Smooth Convex Optimization Using Sub-Zeroth-Order Oracles

Mustafa O. Karabag, Cyrus Neary, Ufuk Topcu
The University of Texas at Austin, Austin, TX
{karabag, cneary, utopcu}@utexas.edu

Abstract

We consider the problem of minimizing a smooth, Lipschitz, convex function over a compact, convex set using sub-zeroth-order oracles: an oracle that outputs the sign of the directional derivative for a given point and a given direction, an oracle that compares the function values for a given pair of points, and an oracle that outputs a noisy function value for a given point. We show that the sample complexity of optimization using these oracles is polynomial in the relevant parameters. The optimization algorithm that we provide for the comparator oracle is the first algorithm with a known rate of convergence that is polynomial in the number of dimensions. We also give an algorithm for the noisy-value oracle that incurs sublinear regret in the number of queries and polynomial regret in the number of dimensions.

Introduction

Derivative-free optimization methods are necessary when explicit access to the objective function is not available, or when the function's gradient is hard to compute (Conn, Scheinberg, and Vicente 2009). Utility functions, a concept from economics, provide an example of a type of objective function which may be hard to explicitly characterize. However, while a consumer may not be able to quantify their utility for a given prospect, they will likely be able to rank the available prospects. From a human's perspective, ranking the prospects may be simple, even if it is difficult to directly assign them values (Abbas and Howard 2015). For example, consider a reinforcement learning scenario in which a robot learns to perform a task via human feedback. The human may not be able to assign explicit rewards to the demonstrations performed by the robot, but she can rank them (Akrour, Schoenauer, and Sebag 2012; Fürnkranz et al. 2012; Wilson, Fern, and Tadepalli 2012). While necessary in a range of applications (Conn, Scheinberg, and Vicente 2009; Audet and Hare 2017), derivative-free optimization methods are usually inferior in theory compared to first-order optimization methods (Conn, Scheinberg, and Vicente 2009). In this work, we leverage the smoothness and convexity of the objective function to provide theoretical guarantees for derivative-free optimization.

We consider the problem of minimizing a smooth, Lipschitz continuous, convex function f on a convex, compact domain $C \subset \mathbb{R}^n$ using sub-zeroth-order oracles: i) the *directional-preference oracle* that outputs the sign of the directional derivative for a given point and direction, ii) the *comparator oracle* that compares the function value for two given points, and iii) the *noisy-value oracle* that outputs the function value plus a subgaussian noise.

For the directional-preference and comparator oracles, we prove an upper bound on the sample complexity that is polynomial in the relevant parameters. Our algorithms take advantage of the convexity and smoothness of the objective function, and rely on gradient estimation. We show that the direction of the gradient can be estimated with high accuracy via the sub-zeroth-order oracles. Having estimated the direction of the gradient, we use a variant of the ellipsoid method (Shor 1972; Yudin and Nemirovskii 1976). We show that the sample complexity is $\tilde{\mathcal{O}}(n^4)$ for the directional-preference and comparator oracles. To the best of our knowledge, the optimization algorithm that we provide for the comparator oracle is the first algorithm with a known polynomial rate of convergence for smooth, convex functions.

We also develop a sublinear regret algorithm for the noisy-value oracle. The algorithm incurs $\tilde{\mathcal{O}}(n^{3.75}T^{0.75})$ regret (ignoring the other factors) with high probability where T is the number of queries. The best known high probability regret bound for the noisy-value oracle is $\tilde{\mathcal{O}}(n^{9.5}\sqrt{T})$ (Bubeck, Lee, and Eldan 2017). While our algorithm requires smoothness, and its regret is not optimal in terms of the dependency on the number of queries, its lower order dependency on the number of dimensions makes it appealing compared to this existing regret bound.

Related Work The bisection method (Burden and Faires 1985) uses the directional-preference oracle to optimize a one-dimensional function. In multiple dimensions, Qian et al. (2015) used the directional-preference oracle to optimize a linear function. Their algorithm uses a predefined set of query directions, whereas we consider a setting where the algorithm is allowed to query any direction at any point. SignSGD (Bernstein et al. 2018) requires the sign of directional derivatives only for fixed orthogonal basis vectors and converges to the optimum for smooth, convex functions.

SignSGD enjoys lower order dependency $\mathcal{O}(n)$ on the number of dimensions. However, it has a sublinear rate of convergence whereas our algorithm has a linear rate of convergence. Additionally, our algorithm for the directional-preference oracle also works for non-smooth functions.

Optimization using the comparator oracle was explored with directional direct search methods (Audet and Dennis Jr 2006) and the Nelson-Mead method (Nelder and Mead 1965). Directional direct search is guaranteed to converge to an optimal solution in the limit for smooth, convex functions. However, the algorithm does not have a known rate of convergence. Meanwhile, the Nelson-Mead method may fail to converge to a stationary point for smooth, convex functions (McKinnon 1998). Convergent variants of the Nelson-Mead method use function values in addition to comparator oracle queries (Price, Coope, and Byatt 2002).

For the regret using the noisy-value oracle, a lower bound of $\Omega(n\sqrt{T})$ has been shown (Shamir 2013). Recently, Lattimore (2020) gave an existence result for an algorithm that achieves $\tilde{\mathcal{O}}(n^{2.5}\sqrt{T})$ regret in the adversarial case. The best known upper bounds with explicit algorithms are $\tilde{\mathcal{O}}(n^{9.5}\sqrt{T})$ (Bubeck, Lee, and Eldan 2017) and $\mathcal{O}(nT^{0.75})$ (Flaxman, Kalai, and McMahan 2005) for Lipschitz, convex functions in the adversarial case. The regret bound $\mathcal{O}(n^{3.75}T^{0.75})$ that we provide is better than $\tilde{\mathcal{O}}(n^{9.5}\sqrt{T})$ regret bound of (Bubeck, Lee, and Eldan 2017) if $T = o(n^{23})$. Our result differs from (Flaxman, Kalai, and McMahan 2005) in that our algorithm succeeds with high probability whereas the algorithm given in (Flaxman, Kalai, and McMahan 2005) succeeds in expectation.

We give the the proofs of the results in the extended version of this paper (Karabag, Neary, and Topcu 2021).

Preliminaries

We denote the unit vectors in $\mathbb{R}^n$ as $e_1, \ldots, e_n$. Let S be a set of vectors in $\mathbb{R}^n$. $Proj_S(x)$ denotes the orthogonal projection of x onto the span of S and $Proj_{S^\perp}(x)$ denotes the orthogonal projection of x onto the complement space of the span of S. The angle between x and y is $\angle(x, y)$.

