Stochastic convex optimization with bandit feedback

Alekh Agarwal
Department of EECS
UC Berkeley
alekh@cs.berkeley.edu

Dean P. Foster
Department of Statistics
University of Pennsylvania
dean.foster@gmail.com

Daniel Hsu
Microsoft Research
New England
dahsu@microsoft.com

Sham M. Kakade
Department of Statistics, University of Pennsylvania
Microsoft Research, New England
skakade@microsoft.com

Alexander Rakhlin
Department of Statistics
University of Pennsylvania
rakhlin@wharton.upenn.edu

Abstract

This paper addresses the problem of minimizing a convex, Lipschitz function f over a convex, compact set $\mathcal{X}$ under a stochastic bandit feedback model. In this model, the algorithm is allowed to observe noisy realizations of the function value $f(x)$ at any query point $x \in \mathcal{X}$. We demonstrate a generalization of the ellipsoid algorithm that incurs $\widetilde{\mathcal{O}}(\text{poly}(d)\sqrt{T})$ regret. Since any algorithm has regret at least $\Omega(\sqrt{T})$ on this problem, our algorithm is optimal in terms of the scaling with T.

1 Introduction

This paper considers the problem of stochastic convex optimization under bandit feedback which is a generalization of the classical multi-armed bandit problem, formulated by Robbins in 1952. Our problem is specified by a mean cost function f which is assumed to be convex and Lipschitz, and a convex, compact domain $\mathcal{X}$. The algorithm repeatedly queries f at points $x \in \mathcal{X}$ and observes noisy realizations of $f(x)$. Performance of an algorithm is measured by *regret*, that is the difference between values of f at the query points and the minimum value of f over $\mathcal{X}$. This specializes to the classical K-armed setting when $\mathcal{X}$ is the probability simplex and f is linear. Several recent works consider the continuum-armed bandit problem, making different assumptions on the *structure* of f over $\mathcal{X}$. For instance, the f is assumed to be linear in the paper [9], a Lipschitz condition on f is assumed in the works [3, 12, 13], and Srinivas et al. [16] exploit the structure of a Gaussian processes. For these "non-parametric" bandit problems, the rates of regret (after T queries) are of the form T^{α}, with exponent α approaching 1 for large dimension d.

The question addressed in the present paper is: How can we leverage *convexity* of the mean cost function as a structural assumption? Our main contribution is an algorithm which achieves, with high probability, an $\widetilde{\mathcal{O}}(\text{poly}(d)\sqrt{T})$ regret after T requests. This result holds for all convex Lipschitz mean cost functions. We observe that the rate with respect to T does not deteriorate with d unlike the non-parametric problems mentioned earlier. Let us also remark that $\Omega(\sqrt{dT})$ lower bounds have been shown for linear mean cost functions, making our algorithm optimal up to factors polynomial in d and logarithmic in T.

Prior Work *Asymptotic* rates of $\sqrt{T}$ have been previously achieved by Cope [8] for unimodal functions under stringent conditions (smoothness and strong convexity of the mean cost function, in addition to the maxima being achieved inside the set). Auer et al. [4] show a regret of $\tilde{\mathcal{O}}(\sqrt{T})$ for a one-dimensional non-convex problem with finite number of maximizers. Yu and Mannor [17] recently studied unimodal bandits in one dimension, but they do not consider higher dimensional cases. Bubeck et al. [7] show $\sqrt{T}$ regret for a subset of Lipschitz functions with certain metric properties. Convex, Lipschitz cost functions have also been looked at in the adversarial model [10, 12], but the best-known regret bounds for these algorithms are $\mathcal{O}(T^{3/4})$. We also note that previous results of Agarwal et al. [1] and Nesterov [15] do not apply to our setting as noted in the full-length version of this paper [2].

The problem addressed in this paper is closely related to noisy zeroth order convex optimization, whereby the algorithm queries a point of the domain $\mathcal{X}$ and receives a noisy evaluation of the function. While the literature on stochastic optimization is vast, we emphasize that an optimization guarantee does not necessarily imply a bound on regret. In particular, we directly build on an optimization method that has been developed by Nemirovski and Yudin [14, Chapter 9]. Given $\epsilon > 0$, the method is guaranteed to produce an ϵ-minimizer in $\widetilde{\mathcal{O}}(\text{poly}(d)\epsilon^{-2})$ iterations, yet this does not immediately imply small regret. The latter is the quantity of interest in this paper, since it is the standard performance measure in decision theory. More importantly, in many applications every query to the function involves a consumption of resources or a monetary cost. A low regret guarantees that the net cost over the entire process is bounded unlike an optimization error bound.

The remainder of this paper is organized as follows. In the next section, we give the formal problem setup and highlight differences between the regret and optimization error settings. We then present a simple algorithm and its analysis for 1-dimension that illustrates some of the key insights behind the general d-dimensional algorithm in Section 3. Section 4 describes our generalization of the ellipsoid algorithm for d dimensions along with its regret guarantee. Proofs of our results can be found in the full-length version [2].

2 Setup

In this section we will give the basic setup and the performance criterion, and explain the differences between the metrics of regret and optimization error.

