# Blackbox optimization of unimodal functions


## Abstract

We provide an intuitive new algorithm for black-box stochastic optimization of unimodal functions, a function class that we observe empirically can capture hyperparameter-tuning loss surfaces. Our method's convergence guarantee automatically adapts to Lipschitz constants and other problem difficulty parameters, recovering and extending prior results. We complement our theoretical development with experimental validation on hyperparameter tuning tasks.


## 1 BLACK-BOX OPTIMIZATION

This paper considers the problem of *black-box stochastic optimization*. Specifically, we are interested in solving a maximization problem of the form:

$$\max_x F(x) = \mathbb{E}_z[f(x, z)]$$

Here, $F$ is the objective function that takes the form $F(x) = \mathbb{E}_z[f(x, z)]$, for some function $f$ and some random variable z. We have access to a black-box evaluation oracle (sometimes called a *stochastic zeroth-order oracle*), which returns a random sample $f(x, z)$ given any $x$. This is the only information we have about $f$; we do not know the function definition or its gradients.

The ability to solve this problem forms a fundamental primitive in more complex systems. For example, it can be used to optimize design of new materials [Terayama et al., 2021], to improve user interfaces, or search for hyperparameters in machine learning [Golovin et al., 2017, Feurer and Hutter, 2019, Hazan et al., 2018]. Although the techniques we present in this work are very general, we focus on the task of optimizing hyperparameters as an important motivating application.

Black-box optimization has been studied from a variety of perspectives, which is reflected in the variety of names that refer to essentially similar concepts. Depending on the author, it may be referred to as "zero-order" or "derivative-free" optimization [Rios and Sahinidis, 2013, Duchi et al., 2015, Jamieson et al., 2012], "bandit" optimization [Agarwal et al., 2011, Agrawal, 1995, Kleinberg et al., 2008, Auer et al., 2007, Shamir, 2013, Jun et al., 2017], sequential experimental design [Chernoff, 1959], or Bayesian optimization [Shahriari et al., 2015, Srinivas et al., 2009, Snoek et al., 2012]. While the details of the methods used to solve the problem are diverse, the overall idea behind all methods is necessarily similar: first, assume some sort of "structure" on either the distribution of $z$ or the shape of $f$ (or both) that constrains the number of possibilities for $F$ - popular examples include convexity, smoothness, a Bayesian prior. Then, use new evaluations $f(x, z)$ to further constrain the possibilities until one can reliable identify a maximizing point.

An algorithm's usefulness can thus be measured on two axes: the degree that whatever structural assumptions it makes are in fact reflected in practice, and the number of samples required by the algorithm to optimize $F$ subject to the assumptions. The goal is to find assumptions expressive enough to capture real problems while still admitting efficient optimization. In this paper, we focus on the assumption of *unimodality*. That is (in the 1D case), the function $F$ has only one local maximum. This assumption is motivated via empirical observation of hyperparameter-tuning problems. For example, in Figure 1 we plot the influence of the learning rate hyperparameter on the accuracy of an AlexNet type architecture trained on the CIFAR100 image classification task [Krizhevsky, 2009]. It is evident that this relationship is at least approximately unimodal, and we describe further evidence for unimodality in Appendix E. While our algorithms simply assume unimodality and so may behave poorly when this condition does not hold, we observe empirically in Section 5 that in practice our approach works well.

Although unimodality is much less well-studied in the literature than its popular and more stringent cousin convexity, we

*Submitted to the 38th Conference on Uncertainty in Artificial Intelligence* (UAI 2022). **To be used for reviewing only**.

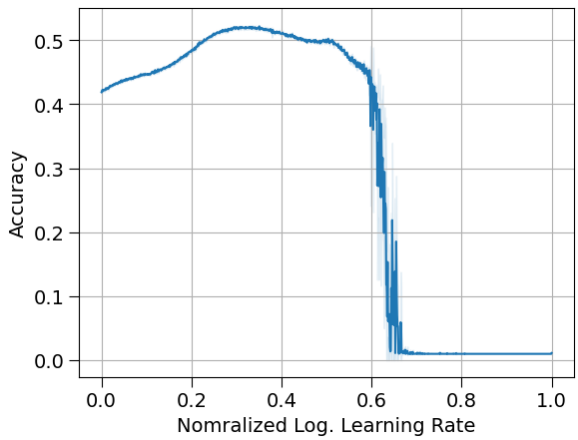



Figure 1: Accuracy of CNN on the CIFAR100 validation set as we vary the learning rate from 1e-6 to 1e-1 on a uniform grid in the log scale. We repeat the experiment 6 times for each of the 1000 learning rate values and show the mean accuracy here, with confidence intervals.

are not the first to consider unimodality. For example, Yu and Mannor [2011], Combes and Proutiere [2014], Combes et al. [2020] all provide algorithms for $F$ that satisfy particular further assumptions in addition to unimodality. We develop a new algorithm that provides theoretical improvements over these works in terms of robustness to these further assumptions: our method simultaneously enjoys provably good performance for these prior function classes as well as other natural classes.

We then conduct an empirical evaluation of our method on a benchmark hyperparameter tuning task involving tuning hyperparameters across a number of different sub-tasks [Arango et al., 2021]. We compare our algorithm to the simple random search baseline as well as the more advanced Gaussian process upper-confidence bound algorithm, which is a standard approach to hyperparameter tuning. Our method not only obtains slightly improved average performance across the different sub-tasks, it also obtains significantly better performance on tasks with less than 6 hyperparameters to tune, forming a very promising initial study of our approach.