A convex function $f : C \to \mathbb{R}$ is said to be *L-Lipschitz* if $\|f(x) - f(y)\| \leq L \|x - y\|$ for all $x, y \in C$. A differentiable convex function $f : C \to \mathbb{R}$ is said to be *β-strongly smooth* if $|f(y) - f(x) - \langle \nabla f(x), y - x \rangle| \leq \beta \|y - x\|^2 / 2$ for all $x, y \in C$.

The *radius* R_C of a compact convex set C is equal to the the radius of the circumscribing ball, i.e., $R_C = \min_{y \in C} \max_{x \in C} \|x - y\|$. A *right circular cone* in $\mathbb{R}^n$ with *semi-vertical angle* $\theta \in [0, \pi/2]$ and *direction* $v \in \mathbb{R}^n$ is $\mathcal{F}(v, \theta) = \{w | W \in \mathbb{R}^n, \angle(v, w) \leq \theta\}$. An *ellipsoid* in $\mathbb{R}^n$ is $\mathcal{E}(A, x_0) = \left\{x | (x - x_0)^T A^{-1} (x - x_0) \leq 1\right\}$ where $x_0 \in \mathbb{R}^n$ and $A \in \mathbb{R}^{n \times n}$ is a positive definite matrix. The *isotropic transformation* T_{A,x_0} of an ellipsoid $\mathcal{E}(A, x_0)$ is $T_{A,x_0}(x) = A^{-1/2}(x - x_0)\sqrt{\lambda_{\max}(A)}$. Intuitively, T_{A,x_0} transforms the ellipsoid into a hypersphere centered at the origin with a radius equal to the ellipsoid's largest radius. The inverse of T_{A,x_0} is $T^{-1}_{A,x_0}(x) = A^{1/2}x/\sqrt{\lambda_{\max}(A)} + x_0$. The *circumscribing ellipsoid* $\mathcal{E}_C = \mathcal{E}(A^*, x_0^*)$ of a compact convex set C satisfies $\det(A^*) = \min_{A, x_0} \det(A)$ where $C \subseteq \mathcal{E}(A, x_0)$. We denote the *identity matrix* by I.

A *σ^2-subgaussian* random variable with mean μ satisfies $\Pr(|X - \mu| > t) \leq 2 \exp\left(-t^2/(2\sigma^2)\right)$.

Smooth Convex Optimization Using Sub-Zeroth-Order Oracles

We consider the minimization of a β-smooth, L-Lipschitz, convex function f on a compact, convex set $C \subseteq \mathbb{R}^n$ where x^* denotes a minimizer of f. We assume x^* is an interior point of C such that $\mathcal{E}(\varepsilon I/L, x^*) \subseteq C$, where ε is the desired suboptimality gap. This assumption is included for simplicity, but can be removed by considering a near-optimal interior point with a sufficiently large neighborhood. Such a point is guaranteed to exist after the isotropic transformation. We also assume $n \geq 2$, but the algorithms that we present generalize to the one-dimensional setting.

Sub-Zeroth-Order Oracles

The first oracle we consider is the directional-preference oracle which outputs a binary value indicating whether the function is increasing on the queried direction at the queried point. The *directional-preference oracle* $\psi^{DP} : C \times \mathbb{R}^n \to \{-1, 1\}$ is a function such that $\psi^{DP}(x, y) = -1$ if $\langle \nabla f(x), y \rangle < 0$, and $\psi^{DP}(x, y) = 1$ otherwise.

We also consider the comparator oracle, which compares the function at a pair of query points. The *comparator oracle* $\psi^C : C \times C \to \{-1, 1\}$ is a function such that $\psi^C(x, y) = -1$ if $f(x) \geq f(y)$, and $\psi^C(x, y) = 1$ otherwise. The comparator oracle is similar to the directional-preference oracle in that $\psi^C(x, x+ky)$ approaches $\psi^{DP}(x, y)$ in the limit as k approaches zero, i.e., $\lim_{k \to 0^+} \psi^C(x, x+ky) = \psi^{DP}(x, y)$ for all $x \in C$ and $y \in \mathbb{R}^n$.

The *noisy value oracle* $\psi^{NV} : C \to \mathbb{R}$ outputs the function value plus a σ^2-subgaussian noise, i.e., $\psi^{NV}(x) = f(x) + Z$ for all $x \in C$, where Z is a σ^2-subgaussian random variable with zero mean.

In addition to the sub-zeroth-order oracles, we also consider the zeroth-order value oracle as preliminary step for the noisy-value oracle. The *value oracle* $\psi^V : C \to \mathbb{R}$ outputs the function value at the queried point, i.e., $\psi^V(x) = f(x)$ for all $x \in C$.

Ellipsoid Method With Approximate Gradients

In this section, we provide optimization algorithms that employ the sub-zeroth-order oracles. We use a variation of the ellipsoid method (Shor 1972; Yudin and Nemirovskii 1976) that uses the approximately correct gradient direction. The ellipsoid method begins each iteration with an ellipsoid containing an optimal point, it then computes the function's gradient at the ellipsoid center and removes all points from the feasible set that lie along an ascent direction. The remaining points in the set are then enclosed in the minimum volume circumscribing ellipsoid, which is used as the starting ellipsoid in the next iteration. The volume of the generated ellipsoid decreases in each iteration. For a Lipschitz, convex function, this method is guaranteed to output a near optimal solution in a finite number of iterations.

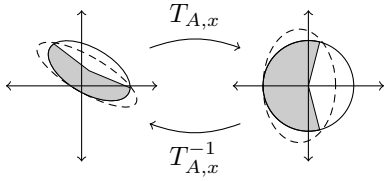


Figure 1: Illustrations of the ellipsoid cuts. The original coordinates are on the left and the isotropic coordinates on the right. The dashed ellipsoids enclose the shaded regions that are the possible descent directions.

While the information on the gradient direction is sufficient to apply the classical ellipsoid method, computing the exact gradient direction would require infinitely many queries to the sub-zeroth-order oracles. On the other hand, if the semi-vertical angle of the cone of possible gradient directions is small enough, i.e., less than $\sin^{-1}(1/n)$, in the isotropic coordinates, one can still find an ellipsoid with a smaller volume that contains all possible descent directions and the optimal solution.