2.1 Problem definition and notation

Let $\mathcal{X}$ be a compact and convex subset of $\mathbb{R}^d$, and let $f\colon \mathcal{X} \to \mathbb{R}$ be a 1-Lipschitz convex function on $\mathcal{X}$, so $f(x) - f(x') \le \|x - x'\|$ for all $x, x' \in \mathcal{X}$. We assume $\mathcal{X}$ is specified in a way so that the algorithm can efficiently construct the smallest Euclidian ball containing $\mathcal{X}$. Furthermore, we assume the algorithm has noisy black-box access to f. Specifically, the algorithm is allowed to query the value of f at any $x \in \mathcal{X}$, and it observes $y = f(x) + \varepsilon$ where ε is an independent σ-subgaussian random variable with mean zero: $\mathbb{E}[\exp(\lambda\varepsilon)] \le \exp(\lambda^2\sigma^2/2)$ for all $\lambda \in \mathbb{R}$. The goal of the algorithm is to minimize its *regret*: after making T queries $x_1, \ldots, x_T \in \mathcal{X}$, the regret of the algorithm compared to any $x^* \in \arg\min_{x\in\mathcal{X}} f(x)$ is

$$R_T = \textstyle\sum_{t=1}^{T}\big[f(x_t) - f(x^*)\big]. \tag{1}$$

We will construct an average and confidence interval (henceforth CI) for the function values at points queried by the algorithm. Letting $\text{LB}_{\gamma_i}(x)$ and $\text{UB}_{\gamma_i}(x)$ denote the lower and upper bounds of a CI of width γ_i for the function estimate of a point x, we will say that CI's at two points are γ-separated if $\text{LB}_{\gamma_i}(x) \ge \text{UB}_{\gamma_i}(y) + \gamma$ or $\text{LB}_{\gamma_i}(y) \ge \text{UB}_{\gamma_i}(x) + \gamma$.

2.2 Regret vs. optimization error

Since f is convex, the average $\bar{x}_T = \frac{1}{T}\sum_{t=1}^{T} x_t$ satisfies $f(\bar{x}_T) - f(x^*) \le R_T/T$ so that low regret (1) also gives a small optimization error. The converse, however, is not necessarily true. An optimization method might can query far from the minimum of the function (that is, *explore*) on most rounds, and output the solution at the last step. Guaranteeing a small regret typically involves a more careful balancing of *exploration* and *exploitation.*

To better understand the difference, suppose $\mathcal{X} = [0,1]$, and let $f(x)$ be one of $xT^{-1/3}, -xT^{-1/3}$ and $x(x-1)$. Let us sample function values at $x = 1/4$ and $x = 3/4$. To distinguish the first two cases, we need $\Omega(T^{2/3})$ points. If f is linear indeed, we only incur $\mathcal{O}(T^{1/3})$ regret on these rounds. However, if instead $f(x) = x(x-1)$, we incur an undesirable $\Omega(T^{2/3})$ regret. For purposes of optimization, it suffices to eventually distinguish the three cases. For the purposes of regret minimization, however, an algorithm has to detect that the function curves between the two sampled points. To address this issue, we additionally sample at $x = 1/2$. The center point acts as a *sentinel*: if it is recognized that $f(1/2)$ is noticeably below the other two values, the region $[0, 1/4]$ or $[3/4, 1]$ can be discarded. Similarly, one of these regions can be discarded if it is recognized that the value of f either at $x = 1/4$ or at $x = 3/4$ is greater than others. Finally, if f at all three points appears to be similar at a given scale, we have a certificate (due to convexity) that the algorithm is not paying regret per query larger than this scale.

This *center-point device* that allows to quickly detect that the optimization method might be paying high regret and to act on this information is the main novel tool of our paper. Unlike discretization-based methods, the proposed algorithm uses convexity in a crucial way. We first demonstrate the device on one-dimensional problems in the next section, where the solution is clean and intuitive. We then develop a version of the algorithm for higher dimensions, basing our construction on the beautiful zeroth order optimization method of Nemirovski and Yudin [14]. Their method does not guarantee vanishing regret by itself, and a careful fusion of this algorithm with our center-point device is required.

3 One-dimensional case

We start with a special case of 1-dimension to illustrate some of the key ideas including the center-point device. We assume wlog that the domain $\mathcal{X} = [0,1]$, and $f(x) \in [0,1]$ (the latter can be achieved by pinning $f(x^*) = 0$ since f is 1-Lipschitz).