In summary, we make the following contributions:

- In Section 3, we propose a novel algorithm for black-box optimization of 1D unimodal functions, with theoretical analysis on a variety of function classes. We simultaneously match the convergence rate of Kleinberg et al. [2008] for Lipschitz (Theorem 1) and smooth functions (Theorem 2), and that of Combes et al. [2020] for functions that meet certain growth conditions (Theorem 6). Our proposed algorithm does *not* need prior knowledge of problem parameters such as Lipschitz or smoothness constants to achieve these results.
- In Section 4, we extend our 1D algorithm to higher dimensions. This extension provably converges to a local maximum.
- In Section 5, we empirically show that our multi-dimensional algorithm is generally competitive with the state-of-the-art Bayesian optimization methods on hyperparameter tuning tasks. For modest dimension counts (i.e. $d \approx 5$), our method significantly outperforms Bayesian optimization methods.

## 2 DEFINITIONS AND SETTING

In this paper we consider exclusively loss functions of the form $f : [0,1]^d \times \mathcal{Z} \to [0,1]$, where $\mathcal{Z}$ is some arbitrary set. Given a $\mathcal{Z}$-valued random variable $Z$, we let $F(x) : [0,1]^d \to [0,1]$ be given by $F(x) = \mathbb{E}[f(x,Z)]$. Our algorithms have access to a stochastic value oracle: given a point $x$, we may generate a new i.i.d. sample $z$ and compute $f(x,z)$. We call such a process a "query" or a "sample" interchangably. Our goal is to find a point $\hat{x}$ such that $F(\hat{x})$ is as large as possible using at most $B$ samples for some given budget $B$.

A 1-D function is unimodal if it has a single local maximum. We extend this notion to arbitrary dimensions by imposing the definition on each coordinate axis.

**Definition 1** (Unimodality)**.** *A function $F : [0,1]^d \to [0,1]$ is unimodal if for all $(x_1, \ldots, x_d) \in [0,1]^d$, and all $i \in \{1, \ldots, d\}$, there exists an $x_i^\star \in [0,1]$ such that for all $a, b \in [0,1]$ with either $x_i^\star \geq a \geq b$ or $x_i^\star \leq a \leq b$, we have*

$$F(x_1, \ldots, x_{i-1}, a, \ldots, x_d) \geq F(x_1, \ldots, x_{i-1}, b, \ldots, x_d)$$

We also define a notion of *local optimality* to non-differentiable functions.

**Definition 2.** *A point $x = (x_1, \ldots, x_d) \in [0,1]^d$ is an $\tau$-approximate local optimum point of $F : [0,1]^d \to [0,1]$ if for all $i \in \{1, \ldots, d\}$, for all $x_i^\star \in [0,1]$,*

$$F(x_1, \ldots, x_{i-1}, x_i^\star, \ldots, x_d) \leq F(x) + \tau$$

In other words, an approximate local optimum point is a point at which only minimal progress can be made by changing only one coordinate.

## 3 ALGORITHM AND ANALYSIS FOR 1-D FUNCTIONS

We first build an algorithm for the special-case of 1-D unimodal functions. We extend it in Section 4 to the high dimensional case by employing coordinate descent. For this section, let $F : [0,1] \to [0,1]$ and $x_\star = \operatorname{argmax}_{x \in [0,1]} F(x)$.

### 3.1 PREVIOUS WORK

*Golden section search* method [Kiefer, 1953] is a minimax optimal algorithm for deterministic 1-D unimodal functions. The key observation is the following:

*If $F(a) \geq F(b)$ for some $a, b \in [0, 1]$, then $a \geq b$ implies $x_\star \geq b$, and $a \leq b$ implies $x_\star \leq b$.*

This suggests a natural strategy: maintain a *candidate interval* $[l, r]$ such that $x_\star \in [l, r]$ and iteratively narrow it down. Combes et al. [2020], Yu and Mannor [2011] generalized the golden section search stochastic settings. They first commit to a small number of possible values $y_1, \ldots, y_K \in [l, r]$, such that each $y_i$ is far from the boundary of the current candidate interval $[l, r]$. Then, they repeatedly sample $f(y_i, z_t)$ until they can estimate $F(y_1), \ldots, F(y_K)$ with high accuracy and identify $y_i$ and $y_j$ where $F(y_i) \geq F(y_j)$.

However, this approach is intuitively wasteful because it puts too many eggs in few baskets. For example, it fails to make progress when $F(y_i)$ is *equal* for all $i$, and all the repeat samples of $f(y_i, z_t)$ are wasted. To address this issue, Combes et al. [2020] imposes additional structural conditions on $F$ that prevent the derivative from being too close to zero. **Our approach does not require such conditions, but it automatically enjoys a faster convergence rate when the conditions are satisfied.**

**Algorithm 1:** One Elimination Round of Unimodal Optimization

**Require:** Confidence parameter $\delta$, interval $\Delta = [l, r]$, threshold $\tau \geq 0$, spacing GAP such that $\frac{u-l}{\text{GAP}}$ is an integer, confidence scaling constant $h > 0$ to be set by Lemma 2.

1: Define the points $l = x^{(1)} < x^{(2)} < \ldots < x^{(N)} = r$ for $N = 1 + \frac{l-r}{\text{GAP}}$ equally spaced in the interval $\Delta$.
2: Compute $f(x^{(k)}, z^{(k)})$ for i.i.d. $z^{(1)}, \ldots, z^{(N)}$.
3: Given any $i, j \in \{1, \ldots, N\}$, define $m_{ij} = \sum_{k=i}^{j} f(x^{(k)}, z^{(k)})/(j - i + 1)$, $s_{ij} = h\sqrt{\log(2N/\delta)}/\sqrt{j - i + 1}$.
4: Define upper confidence bound $U(i, j) = m_{ij} + s_{ij}$ for each $i, j \in \{1, \ldots, N\}$
5: Define lower confidence bound $L(i, j) = m_{ij} - s_{ij}$ for each $i, j \in \{1, \ldots, N\}$
6: Let $S_l = \max\{x^{(i)} \,|\exists i \leq j \leq k \leq l \; s.t. \; U(i, j) < L(k, l) - \tau\} \cup \{x^{(1)}\}$.
7: Let $S_r = \min\{x^{(l)} \,|\exists i \leq j \leq k \leq l \; s.t. \; U(k, l) < L(i, j) - \tau\} \cup \{x^{(N)}\}$.
8: Let $I_{\text{best}} = \text{argmax}_{[x^{(i)}, x^{(j)}]} L(i, j)$
9: Return new interval $[S_l, S_r] \subset \Delta$ and the best interval $I_{\text{best}}$.