Lemma 1. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a differentiable, convex function. For $\theta \in [0, \sin^{-1}(1/n)]$ and $p \in \mathbb{R}^n$, if $\nabla f(0) \in \mathcal{F}(p, \theta)$, then $f(x') \geq f(0)$ for all $x' \in \mathcal{E}(I, 0) \cap \{x | \langle p/\|p\|, x\rangle > \sin\theta\}$, and there exists an ellipsoid $\mathcal{E}^*$ such that $\mathcal{E}^* \supseteq \mathcal{E}(I, 0) \cap \{x | \langle p/\|p\|, x\rangle \leq \sin\theta\}$ and*

$$\frac{Vol(\mathcal{E}^*)}{Vol(\mathcal{E}(I,0))} = \left(\frac{n^2(1-\sin^2(\theta))}{n^2-1}\right)^{(n-1)/2} \frac{n(1+\sin(\theta))}{n+1}$$

If $\theta = \sin^{-1}(1/(2n))$, then

$$Vol(\mathcal{E}^*) \leq Vol(\mathcal{E}(I,0))e^{-\frac{1}{8(n+1)}} < Vol(\mathcal{E}(I,0)).$$

Lemma 1 shows that if the semi-vertical angle is small enough, there exists an ellipsoid with a smaller volume that contains the intersection of the possible descent directions and the initial ellipsoid as shown in Figure 1. Since the isotropic transformation is affine, it preserves the ratio of volumes. Thus, there also exists an ellipsoid with a smaller volume in the original coordinates as shown in Figure 1.

We need to approximately estimate the direction of the gradient in order to employ Lemma 1. For the value and the noisy-value oracles, we can estimate the direction of the gradient by sampling the function on a fixed set of basis vectors. However, to estimate the gradient direction using the comparator and directional-preference oracles, we need to successively select different collections of vectors along which to sample the function. In the following two sections, we describe in detail how to estimate the direction of the gradient using the sub-zeroth-order oracles, and how to use these estimations for optimization.

Optimization Using the Directional-Preference Oracle For the directional preference oracle, we can estimate direction of the gradient by iteratively sampling the function along different sets of basis vectors. Consider Figure 2 as an example. Assume that the gradient $\nabla f(x)$ lies in $\mathcal{F}(d_1, \gamma)$ shown in Figure 2a. We can use $\psi^{DP}(x, d_2)$ and $\psi^{DP}(x, d_3)$

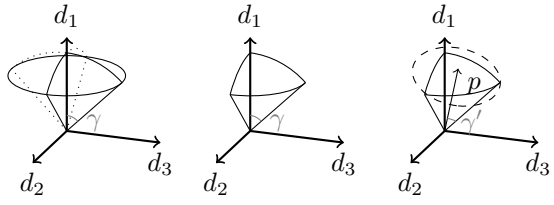


Figure 2: Illustrations the gradient pruning method by directional-preferences. (a) The cone $\mathcal{F}(d_1, \gamma)$ is the possible gradient directions. (b) The quarter cone is the possible gradient directions after the queries. (c) The dashed cone $\mathcal{F}(p, \gamma')$ overapproximates possible gradient directions.

to prune the direction estimation. The query directions slice the n-dimensional space into 2^n hyperoctants that are symmetric around the direction of the cone. The query results determine the hyperoctant that the gradient lies in. For example, if $\psi^{DP}(x, d_2) = 1$ and $\psi^{DP}(x, d_3) = 1$, the gradient lies in the quarter cone given in Figure 2b. Before the next set of queries, we limit the possible set of gradient directions with $\mathcal{F}(p, \gamma')$ such that $\gamma' < \gamma$ as shown in Figure 2c.

Lemma 2. *Let $\gamma \in (0, \pi/2]$, $d_1 = e_1$, $d_i = \cos(\gamma)e_1 + \sin(\gamma)e_i$, for all $i \in \{2, \ldots, n\}$, $p = \sum_{i=1}^n d_i$, and $\gamma' = \cos^{-1}(\langle p, d_2\rangle/\|p\|))$. Then, $\mathcal{F}(p, \gamma') \supseteq \mathcal{F}(d_1, \gamma) \cap \{x | x_i \geq 0\}$ and $\sin(\gamma')/\sin(\gamma) \leq \sqrt{n-1}/\sqrt{n}$.*

Lemma 2 shows that if we choose the direction of the new cone as the average of the extreme points of the intersection of the previous cone and the hyperoctant as in Figure 2c, then the semi-vertical angle of the cone of possible gradient directions is a fraction of the previous angle depending on the number of dimensions. For the directional-preference and the comparator oracles, we repeat this process until the cone of possible gradient directions is sufficiently small, i.e., less than $\sin^{-1}(1/(2n))$.

Algorithm 1 obtains a near-optimal solution for a given smooth, Lipschitz, convex function. At each iteration, we estimate the gradient direction using the direction pruning algorithm PD-DP, which implements the procedure described above. After the gradient direction estimation, we remove the ascent directions from the feasible set and proceed to the next iteration by enclosing the feasible set using an ellipsoid.

In the classical ellipsoid method, the output is the ellipsoid center with the smallest function value. The directional-preference oracle cannot compare the function values for a given pair of points, x_l and x_r. However, we can use the bisection method to find a point x' such that $f(x') \leq \min(f(x_l), f(x_r)) + \delta$ for a given δ. Since the function is Lipschitz, the search stops after a finite number of iterations. To find a point whose function value is close to the function value of the optimal ellipsoid center, we can remove x^l and x^r from the set of candidate points and add x' to the set of the set of candidate points. Hence, the sample complexity of finding a point x'' such that $f(x'') \leq \min_{x \in X} f(x) + \varepsilon/2$ is linear in the size of X. The function COMPARE-DP implements the bisection search method on a given set X.

Algorithm 1 The optimization algorithm OPTIMIZE-DP(X, ψ^{DP}) for the directional preference oracle

Find $\underline{\mathcal{E}}_C = \mathcal{E}(A^{(k)}, x^{(1)})$ of C.
Set $X = \{x^{(1)}\}$, $C^{(1)} = C$, $K = \left\lceil 8n(n+1)\log\left(\frac{2R_C L}{\varepsilon}\right)+1\right\rceil$.
for $k = 1 \dots K$ **do**
  Set $p = \text{PD-DP}\left(\psi^{DP}, x^{(k)}, \sin^{-1}\left(1/(2n)\right), A^{(k)}\right)$.
  Set $C^{(k+1)} = C^{(k)} \cap \mathcal{E}(A^{(k)}, x^{(k)}) \cap T^{-1}_{A^{(k)}, x^{(k)}}\left(\{x \mid \langle p/\|p\|, x\rangle \le 1/(2n)\}\right)$.
  Find $\underline{\mathcal{E}}_{C^{(k+1)}} = \mathcal{E}(A^{(k+1)}, x^{(k+1)})$ of $C^{(k+1)}$.
  Set $X = X \cup \{x^{(k+1)}\}$.
end for
return COMPARE-DP($X, \psi^{DP}, \varepsilon/2$).