3.1 Algorithm description

Algorithm 1 One-dimensional stochastic convex bandit algorithm

```
input noisy black-box access to $f\colon [0,1] \to \mathbb{R}$, total number of queries allowed $T$.
 1: Let $l_1 := 0$ and $r_1 := 1$.
 2: for epoch $\tau = 1, 2, \ldots$ do
 3:   Let $w_\tau := r_\tau - l_\tau$.
 4:   Let $x_l := l_\tau + w_\tau/4$, $x_c := l_\tau + w_\tau/2$, and $x_r := l_\tau + 3w_\tau/4$.
 5:   for round $i = 1, 2, \ldots$ do
 6:     Let $\gamma_i := 2^{-i}$.
 7:     For each $x \in \{x_l, x_c, x_r\}$, query $f(x)$ $\frac{2\sigma}{\gamma_i^2}\log T$ times.
 8:     if $\max\{\mathrm{LB}_{\gamma_i}(x_l), \mathrm{LB}_{\gamma_i}(x_r)\} \geq \min\{\mathrm{UB}_{\gamma_i}(x_l), \mathrm{UB}_{\gamma_i}(x_r)\} + \gamma_i$ then
 9:                          {Case 1: CI's at $x_l$ and $x_r$ are $\gamma_i$ separated}
10:       if $\mathrm{LB}_{\gamma_i}(x_l) \geq \mathrm{LB}_{\gamma_i}(x_r)$ then let $l_{\tau+1} := x_l$ and $r_{\tau+1} := r_\tau$.
11:       if $\mathrm{LB}_{\gamma_i}(x_l) < \mathrm{LB}_{\gamma_i}(x_r)$ then let $l_{\tau+1} := l_\tau$ and $r_{\tau+1} := x_r$.
12:       Continue to epoch $\tau + 1$.
13:     else if $\max\{\mathrm{LB}_{\gamma_i}(x_l), \mathrm{LB}_{\gamma_i}(x_r)\} \geq \mathrm{UB}_{\gamma_i}(x_c) + \gamma_i$ then
14:                  {Case 2: CI's at $x_c$ and $x_l$ or $x_r$ are $\gamma_i$ separated}
15:       if $\mathrm{LB}_{\gamma_i}(x_l) \geq \mathrm{LB}_{\gamma_i}(x_r)$ then let $l_{\tau+1} := x_l$ and $r_{\tau+1} := r_\tau$.
16:       if $\mathrm{LB}_{\gamma_i}(x_l) < \mathrm{LB}_{\gamma_i}(x_r)$ then let $l_{\tau+1} := l_\tau$ and $r_{\tau+1} := x_r$.
17:       Continue to epoch $\tau + 1$.
18:     end if
19:   end for
20: end for
```

Algorithm 1 proceeds in a series of *epochs* demarcated by a working feasible region (the interval $\mathcal{X}_\tau = [l_\tau, r_\tau]$ in epoch τ). In each epoch, the algorithm aims to discard a portion of $\mathcal{X}_\tau$ determined to only contain suboptimal points. To do this, the algorithm repeatedly makes noisy queries to f at three different points in $\mathcal{X}_\tau$. Each epoch is further subdivided into *rounds*, where we query the function $(2\sigma \log T)/\gamma_i^2$ times in round i at each of the points. By Hoeffding's inequality, this implies that we know the function value to within γ_i with high probability. The value γ_i is halved at every round. At the end of an epoch τ, $\mathcal{X}_\tau$ is reduced to a subset $\mathcal{X}_{\tau+1} = [l_{\tau+1}, r_{\tau+1}] \subset [l_\tau, r_\tau]$ of the current region for the next epoch $\tau + 1$, and this reduction is such that the new region is smaller in size by a constant fraction. This geometric rate of reduction guarantees that only a small number of epochs can occur before $\mathcal{X}_\tau$ only contains near-optimal points.

For the algorithm to identify a sizable portion of $\mathcal{X}_\tau$ to discard, the queries in each epoch should be suitably chosen, and the convexity of f must be exploited. To this end, the algorithm makes its queries at three equally-spaced points $x_l < x_c < x_r$ in $\mathcal{X}_\tau$ (see Section 4.1 of the full-length version for graphical illustrations of these cases).

Case 1: If the CIs around $f(x_l)$ and $f(x_r)$ are sufficiently separated, the algorithm discards a fourth of $[l_\tau, r_\tau]$ (to the left of x_l or right of x_r) which does not contain x^*.

Case 2: If the above separation fails, the algorithm checks if the CI around $f(x_c)$ is sufficiently below at least one of the other CIs (for $f(x_l)$ or $f(x_r)$). If that happens, the algorithm again discards a quartile of $[l_\tau, r_\tau]$ that does not contain x^*.

Case 3: Finally, if none of the earlier cases is true, then the algorithm is assured (by convexity) that the function is sufficiently flat on $\mathcal{X}_\tau$ and hence it has not incurred much regret so far . The algorithm continues the epoch, with an increased number of queries to obtain smaller confidence intervals at each of the three points.

3.2 Analysis

The analysis of Algorithm 1 relies on the function values being contained in the confidence intervals we construct at each round of each epoch. To avoid having probabilities throughout our analysis, we define an event $\mathcal{E}$ where at each epoch τ, and each round i, $f(x) \in [\mathrm{LB}_{\gamma_i}(x), \mathrm{UB}_{\gamma_i}(x)]$ for $x \in \{x_l, x_c, x_r\}$. We will carry out the remainder of the analysis conditioned on $\mathcal{E}$ and bound the probability of $\mathcal{E}^c$ at the end.

The following theorem bounds the regret incurred by Algorithm 1. We note that the regret would be maintained in terms of the points x_t queried by the algorithm at time t. Within any given round, the order of queries is immaterial to the regret.

