{j}{2^{i-1}}\right)}{\frac{1}{2^{i-1}}}$$

and the average

$$m_t = \frac{1}{2} \cdot \left(f\left(\frac{j}{2^{i-1}}\right) + f\left(\frac{j+1}{2^{i-1}}\right) \right).$$

To choose $f(x_t)$, there are two cases:

- Case 1: If $|d_t| + L\delta > L$, $f(x_t) = m_t$.

- Case 2: Otherwise, $f(x_t) = m_t + \frac{L\delta}{2^i}$ with probability $1/2$, and $f(x_t) = m_t - \frac{L\delta}{2^i}$, with probability $1/2$.

It is not hard to check that after defining $f(x_t)$, there still exist a L-Lipschitz function that are consistent with the known $(x, f(x))$ pairs (in particular, this is true since every pair of neighboring x_t's satisfies the Lipschitz constraint).

We will now show that in each level, a constant fraction of x_t are defined via case 2. For each x_t in level i, it has two children x_{2t} and x_{2t+1} in level $i+1$ (as in Figure 3). By the construction, we know that if $|d_t| + L\delta \leq L$, then one of d_{2t} and d_{2t+1} is $d_t + L\delta$ and the other one is $d_t - L\delta$. If we take a random path from root to leaves as in Figure 3, the sequence of d_t's on the path behave like a random walk. Specifically this random walk has step length $L\delta$ and it stops when hits L or $-L$. We know if the random walk does not hit L or $-L$, it means that all the $f(x_t)$'s on the path is defined in case 2. Since the path has length $m = O(\log T)$ and we pick $\delta = 1/\sqrt{\log T \log\log T}$, we know that at least constant fraction of the paths never hit L or $-L$. It follows that in each level, there are at least a constant fraction of x_t's which are defined in case 2.

For any x_t's in level i, if $f(x_t)$ is defined in case two, any algorithm will get expected loss at least $\Omega(L\delta/2^i)$ in round t. We know that in each level, at least constant fraction of x_t's are defined in case 2. Therefore any algorithm will get expected regret

$$\Omega\left(\sum_{i=1}^{m} 2^{i-1} \cdot \frac{L\delta}{2^i}\right) = \Omega(L\delta m) = \Omega\left(L\sqrt{\frac{\log T}{\log\log T}}\right).$$

For the case when $L > 1$, if we directly use the above construction, the function value might exceed $[0, 1]$. Wlog we assume L is an integer. Divide x-axis into L regions: $[0, 1/L]$, $[1/L, 2/L]$, ...,$[(L-1)/L, 1]$. In i-th region, freshly sample a 1-Lipschitz function f with T/L locations according to the above procedure and construct function $f_i(x) = f\left(\left(x - \frac{i-1}{L}\right) \cdot L\right)$ for $x \in [\frac{i-1}{L}, \frac{i}{L}]$. Now consider function f' which is defined as combination of $f_1, ..., f_L$: $f'(x) = f_i(x)$ for $x \in [\frac{i-1}{L}, \frac{i}{L}]$. It's not hard to see that f' is L-Lipschitz. Also using the argument above, we get that any algorithm will have $\Omega\left(\sqrt{\frac{\log(T/L)}{\log\log(T/L)}}\right) = \Omega\left(\sqrt{\frac{\log T}{\log\log T}}\right)$ in each region $[(i-1)/L, i/L]$ for $i = 1, ..., L$. Since any algorithm does not learn anything about region j from function values in region i for $i \neq j$, any algorithm will have $\Omega\left(L\sqrt{\frac{\log T}{\log\log T}}\right)$ expected regret on function f'.

□

Proof of Lemma 10

Proof. First, note that $\sin x$ and $\cos x$ are both 1-Lipschitz functions, and that the product of d 1-Lipschitz functions bounded in $[-1, 1]$ is d-Lipschitz. From this, it follows that the ith component of $x(\theta)$ is i-Lipschitz.

Now, note that $F(\theta) = \langle x(\theta), v_\theta \rangle$ for some v_θ. Now, for any θ',

$$\begin{aligned}
F(\theta') &= \max_{v \in S} \langle x(\theta'), v \rangle \\
&\geq \langle x(\theta'), v_\theta \rangle \\
&= \langle x(\theta), v_\theta \rangle + \langle x(\theta') - x(\theta), v_\theta \rangle \\
&\geq F(\theta) - ||x(\theta') - x(\theta)||_1 \\
&= F(\theta) - \sum_{i=1}^{d-1} |x(\theta')_i - x(\theta)_i| \\
&\geq F(\theta) - \sum_{i=1}^{d-1} i||\theta' - \theta||_\infty \\
&= F(\theta) - \frac{d(d-1)}{2}||\theta' - \theta||_\infty
\end{aligned}$$

It follows that

$$F(\theta) - F(\theta') \leq \frac{d(d-1)}{2}||\theta' - \theta||_\infty.$$

By symmetry, it is also the case that

$$F(\theta') - F(\theta) \leq \frac{d(d-1)}{2}||\theta' - \theta||_\infty,$$

and therefore F is $d(d-1)/2$-Lipschitz. □

Proof of Lemma 13

Proof. We show how to construct such a set of unit vectors in $\mathbb{R}^d$. By then taking the orthant with the largest number of elements of this set (and rotating this orthant appropriately), this then leads to a construction of such a set of unit vectors in $(\mathbb{R}^+)^d$ (at the cost of a factor of 2^d).

Let $\mathbb{S}^{d-1}$ be the unit sphere in $\mathbb{R}^d$, and let $B(\alpha, u) \subset \mathbb{S}^{d-1}$ be the subset of the unit sphere of points that form an angle of at most α with u. Note that if μ is the boundary measure of $\mathbb{S}^{d-1}$, then simple calculus shows that $\mu(B(\alpha, u))/\mu(\mathbb{S}^{d-1}) = \Theta_d(\alpha^{(d-1)})$.

We will now construct our set of unit vectors iteratively in the following way. Assume our set already includes the unit vectors $u_1, u_2, \ldots, u_n$. Now choose u_{n+1} to be any point in $S_{n+1} = \mathbb{S}^{d-1} \setminus \bigcup_{i=1}^n B(\alpha, u_i)$. Note that any point in S_{n+1} forms an angle of at least α with all of the u_i. In addition, as long as $n\mu(B(\alpha, u_i)) \leq \mu(\mathbb{S}^{d-1})$, S_{n+1} is guaranteed to be nonempty. Therefore we can continue this procedure until $n \geq \mu(\mathbb{S}^{d-1})/\mu(B(\alpha, u_i)) = \Theta_d(\alpha^{-(d-1)})$. The result follows.

□