Master research Internship

Internship report

Hierarchical Bandits for "Black Box" Optimization

Domain: Optimization - Machine Learning

Author:
Xuedong Shang

Supervisor:
Emilie Kaufmann
Michal Valko
SequeL - Inria Lille

Abstract Heuristics like Monte-Carlo Tree Search (MCTS), which trades off well *exploration* and *exploitation*, are widely used for sequential global optimization problems, and has led to some great success especially in game AI designing. In many cases, the exploration phase follows the famous *optimism in the face of uncertainty* principle, which is encountered in the so-called *multi-armed bandit* problem. However, recent studies on these models shows that they are not optimal for the optimization purpose, and that methods based on *best arm identification* are preferred. During this internship, some approaches based on these new statistic tools are investigated both in a practical and theoretical way.

Keywords —- reinforcement learning, global optimization, multi-armed bandits, simple regret, cumulative regret, hyperparameter optimization.

Contents

Introduction

Sequential global optimization consists in optimizing some complicated function by using a sequence of (noisy) observations of this function. Optimization could be for example maximization of a *reward* or minimization of a cost. It is a crucial problem in many different domains such as biology and chemistry Floudas and Pardalos 2000, engineering Wang and Shan 2007, bioinformatics Moles et al. 2003, finance Ziemba and Vickson 2006, etc. In these cases, we often do not make extra hypothesis on the regularity of the function we want to optimize (so-called "black box"). Thus it can be very costly to evaluate the function and a good strategy for choosing the next observation is needed so that we can find the optimum as quickly as possible. Recently, this kind of black box optimization is motivated in particular by applications in automatic hyper-parameter configuration of machine learning algorithms Hoffman et al. 2014, Li et al. 2016.

In the past few years, such optimization algorithms have been widely inspired by literature of multi-armed bandit models. These algorithms are based on a hierarchical exploration (in a tree form) of the domain of the function, with the help of the optimism principle for choosing which part of the tree to explore. This work brought some breakthroughs especially in the field of AI designing, e.g. for the game of Go Silver et al. 2016.

A simple way to describe the multi-armed bandit scenario is to consider K arms labeled by integers from 1 to K. Each arm $k \in \{1, \dots, K\}$ is characterized by an unknown distribution ν_k. At each step t, an arm k_t is selected and some reward $r_t \sim \nu_{k_t}$ is returned. The optimism principle is used when we are interested in maximizing the sum of rewards. Meanwhile, another objective could have been preferred, which is to decide as quickly as possible which arm has the highest reward on average. Recent works showed that optimal algorithms for this kind of best arm identification problems are totally different from those for reward maximization problems.

Dealing with the sequential global optimization problem using the hierarchical exploration with the help of best arm identification techniques instead of the classic optimistic approaches has raised much attention recently. For instance, most existing algorithms for the MCTS problem are variants of the Upper-Confidence Tree (UCT) algorithm in Kocsis and Szepesvári 2006 in which the exploration phase follows the optimism in the face of uncertainty principle. The goal is to use a different approach, in which the exploration phase will be based on a process of best arm identification, like LUCB Kalyanakrishnan et al. 2012. Several methods can be considered for this purpose, in particular methods of hierarchical optimization (e.g. hierarchical optimistic optimization Grill et al. 2015, Bubeck et al. 2011) and methods using Gaussian processes (e.g. Bayesian optimization Brochu et al. 2010). The main objective of this internship is to investigate these two types of approaches and their applications.

Contribution The contribution of this internship is twofolds:

(i) The theoretical part focuses on methods of hierarchical optimization. We show that milder assumptions on the target function can be applied to one of the algorithms and we explain why it is important.

(ii) Regarding the numerical part, we first compare hierarchical methods with Bayesian methods which is rarely done before (for some reasons we explain later). Then we focus on their application to hyper-parameter optimization problem.

Structure For the rest of this report, we begin by a more formal definition of the bandit problem and some classic methods as well. Then we concentrate on black box optimization, for which we present two kinds of methods: Bayesian optimization and hierarchical optimization. Finally, we compare these techniques and discuss about how they can be used in the context of hyper-parameter optimization for machine learning problems before concluding.

1 Multi-armed Bandit Problem

In this section, we formulate the bandit problem in a more rigorous way and present a classic algorithm for this problem: Upper-Confidence Bound (UCB) algorithm Auer et al. 2002.

1.1 Problem Formulation

We consider K arms that follow K unknown $[0,1]$-valued distributions $(\nu_k)_{1\leq k\leq K}$. At time t, a player plays one arm $k_t \in \{1,\dots,K\}$ and receives a reward $r_t \sim \nu_{k_t}$. Here x_t is an independent observation of the distribution corresponding to the chosen arm.

Let μ_k be the expectation of the unknown distribution ν_k and μ^* be the expectation of the optimal arm. Since the laws of each arm are unknown, one must explore every arm to collect information, bu in the mean time one may want to exploit the most profitable arm as much as possible. This is the famous *exploration-exploitation dilemma*.

A *policy* or an *allocation strategy* is an algorithm which chooses at step n an arm k_n to play based on the past plays. Sometimes, this allocation strategy is coupled with a recommendation strategy, that selects an arm j_n as a guess for the best arms (notice that j_n can surely be different from k_n). A simple example of recommendation could be the arm with the highest empirical mean until now.

In order to evaluate the performance of a given policy, two criteria related to the notion of regret are proposed, which allow us to define the speed of convergence of the average reward obtained by the policy towards the average optimal reward.

Definition 1 (Simple regret)**.** *At time n, a given policy which observes a sequence of rewards $(r_t)_{1\leq t\leq n}$ and recommendations $(j_t)_{1\leq t\leq n}$ suffers from a simple regret:*

$$S_n \coloneqq \mu^* - \mu_{j_n}.$$

The simple regret measures the difference between the expected reward of the optimal arm and the recommended arm at this moment. In practice, another criterion is often considered as well.

Definition 2 (Cumulative regret)**.** *At time n, a given policy which observes a sequence of rewards $(r_t)_{1\leq t\leq n}$ suffers from a cumulative regret:*

$$R_n \coloneqq n\mu^* - \sum_{1\leq t\leq n} \mu_{k_t}.$$

The cumulative regret measures the difference between the expected cumulative reward obtained by the optimal arm and the cumulative reward returned by the given policy.

We are particularly interested by the expectation of the cumulative regret $\mathbb{E}[R_n]$. Let us denote $\Delta_k = \mu^* - \mu_k$ as the difference between the optimal arm and arm k and $T_k(n)$ as the number of

times that arm k has been played until time n. Then the expectation of the cumulative regret can be reformulated as follows:

$$\mathbb{E}[R_n] = \mathbb{E}\left[\sum_{k=1}^{K} T_k(n)\Delta_k\right].$$

Basically speaking, a good policy for (cumulative) regret minimization should choose any sub-optimal arm as rarely as possible.