Our approach is inspired by prior work on continuum armed-bandits [Auer et al., 2007] that tries to identify an interval containing $x_\star$. We extend their technique to unimodality so that the intervals can be used for elimination.

### 3.2 ALGORITHM

Our algorithm samples a large number of points only *once* (Line 5 of Algorithm 1). As a result, our algorithm cannot build good enough estimates of $F$ at a single point to identify *a pair of points* $(a, b)$ with $F(a) \geq F(b)$. Instead, **our algorithm identifies a pair of intervals** $I_a = [a_1, a_2]$ and $I_b = [b_1, b_2]$ for which there is some (unknown) $a \in I_a, b \in I_b$ satisfying $F(a) \geq F(b)$. Thus, we can either eliminate all points less than $b_1$ if $b_2 < a_1$, or all points greater than $b_2$ if $a_2 < b_1$. (Line 5-5).

Note that since $b \in [b_1, b_2]$, we do not eliminate as much as we would be able to if we had found the exact point $b$. That is, our algorithm operates with a potentially larger candidate interval than the existing algorithms at first, but casts a wider net for finding sub-intervals to eliminate. Intuitively speaking, our algorithm explores the geometry of the function $F$ more broadly.

In order to find these intervals, we start by generating a grid of uniformly spaced points $x^{(1)}, x^{(2)}, \ldots, x^{(N)}$ with $|x^{(i)} - x^{(i+1)}| = \Delta$ for some grid-spacing parameter $\Delta$. Then, we evaluate $f(x^{(i)}, z^{(i)})$ for each $i \in \{1, \ldots, N\}$ for independent samples $z_1, \ldots, z_N$. Now, for each interval $I = [x_i, x_j]$, we can form the intersection $S_I = I \cap \{x^{(1)}, \ldots, x^{(N)}\}$ and estimate the average $F_I = \frac{1}{|S_I|} \sum_{x^{(i)} \in S} F(x^{(i)})$ by $\frac{1}{|S_I|} \sum_{x^{(i)} \in S_I} f(x^{(i)}, z^{(i)})$. Moreover, for each $I$ we can also form *confidence intervals* of width $O(\sqrt{\log(N^2/\delta)}/\sqrt{|S_I|})$ valid with probability $1 - \delta/N^2$. Since there are $O(N^2)$ possible sets $S_I$, the confidence intervals are simultaneously valid with probability $1 - \delta$. Finally, if any two intervals $I_a$ and $I_b$ are such that the confidence interval for $I_a$ is larger and disjoint from that for $I_b$, then there must be $a \in I_a$ and $b \in I_b$ with $F(a) \geq F(b)$.

We formally specify this algorithm in Algorithm 1 and provide analysis in Lemma 1. The algorithm includes a "threshold" parameter $\tau$, which is used for extensions to multiple dimensions in Section 4. For the remainder of this section, we will assume $\tau = 0$.

**Lemma 1.** *There is a universal constant $h$ that can be provided to Algorithm 1 such that with probability at least $1 - \delta$, if $x_\star \in \Delta$ and $[S_l, S_r]$ is the output of Algorithm 1 with any* GAP *and $\tau$, then $x_\star \in [S_l, S_r]$, and $\frac{1}{j-i+1} \sum_{k=i}^{j} F(x^{(k)}) \in [L(i, j), U(i, j)]$ for all $i, j$.*

Now, our full 1-D algorithm combines this interval shrinking routine with a doubling argument: we repeatedly call Algorithm 1 with $\Delta$ set to the previously returned $[S_l, S_r]$ and GAP set to the previous GAP divided by two. Intuitively, if Algorithm 1 does *not* significantly shrink the interval $\Delta$, then this approach re-runs the algorithm with double the number of samples. Otherwise, if $\Delta$ shrinks by a constant factor, we are making sure to still take a reasonable number of samples in this new smaller interval.

The final ingredient in our algorithm is how to select the

final output $\hat{x}$. One plausible approach would be to pick a random element of the interval $\Delta$. However, we opt for a more refined method: we pick a random element of the interval that has the largest lower confidence bound. This enables us to make claims about the quality of our final output even in the case that the algorithm never eliminates any intervals at all (i.e. $\Delta = [0, 1]$ always). The formal description is provided in Algorithm 2.

**Algorithm 2:** 1D Unimodal Optimization

**Require:** Confidence parameter $\delta$, threshold $\tau \geq 0$, budget $B$.

Initialize $t = 1$, $\Delta_1 = [0, 1]$, $\text{GAP}_1 = 0.5$, $N = N_1 = 3$.
**while** $N \leq B$ **do**
  Call Algorithm 1 with input $6\delta/\pi^2 t^2, \Delta_t, \tau, \text{GAP}_t$ to obtain outputs $[S_l, S_r]$, $I_{\text{best}}$.
  Let $\Delta_{t+1} = [S_l, S_r]$.
  Set $\text{GAP}_{t+1} = \text{GAP}_t/2$.
  Set $N_{t+1} = 1 + \frac{S_r - S_l}{\text{GAP}}$ //Budget to be consumed by next iteration.
  Set $N = N + N_t$. //Total budget consumed at end of next iteration.
  Set $t = t + 1$.
**end while**
Return a random element of $I_{\text{best}}$.

### 3.3 CONVERGENCE ANALYSIS

In order to prove convergence rates, we will need to make some additional assumptions about the shape of $F$ beyond just unimodality. To see this, consider the function $F(0.7) = 1$ and $F(x) = 0$ for $x \neq 0.7$. Such an $F$ is certainly unimodal, and yet clearly no algorithm can obtain any non-trivial bound on $F(x_\star) - F(\hat{x})$. As our first assumption, we consider lipschitz functions.