Function PD-DP($x, \theta, T_{A,x}$)

$p = e_1$, $r = 1$, $\gamma = \pi/2$.
while $\gamma > \theta$ **do**
  Find d_i such that $d_1 = p$, $d_i \perp d_j$ for all $i \neq j \in [n]$, and $\|d_i\| = 1$ for all $i \in [n]$.
  Query $\psi^{DP}(x, A^{-1/2}d_1), \dots, \psi^{DP}(x, A^{-1/2}d_n)$.
  Set $w_1 = d_1$ and for all $i \in \{2, \dots, n\}$, set $w_i = d_1\cos(\gamma) + d_i\psi^{DP}(x, A^{-1/2}d_i)\sin(\gamma)$.
  Set $p = \left(\sum_{i=1}^n w_i/n\right) / \left\|\sum_{i=1}^n w_i/n\right\|$,
  Set $\gamma = \cos^{-1}(\langle p, w_2\rangle)$.
  Set $r = \sin^{-1}(\gamma)$.
end while
return p.

Theorem 1. *Let $K = \left\lceil 8n(n+1)\log\left(\frac{2R_{CL}}{\varepsilon}\right)\right\rceil$. For an L-Lipschitz, β-smooth, convex function $f : C \to \mathbb{R}$, Algorithm 1 makes at most*

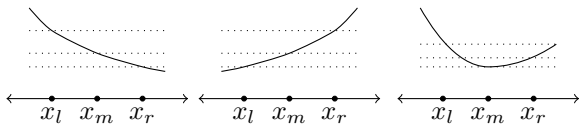
$$nK\left\lceil 2n\log(2n)\right\rceil + K\log_2\left(\frac{R_C L(K+1)}{\varepsilon}\right)$$

queries to ψ^{DP} and the output x' of Algorithm 1 satisfies $f(x') \le \min_{x\in C} f(x) + \varepsilon$.

The sample complexity and the correctness of Algorithm 1 follows from Lemmas 1 and 2. The sample complexity using the directional-preference oracle is $\tilde{\mathcal{O}}(n^2)$ of the classical ellipsoid algorithm. An invetable factor of $\mathcal{O}(n)$ is required to query the function in all dimensions, i.e., to slice the cone of the possible gradient directions into hyperoctants. By Lemma 2, a factor of $\mathcal{O}(n\log(n))$ is due to the number of iterations of the gradient pruning algorithm. While the gradient pruning method is optimal when the semi-vertical angle of the possible gradient directions is large, it is suboptimal when the semi-vertical angle is close to 0. One may improve the dependency of $\mathcal{O}(n\log(n))$ by treating this small angle regime differently. We remark that optimization using the directional-preference oracle is still possible in the absence of smoothness. One can use the same optimization method with an oracle that outputs the sign of an arbitrary directional subgradient.

Function COMPARE-DP(X, ε)

Set $X^* = X$ and $m = |X|$.
while $|X^*| > 1$ **do**
  Arbitraritly pick $x^1, x^2 \in X$ such that $x^1 \neq x^2$.
  Set $X^* = X^* \setminus \{x^1, x^2\}$.
  Set $x^l = x^1$ and $x^r = x^2$.
  while $\|x^r - x^l\| \le 2\varepsilon/(Lm)$ **do**
    Query $\psi^{DP}((x^r + x^l)/2, (x^r - x^l)/2)$.
    if $\psi^{DP}((x^r + x^l)/2, (x^r - x^l)/2) = 0$ **then**
      $x^l = (x^r + x^l)/2$.
    else
      $x^r = (x^r + x^l)/2$.
    end if
  end while
  $X^* = X^* \cup \{(x^r + x^l)/2\}$
end while
return $x^* \in X^*$.



(a) (b) (c)

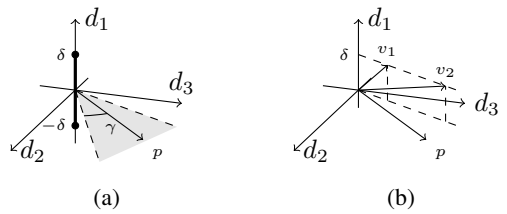
Figure 3: Possible orderings for a convex function at three points on a line.

Optimization Using the Comparator Oracle The optimization algorithm that we provide for the comparator oracle is similar to the optimization algorithm for the directional preference oracle. To solve the optimization problem, we begin by using comparisons to infer the sign of the directional derivative, i.e., we use the comparator oracle ψ^C to infer the directional-preference oracle ψ^{DP}. Then, we approximately find the direction of the gradient using the signs of the directional derivatives.

Suppose function g is in isotropic coordinates and we compare the function values at three points on a line, x_r, x_m, and x_l. We can get the directional derivative information at the middle point if the values of the function at x_r, x_m, and x_l are ordered as in Figures 3a and 3b. If the queried points are not ordered, i.e., the function value at x_m is lower than or equal to the function values at both x_r and x_l as in Figure 3c, the sign of the directional derivative is unknown at x_m. Function FDD-C takes the isotropic transformation information and outputs directional derivative information.

Function FDD-C(A, x_0, d, t)

Query $\psi^C(x - tA^{-1/2}d, x), \psi^C(x, x + tA^{-1/2}d)$.
if $f(x - tA^{-1/2}d) \le f(x) \wedge f(x) \le f(x + tA^{-1/2}d)$ **then return** 1.
else if $f(x - tA^{-1/2}d) < f(x) \wedge f(x) < f(x + tA^{-1/2}d)$ **then return** -1.
else return $unknown$.



(a) (b)

Figure 4: Possible cases for Algorithm 2. (a) The uncertainty sets for the unknown direction, d_1, and the known directions, d_2 and d_3. (b) Two possible cases for the gradient estimation in Algorithm 2.