Remark 1. It is easy to see that for any regret minimization strategy, results on cumulative regret can be readily extended to simple regret if we choose some specific recommendation strategy. For instance, at time n, if the algorithm recommends a sampled arm uniformly at random, then we have:

$$\mathbb{E}[S_n] \leq \mathbb{E}\left[\frac{R_n}{n}\right].$$

1.2 UCB Algorithm

The UCB algorithm, popularized by Auer et al. 2002, is one of the first strategies that achieves a uniform logarithmic regret over n and it follows the optimism in the face of uncertainty principle. That is to say, despite lack of knowledge in which actions are best, we can still construct an optimistic guess that picks an optimal arm in the most favorable environments that are compatible with the observations. Here by "compatible environments" we mean the set of possible distributions of the arms that are likely to have generated the observed rewards. The simplest version of UCB is given below.

Algorithm 1: UCB_1

Initialize: At first, the UCB policy pulls each arm once.

1 **Loop:** At time $n+1$, the UCB policy pulls the arm with largest B-values:

$$x_{n+1} \in \arg\max_{1\leq k\leq K} B_{(n+1),T_k(n)(k)},$$

where the B-value of an arm k is defined as:

$$B_{t,s}(k) \coloneqq \hat{\mu}_{k,s} + \sqrt{\frac{3\log t}{2s}},$$

where $\hat{\mu}_{k,s}$ is the average reward of the first s rewards obtained by arm k.

The policy above follows the optimism in the face of uncertainty principle and the B-values defined are actually upper-confidence bounds on μ_k. Formally speaking, for the UCB algorithm, it is proved that

$$\mathbb{P}(B_{t,s}(k) \geq \mu_k) \geq 1 - t^{-3}, \forall s \in \{1, \ldots, t\}.$$

This property is ensured by the famous Chernoff-Hoeffding inequality Hoeffding 1963. We can then deduce an upper bound on the average number of times that a sub-optimal arm is played.

Proposition 3. *Each sub-optimal arm k is played at most*

$$\mathbb{E}[T_n(k)] \leq 6\frac{\log n}{\Delta_k^2} + \frac{\pi^2}{3} + 1$$

times on average.

The expectation of the cumulative regret can then be easily bounded as shown in the corollary below.

Corollary 4. *The cumulative regret of the UCB policy is bounded as*

$$\mathbb{E}[R_n] = \mathbb{E}\left[\sum_{k=1}^{K} T_k(n)\Delta_k\right] = \sum_{k=1}^{K} \Delta_k \mathbb{E}[T_k(n)] \leq 6 \sum_{\Delta_k > 0} \frac{\log n}{\Delta_k} + K(\frac{\pi^2}{3} + 1).$$

UCB is quite simple to understand and implement, and it has a great influence on its successors. The two terms introduced in the definition of B-value here actually represent a trade-off between exploration and exploitation. And this is also the most crucial point to be taken into account while designing other algorithms. Later in this report, we will see that all algorithms presented follow this principle.

Remark 2. UCB aims at minimizing the cumulative regret (i.e. maximizing the reward), which may not be an appropriate strategy in some circumstances. An alternative philosophy, say best arm identification, can be considered as well. And this will also be our focus for the future work.

2 Hierarchical Optimization

In this section, we first reformulate the definitions given previously in the context of multi-armed bandit problem in a more general way. We consider the problem of optimizing sequentially an unknown noisy function $f : \mathcal{X} \to \mathbb{R}$ of which the cost of function evaluation is high.

At each step t, a strategy picks an action $\mathbf{x}_t \in \mathcal{X}$ and receives a reward $r_t = f(\mathbf{x}_t) + \epsilon_t$ where ϵ_t is the noise. After n steps, the strategy returns a guess for a point maximizing f, denoted by $\mathbf{x}(n)$ (which is the counterpart of the recommended arm j_n in the bandit case, which corresponds to a finite set $\mathcal{X}$). We can then define the simple regret, which is also called *optimization error* in an optimization setting, in the same way as in the bandit setting,

$$S_n \coloneqq \sup_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x}) - f(\mathbf{x}(n)).$$

Suppose that f reaches its upper bound, thus we can denote $f^* \coloneqq f(\mathbf{x}^*) = \sup_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x})$ throughout the rest of this report. The cumulative regret is simply defined as

$$R_n \coloneqq \sum_{1\leq t\leq n} (f(\mathbf{x}^*) - f(\mathbf{x}_t)).$$

Notice that a desirable property of an optimization algorithm would be *no-regret*, i.e. $\lim_{n\to\infty} R_n/n = 0$.

In the rest of this section as well as in the next one, two dominant types of approaches for this problem are introduced, i.e. hierarchical optimization and Bayesian optimization. We present different algorithms for each of these two types and we are particularly interested in their theoretical guarantees.

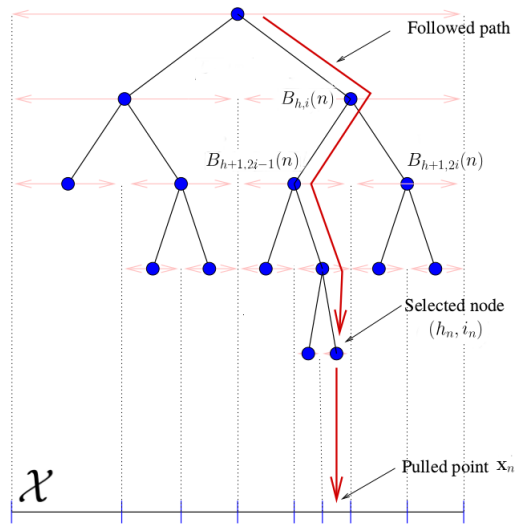


Figure 1: Illustration of the node selection in round n. In this example, $B_{h+1,2i-1}(n) > B_{h+1,2i}(n)$, thus the current optimistic path includes the node $(h+1, 2i-1)$ rather than the node $(h+1, 2i)$.

2.1 Hierarchical Optimistic Optimization

Indeed, the previous problem setting can be seen as a so-called $\mathcal{X}$-armed bandits problem, which means the set of arms $\mathcal{X}$ is allowed to be a generic measurable space. One of the solutions to this problem is the hierarchical optimistic optimization (HOO) algorithm Bubeck et al. 2011. The main idea of HOO is to exploit as much knowledge of f as possible around its maxima, while it does not care a lot about other parts of the space. For that purpose, a tree-formed partitioning is proposed such that nodes which are deeper in the tree represent smaller measurable sub-regions of $\mathcal{X}$.