**Definition 3** (Lipschitz)**.** *$F(x)$ is L-lipschitz (i.e. $|F(x) - F(y)| \leq L|x - y|$ for all $x, y \in [0, 1]$).*

**Theorem 1.** *Suppose $F$ is L-lipschitz. Let $\hat{x}$ be the output of Algorithm 2 with total sample budget $B$ and input failure probability $\delta$. Then, for any $\delta < 1/2$, there is a constant $C$ and an event $E$ that occurs with probability at least $1 - \delta$ such that:*

$$\mathbb{E}[F(\hat{x})|E] \geq F(x_\star) - 3(LC^2 \text{GAP}_T \log(B/\delta))^{1/3}$$
$$\geq F(x_\star) - \frac{3(12LC^2 \log(B/\delta))^{1/3}}{B^{1/3}}$$

*where $C$ is an absolute constant.*

This result shows that the Algorithm 2 converges at a $O(1/B^{1/3})$ rate, or equivalently that $O(1/\epsilon^3)$ samples suffice to find an $\epsilon$-suboptimal point for a Lipschitz objective $F$, matching classical rates for this setting [Kleinberg et al., 2008]. The proof proceeds by observing that since Algorithm 2 never discards $x_\star$, eventually it will explore a fine enough grid of points that there will be an interval whose lower-confidence-bound is within $O(1/B^{1/3})$ of $F(x_\star)$, so that picking a random $\hat{x}$ from the interval with largest lower-confidence-bound is guaranteed to be $O(1/B^{1/3})$ suboptimal.

Next, we consider the case of *smoooth* rather than Lipschitz $F$. For smooth $F$, we can improve upon the Lipschitz analysis. The key idea is that the average value of $F$ over an interval of width $W$ is within $O(W^2)$ of the value of $F$ at the midpoint of the interval. In contrast, if we assume only Lipschitz $F$, the average value may be $O(W)$ away from any given point in the interval. This improvement translates into a suboptimality of $O(1/B^{2/5})$ rather than $O(1/B^{1/3})$, as described formally in Theorem 2.

**Definition 4.** *A function $F$ is $\beta$-smooth if $F(x)$ is differentiable and $F'(x)$ is $\beta$-Lipschitz.*

**Theorem 2.** *Suppose $F$ is $\beta$-smooth and that $F'(x_\star) = 0$ (i.e. $x_\star$ is not on the boundary). Let $\hat{x}$ be the output of Algorithm 2 with total sample budget $B$ and input failure probability $\delta$. Then, there is a constant $C$ such that for any $\delta < 1/2$ there is an event $E$ that occurs with probability at least $1 - \delta$ such that:*

$$\mathbb{E}[F(\hat{x})|E] \geq F(x_\star) - \frac{5(\beta C^4 \log(B/\delta)^2)^{1/5}}{B^{2/5}}$$

Theorem 1 and 2 establish baseline convergence rates for Lipschitz or smooth $F$. They have the desirable property that the algorithm need not use any knowledge of the Lipschitz or smoothness parameters $L$ and $\beta$, so that the bounds hold with the tightest possible (unknown) values. On the other hand, the result is somewhat naive: we did not use any of the "elimination" power of the algorithm, and in fact the same result would hold if we simply applied one single round of Algorithm 2 and did not even require unimodality.

The power of our algorithm is that it automatically enjoys a better convergence rate if the function meets certain conditions, without any changes to the input or prior knowledge. The first such condition we explore is the following:

**Assumption 1** ($(\gamma, M)$-Lipschitz lower bound)**.** *There is some $\gamma$ and $M$ such that for all $x, y$ with $|x - x_\star| \geq \gamma$ and $|y - x_\star| \geq \gamma$, and $\text{sign}(x - x_\star) = \text{sign}(y - x_\star)$, we have $|F(x) - F(y)| \geq M|x - y|$ (i.e. if $F$ is differentiable, then $|F'(x)| \geq M$ whenever $|x - x_\star| \geq \gamma$).*

Intuitively, these functions have a "difficult" interval of size $2\gamma$ around the optimum $x_\star$. The lower bound on the slope of $F$ outside this region allows our algorithm to rapidly eliminate subintervals and converge to a small interval near the optimum. We can improve upon Theorem 1, and establish that the asymptotic error is $\tilde{O}((\gamma/B)^{1/3} + 1/\sqrt{B})$. Notice

that the algorithm at no point requires knowledge of the parameter $\gamma$.

**Theorem 3.** *Suppose $F$ is $L$-Lipschitz and satisfies Assumption 1. Let $\hat{x}$ be the output of Algorithm 2 with total sample budget $B > 78$ and input failure probability $\delta$. Then, there is a constant $C$ such that for any $\delta < 1/2$, there is an event $E$ that occurs with probability at least $1 - \delta$ such that:*

$$\begin{aligned}\mathbb{E}[F(\hat{x})|E] &\geq F(x_\star) - 3(L\textsc{Gap}_T C^2 \log(B/\delta))^{1/3} \\ &\geq F(x_\star) - \tilde{O}\left(\frac{L\gamma^{1/3}}{B^{1/3}} + \frac{1}{\sqrt{B}M^{1/3}}\right)\end{aligned}$$

*where $C$ is an absolute constant.*

Finally, we consider the class of functions satisfying Assumption 2, which is a strict superset of function class studied by Combes et al. [2020] (denoted $\mathcal{U}_{[0,1]}$ in their paper). We show that our Algorithm 2 recovers the same $\tilde{O}(1/\sqrt{B})$ convergence rate in this setting, again without requiring any knowledge of problem parameters.