Function PD-C(x, θ, A, t)

Set $r = 1, \gamma = \pi/2, m = 0, UD = \emptyset, p = e_1$.
while $\gamma > \theta \wedge m < n$ **do**
  Set $\{d_1, \ldots, d_m\} = UD$.
  Find d_i such that $d_{m+1} = p$, $d_i \perp d_j$ for all $i \neq j \in [n]$, and $\|d_i\| = 1$ for all $i \in [n]$.
  Set $\psi^{DP}(x, A^{-1/2}d_i) =$ FDD-C(A, x_0, d, t) for all $i \in [n]$.
  if $\exists i \in \{m+1, \ldots, n\}$, such that $\psi^{DP}(x, A^{-1/2}d_i) = unknown$ **then**
    Set $UD = UD \cup d_i$, and $m = m + 1$.
  else
    Set $w_i = d_{m+1}\psi^{DP}(x, A^{-1/2}d_{m+1})\cos(\gamma) + d_i\psi^{DP}(x, A^{-1/2}d_i)\sin(\gamma)$ for all $i \in [n]$.
    Set $p = \left(\sum_{i=m+1}^{n} w_i/n\right) / \left\|\sum_{i=m+1}^{n} w_i/n\right\|$, $\gamma = \cos^{-1}(\langle p, w_{m+2}\rangle)$, $r = \sin^{-1}(\gamma)$.
  end if
end while
if $m \neq n$ **then return** p, **else return** e_1.

In cases when the sign of the directional derivative is unknown, we can use the smoothness of the objective function to bound the magnitude of the derivative as follows. In the case shown in Figure 3c, there exists a point x' such that $\left\langle \nabla g(x'), x^r - x^l \right\rangle = 0$ and $x' = \alpha x^r + (1-\alpha)x^l$ for some $\alpha \in [0, 1]$. Due to the smoothness property, we have $\left\langle \nabla g(x^m), x^r - x^l \right\rangle \leq \beta \left\| x^r - x^l \right\| /2$.

The function PD-C prunes the cone of the possible gradient directions by inferring the directional derivative information on different sets of basis vectors. At each iteration, the algorithm starts with a cone of possible gradient directions. Based on the query results the algorithm identifies the unknown directions UD and finds an approximate direction for the projection $Proj_{UD^\perp}(\nabla g(x))$ of the gradient onto the span of the known directions. In the next iteration, the algorithm uses the $Proj_{UD^\perp}(\nabla g(x))$ as the direction of the cone of the possible gradient directions. When the semi-vertical angle of the cone of the possible gradient directions is sufficiently small or the number of unknown directions is equal to the number of dimensions, the function returns the estimation for the direction of the gradient.

Algorithm 2, used for optimization with the comparator oracle, has two steps in each iteration. In the first step, the algorithm identifies a candidate approximate gradient direction in the isotropic coordinates using the direction pruning function PD-C. In the second step, the algorithm performs a cut as in the classical ellipsoid method.

Algorithm 2 The optimization algorithm OPTIMIZE-C(ε) for the comparator oracle

Set $C^{(1)} = C$. Find $\underline{\mathcal{E}}_{C^{(1)}} = \mathcal{E}(A^{(1)}, x^{(1)})$ of $C^{(1)}$.
Set $X = \{x^{(1)}\}$, $K = \left\lceil 8n(n+1)\log\left(\frac{R_C L}{\varepsilon}\right)\right\rceil$, $\kappa = \max\left(\frac{4}{4n - \sqrt{2}n\sqrt{\frac{4n^2-1}{4n^2}}}, 1\right)$.
for $k = 1 \ldots K$ **do**
  Set $t^{(k)} = \frac{\min(\varepsilon, \sqrt{\lambda_{\max}(A^k)})}{\kappa n^{5/2}\max(\beta, 1)\max(R_C, 1)}$.
  Set $p =$ PD-C$\left(x^{(k)}, \sin^{-1}\left(\frac{1}{2\sqrt{2}n}\right), A^{(k)}, t^{(k)}\right)$.
  Set $C^{(k+1)} = C^{(k)} \cap \mathcal{E}(A^{(k)}, x^{(k)}) \cap T^{-1}_{A^{(k)}, x^{(k)}}(\{x | \langle p/\|p\|, x\rangle \leq 1/(2n)\})$.
  Find $\underline{\mathcal{E}}_{C^{(k+1)}} = \mathcal{E}(A^{(k+1)}, x^{(k+1)})$ of $C^{(k+1)}$.
  Set $X = X \cup \{x^{(k+1)}\}$.
end for
Find $x' = \min_{x \in X} f(x)$ using ψ^C.
return x'.

In order to find a near-optimal point, the algorithm exploits the fact that the direction of the projection of the gradient onto the linear subspace $Span(UD)^\perp$ of $\mathbb{R}^n$ is approximately correct, and the magnitude of the projection $\|Proj_{UD}(\nabla g(x))\|$ of the gradient onto the complement subspace is small. For example, in Figure 4a, direction d_1 is the unknown direction and $\|Proj_{UD}(\nabla g(x))\| \leq \delta$. Directions d_2 and d_3 are the known directions and $\angle(Proj_{UD}(\nabla g(x)), p) \leq \gamma$. There are two possible cases:

1. The angle between $Proj_{UD^\perp}(\nabla g(x))$ and $\nabla g(x)$ is sufficiently small.
2. The angle between $Proj_{UD^\perp}(\nabla g(x))$ and $\nabla g(x)$ is not sufficiently small.

Case 1 happens if $\|Proj_{UD^\perp}(\nabla g(x))\|$ is large enough. In this case, the estimation for the direction of the gradient $\nabla g(x)$ is approximately correct since the estimation p for the direction of $Proj_{UD^\perp}(\nabla g(x))$ is approximately correct. In this case, the ellipsoid algorithm proceeds normally. For example, if $\nabla g(x) = v_2$ in Figure 4b, then $\angle(\nabla g(x), p)$ is small enough, say less than $\sin^{-1}(1/(2n))$. If Case 2 happens, the gradient approximation is not accurate, i.e., $\angle(\nabla g(x), p)$ might be larger than $\sin^{-1}(1/(2n))$. However, if Case 2 happens, it implies that $\|Proj_{UD^\perp}(\nabla g(x))\|$ is not large enough compared to $\|Proj_{UD}(\nabla g(x))\|$. Consequently, the magnitude $\|\nabla g(x)\|$ of the gradient is not large, say less than $\varepsilon/(nR_C)$. For example, if $\nabla g(x) = v_1$ in Figure 4b, then $\|\nabla g(x)\|$ is small enough. We carefully choose the sampling distance so that the current ellipsoid center x is near optimal if $\|Proj_{UD^\perp}(\nabla g(x))\|$ is not large enough. Algorithm 2 is agnostic to whichever case happens: The algorithm always assumes that the direction estimation approximately correct. However, the output point is near optimal