2.1.1 HOO Algorithm

HOO is an *anytime* algorithm that relies on a tree-formed hierarchical partitioning $\mathcal{P} = \{\mathcal{P}_{h,i}\}$ defined recursively as follows,

$$\mathcal{P}_{0,1} = \mathcal{X},$$

$$\mathcal{P}_{h,i} = \mathcal{P}_{h+1,2i-1} \cup \mathcal{P}_{h+1,2i}.$$

As a corollary, at every depth $h \geq 0$, all the nodes form a partition of the input space $\mathcal{X}$,

$$\mathcal{X} = \bigcup_{i=1}^{2^h} \mathcal{P}_{h,i}.$$

Remark 3. Here a simple example of binary tree partitioning is given, notice that each $\mathcal{P}_{h,i}$ can also be split into several regions of the same size. In the rest of this section, we remain in this binary tree situation for the sake of simplicity.

At each round of HOO, the function is evaluated at one point within an unexplored node at some level h. And it always chooses the node whose B-value $B_{h,i}(t)$ is the highest. $B_{h,i}(t)$ is defined

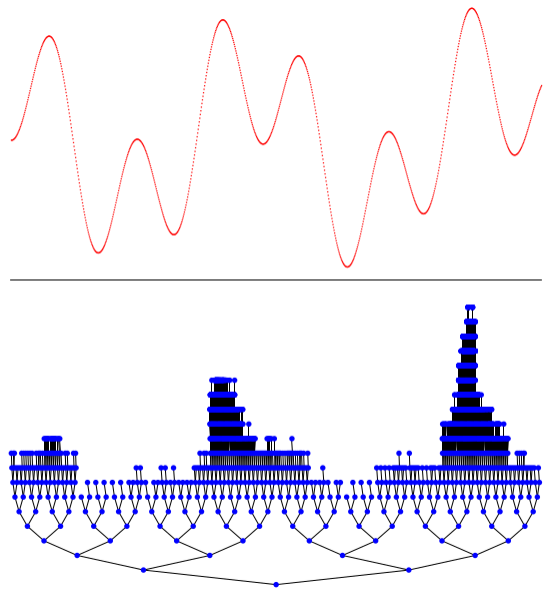


Figure 2: Extracted from Bubeck et al. 2011, the tree built by HOO over 10,000 rounds of the function $x \in [0,1] \mapsto 1/2(\sin(13x)\sin(27x)+1$.

as follows,

$$B_{h,i}(t) := \begin{cases} \min\{U_{h,i}(t), \max\{B_{h+1,2i-1}(t), B_{h+1,2i}(t)\}\} & \text{if } (h,i) \in \mathcal{T}_n \\ \infty & \text{otherwise} \end{cases}$$

where $\mathcal{T}_n$ is the covering tree explored at step n and $U_{h,i}(t)$ is defined as,

$$U_{h,i}(t) := \begin{cases} \hat{\mu}_{h,i}(t) + \sqrt{\frac{2\ln(t)}{T_{h,i}(t)}} + \nu\rho^h & \text{if } T_{h,i}(t) > 0 \\ \infty & \text{otherwise} \end{cases}$$

where $\hat{\mu}_{h,i}(t)$ is the empirical mean of all evaluations done in the cell $\mathcal{P}_{h,i}$ (and its descendants), and $T_{h,i}(t)$ is the number of these evaluations. Here $U_{h,i}(t)$ can be interpreted as an upper confidence bound on $f(\mathbf{x})$ where $\mathbf{x} \in \mathcal{P}_{h,i}$, and it is easy to see that $B_{h,i}(t)$ is also an upper confidence bound, but tighter.

HOO chooses a path from root to leaf that maximizes the minimum value $U_{h,i}(t)$ among all cells at level h. At the leaf, a new point is then sampled randomly. More details can be found in Algorithm 2. Fig. 1 illustrates how HOO chooses its path and Fig. 2 shows the first $10,000$ rounds of the tree construction by HOO for a specific function.

Here the third term $\nu\rho^h$ in $U_{h,i}(t)$ is supposed to be a bound on the difference $f(\mathbf{x}^*) - f(\mathbf{x})$ over a region at depth h that contains one of the maxima of the function. As a consequence, the HOO algorithm needs some prior knowledge on the smoothness of the function as it needs some information on $\nu\rho^h$.

Remark 4. In practice, if we know the number of rounds n in advance, we can change the $\log(t)$ term in the line 18 into $\log(n)$, which can significantly accelerate the processing time since at each round, only the cells along the current optimistic path need to be updated.

Algorithm 2: Hierarchical Optimistic Optimization

Input : $\nu_1 > 0$; $\rho \in (0,1)$; $c > 0$; tree partition $\{\mathcal{P}_{h,i}\}$.
Initialize: $\mathcal{T}_1 = \{(0,1)\}$; $B_{1,1}(1) = B_{1,2}(1) = +\infty$

```
for t ← 1 to n do
    (h, i) ← (0, 1); P_t ← {(h, i)};
    while (h, i) ∈ T_t do
        if B_{h+1,2i-1} ≥ B_{h+1,2i} then
            (h, i) ← (h + 1, 2i − 1);
        else
            (h, i) ← (h + 1, 2i);
        end
    end
    (h_t, i_t) ← (h, i);
    Pull an arbitrary arm in P_{h_t,i_t} and obtain r_t;
    T_t ← T_t ∪ {(h_t, i_t)};
    for (h, i) ∈ P_t do
        T_{h,i}(t) ← T_{h,i}(t) + 1;
        Update μ̂_{h,i}(t);
    end
    for (h, i) ∈ T_t do
        U_{h,i}(t) ← μ̂_{h,i}(t) + sqrt(2 log(t) / T_{h,i}(t)) + ν_1 ρ^h;
    end
    B_{h_t+1,2i_t−1}(t) ← ∞; B_{h_t+1,2i_t}(t) ← ∞;
    T ← T_t;
    while T ≠ {(0, 1)} do
        Take any leaf (h, i);
        B_{h,i}(t) ← min{U_{h,i}(t), max{B_{h+1,2i−1}(t), B_{h+1,2i}(t)}};
        T ← T − {(h, i)};
    end
end
```

2.1.2 Analysis

A global optimization problem without any assumptions on the regularity of f would be almost a "mission impossible". Most of the algorithms make a very weak hypothesis that f possesses at least some local smoothness. This smoothness is often quantified by a certain semi-metric l, e.g. $l(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|^\alpha$ with $\alpha < 1$. One simple example of smoothness assumption could be:

$$f(\mathbf{x}^*) - f(\mathbf{x}) \leq l(\mathbf{x}^*, \mathbf{x})$$

for x close to one of the function maxima.