Theorem 1 (Regret bound for Algorithm 1)**.** *Suppose Algorithm 1 is run on a convex, 1-Lipschitz function f bounded in [0,1]. Suppose the noise in observations is i.i.d. and σ-subGaussian. Then with probability at least $1 - 1/T$ we have*

$$\sum_{t=1}^{T} f(x_t) - f(x^*) \leq 108\sqrt{\sigma T \log T} \log_{4/3}\left(\frac{T}{8\sigma \log T}\right).$$

Remarks: As stated Algorithm 1 and Theorem 1 assume knowledge of T, but we can make the algorithm adaptive to T by a standard doubling argument. We remark that $\mathcal{O}(\sqrt{T})$ is the smallest possible regret for any algorithm even with noisy gradient information. Hence, this result shows that for purposes of regret, noisy zeroth order information is no worse than noisy first-order information apart from logarithmic factors.

Theorem 1 is proved via a series of lemmas below. The key idea is to show that the regret on any epoch is small and the total number of epochs is bounded. To bound the per-epoch regret, we will show that the total number of queries made on any epoch depends on how flat the function is on $\mathcal{X}_\tau$. We either take a long time, but the function is very flat, or we stop early when the function has sufficient slope, never accruing too much regret. We start by showing that the reduction in $\mathcal{X}_\tau$ after each epoch always preserves near-optimal points.

Lemma 1 (Survival of approx. minima)**.** *If epoch τ ends in round i, then $[l_{\tau+1}, r_{\tau+1}]$ contains every $x \in [l_\tau, r_\tau]$ such that $f(x) \leq f(x^*) + \gamma_i$. In particular, $x^* \in [l_\tau, r_\tau]$ for all τ.*

The next two lemmas bound the regret incurred in any single epoch. To show this, we first establish that an algorithm incurs low regret in a round as long as it does not end an epoch. Then, as a consequence of the doubling trick, we show that the regret incurred in an epoch is on the same order as that incurred in the last round of the epoch.

Lemma 2 (Certificate of low regret)**.** *If epoch τ continues from round i to round $i+1$, then the regret incurred in round i is at most* $72\gamma_i^{-1}\sigma\log T$.

Lemma 3 (Regret in an epoch)**.** *If epoch τ ends in round i, then the regret incurred in the entire epoch is at most* $216\gamma_i^{-1}\sigma\log T$.

To obtain a bound on the overall regret, we bound the number of epochs that can occur before $\mathcal{X}_\tau$ only contains near-optimal points. The final regret bound is simply the product of the number of epochs and the regret incurred in any single epoch.

Lemma 4 (Bound on the number of epochs)**.** *The total number of epochs τ performed by Algorithm 1 is bounded as* $\tau \le \frac{1}{2}\log_{4/3}\left(\frac{T}{8\sigma\log T}\right)$.

4 Algorithm for optimization in higher dimensions

We now move to present the general algorithm that works in d-dimensions. The natural approach would be to try and generalize Algorithm 1 to work in multiple dimensions. However, the obvious extension requires querying the function along every direction in a covering of the unit sphere so that we know the behavior of the function along every direction. Such an approach yields regret and running time that scales exponentially with the dimension d. Nemirovski and Yudin [14] address this problem in the setup of zeroth order optimization by a clever construction to capture all the directions in polynomially many queries. We define a pyramid to be a d-dimensional polyhedron defined by $d+1$ points; d points form a d-dimensional regular polygon that is the base of the pyramid, and the apex lies above the hyperplane containing the base (see Figure 1 for a graphic illustration in 3 dimensions). The idea of Nemirovski and Yudin is to build a sequence of pyramids, each capturing the variation of function in certain directions, in such a way that with $\mathcal{O}(d\log d)$ pyramids we can explore all the directions. However, as mentioned earlier, their approach fails to give a low regret. We combine their geometric construction with ideas from the one-dimensional case to obtain Algorithm 2 which incurs a bounded regret.

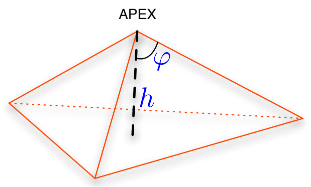

Figure 1: Pyramid in 3-dimensions

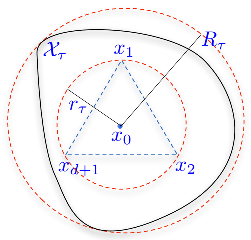

Figure 2: The regular simplex constructed at round i of epoch τ with radius r_τ, center x_0 and vertices $x_1,\ldots,x_{d+1}$.

Just like the 1-dimensional case, Algorithm 2 proceeds in *epochs*. We start with the optimization domain $\mathcal{X}$, and at the beginning we set $\mathcal{X}_0 = \mathcal{X}$. At the beginning of epoch τ, we have a current feasible set $\mathcal{X}_\tau$ which contains at least one approximate optimum of the convex function. The epoch ends with discarding some portion of the set $\mathcal{X}_\tau$ in such a way that we still retain at least one approximate optimum in the remaining set $\mathcal{X}_{\tau+1}$.