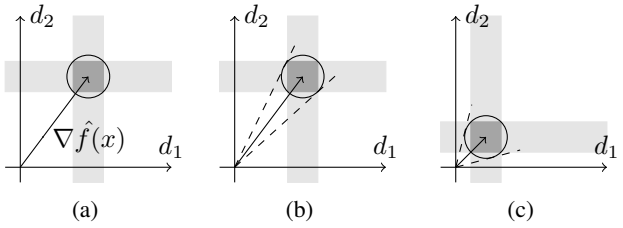


Figure 5: Illustrations of possible cases for the gradient $\nabla g(x)$. (a) The light gray stripes are the uncertainty sets for the directional derivatives. The dark gray squares are the uncertainty sets for $\nabla g(x)$ and, the circles overapproximate the uncertainty sets. In (b) and (c), the angle between the empirical gradient $\nabla \hat{g}(x)$ and the dashed lines is the maximum angle between $\nabla \hat{g}(x)$ and $\nabla g(x)$.

since we compare the ellipsoid centers and output the best point before the termination.

Theorem 2. *Let $K = \left\lceil 8n(n+1)\log\left(\frac{R_C L}{\varepsilon}\right)\right\rceil$. For an L-Lipschitz, β-smooth, convex function $f : C \to \mathbb{R}$, Algorithm 2 makes at most*

$$2n\left\lceil 2n\log(2\sqrt{2}n) + n\right\rceil K + K$$

queries to ψ^C and the output x' of Algorithm 2 satisfies $f(x') \leq \min_{x\in C} f(x) + \varepsilon$.

Theorem 2 shows that using the comparator oracle we can find a near optimal point with $\tilde{\mathcal{O}}(n^4)$ queries, which is at the same order with the sample complexity of optimization using the directional preference oracle. We also remark that while the smoothness of the function is required to determine the sampling distance, the sample complexity is not dependent on the smoothness constant.

Optimization Using the Value Oracle The value oracle is more informative than the comparator and directional-preference oracles; we can query the function in orthogonal directions near the center point and estimate the gradient. In the limit, i.e., the sampling distance goes to 0, the gradient estimate converges to the true gradient.

Under the smoothness assumption, we can get a provably good approximation of the gradient with a finite sampling distance. Let g be a β-smooth function in the isotropic coordinates. Formally, we have $g(x) - g(y) - \beta\|x-y\|^2/2 \leq \langle \nabla g(x), x-y\rangle \leq g(x) - g(y) + \beta\|x-y\|^2/2$.

Assume that we sample the points that have a distance of d from the center point in the isotropic coordinates. After $n+1$ queries we can bound the gradient in a hypercube with the edge length of βd. The hypercube can be contained in a hypersphere with radius $\beta\sqrt{n}d/2$. For example, consider the case shown in Figure 5a. Let $\nabla \hat{g}(x)$ be the empirical gradient estimate, i.e., the center of the hypercube. We can have two cases, either the gradient is in $\mathcal{F}(0, \sin^{-1}(1/(2n))$ or the magnitude of the gradient is smaller than $(2n+1)\sqrt{n}\beta d$. The former and latter cases are illustrated in Figures 5b and 5c, respectively. In the latter case, if d is sufficiently small, i.e., lower than $\varepsilon/((2n+1)\sqrt{n}\beta R_C)$, the center point is near optimal. Overall, the sample complexity of optimization using the value oracle with the ellipsoid method is $\tilde{\mathcal{O}}(n^3)$. (Nemirovsky and Yudin 1983) provided a randomized optimization algorithm that succeeds with probability at least $1-\delta$ (where δ can be chosen to be arbitrarily small) and has a sample complexity of $\tilde{\mathcal{O}}(n^3)$ for Lipschitz continuous, convex functions. With an additional smoothness assumption, the method that we describe deterministically succeeds with the same complexity. We also remark that these bounds are inferior to the $\mathcal{O}(n^2)$ sample complexity result given in (Lee, Sidford, and Vempala 2018).

Optimization Using the Noisy-Value Oracle For the noisy-value oracle, we can use the same gradient direction estimation method as in the value oracle. Different from the value oracle, we also need to consider the stochasticity of the oracle outputs since the empirical estimate $(\psi^{NV}(x) - \psi^{NV}(y))/\|x-y\|$ of directional derivative is a $2\sigma^2/\|x-y\|^2$-subgaussian random variable.

We need $\tilde{\mathcal{O}}(\sigma^2/(\beta^2\|x-y\|^4))$ samples to obtain a confidence interval of $\mathcal{O}(\beta\|x-y\|)$ for the directional derivative estimate. By letting $\|x-y\| = \mathcal{O}\left(\varepsilon/(2(2n+1)\sqrt{n}\beta R_C)\right)$, we can ensure either that the ellipsoid method proceeds normally or that the current ellipsoid center is near optimal. Overall, the sample complexity of optimization using this method is $\tilde{\mathcal{O}}(n^{13}/\varepsilon^4)$. We remark that Belloni et al. (2015) provived an algorithm that has $\tilde{\mathcal{O}}(n^{7.5}/\varepsilon^2)$ sample complexity and ε-suboptimality in expectation.

A Sublinear Regret Algorithm for the Noisy-Value Oracle

The regret of an optimization algorithm measures the performance of the algorithm during optimization. Define x_i as the query point at time i, and $\hat{f}(x_i)$ as the output of the oracle. For a given number of queries T, the regret of an algorithm $\mathcal{A}$ is $\mathcal{R}^{\mathcal{A}}(T) = \sum_{t=1}^{T} f(\mathcal{A}(h_t)) - \sum_{t=1}^{T} f(x^*)$ where $h_t = (x_0, \hat{f}(x_0))\ldots(x_{t-1}, \hat{f}(x_{t-1}))$ is the history of the algorithm. As in the previous section, we assume that x^* is an interior point of C such that $\mathcal{E}(T^{-0.25}I/L, x^*) \subseteq C$.