**Assumption 2.** *There exists $A_1, A_2$ and $z$ such that for all $x$, $A_2|x - x_\star|^z \geq F(x_\star) - F(x) \geq A_1|x - x_\star|^z$. Such functions need be neither Lipschitz nor smooth.*

**Theorem 4.** *Suppose $F$ satisfies Assumption 2. Then, there is a constant $C$ such that for all $\delta < 1/2$, for any $B$ satisfying $B \geq \frac{24A_2^{1/z}}{C^{1/z}\log(B/\delta)^{1/2z}}$, with probability at least $1 - \delta$ Algorithm 2 with $\tau = 0$ guarantees:*

$$\begin{aligned}F(\hat{x}) &\geq F(x_\star) - F(x_\star) \\ &\quad - \max\Bigg(12^z \frac{C^{\frac{2z}{2z+1}} A_2^{\frac{2z+2}{2z+1}} \log(B/\delta)^{\frac{z}{2z+1}}}{A_1 2^{\frac{zB}{24z+12}}}, \\ &\qquad \frac{12^z 2^{\frac{z}{2z+1}}\sqrt{72}}{\sqrt{2^{\frac{2z}{2z+1}} - 1}} \frac{C A_2^{1+\frac{1}{2z}} \sqrt{\log(B/\delta)}}{A_1^{1+\frac{1}{2z}}\sqrt{B}}\Bigg) \\ &\geq F(x_\star) - \tilde{O}(1/\sqrt{B})\end{aligned}$$

It should be noted that the dependencies of Theorem 4 on the parameter $z$ are slightly worse than in Combes et al. [2020]. However, this is mitigated by the fact that our function class is actually somewhat larger (Combes et al. [2020] consider a stronger version of Assumption 2 that controls local behavior of $F$ even far from $x_\star$). Further, due to the greater range of exploration of our algorithm, we are able to handle more general Lipschitz or smooth losses as described previously.

We would also like to note that our algorithm would be better than an algorithm which provides a third of the budget to CAB1 [Kleinberg, 2004], a third to [Combes et al., 2020] and picks the best between the two returned answers by testing on the remaining trials. In order to see this, let us look at the following informal function class.

**Assumption 3** (Informal)**.** *Let $F$ be the class of unimodal functions that satisfy Assumption 2 only on a sub-interval $R \in [0, 1]$ containing the optimum, and outside of this sub-interval we make no further assumptions other than unimodality.*

Under Assumption 3, our algorithm will eliminate everything outside of $R$ in $\mathrm{poly}(1/|R|)$ time, using essentially a discretization style analysis. However, once this elimination occurs, the algorithm will automatically switch to the faster $O(1/\sqrt{B})$ rate. Note that for any $|R| > 0$, CAB1 will not achieve a $O(1/\sqrt{B})$ asymptotic rate (due to its discretization it cannot hope for better than $O(1/B^{1/3})$), while for $|R| < 1/4$, the algorithm in [Combes et al., 2020] may not converge. Thus, this is a class of functions for which we outperform the minimum of the two baselines. Moreover, this class of functions that are "well-behaved near the optimum" is reasonably natural: indeed our Figure 1 seems to exhibit such behavior.

# 4 MULTIDIMENSIONAL PROBLEMS VIA COORDINATE ASCENT

In this section, we describe a simple approach to generalize our unimodal optimization algorithm to higher dimensions. While in the previous section the threshold parameter $\tau$ was essentially a nuisance factor and could be safely set to zero, here we will need the threshold to overcome a technical difficulty: we would like to ensure that any time a point $x$ is eliminated by Algorithm 2, it is possible to identify a $x^+$ for which $F(x_\star) - F(x^+)$ is significantly smaller than $F(x_\star) - F(x)$. We conjecture that $\tau$ is not actually required for this task, but this is left as an open question.

We consider functions $F : [0, 1]^d \to [0, 1]$ such that for every point $x \in [0, 1]$, the $d$ 1-D functions given by restricting $F$ to each coordinate axis through $x$ are all unimodal. This assumption suggests a natural iterative strategy: we will initialize our algorithm at some $x_1 \in [0, 1]^d$. Then, we run a copy of our 1-D algorithm on each coordinate independently, choosing samples along each coordinate in a round-robin fashion. If one of the copies is able to successfully eliminate the original point $w_1$, then we let the output of this copy be a next iterate $w_2$ and repeat the process.

The analysis of this method proceeds by showing that (1) $F(x_{t+1}) \geq F(x_t) + \tau$ for all $t$, and that (2) the number of samples required to identify $x_{t+1}$ is not too large, so long as an appropriate $x_{t+1}$ exists. Combined, this means that after at most roughly $1/\tau$ iterations of this procedure, we will converge to a $\tau$-approximate local optimum (Definition 2)

We provide an analysis of this algorithm for the case of Lipschitz $F$ in Theorem 5. It follows from an alternative analysis of Algorithm 2, which tells us that Algorithm 2 will quickly eliminate any non-local minimum point. For details, please see Lemma 5 in the appendix. Informally, this result tells us that with a budget of $B$ samples, we will

**Algorithm 3:** Unimodal Coordinate Ascent

$t = 1$
Choose an arbitrary $w^t = (w_1^t, \dots, w_d^t) \in [0,1]^d$
**while** $N \le B - d$ **do**
  Given a starting point $w^t \in [0,1]^d$, initialize $d$ copies of Algorithm 2 with $\delta = 6/dt^2\pi^2$, budget set to $\infty$ and threshold $\tau$ where the $i$th copy considers the function $F_i : [0,1] \to [0,1]$
  $F_i(x) = F(w_1^t, \dots, w_{i-1}^t, x, w_{i+1}^t, \dots, w_d^t)$.
  Let $\Delta_{ti}^t$ be the active interval associated with the $i$th copy.
  Let $I_{\text{best},i}^t$ be the current value of $I_{\text{best}}$ maintained by the $i$th copy.
  **while** $w_i^t \in \Delta_i^t$ for all $i$ **do**
    **if** $N \le B - d$ **then**
      Returns $w^t$.
    **end if**
    Let each copy sample one additional point.
    $N = N + d$.
  **end while**
  Suppose the $j$th copy eliminated $w_j^t$ ($w_j^t \notin \Delta_j^t$).
  Let $w_j^{t+1}$ be a randomly selected point in $I_{\text{best},j}^t$.
  Set $w^{t+1} = (w_1^t, \dots, w_{j-1}^t, w_j^{t+1}, w_{j+1}^t, \dots)$.
  $t \leftarrow t + 1$.
**end while**
Return $w^t$.

find an $\tilde{O}(d^{1/4}/B^{1/4})$-approximate local minimum for a $d$-dimensional Lipschitz unimodal function.