At the start of the epoch τ, we apply an affine transformation to $\mathcal{X}_\tau$ so that the smallest volume ellipsoid containing it is a Euclidian ball of radius R_τ (denoted as $\mathbb{B}(R_\tau)$). We define $r_\tau = R_\tau/c_1 d$ for a constant $c_1 \ge 1$, so that $\mathbb{B}(r_\tau) \subseteq \mathcal{X}_\tau$ (see e.g. Lecture 1, p. 2 [5]). We will use the notation $\mathcal{B}_\tau$ to refer to the enclosing ball. Within each epoch, the algorithm proceeds in several rounds, each round maintaining a value γ_i which is successively halved.

Algorithm 2 Stochastic convex bandit algorithm

```
input feasible region $\mathcal{X} \subset \mathbb{R}^d$; noisy black-box access to $f\colon \mathcal{X} \to \mathbb{R}$, constants $c_1$ and $c_2$, functions
    $\Delta_\tau(\gamma)$, $\overline{\Delta}_\tau(\gamma)$ and number of queries $T$ allowed.
 1: Let $\mathcal{X}_1 := \mathcal{X}$.
 2: for epoch $\tau = 1, 2, \ldots$ do
 3:   Round $\mathcal{X}_\tau$ so $\mathbb{B}(r_\tau) \subseteq \mathcal{X}_\tau \subseteq \mathbb{B}(R_\tau)$, $R_\tau$ is minimized, and $r_\tau := R_\tau/(c_1 d)$. Let $\mathcal{B}_\tau = \mathbb{B}(R_\tau)$.
 4:   Construct regular simplex with vertices $x_1, \ldots, x_{d+1}$ on the surface of $\mathbb{B}(r_\tau)$.
 5:   for round $i = 1, 2, \ldots$ do
 6:     Let $\gamma_i := 2^{-i}$.
 7:     Query $f$ at $x_j$ for each $j = 1, \ldots, d+1$ $\frac{2\sigma \log T}{\gamma_i^2}$ times.
 8:     Let $y_1 := \arg\max_j \mathrm{LB}_{\gamma_i}(x_j)$.
 9:     for $k = 1, 2, \ldots$ do
10:       Construct pyramid $\Pi_k$ with apex $y_k$; let $z_1, \ldots, z_d$ be the vertices of the base of $\Pi_k$ and
          $z_0$ be the center of $\Pi_k$.
11:       Let $\widehat{\gamma} := 2^{-1}$.
12:       loop
13:         Query $f$ at each of $\{y_k, z_0, z_1, \ldots, z_d\}$ $\frac{2\sigma \log T}{\widehat{\gamma}^2}$ times.
14:         Let CENTER := $z_0$, APEX := $y_k$, TOP be the vertex $v$ of $\Pi_k$ maximizing $\mathrm{LB}_{\widehat{\gamma}}(v)$,
            BOTTOM be the vertex $v$ of $\Pi_k$ minimizing $\mathrm{LB}_{\widehat{\gamma}}(v)$.
15:         if $\mathrm{LB}_{\widehat{\gamma}}(\text{TOP}) \geq \mathrm{UB}_{\widehat{\gamma}}(\text{BOTTOM}) + \Delta_\tau(\widehat{\gamma})$ and $\mathrm{LB}_{\widehat{\gamma}}(\text{TOP}) \geq \mathrm{UB}_{\widehat{\gamma}}(\text{APEX}) + \widehat{\gamma}$ then
16:           {Case (1a)}
17:           Let $y_{k+1} :=$ TOP, and immediately continue to pyramid $k+1$.
18:         else if $\mathrm{LB}_{\widehat{\gamma}}(\text{TOP}) \geq \mathrm{UB}_{\widehat{\gamma}}(\text{BOTTOM}) + \Delta_\tau(\widehat{\gamma})$ and $\mathrm{LB}_{\widehat{\gamma}}(\text{TOP}) < \mathrm{UB}_{\widehat{\gamma}}(\text{APEX}) + \widehat{\gamma}$ then
19:           {Case (1b)}
20:           Set $(\mathcal{X}_{\tau+1}, \mathcal{B}'_{\tau+1}) =$ CONE-CUTTING$(\Pi_k, \mathcal{X}_\tau, \mathcal{B}_\tau)$, and proceed to epoch $\tau+1$.
21:         else if $\mathrm{LB}_{\widehat{\gamma}}(\text{TOP}) < \mathrm{UB}_{\widehat{\gamma}}(\text{BOTTOM}) + \Delta_\tau(\widehat{\gamma})$ and $\mathrm{UB}_{\widehat{\gamma}}(\text{CENTER}) \geq \mathrm{LB}_{\widehat{\gamma}}(\text{BOTTOM}) -$
            $\overline{\Delta}_\tau(\widehat{\gamma})$ then
22:           {Case (2a)}
23:           Let $\widehat{\gamma} := \widehat{\gamma}/2$.
24:           if $\widehat{\gamma} < \gamma_i$ then start next round $i+1$.
25:         else if $\mathrm{LB}_{\widehat{\gamma}}(\text{TOP}) < \mathrm{UB}_{\widehat{\gamma}}(\text{BOTTOM}) + \Delta_\tau(\widehat{\gamma})$ and $\mathrm{UB}_{\widehat{\gamma}}(\text{CENTER}) < \mathrm{LB}_{\widehat{\gamma}}(\text{BOTTOM}) -$
            $\overline{\Delta}_\tau(\widehat{\gamma})$ then
26:           {Case (2b)}
27:           Set $(\mathcal{X}_{\tau+1}, \mathcal{B}'_{\tau+1})=$ HAT-RAISING$(\Pi_k, \mathcal{X}_\tau, \mathcal{B}_\tau)$, and proceed to epoch $\tau+1$.
28:         end if
29:       end loop
30:     end for
31:   end for
32: end for
```