The optimization algorithm mentioned in the previous section incurs sublinear regret when $\varepsilon = \mathcal{O}(T^{-0.2})$. However, this approach yields a regret that has high order dependencies on the other parameters since the algorithm only relies on finding a near-optimal point with a regret of $\mathcal{O}(T^{-0.2})$ if the gradient estimation fails. We give Algorithm 3 that incurs $\tilde{\mathcal{O}}(n^{3.75}R_C\sqrt{\beta\sigma}T^{0.75})$ regret with high probability when $T = \Omega(n^3L^{4/3}\sigma^2 + L^4\sigma^6)$ and $nR_C, \beta, L, \sigma \geq 1$. Different from the optimization algorithm, Algorithm 3 does not find a near-optimal point if the gradient estimation fails. Instead, Algorithm 3 finds a point with a regret that incurs the half of the regret of the previous query point. While this approach increases the number of queries for optimization purposes, it yields a low regret.

Algorithm 3 consists of three phases. In Phase 1, we start by limiting the current convex set with the circumscribing ellipsoid and apply the isotropic transformation. We query

the oracle in every dimension at the center of the ellipsoid and at the points that are close to the center. Then, we estimate the gradient within a confidence interval and limit the possible gradient directions to a cone in the isotropic coordinates. There are two possible cases:

1. If the semi-vertical angle of the possible gradient directions is small enough, i.e., less than $\sin^{-1}(1/(2n))$, we cut the current ellipsoid, and start the process from the beginning using the remaining set.
2. If the semi-vertical angle of the possible gradient directions is not small enough, we halve the sampling distance and confidence interval, and start querying with the new sampling distance and confidence interval.

If Case 2 happens, it implies that the gradient at the current ellipsoid center has a small magnitude, and the regret of the next set of queries is low. If Case 1 happens sufficiently many times, then one of the ellipsoid centers is a near optimal point with low regret as in the classical ellipsoid method. After Case 1 happens sufficiently many times, the algorithm proceeds to Phase 2. In this phase, we compare the the ellipsoid centers and find an ellipsoid center with a low regret, i.e., $\mathcal{O}(T^{-0.25})$. In Phase 3, we repeatedly query the ellipsoid center with a low regret.

Theorem 3. *Let* $K = \lceil 8n(n+1)\log\left(2R_C L T^{0.25}\right)\rceil$, $\delta' = \delta/\left(4nK\log_{16}\left(\frac{15T}{2n}\right)\right)$, *and* $\tau = \left\lceil 8\sigma^2\beta^2 n^4 \log\left(\frac{2}{\delta'}\right)\right\rceil$. *For an L-Lipschitz, β-smooth, convex function $f : C \to \mathbb{R}$, a given failure probability $\delta > 0$, and a time horizon T, Algorithm 3 has a regret of at most* $K\left(R_C L\tau + 5T^{0.75}n^{-0.25}\max(nR_C, 1)(1+\beta)\tau^{0.25}\right) + (K+1)\left\lceil 32\sigma^2\sqrt{T}\log\left(\frac{2(K+1)}{\delta}\right)\right\rceil R_C L + T^{0.75}$ *with probability at least* $1-\delta$.

For a given L-Lipschitz, β-smooth function $f : C \to \mathbb{R}$, we can define $f' : C' \to \mathbb{R}$ such that $C' = \left\{x' | x' = x\sqrt{\beta}, x \in C\right\}$ and $f'(\sqrt{\beta}x) = f(x)$ for all $x \in C$. f' is $L/\sqrt{\beta}$-Lipschitz, 1-smooth, and $R_{C'} = \sqrt{\beta}R_C$. If Algorithm 3 operates with the parameters of f', the regret is $\tilde{\mathcal{O}}(n^{3.75}R_C\sqrt{\beta}\sigma T^{0.75})$ when $T = \Omega(n^3 L^{4/3}\sigma^2 + L^4\sigma^6)$ and $nR_C\sqrt{\beta}, L, \sigma \geq 1$.

Discussion

We consider the problem of minimizing a smooth, Lipschitz, convex function using sub-zeroth-order oracles. We leverage the smoothness property of the objective function and build variants of the ellipsoid method based on gradient estimation. We show that the sample complexities of optimization using sub-zeroth-order oracles are polynomial.

We remark that the main concern of this paper is the sample complexity of the optimization problems. The computational and space complexities of the provided algorithms are the same as those of the classical ellipsoid method. However, we remark that finding the exact minimum volume circumscribing ellipsoid for an arbitrary convex set is computationally intractable. In practice, we can avoid the computation of the minimum volume ellipsoids by finding an approximate enclosing ellipsoid using a separation oracle and analytical expressions involving the ellipsoid found at the previous step (Goldfarb and Todd 1982). We note that in the case of the comparator and noisy-value oracles, we use a property of the minimum volume circumscribing ellipsoid to give optimality and regret guarantees. We can show that a similar property holds for the approximate ellipsoids through additional feasibility cuts.

For the directional-preference and comparator oracles, since the sampling distance can be arbitrarily reduced, through small modifications in the presented algorithms, the given sample complexities can be achieved with polynomial time complexities. For the noisy-value oracle we expect that polynomial time complexities can be achieved while maintaining a polynomial regret in the number of dimensions.