As for HOO, we only need a notion of dissimilarity.

Assumption 1 (Dissimilarity)**.** *The arm space $\mathcal{X}$ is equipped with a non-negative function $l : \mathcal{X}^2 \to \mathbb{R}$ such that $\forall \boldsymbol{x} \in \mathcal{X}, l(\boldsymbol{x}, \boldsymbol{x}) = 0$.*

With this dissimilarity function, we can define the diameter of a subset $A \subseteq \mathcal{X}$ as diam(A) $:= \sup_{\mathbf{x},\mathbf{y}\in A} l(\mathbf{x},\mathbf{y})$, while an l-ball of radius ϵ and center $\mathbf{x} \in \mathcal{X}$ can be defined as $\mathcal{B}(\mathbf{x},\epsilon) := \{\mathbf{x}' \in \mathcal{X} : l(\mathbf{x},\mathbf{x}') \leq \epsilon\}$. We then make the following assumptions on the local smoothness.

Assumption 2 (Local smoothness)**.** *Assume that there exist $\nu_1, \nu_2 > 0$ and $\rho \in (0,1)$ s.t. for all nodes $(h,i),(h,j)$ and for all $\boldsymbol{x},\boldsymbol{y} \in \mathcal{X}$:*

(i) $\mathrm{diam}(\mathcal{P}_{h,i}) \leq \nu_1 \rho^h$,

(ii) $\exists \boldsymbol{x}^o_{h,i} \in \mathcal{P}_{h,i}$ *s.t.* $\mathcal{B}_{h,i} := \mathcal{B}(\boldsymbol{x}^o_{h,i}, \nu_2\rho^h) \subset \mathcal{P}_{h,i}$,

(iii) $\mathcal{B}_{h,i} \cap \mathcal{B}_{h,j} = \emptyset$,

(iv) $f^* - f(\boldsymbol{y}) \leq f^* - f(\boldsymbol{x}) + \max\{f^* - f(\boldsymbol{x}), l(\boldsymbol{x},\boldsymbol{y})\}$.

For all nodes (h,i), let us denote $f^*_{h,i} := \sup_{\mathbf{x}\in\mathcal{P}_{h,i}} f(\mathbf{x})$ and $\Delta_{h,i} := f^* - f^*_{h,i}$. With Assumption 1 and Assumption 2, we can prove a crucial lemma for the analysis of HOO (see Appendix A for the proof).

Lemma 5. *If $\Delta_{h,i} \leq c\nu_1\rho^h$ for some constant $c \geq 0$, then all arms in $\mathcal{P}_{h,i}$ are $\max\{2c, c+1\}\nu_1\rho^h$-optimal.*

The regret of an $\mathcal{X}$-armed bandits problem should depend on how fast the volumes of ϵ-optimal arms shrink as ϵ tends toward 0 as observed by Auer et al. 2007. Thus they defined a notion of *near-optimality dimension*, which measures the size of the ϵ-optimal space $\mathcal{X}_\epsilon := \{\mathbf{x} \in \mathcal{X} : f(\mathbf{x}) > f^* - \epsilon\}$ in terms of the *packing number* $\mathcal{N}(A, l, \epsilon)$ that represents the maximum number of disjoint l-balls of radius ϵ contained in the subset $A \subseteq \mathcal{X}$.

Definition 6 (Near-optimality dimension)**.** *For any constant $c > 0, \epsilon_0 > 0$, the (c,ϵ_0)-near-optimality dimension d_1 of f with respect to l is defined as*

$$d_1(c,\epsilon_0) := \inf\{d' \in \mathbb{R}^+ : \exists C > 0, \forall \epsilon \geq \epsilon_0, \mathcal{N}(\mathcal{X}_{c\epsilon}, l, \epsilon) \leq C\epsilon^{-d'}\}.$$

The cumulative regret bound of HOO is stated as follow.

Theorem 7. *Under Assumptions 1 and 2, let d be the $4\nu_1/\nu_2$-near-optimality dimension of f w.r.t l, then there exists γ s.t.*

$$\mathbb{E}[R_n] \leq \gamma n^{(d+1)/(d+2)} (\log(n))^{1/(d+2)}.$$

This result suggests that in the case of $d = 0$, the regret bound is $O(\sqrt{n\log(n)})$.

2.2 Parallel Optimistic Optimization

Parallel Optimistic Optimization (POO) Grill et al. 2015 is somewhat a "*meta-algorithm*" on top of HOO, and is also designed to be an anytime algorithm. It is under some milder assumptions compared to HOO. In fact, the semi-metric assumption made in HOO is not necessary. It does not exploit full information of the metric value, but only uses ν and ρ, thus a single assumption that relates directly f to the partitioning could be preferred.

2.2.1 POO Algorithm

The idea of POO is very simple, it uses HOO as subroutine in which several instances of HOO are run at the same time, hence the name of the algorithm. Each instance of HOO is run with a different (ν, ρ). At the end, it chooses the (ν^*, ρ^*) that performs the best and returns one of the points chosen randomly by the corresponding HOO instance.

Running an instance of HOO with (ν, ρ) that are far from the best one may cause underperformance of HOO. However POO analysis shows surprisingly that this *suboptimality gap* does not decrease too fast while the chosen (ν, ρ) roll away from the best one. It shows also that only $\ln(n)$ instances of HOO are needed to make sure that one of the instances performs well. Here n is the current number of function evaluations, meaning that we do not need to know the total number of rounds in advance, which makes POO an anytime algorithm.

2.2.2 Analysis

Given a global maximum $\mathbf{x}^*$, let us denote i_h^* the index of the only cell at depth h that contains $\mathbf{x}^*$. As we already mentioned, the POO analysis relates directly f to the partitioning, thus we make the following assumption on the local smoothness.

Assumption 3 (Local smoothness)**.** *There exist $\nu > 0$ and $\rho \in (0, 1)$ s.t.,*

$$\forall h \geq 0, \forall \boldsymbol{x} \in \mathcal{P}_{h,i_h^*}, f(\boldsymbol{x}) \geq f(\boldsymbol{x}^*) - \nu\rho^h.$$

In order to measure the complexity of the optimization problem directly in terms of partitioning, a new definition of *near-optimality dimension* is needed.