**Theorem 5.** *There is a constant $C$ such that, given a budget of $B$ and failure probability $\delta < 1/2$, if we set $\tau = \max\left(\frac{2d\log(B/\delta)}{B}, \frac{37\cdot(dC^2L\log(B/\delta)^2)^{1/4}}{B^{1/4}}\right)$, Then with probability at least $1 - 2\delta$, Algorithm 3, returns a point $\hat{w}$ that is a $3\tau$-approximate local minimum. That is, for each coordinate $i$, for any $w_i^\star$ that differs from $\hat{w}$ only in the $i$th coordinate, we have $F(w_i^\star) \le F(\hat{w}) + 3\tau$*

## 5 EXPERIMENTS

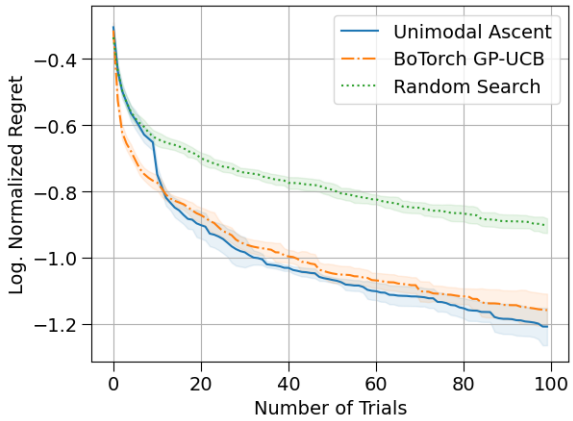
We tested our algorithm on machine learning hyperparameter optimization (HPO) problems. We used the continuous variant of HPO-B benchmark by [Arango et al., 2021, Sec 5.4]. The blackbox functions being optimized are the (approximated) validation accuracy of models for different hyper-parameter settings. The models vary across several tasks like training SVMs, GLMNets etc on various datasets. The approximation is done through XGBoost surrogate functions. We used the "HPO-B-v3 Meta-test split" as detailed in Arango et al. [2021]. It consists of 86 tasks in 16 distinct search spaces. We followed the exact test protocols as defined in their open source package.

**Benchmark algorithms.** The state of the art for HPO often involves using Gaussian Process (GP) regressors to derive surrogate functions and optimize them. We used the GP-UCB implementation included in the HPO-B open source package. It uses the GP implementation in BoTorch [Balandat et al., 2019] with UCB coefficient of 0.1 (this method was shown to have strong performance on the benchmarks [Arango et al., 2021]). We also use random search over the domain as another baseline. In order to be fair to all the algorithms, we initialize each of them at a random point for any of the benchmark tasks. All algorithms are run five times for every task.

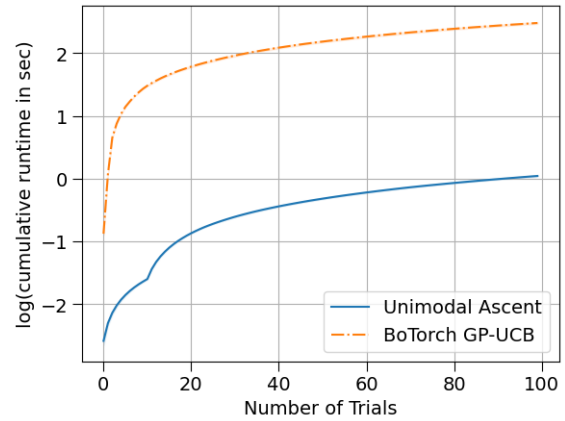
**Algorithm 4:** Modified Unimodal Coordinate Ascent

Set $N, t = 10$. Choose 10 random points $[0,1]^d$ and set arbitrary $w^t$ as the one with the best observed value.
**while** $N \le B - d$ **do**
  Given a starting point $w^t \in [0,1]^d$, initialize $d$ copies of Algorithm 6 with $\delta = 6/dt^2\pi^2$, budget set to $\infty$ where the $i$th copy considers the function $F_i : [0,1] \to [0,1]$
  $F_i(x) = F(w_1^t, \dots, w_{i-1}^t, x, w_{i+1}^t, \dots, w_d^t)$.
  Let $\Delta_{ti}^t$ be the active interval associated with the $i$th copy.
  Let $I_{\text{best},i}^t$ be the current value of $I_{\text{best}}$ maintained by the $i$th copy.
  **while** $w_i^t \in \Delta_i^t$ for all $i$ **do**
    **if** $N \le B - d$ **then**
      Return $w^t$.
    **end if**
    Let $i$ be the dimension sampled from distribution proportional to $\{\exp(s_1), \cdots, \exp(s_d)\}$. Here $s_i$ is the observed standard deviation of the values for dimension $i$ so far.
    Run one more epoch of Algorithm 6 for dimension $i$.
    $N = N +$ <#new points sampled>.
  **end while**
  Suppose the $j$-th copy is one such copy that eliminated $w_j^t$ ($w_j^t \notin \Delta_j^t$) and $\Delta_{t,j}$ is smallest among all eliminating dimensions.
  Let $w_j^{t+1}$ be the best point selected in $I_{\text{best},j}^t$, in terms of observed value.
  Set $w^{t+1} = (w_1^t, \dots, w_{j-1}^t, w_j^{t+1}, w_{j+1}^t, \dots)$ and begin a new round (reuse points if available).
**end while**
Return $w^t$.