Algorithm 3 Cone-cutting

```
input pyramid $\Pi$ with apex $y$, (rounded) feasible region $\mathcal{X}_\tau$ for epoch $\tau$, enclosing ball $\mathcal{B}_\tau$
 1: Let $z_1, \ldots, z_d$ be the vertices of the base of $\Pi$, and $\bar{\varphi}$ the angle at its apex.
 2: Define the cone
      $$\mathcal{K}_\tau = \{x \mid \exists \lambda > 0, \alpha_1, \ldots, \alpha_d > 0, \sum_{i=1}^d \alpha_i = 1 \ : \ x = y - \lambda \sum_{i=1}^d \alpha_i (z_i - y)\}$$
 3: Set $\mathcal{B}'_{\tau+1}$ to be the min. volume ellipsoid containing $\mathcal{B}_\tau \setminus \mathcal{K}_\tau$.
 4: Set $\mathcal{X}_{\tau+1} = \mathcal{X}_\tau \cap \mathcal{B}'_{\tau+1}$.
output new feasible region $\mathcal{X}_{\tau+1}$ and enclosing ellipsoid $\mathcal{B}'_{\tau+1}$.
```

Algorithm 4 Hat-raising

```
input pyramid $\Pi$ with apex $y$, (rounded) feasible region $\mathcal{X}_\tau$ for epoch $\tau$, enclosing ball $\mathcal{B}_\tau$.
 1: Let CENTER be the center of $\Pi$.
 2: Set $y' = y + (y -$ CENTER$)$.
 3: Set $\Pi'$ to be the pyramid with apex $y'$ and same base as $\Pi$.
 4: Set $\mathcal{X}_{\tau+1}, \mathcal{B}'_{\tau+1} =$ CONE-CUTTING$(\Pi', \mathcal{X}_\tau, \mathcal{B}_\tau)$.
output new feasible region $\mathcal{X}_{\tau+1}$ and enclosing ellipsoid $\mathcal{B}'_{\tau+1}$.
```

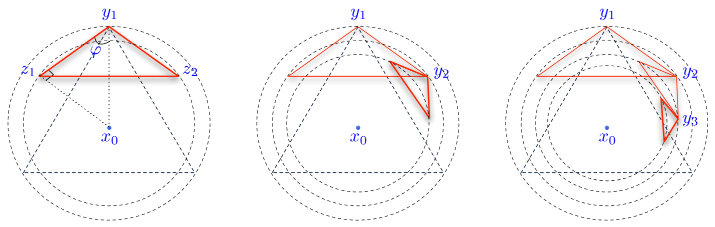

Figure 3: Sequence of pyramids constructed by Algorithm 2

Let x_0 be the center of the ball $\mathbb{B}(R_\tau)$ containing $\mathcal{X}_\tau$. At the start of a round i, we construct a regular simplex centered at x_0 and contained in $\mathbb{B}(r_\tau)$. The algorithm queries the function f at all the vertices of the simplex, denoted by $x_1.\dots, x_{d+1}$, until the CI's at each vertex shrink to γ_i. The algorithm picks the point y_1 that maximizes $\mathrm{LB}_{\gamma_i}(x_i)$. By construction, $f(y_1) \geq f(x_j) - \gamma_i$ for all $j = 1, \dots, d+1$. This step is depicted in Figure 2.

The algorithm now successively constructs a sequence of pyramids, with the goal of identifying a region of the feasible set $\mathcal{X}_\tau$ such that at least one approximate optimum of f lies outside the selected region. This region will be discarded at the end of the epoch. The construction of the pyramids follows the construction from Section 9.2.2 of Nemirovski and Yudin [14]. The pyramids we construct will have an angle 2φ at the apex, where $\cos\varphi = c_2/d$. The base of the pyramid consists of vertices $z_1, \dots, z_d$ such that $z_i - x_0$ and $y_1 - z_i$ are orthogonal. We note that the construction of such a pyramid is always possible—we take a sphere with $y_1 - x_0$ as the diameter, and arrange $z_1, \dots, z_d$ on the boundary of the sphere such that the angle between $y_1 - x_0$ and $y_1 - z_i$ is φ. The construction of the pyramid is depicted in Figure 3. Given this pyramid, we set $\widehat{\gamma} = 1$, and sample the function at y_1 and $z_1, \dots, z_d$ as well as the center of the pyramid until the CI's all shrink to $\widehat{\gamma}$. Let TOP and BOTTOM denote the vertices of the pyramid (including y_1) with the largest and the smallest function value estimates resp. For consistency, we will also use APEX to denote the apex y_1. We then check for one of the following conditions (see Section 5 of the full-length version [2] for graphical illustrations of these cases):

(1) If $\mathrm{LB}_{\widehat{\gamma}}(\text{TOP}) \geq \mathrm{UB}_{\widehat{\gamma}}(\text{BOTTOM}) + \Delta_\tau(\widehat{\gamma})$, we proceed based on the separation between TOP and APEX CI's.