Definition 8 (Near-optimality dimension)**.**

$$d_2(\nu, \rho) = \inf\{d' \in \mathbb{R}^+ : \exists C > 0, \forall h \geq 0, \mathcal{N}_h(2\nu\rho^h) \leq C\rho^{-d'h}\},$$

where $\mathcal{N}_h(2\nu\rho^h)$ is the number of cells $\mathcal{P}_{h,i}$ s.t.,

$$\sup_{x \in \mathcal{P}_{h,i}} f(\boldsymbol{x}) \geq f(\boldsymbol{x}^*) - 2\nu\rho^h.$$

Intuitively, $\mathcal{N}_h(2\nu\rho)$ represents the number of cells that any algorithm needs to sample in order to find the maximum. Thus, a small near-optimality dimension makes the function easier to optimize.

An upper bound on the simple regret of POO can be then given in terms of the near-optimality dimension.

Theorem 9. *At step n, for any (ν, ρ) that verifies the smoothness assumption for POO such that $\nu \leq \nu_{\max}$ and $\rho \leq \rho_{\max}$, there exists κ s.t.,*

$$\mathbb{E}[S_n] \leq \kappa \left(\frac{\log(n)^2}{n} \right)^{\frac{1}{d(\nu,\rho)+2}},$$

where $\nu_{\max}$ and $\rho_{\max}$ are two optional parameters for the POO algorithm that can be set automatically as functions of n.

Notice that here only simple regret bound is reported for POO, while for HOO we can bound both cumulative and simple regret. The reason is that POO runs several non-optimal instances of HOO which can dramatically influence the cumulative regret.

2.2.3 Discussion

Assumptions made for the POO analysis (see Assumption 3) are milder than those of HOO (see Assumption 2). However it is assumed that the regret bound of HOO remains valid under these weaker assumptions. The reason why milder assumptions are proposed is quite simple since HOO does not exploit full information about the metric l. One natural question is whether we can indeed adapt POO assumptions to the HOO analysis.

Unfortunately, it seems to be a bit hard to do so. The main reason behind this relates to Lemma 5. Indeed, Lemma 5 ensures that once the local optimal point in a cell $\mathcal{P}_{h,i}$ is not far from the global optimum, then all the points in this very cell would not be very far from the global optimum neither. This, however, cannot be ensured only under Assumption 3.

As we already mentioned, POO is some kind of "meta-algorithm", which means it can be used on top of other hierarchical optimization algorithms than HOO. Therefore, as for the theoretical contribution of this report, we try to find another underlying algorithm which is able to keep its theoretical guarantee under POO assumptions.

2.3 High-Confidence Tree

The high-confidence tree (HCT) algorithm Azar et al. 2014 is another algorithm for the $\mathcal{X}$-armed bandits problem. The main advantage of HCT with respect to HOO is that it may handle the case of correlated bandit reward. However, it will not be our focus in this report, and we concentrate on the i.i.d variant of HCT (HCT-*iid*).

2.3.1 HCT Algorithm

We consider a binary covering tree $\mathcal{T}_n$ for the sake of simplicity. Unlike in HOO where an arbitrary arm $\mathbf{x} \in \mathcal{P}_{h,i}$ is pulled whenever the node (h,i) has been selected by the algorithm, in HCT, a representative point $\mathbf{x}_{h,i}$ is associated to every cell $\mathcal{P}_{h,i}$. Algorithm 3 shows the structure of HCT for which we have to clarify some notations.

Indeed, at each node (h,i) of the covering tree $\mathcal{T}_n$, the algorithm keeps track of some statistics regarding its representative arm $\mathbf{x}_{h,i}$. These include the empirical mean $\hat{\mu}_{h,i}(t)$ which is defined as

$$\hat{\mu}_{h,i}(t) \coloneqq \frac{1}{T_{h,i}(t)} \sum_{s=1}^{T_{h,i}(t)} r_{(h,i),s}$$

where $T_{h,i}(t)$ is the number of pulls of (h,i) (notice that this is different from HOO), and $r_{(h,i),s}$ denotes the s-th reward received by $\mathbf{x}_{h,i}$. Similar to HOO, the HCT algorithm keeps track of an upper confidence bound U-value

$$U_{h,i}(t) \coloneqq \hat{\mu}_{h,i}(t) + \nu\rho^h + c\sqrt{\frac{\log(1/\tilde{\delta}(t^+))}{T_{h,i}(t)}}$$

where $t^+ = 2^{\lceil \log(t) \rceil}$, $\tilde{\delta}(t) \coloneqq \min\{c_1\delta/t, 1/2\}$, and its corresponding B-value

$$B_{h,i}(t) \coloneqq \begin{cases} \min\{U_{h,i}(t), \max\{B_{h+1,2i-1}(t), B_{h+1,2i}(t)\}\} & \text{if } (h,i) \text{ is a leaf} \\ \infty & \text{otherwise} \end{cases}$$

Algorithm 3: High-Confidence Tree

Input : $\nu > 0$; $\rho \in (0,1)$; $c > 0$; tree partition $\{\mathcal{P}_{h,i}\}$; confidence δ.

Initialize: $\mathcal{T}_1 = \{(0,1),(1,1),(1,2)\}$; $H(1) = 1$; $U_{1,1}(1) = U_{1,2}(1) = +\infty$

1 **for** $t \leftarrow 1$ **to** n **do**
2 **if** $t = t^+$ **then**
3 **for** $(h,i) \in \mathcal{T}_t$ **do**
4 $U_{h,i}(t) \leftarrow \hat{\mu}_{h,i}(t) + \nu\rho^h + c\sqrt{\frac{\log(1/\tilde{\delta}(t^+))}{T_{h,i}(t)}}$;
5 **end**
6 **for** $(h,i) \in \mathcal{T}_t$ *backward from* $H(t)$ **do**
7 **if** (h,i) *is a leaf* **then**
8 $B_{h,i}(t) \leftarrow U_{h,i}(t)$;
9 **else**
10 $B_{h,i}(t) \leftarrow \min\{U_{h,i}(t), \max\{B_{h+1,2i-1}(t), B_{h+1,2i}(t)\}\}$;
11 **end**
12 **end**
13 **end**
14 $(h_t, i_t), P_t \leftarrow$ *OptTraverse*$(\mathcal{T}_t)$;
15 Pull $\mathbf{x}_{h_t,i_t}$ and obtain r_t;
16 $t \leftarrow t + 1$;
17 $T_{h_t,i_t}(t) \leftarrow T_{h_t,i_t}(t) + 1$;
18 Update $\hat{\mu}_{h_t,i_t}(t)$;
19 $U_{h_t,i_t}(t) \leftarrow \hat{\mu}_{h_t,i_t}(t) + \nu\rho^{h_t} + c\sqrt{\frac{\log(1/\tilde{\delta}(t^+))}{T_{h_t,i_t}(t)}}$;
20 *UpdateB*$(\mathcal{T}_t, P_t, (h_t, i_t))$;
21 $\tau_{h_t}(t) \leftarrow \lceil \frac{c^2 \log(1/\tilde{\delta}(t^+))}{\nu^2} \rho^{-2h_t} \rceil$;
22 **if** $T_{h_t,i_t}(t) \geq \tau_{h_t}(t)$ *and* (h_t, i_t) *is a leaf* **then**
23 *Expand*$((h_t, i_t))$;
24 **end**
25 **end**

which is designed to be a tighter upper confidence bound. Here, ν and ρ are the same kind of parameters as for HOO, and c, c_1 are two constants (see Appendix B for further discussion).