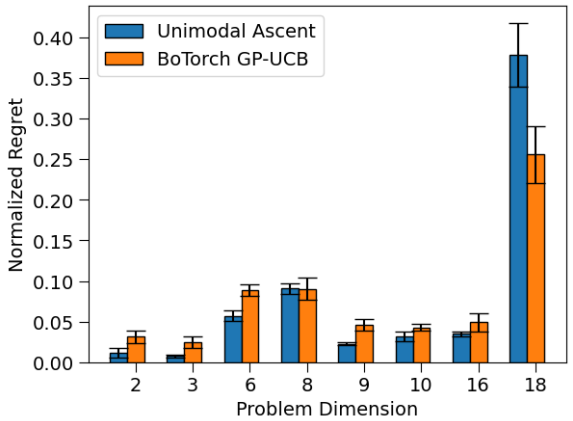
**Our implementation.** We make some practical modifications to Algorithm 3 for the real world benchmarks, both to improve runtime and also to acknowledge that the true black-box functions might not exactly adhere to our assumptions. First, instead of comparing every possible pair of disjoint intervals $(I_1, I_2)$, we compare only pairs where $|I_1| = |I_2|$ and both are powers of 2. This significantly reduces the number of pairs to compare, and an inspection of our proof techniques shows that it will only harm constants



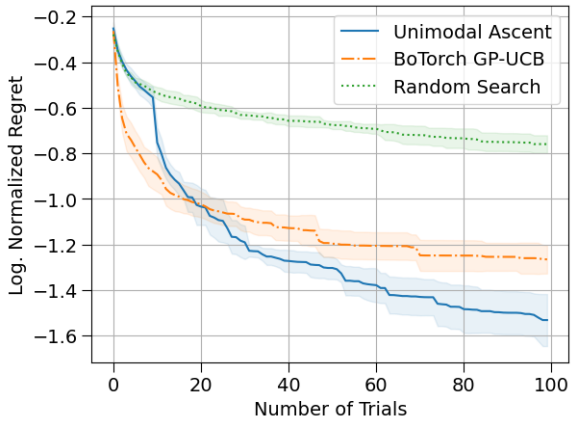

(a) Regret across all tasks




(b) Cumulative runtime




(c) Regret vs Dimension




(d) Regret on lower dimensional tasks

Figure 2: In (a), we plot the normalized regret as a function of number of trials (observations) averaged over 86 benchmark tasks. Each task is repeated 5 times and we plot the corresponding standard errors. It can be seen that our algorithm (Unimodal Ascent) outperforms the baselines over the majority of the x-axis. In (b), we compare the run-times of our algorithm and BoTorch GP-UCB as a function of number of trials. In (c), we plot the average regret of our algorithm and BoTorch-GP as a function of problem dimension. It can be seen that we outperform the GP on all dimensions except 8 and 18. In (d), we plot the normalized regret as a function of number of trials for problems with dimensions $\leq 6$.

in the bounds. Further, we never sample the same point twice, but instead *re-use* the previous sample every time it is requested in Algorithm 1. This again only harms constants in the bounds. Next, in all tasks, we start our algorithm with 10 random points in the domain, and then choose the point with the best observed value as the starting point for the unimodal coordinate ascent. In practice this leads to a good initialization of our algorithm. Note that we count those ten points as trials expended by our algorithm so that it is a fair comparison with the benchmarks. In Algorithm 3, if at any point there are multiple coordinates that can eliminate the current point (line 12), we choose the coordinate along which the maximum area has been eliminated. Finally, instead of letting each coordinate sample a point in line 9, we choose the coordinate that can sample the next point from a distribution that up-weights coordinates that have shown more variation in past trials. We also set the threshold parameter $\tau$ to zero. In the interest of space, we provide the pseudo-code of this modified algorithm as Algorithm 4 (more details in Appendix D).

**Regret on the full benchmark.** In Figure 2a, we plot the *normalized regret* as a function of number of trials/ observations. We adopt the definition of normalized regret from [Arango et al., 2021] i.e given the points $\{x_1, x_2, \cdots, x_t\}$ chosen by the algorithm till time $t$, the normalized regret is

$$r(t) = \frac{F(x_*) - \max_{s \in [t]} F(x_s)}{F(x_*) - F(x_{min})}$$

where $x_{min}$ is the point with the lowest objective value. It can be observed that our simple unimodal ascent algorithm outperforms a state of the art GP averaged over all the 86 real world benchmark HPO tasks. This is quite remarkable since the benchmark tasks might not strictly adhere to our assumptions and the GP has been carefully tuned by Arango et al. [2021] for good performance on the HPO-B tasks. We also provide a comparison w.r.t a version of the Zooming Algorithm [Kleinberg et al., 2008] in Appendix D.1.

**Runtime.** Figure 2b shows the cumulative runtime of our algorithm against the BoTorch GP as a function of number of observations. The logarithmic scale in the y-axis shows that our algorithm is orders of magnitude faster than the GP, owing to its simplicity. All our experiments were performed on Intel(R) Xeon(R) CPU E5-2620 v4 @ 2.10GHz machines with NVIDIA Tesla P100 GPU.

Indeed, it is well-known that generating the $N$-th suggestion from the GP takes $O(A(DN^2 + N^3) + B(DN + N^2))$ time, where $D$ is search space dimension, $A$ is the number of kernel hyper-parameter optimization (also known as *ARD*) iterations, and $B$ is the number of acquisition function optimization iterations [Garnett, 2022].

In comparison, generating the $N$-th suggestion from Algorithm 4 takes $O(N^2)$ time, up to logarithmic factors. This can be a significant advantage when tuning a lightweight model or optimizing with a large budget on the number of evaluations.