  (a) If $\mathrm{LB}_{\widehat{\gamma}}(\text{TOP}) \geq \mathrm{UB}_{\widehat{\gamma}}(\text{APEX}) + \widehat{\gamma}$, then we know that with high probability

$$f(\text{TOP}) \geq f(\text{APEX}) + \widehat{\gamma} \geq f(\text{APEX}) + \gamma_i. \tag{2}$$

  In this case, we set TOP to be the apex of the next pyramid, reset $\widehat{\gamma} = 1$ and continue the sampling procedure on the next pyramid.

  (b) If $\mathrm{LB}_{\widehat{\gamma}}(\text{TOP}) \leq \mathrm{UB}_{\widehat{\gamma}}(\text{APEX}) + \widehat{\gamma}$, then we know that $\mathrm{LB}_{\widehat{\gamma}}(\text{APEX}) \geq \mathrm{UB}_{\widehat{\gamma}}(\text{BOTTOM}) + \Delta_\tau(\widehat{\gamma}) - 2\widehat{\gamma}$. In this case, we declare the epoch over and pass the current apex to the cone-cutting step.

(2) If $\mathrm{LB}_{\widehat{\gamma}}(\text{TOP}) \leq \mathrm{UB}_{\widehat{\gamma}}(\text{BOTTOM}) + \Delta_\tau(\widehat{\gamma})$, then one of the following happens:

  (a) If $\mathrm{UB}_{\widehat{\gamma}}(\text{CENTER}) \geq \mathrm{LB}_{\widehat{\gamma}}(\text{BOTTOM}) - \overline{\Delta}_\tau(\widehat{\gamma})$, then all of the vertices and the center of the pyramid have their function values within a $2\Delta_\tau(\widehat{\gamma}) + 3\widehat{\gamma}$ interval. In this case, we set $\widehat{\gamma} = \widehat{\gamma}/2$. If this sets $\widehat{\gamma} < \gamma_i$, we start the next round with $\gamma_{i+1} = \gamma_i/2$. Otherwise, we continue sampling the current pyramid with the new value of $\widehat{\gamma}$.

  (b) If $\mathrm{UB}_{\widehat{\gamma}}(\text{CENTER}) \leq \mathrm{LB}_{\widehat{\gamma}}(\text{BOTTOM}) - \overline{\Delta}_\tau(\widehat{\gamma})$, then we terminate the epoch and pass the center and the current apex to the hat-raising step.

Hat-Raising: This step happens when the algorithm enters case 2(b). In this case, we will show that if we move the apex of the pyramid a little from y_i to $y_i^{'}$, then $y_i^{'}$'s CI is above the TOP CI while the angle of the new pyramid at $y_i^{'}$ is not much smaller than φ. Letting CENTER_i denote the center of the pyramid, we set $y_i^{'} = y_i + (y_i - \text{CENTER}_i)$ and denote the angle at the apex $y_i^{'}$ by $2\bar{\varphi}$. Figure 4 shows the transformation involved in this step.

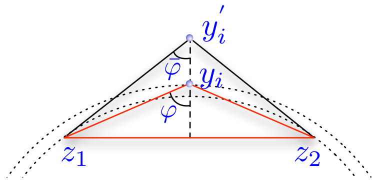

Figure 4: Transformation of the pyramid Π in the hat-raising step.

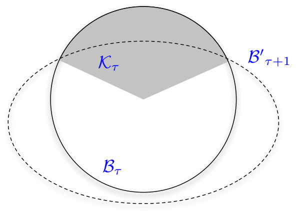

Figure 5: Cone-cutting step at epoch τ. Solid circle is the enclosing ball $\mathcal{B}_\tau$. Shaded region is the intersection of $\mathcal{K}_\tau$ with $\mathcal{B}_\tau$. The dotted ellipsoid is the new enclosing ellipsoid $\mathcal{B}'_{\tau+1}$.

Cone-cutting: This step concludes an epoch. The algorithm gets here either through case 1(b) or through the hat-raising step. In either case, we have a pyramid with an apex y, base $z_1, \ldots, z_d$ and an angle $2\bar{\varphi}$ at the apex, where $\cos(\bar{\varphi}) \leq 2c_2/d$. We now define a cone

$$\mathcal{K}_\tau = \{x \mid \exists \lambda > 0, \alpha_1, \ldots, \alpha_d > 0, \sum_{i=1}^{d} \alpha_i = 1 \ : \ x = y - \lambda \sum_{i=1}^{d} \alpha_i (z_i - y)\} \tag{3}$$

which is centered at y and a reflection of the pyramid around the apex. By construction, the cone $\mathcal{K}_\tau$ has an angle $2\bar{\varphi}$ at its apex. We set $\mathcal{B}'_{\tau+1}$ to be the ellipsoid of minimum volume containing $\mathcal{B}_\tau \setminus \mathcal{K}_\tau$ and define $\mathcal{X}_{\tau+1} = \mathcal{X}_\tau \cap \mathcal{B}'_{\tau+1}$. This is illustrated in Figure 5. Finally, we put things back into an isotropic position and $\mathcal{B}_{\tau+1}$ is the ball containing $\mathcal{X}_{\tau+1}$ is in the isotropic coordinates, which is just obtained by applying an affine transformation to $\mathcal{B}'_{\tau+1}$.