From line 2 to line 13 in Algorithm 3 we can see a refreshing phase of the HCT algorithm. Indeed, since HCT is an anytime algorithm just like HOO and POO, we need to update periodically the upper bounds statistics on every node in the covering tree. However, it is not necessary to refresh at every round since uncertainty terms depend on $\tilde{\delta}(t)^+$ which changes only at $t = 1, 2, 4, \ldots$.

Now the tricky part in this algorithm is the subroutine *OptTraverse* described in Algorithm 4 which tells us how to explore the covering tree, and returns a selected node and the current optimistic path. Indeed, unlike in HOO where every cell is sampled only once at each level, HCT proposes to sample one node a sufficient number of times so as to reduce the uncertainty sufficiently before expanding this very node. Here $\tau_h(t)$ is considered to be a threshold when the expansion of a node

happens,

$$\tau_h(t) = \lceil \frac{c^2 \log(1/\tilde{\delta}(t^+))}{\nu^2} \rho^{-2h} \rceil.$$

Intuitively, the expansion of a node occurs when the uncertainty over the rewards in $\mathcal{P}_{h,i}$ becomes smaller than the resolution of the node, which means the third term $c\sqrt{\log(1/\tilde{\delta}(t^+))/T_{h,i}(t)}$ in the upper confidence bound $U_{h,i}(t)$ is roughly equal to the second term $\nu\rho^h$ (see Appendix B for further details).

Algorithm 4: The *OptTraverse* function

Input : $\mathcal{T}_t$.
Initialize: $(h,i) \leftarrow (0,1)$; $P \leftarrow \{(0,1)\}$; $T_{0,1}(t) = \tau_0(t) = 1$
while (h,i) *is not a leaf and* $T_{h,i}(t) \geq \tau_h(t)$ **do**
  if $B_{h+1,2i-1}(t) \geq B_{h+1,2i}(t)$ **then**
    $(h,i) \leftarrow (h+1, 2i-1)$;
  else
    $(h,i) \leftarrow (h+1, 2i)$;
  end
  $P \leftarrow P \cup \{(h,i)\}$;
end
return (h,i) *and* P

Finally, the *UpdateB* function is used to update the B-values along the current optimistic path, and the *Expand* function is used to expand a leaf when it has been sampled enough times.

2.3.2 Analysis

Now we give the regret bound of HCT-*iid* using slightly modified near-optimality dimension and the same local smoothness assumption as Assumption 2. And we demonstrate that it can achieve almost the same guarantee as the original HCT paper.

Definition 10 (Near-optimality dimension)**.**

$$d_3(\nu, \rho) = \inf\{d' \in \mathbb{R}^+ : \exists C > 0, \forall h \geq 0, \mathcal{N}_h(3\nu\rho^h) \leq C\rho^{-d'h}\},$$

where $\mathcal{N}_h(3\nu\rho^h)$ *is the number of cells* $\mathcal{P}_{h,i}$ *s.t.,*

$$\sup_{x \in \mathcal{P}_{h,i}} f(\boldsymbol{x}) \geq f(\boldsymbol{x}^*) - 3\nu\rho^h.$$

We now give the regret bound for HCT as follows, and the proof of the following theorem is reported in Appendix B.

Theorem 11. *If at each step t, the reward y_t is independent of all prior random events, then the regret of HCT-iid in n steps is,*

$$R_n \leq O((\log(n/\delta))^{1/(d+2)} n^{(d+1)/(d+2)}),$$

with probability $1 - \delta$.

This bound matches the regret bound for HOO up to constants. It is thus a reasonable underlying algorithm to use within POO.

2.3.3 Discussion

We proved that POO assumptions can be adapted to HCT. Indeed, the main difference between HCT and HOO is the fact that HCT expands leaves only when they are pulled enough number of times such that we have small enough uncertainty. The trick in this case is that we can bound the depth of the covering tree constructed by the algorithm to $O(\log(n))$, while for HOO, the depth of the covering tree can be linear w.r.t to n.

In general, hierarchical optimization algorithms shows efficiency in regret minimization, and is also time-consumption friendly if they are implemented smartly.

2.4 Comparison

One may ask naturally which one performs better, HCT or HOO? Thus in this part of the report, we show some numerical simulations of HCT and HOO in order to compare them.

We compare HOO and HCT on some different 1D and 2D benchmark functions, we show the averaged cumulative regret R_n/n as a function of the number of evaluations n. Notice that R_n/n can be considered as an approximation of the simple regret if we use the recommendation strategy as mentioned in Remark 1. Functions used for our experiments are shown in Fig. 3.

Fig. 4 reports the comparison between HCT and HOO on these functions. For each function, we ran experiments 10 times over 5000 evaluations. We can see that HOO beats HCT in most cases, thus for the rest of this work, we use essentially HOO and POO on top of HOO for numerical simulations.

3 Bayesian Optimization

In this section, we present another dominant type of approaches for black box optimization which is Bayesian optimization. Bayesian optimization algorithms usually consist of two main components: prior over the target function and an *acquisition function* which guides the search for the optimum. We explain in details their roles in the Bayesian optimization approach and we also discuss about some encountered drawbacks.