**Dimension of search-space.** Now we dive deeper into the behavior of the algorithms in search-spaces of varying dimensions. The HPO-B benchmarks contain search spaces of dimensions ranging from 2 to 18. We expect our algorithm to work significantly better than the GP for lower dimensional problems in the presence of approximately unimodal structure. This phenomenon can be observed in Figure 2c where we plot the average normalized regret achieved at the end of 100 trials, broken down by search-space dimension. Our gains are much more significant for lower dimensions. The BoTorch GP is only better than us for problems with dimension 18 and comparable to us on problems with dimension 8.

In Figure 2d, we plot regret as a function of number of trials, for tasks with dimension less than or equal to 6 (more than 38% of the search-spaces). This shows that we are significantly better than the GP for these tasks. In conclusion our algorithm is simpler and faster than the state of the art BoTorch GP, while being more performant on average on the HPO-B benchmark.

## 6 CONCLUSION

We describe a new algorithm for black-box optimization of unimodal functions. Our algorithm is based upon the intuitive idea that one can estimate the objective value at any given point by sampling several nearby points, rather than sampling the same point many times. This allows us to uniformly cover the input space and gain more information about the shape of the function while still employing a simple elimination test to remove suboptimal regions of the input quickly. We demonstrate theoretically that this method matches prior work, and empirically we are able to outperform more advanced methods based upon Gaussian processes. We conjecture that this capability arises from the unimodality assumption providing an informative prior about the overall shape of the objective (even if it is not strictly true in all practical settings).

Our work suggests at least three natural avenues for future investigation. Perhaps the most pressing is improving the method of extension to high dimensional problems. We adopt a relatively simple coordinate ascent strategy, while one might hope for a more refined algorithm that views all dimensions at once. Nevertheless, even this naive approach performs surprisingly well in our experimental study. Next, although our theoretical development matches prior work in the 1D setting, there are classes of functions missing from our analysis, notably the *convex* $F$, which are of course also unimodal. Thus, it would be valuable to improve the analysis or algorithm to achieve the optimal $O(1/\sqrt{T})$ convergence rates for convex functions.

## References


Alekh Agarwal, Dean P Foster, Daniel J Hsu, Sham M Kakade, and Alexander Rakhlin. Stochastic convex optimization with bandit feedback. *Advances in Neural Information Processing Systems*, 24, 2011.

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

Sebastian Pineda Arango, Hadi S Jomaa, Martin Wistuba, and Josif Grabocka. Hpo-b: A large-scale reproducible benchmark for black-box hpo based on openml. *arXiv preprint arXiv:2106.06257*, 2021.

Peter Auer, Ronald Ortner, and Csaba Szepesvári. Improved rates for the stochastic continuum-armed bandit problem. In *International Conference on Computational Learning Theory*, pages 454–468. Springer, 2007.

Maximilian Balandat, Brian Karrer, Daniel R Jiang, Samuel Daulton, Benjamin Letham, Andrew Gordon Wilson, and Eytan Bakshy. Botorch: Programmable bayesian optimization in pytorch. 2019.

Sébastien Bubeck et al. Convex optimization: Algorithms and complexity. *Foundations and Trends® in Machine Learning*, 8(3-4):231–357, 2015.

Herman Chernoff. Sequential design of experiments. *The Annals of Mathematical Statistics*, 30(3):755–770, 1959.

Richard Combes and Alexandre Proutiere. Unimodal bandits: Regret lower bounds and optimal algorithms. In *International Conference on Machine Learning*, pages 521–529. PMLR, 2014.

Richard Combes, Alexandre Proutière, and Alexandre Fauquette. Unimodal bandits with continuous arms: Order-optimal regret without smoothness. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 4(1):1–28, 2020.

John C Duchi, Michael I Jordan, Martin J Wainwright, and Andre Wibisono. Optimal rates for zero-order convex optimization: The power of two function evaluations. *IEEE Transactions on Information Theory*, 61(5):2788–2806, 2015.

Matthias Feurer and Frank Hutter. Hyperparameter optimization. In *Automated machine learning*, pages 3–33. Springer, Cham, 2019.

Roman Garnett. *Bayesian Optimization*. Cambridge University Press, 2022. in preparation.

Daniel Golovin, Benjamin Solnik, Subhodeep Moitra, Greg Kochanski, John Karro, and David Sculley. Google vizier: A service for black-box optimization. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1487–1495, 2017.

Elad Hazan, Adam Klivans, and Yang Yuan. Hyperparameter optimization: a spectral approach. In *International Conference on Learning Representations*, 2018.

Kevin G Jamieson, Robert Nowak, and Ben Recht. Query complexity of derivative-free optimization. *Advances in Neural Information Processing Systems*, 25, 2012.

Kwang-Sung Jun, Aniruddha Bhargava, Robert Nowak, and Rebecca Willett. Scalable generalized linear bandits: Online computation and hashing. *Advances in Neural Information Processing Systems*, 30, 2017.

Jack Kiefer. Sequential minimax search for a maximum. *Proceedings of the American mathematical society*, 4(3): 502–506, 1953.

Robert Kleinberg. Nearly tight bounds for the continuum-armed bandit problem. *Advances in Neural Information Processing Systems*, 17, 2004.

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

Alex Krizhevsky. Learning multiple layers of features from tiny images. 2009.

Luis Miguel Rios and Nikolaos V Sahinidis. Derivative-free optimization: a review of algorithms and comparison of software implementations. *Journal of Global Optimization*, 56(3):1247–1293, 2013.

Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando De Freitas. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2015.

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

Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. *Advances in neural information processing systems*, 25, 2012.

Niranjan Srinivas, Andreas Krause, Sham M Kakade, and Matthias Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. *arXiv preprint arXiv:0912.3995*, 2009.

Kei Terayama, Masato Sumita, Ryo Tamura, and Koji Tsuda. Black-box optimization for automated discovery. *Accounts of Chemical Research*, 54(6):1334–1346, 2021.

Jia Yuan Yu and Shie Mannor. Unimodal bandits. In *International Conference on Machine Learning*. PMLR, 2011.