Algorithm 3 The low regret algorithm REGRET-NV(T, δ) for the noisy value oracle

Set $C^{(1)} = C$. Find $\underline{\mathcal{E}}_{C^{(1)}} = \mathcal{E}(A^{(1)}, x^{(1)})$. Set $X = \{x^{(1)}\}$.
Set $K = \lceil 8n(n+1)\log\left(2R_C L T^{0.25}\right)\rceil$, $\tau = \lceil 32\sigma^2 n^4 \log\left(\frac{2}{\delta'}\right)\rceil$, $\delta' = \frac{\delta}{4nK\log_{16}\left(\frac{15T}{2n}\right)}$.
for $k = 1, \ldots, K$ **do** ▷ Phase 1
  Set $d = \frac{\min\left(\sqrt{\lambda_{\max}(A^{(k)})}, 1\right)}{2n}$.
  Set $\Delta = \frac{d\left(2+\beta\lambda_{\max}(A^{(k)})\right)}{2\lambda_{\max}(A^{(k)})}$.
  for $i = 0, 1, \ldots$ **do** ▷ Case 2
    Set $d_i = d/2^i$, $\Delta_i = \Delta/2^i$, $\tau_i = 2^{4i}\tau$
    Query τ_i times $\psi^{NV}(x^{(k)})$ and $\psi^{NV}(T^{-1}_{A^{(k)}, x^{(k)}}(de_j))$ for all $i \in [n]$.
    For every query point x, set $\hat{\psi}^{NV}(x)$ as the mean of queries for point x.
    Estimate the gradient p using the mean values $\hat{\psi}^{NV}(x)$.
    if $(\|p\| > \sqrt{n}\Delta_i) \wedge \left(\sin^{-1}\left(\frac{\sqrt{n}\Delta_i}{\|p\|}\right) \leq \sin^{-1}\left(\frac{1}{2n}\right)\right)$ **then** ▷ Case 1
      Set $C^{(k+1)} = C^{(k)} \cap \mathcal{E}(A^{(k)}, x^{(k)}) \cap T^{-1}_{A, x^{(k)}}\left(\{x | \langle p/\|p\|, x\rangle \leq 1/(2n)\}\right)$.
      Find $\underline{\mathcal{E}}_{C^{(k+1)}} = \mathcal{E}(A^{(k+1)}, x^{(k+1)})$.
      Set $X = X \cup \{x^{(k+1)}\}$.
      break
    end if
  end for
end for
For all $x \in X$, query $\psi^{NV}(x)$, $\left\lceil 32\sigma^2\sqrt{T}\log\left(\frac{2(K+1)}{\delta}\right)\right\rceil$ times. Set x' to the point with the highest empirical mean. ▷ Phase 2
Repeatedly query $\psi^{NV}(x')$. ▷ Phase 3

Acknowledgements

This work was supported in part by AFOSR FA9550-19-1-0005, DARPA D19AP00004, NSF 1646522, and NSF 1652113.

Ethics Statement

The results of this paper are theoretical. The impacts of these results are thus dependent on the chosen application area.

Learning human preferences is one of the possible applications of the optimization algorithms that we provide. While humans may not be able to express their utility functions or want to share their utility functions, they can rank the available prospects or evaluate a given prospect. The optimization algorithms provided in this paper can optimize a system based on human feedback. This application would positively affect the quality of human life. On the other hand, gradient-free optimization methods have practically high sample complexities. The use of these optimization methods may overwhelm humans due to the high number of queries and raise privacy concerns due to the high amount of collected data. We remark that while the sample complexities are high, the optimization algorithms that we provide do not need to store the query results. Hence, these algorithms are robust against data breaches.

References

Abbas, A. E.; and Howard, R. A. 2015. *Foundations of decision analysis*. Pearson Higher Ed.

Akrour, R.; Schoenauer, M.; and Sebag, M. 2012. APRIL: Active preference learning-based reinforcement learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 116–131.

Audet, C.; and Dennis Jr, J. E. 2006. Mesh adaptive direct search algorithms for constrained optimization. *SIAM Journal on optimization* 17(1): 188–217.

Audet, C.; and Hare, W. 2017. *Derivative-free and blackbox optimization*. Springer.

Belloni, A.; Liang, T.; Narayanan, H.; and Rakhlin, A. 2015. Escaping the local minima via simulated annealing: Optimization of approximately convex functions. In *Conference on Learning Theory*, 240–265.

Bernstein, J.; Wang, Y.-X.; Azizzadenesheli, K.; and Anandkumar, A. 2018. signSGD: Compressed Optimisation for Non-Convex Problems. In *International Conference on Machine Learning*, 560–569.

Bubeck, S.; Lee, Y. T.; and Eldan, R. 2017. Kernel-based methods for bandit convex optimization. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, 72–85.

Burden, R. L.; and Faires, J. D. 1985. *Numerical analysis*. PWS Publishers.

Conn, A. R.; Scheinberg, K.; and Vicente, L. N. 2009. *Introduction to derivative-free optimization*, volume 8. SIAM.

Flaxman, A. D.; Kalai, A. T.; and McMahan, H. B. 2005. Online Convex Optimization in the Bandit Setting: Gradient Descent without a Gradient. In *Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, 385–394.

Fürnkranz, J.; Hüllermeier, E.; Cheng, W.; and Park, S.-H. 2012. Preference-based reinforcement learning: a formal framework and a policy iteration algorithm. *Machine learning* 89(1-2): 123–156.

Goldfarb, D.; and Todd, M. J. 1982. Modifications and implementation of the ellipsoid algorithm for linear programming. *Mathematical Programming* 23(1): 1–19.

Karabag, M. O.; Neary, C.; and Topcu, U. 2021. Smooth Convex Optimization using Sub-Zeroth-Order Oracles. *arXiv preprint arXiv:2103.00667* .

Lattimore, T. 2020. Improved Regret for Zeroth-Order Adversarial Bandit Convex Optimisation. *arXiv preprint arXiv:2006.00475* .

Lee, Y. T.; Sidford, A.; and Vempala, S. S. 2018. Efficient convex optimization with membership oracles. In *Conference On Learning Theory*, 1292–1294. PMLR.

McKinnon, K. I. 1998. Convergence of the Nelder–Mead Simplex Method to a Nonstationary Point. *SIAM Journal on optimization* 9(1): 148–158.

Nelder, J. A.; and Mead, R. 1965. A simplex method for function minimization. *The computer journal* 7(4): 308–313.

Nemirovsky, A. S.; and Yudin, D. B. 1983. *Problem complexity and method efficiency in optimization.* John Wiley & Sons.

Price, C. J.; Coope, I. D.; and Byatt, D. 2002. A convergent variant of the Nelder–Mead algorithm. *Journal of optimization theory and applications* 113(1): 5–19.

Qian, L.; Gao, J.; and Jagadish, H. 2015. Learning user preferences by adaptive pairwise comparison. *Proceedings of the VLDB Endowment* 8(11): 1322–1333.

Shamir, O. 2013. On the complexity of bandit and derivative-free stochastic convex optimization. In *Conference on Learning Theory*, 3–24.

Shor, N. Z. 1972. Utilization of the operation of space dilatation in the minimization of convex functions. *Cybernetics* 6(1): 7–15.

Wilson, A.; Fern, A.; and Tadepalli, P. 2012. A Bayesian Approach for Policy Learning from Trajectory Preference Queries. In *Advances in Neural Information Processing Systems*, 1133–1141.

Yudin, D. B.; and Nemirovskii, A. S. 1976. Evaluation of the information complexity of mathematical programming problems. *Ekonomika i Matematicheskie Metody* 12: 128–142.