3.1 The Bayesian Optimization Approach

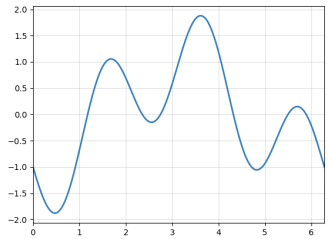
As we already mentioned in the introduction, the objective of sequential global optimization is to approximate the optimum as quickly as possible. Bayesian optimization technique is another powerful tool to achieve this goal. It is called Bayesian optimization since it is derived from the famous "Bayes' theorem":

$$P[M|E] \propto P[E|M]P[M],$$

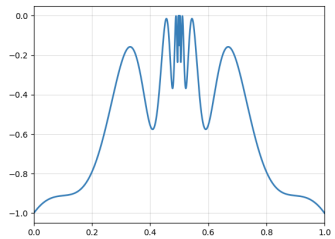
where M represents the model and E represents the given evidence. Hence, Bayesian optimization depends on some prior distribution over f given some observations $\mathcal{D}_t \coloneqq \{\mathbf{x}_{1:t}, \mathbf{r}_{1:t}\}$ where

$$\forall i \in \{1, \dots, t\}, r_i = f(\mathbf{x}_i) + \epsilon_i,$$

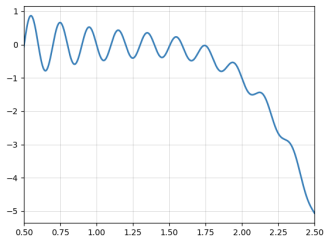
and defines a posterior distribution over the space of functions. Then, it depends on an utility function/acquisition function to decide where to sample next by optimizing the expectation of utility w.r.t the posterior distribution of the target function. Algorithm 5 shows the pseudo-code of



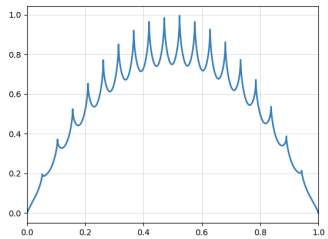
(a) $x \mapsto -\cos(x) - \sin(3x)$



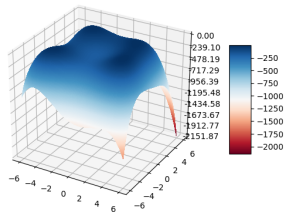
(b) 1D Double Sine Function



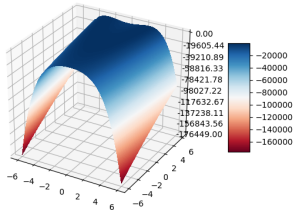
(c) 1D Grammacy Function



(d) 1D Garland Function



(e) 2D Himmelblau Function



(f) 2D Rosenbrock Function

Figure 3: Functions used for HOO vs HCT comparisons.

a Bayesian optimization-like approach and Fig. 5 shows a simple example of Bayesian optimization with 5 observations.

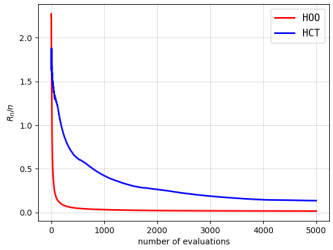
In the next, we study a special case of Bayesian optimization where the prior on f is a Gaussian process (GP).

Remark 5. GP priors are the most widely used priors, but other priors exist as well like Wiener process Močkus et al. 1978.

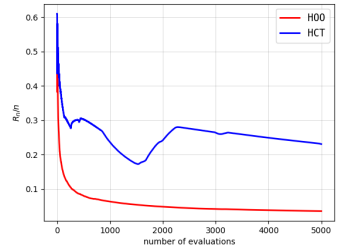
3.2 Priors over Functions

Unlike in the hierarchical setting, we can make some implicit assumptions on the smoothness without explicit parametric assumptions in Bayesian optimization. For example, in a Gaussian process setting, we assume that the target function is a sample from a Gaussian process.

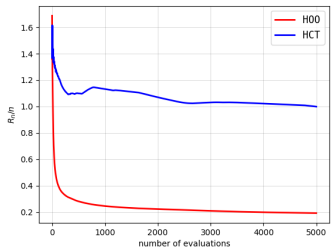
A Gaussian process $GP(\mu(\mathbf{x}), k(\mathbf{x}, \mathbf{x}'))$ is characterized by its mean function $\mu(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})]$ and its covariance function $k(\mathbf{x}, \mathbf{x}') = \mathbb{E}[(f(\mathbf{x}) - \mu(\mathbf{x}))(f(\mathbf{x}') - \mu(\mathbf{x}'))]$. In our case, we enforce $k(\mathbf{x}, \mathbf{x}') \leq 1$ and we suppose that GPs are not conditioned on data, thus we can assume that $\mu \equiv 0$.



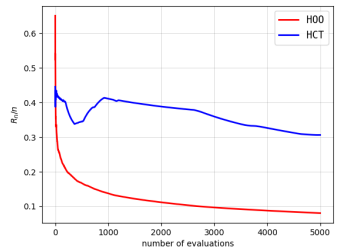
(a) $x \mapsto -\cos(x) - \sin(3x)$



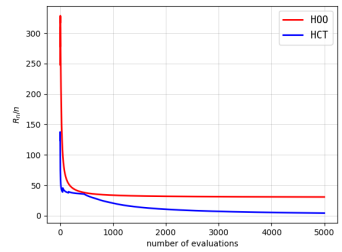
(b) 1D Double Sine Function



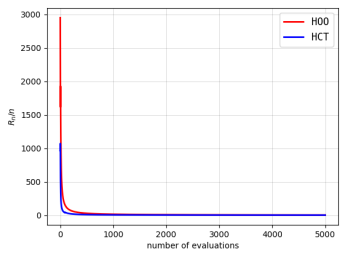
(c) 1D Grammacy Function



(d) 1D Garland Function



(e) 2D Himmelblau Function



(f) 2D Rosenbrock Function

Figure 4: HOO vs HCT.

One main advantage of Gaussian process priors is that the posterior distribution has analytic expressions for mean and variance. Assume that we use $GP(0, k(\mathbf{x}, \mathbf{x}'))$ as the prior distribution over the target function f and we have some noisy samples $\mathbf{r}_{1:t} = [r_1, r_2, \ldots, r_t]^T$ evaluated over points $[\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_t]^T$ with Gaussian noises. Now if we query a new point $\mathbf{x}_{t+1}$ with reward r_{t+1}, then the joint distribution is given by

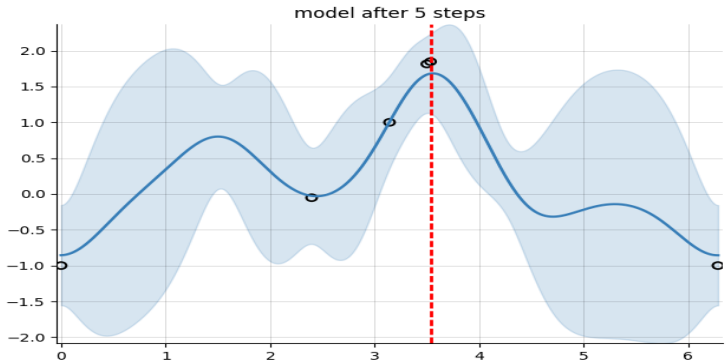
$$\begin{bmatrix} \mathbf{r}_{1:t} \\ r_{t+1} \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \mathbf{K}(\mathbf{x}_{1:t}, \mathbf{x}_{1:t}) & \mathbf{K}(\mathbf{x}_{1:t}, \mathbf{x}_{t+1}) \\ \mathbf{K}(\mathbf{x}_{1:t}, \mathbf{x}_{t+1})^T & \mathbf{K}(\mathbf{x}_{t+1}, \mathbf{x}_{t+1}) \end{bmatrix}\right).$$