Let us end with a brief discussion regarding the computational aspects of this algorithm. Clearly, the most computationally intensive steps of this algorithm are cone-cutting and the isotropic transformation at the end. However, these are exactly analogous to the classical ellipsoid method. In particular, the equation for $\mathcal{B}'_{\tau+1}$ is known in closed form [11]. Furthermore, the affine transformations needed to the reshape the set can be computed via rank-one matrix updates and hence computation of inverses can be done efficiently as well (see e.g. [11] for the relevant implementation details of the ellipsoid method).

The following theorem states our regret guarantee on the performance of Algorithm 2.

Theorem 2. *Suppose Algorithm 2 is run with $c_1 \geq 64$, $c_2 \leq 1/32$ and parameters*

$$\Delta_\tau(\gamma) = \left(\frac{6c_1 d^4}{c_2^2} + 3\right)\gamma \quad \textit{and} \quad \overline{\Delta}_\tau(\gamma) = \left(\frac{6c_1 d^4}{c_2^2} + 5\right)\gamma.$$

Then with probability at least $1 - 1/T$, the regret incurred by the algorithm is bounded by

$$768 d^3 \sigma \sqrt{T} \log^2 T \left(\frac{2d^2 \log d}{c_2^2} + 1\right)\left(\frac{4d^7 c_1}{c_2^3} + \frac{d(d+1)}{c_2}\right)\left(\frac{12 c_1 d^4}{c_2^2} + 11\right) = \tilde{\mathcal{O}}(d^{16}\sqrt{T}).$$

Remarks: Theorem 2 is again optimal in the dependence on T. The large dependence on d is also seen in Nemirovski and Yudin [14] who obtain a d^7 scaling in noiseless case and leave it an unspecified polynomial in the noisy case. Using random walk ideas [6] to improve the dependence on d is an interesting question for future research.

Acknowledgments

Part of this work was done while AA and DH were at the University of Pennsylvania. AA was partially supported by MSR and Google PhD fellowships and NSF grant CCF-1115788 while this work was done. DH was partially supported under grants AFOSR FA9550-09-1-0425, NSF IIS-1016061, and NSF IIS-713540. AR gratefully acknowledges the support of NSF under grant CAREER DMS-0954737.

References

[1] A. Agarwal, O. Dekel, and L. Xiao. Optimal algorithms for online convex optimization with multi-point bandit feedback. In *COLT*, 2010.

[2] A. Agarwal, D. Foster, D. Hsu, S. Kakade, and A. Rakhlin. Stochastic convex optimization with bandit feedback. URL http://arxiv.org/abs/1107.1744, 2011.

[3] R. Agrawal. The continuum-armed bandit problem. *SIAM journal on control and optimization*, 33:1926, 1995.

[4] P. Auer, R. Ortner, and C. Szepesvári. Improved rates for the stochastic continuum-armed bandit problem. *Learning Theory*, pages 454–468, 2007.

[5] K. Ball. An elementary introduction to modern convex geometry. In *Flavors of Geometry*, number 31 in Publications of the Mathematical Sciences Research Institute, pages 1–55. 1997.

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

[7] S. Bubeck, R. Munos, G. Stolz, and C. Szepesvári. $\mathcal{X}$-armed bandits. *Journal of Machine Learning Research*, 12:1655–1695, 2011.

[8] E.W. Cope. Regret and convergence bounds for a class of continuum-armed bandit problems. *Automatic Control, IEEE Transactions on*, 54(6):1243–1253, 2009.

[9] V. Dani, T.P. Hayes, and S.M. Kakade. Stochastic linear optimization under bandit feedback. In *Proceedings of the 21st Annual Conference on Learning Theory (COLT)*, 2008.

[10] A. D. Flaxman, A. T. Kalai, and B. H. Mcmahan. Online convex optimization in the bandit setting: gradient descent without a gradient. In *Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 385–394, 2005.

[11] Donald Goldfarb and Michael J. Todd. Modifications and implementation of the ellipsoid algorithm for linear programming. *Mathematical Programming*, 23:1–19, 1982.

[12] R. Kleinberg. Nearly tight bounds for the continuum-armed bandit problem. *Advances in Neural Information Processing Systems*, 18, 2005.

[13] R. Kleinberg, A. Slivkins, and E. Upfal. Multi-armed bandits in metric spaces. In *Proceedings of the 40th annual ACM symposium on Theory of computing*, pages 681–690. ACM, 2008.

[14] A. Nemirovski and D. Yudin. *Problem Complexity and Method Efficiency in Optimization.* Wiley, New York, 1983.

[15] Y. Nesterov. Random gradient-free minimization of convex functions. Technical Report 2011/1, CORE DP, 2011.

[16] N. Srinivas, A. Krause, S.M. Kakade, and M. Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. *Arxiv preprint arXiv:0912.3995*, 2009.

[17] J. Y. Yu and S. Mannor. Unimodal bandits. In *ICML*, 2011.