Using the Sherman-Morrison-Woodbury formula Sherman and Morrison 1950, one can easily obtain

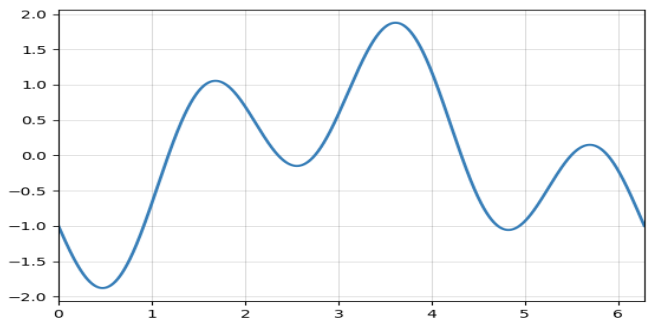
$$r_{t+1}|\mathcal{D}_t, \mathbf{x}_{t+1} \sim \mathcal{N}(\mu(\mathbf{x}_{t+1}|\mathcal{D}_t), \sigma^2(\mathbf{x}_{t+1}|\mathcal{D}_t))$$

where

$$\mu(\mathbf{x}_{t+1}|\mathcal{D}_t) = \mathbf{K}(\mathbf{x}_{1:t}, \mathbf{x}_{t+1})^T (\mathbf{K}(\mathbf{x}_{1:t}, \mathbf{x}_{1:t}))^{-1} r_{1:t},$$
$$\sigma^2(\mathbf{x}_{t+1}|\mathcal{D}_t) = \mathbf{K}(\mathbf{x}_{t+1}, \mathbf{x}_{t+1}) - \mathbf{K}(\mathbf{x}_{1:t}, \mathbf{x}_{t+1})^T (\mathbf{K}(\mathbf{x}_{1:t}, \mathbf{x}_{1:t}))^{-1} \mathbf{K}(\mathbf{x}_{1:t}, \mathbf{x}_{t+1}),$$



(a) Prediction after 5 steps



(b) Target function

Figure 5: A toy 1D example of Bayesian optimization after 5 observations on a simple target function $x \in [0, 2\pi] \mapsto -\cos(x) - \sin(3x)$. The solid blue line in (a) is the GP prediction of the target function given the data, the cyan-shaded part shows the mean plus and minus the variance, and the small rounds are the observations. (b) shows the objective function.

where $\mathbf{K}(\mathbf{x}_{1:n}, \mathbf{x}'_{1:m})$ is the covariance matrix between vectors $\mathbf{x}_{1:n}$ and $\mathbf{x}'_{1:m}$ (see Appendix C for more details).

3.3 Kernel Functions

The covariance function of a GP is very important since it defines the smoothness properties of sample functions drawn from it. Here we list some popular choices of kernel functions that are used in our experiments as well.

Linear kernel Linear kernel is probably the simplest kernel which has the form

$$k(\mathbf{x}, \mathbf{x}') = \mathbf{x}^T \mathbf{x}',$$

and its corresponding sample functions has the form

$$f(\mathbf{x}) = \mathbf{w}^T \mathbf{x},$$

where $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$.

Algorithm 5: Bayesian Optimization

Input: input space $\mathcal{X}$; prior information.

for $t \leftarrow 1$ **to** n **do**

  Pick $\mathbf{x}_t$ by maximizing the acquisition function: $\mathbf{x}_t = \arg\max u(\mathbf{x}|\mathcal{D}_{1:t-1})$;

  Sample the target and obtain a new reward: $r_t = f(\mathbf{x}_t) + \epsilon_t$;

  Perform Bayesian update and augment the data.

end

Squared Exponential kernel Rasmussen and Williams 2006 proposed a hyper-parameterized version of squared exponential kernel with a vector of *automatic relevance determination* (ARD) for anisotropic models,

$$k(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{1}{2}(\mathbf{x} - \mathbf{x}')^T \operatorname{diag}(\boldsymbol{\theta})^{-2}(\mathbf{x} - \mathbf{x}')\right).$$

Remark 6. An anisotropic model simply means its covariance matrix does not equal to identity matrix.

Matérn kernel Another popular choice of kernel is the Matérn kernel Matérn 1960 which involves a smoothness parameter ν,

$$k(\mathbf{x}, \mathbf{x}') = \frac{1}{2^{\nu-1}\Gamma(\nu)}(2\sqrt{\nu}\|\mathbf{x} - \mathbf{x}'\|)^{\nu} B_{\nu}(2\sqrt{\nu}\|\mathbf{x} - \mathbf{x}'\|).$$

Here Γ and B_ν represent the Gamma function and the Bessel function respectively. In practice, ν is usually set to 1/2, 3/2 and 5/2.

3.4 Acquisition Functions

Now we introduce some popular acquisition functions used in our experiments such as *probability of improvement* (PI), *expected improvement* (EI) and Gaussian Process-Upper Confidence Bound (GP-UCB). And we also discuss about the problem of optimizing the acquisition function.

3.4.1 Probability of Improvement

One of the first acquisition functions for GP priors is suggested by Kushner 1964. It considers the probability of improvement over the incumbent maximum of the acquisition function $f(\mathbf{x}^+)$ where $\mathbf{x}^+ := \arg\max_{\mathbf{x}\in\mathbf{x}_{1:t}} f(\mathbf{x})$. And the PI function is defined as

$$\begin{aligned}\mathrm{PI}(x) &:= \mathbb{P}(f(\mathbf{x}) \geq f(\mathbf{x}^+) + \xi) \\ &= \Phi\left(\frac{\mu(\mathbf{x}) - f(\mathbf{x}^+) - \xi}{\sigma(\mathbf{x})}\right),\end{aligned}$$

where Φ is the normal cumulative distribution function and ξ is an exploration-exploitation trade-off parameter.

3.4.2 Expected Improvement

An alternative acquisition function would take into account also the magnitude of the improvement that a point can potentially make. Močkus et al. 1978 suggested an improvement function,

$$\mathrm{I}(\mathbf{x}) := \max\{0, f_t(\mathbf{x}) - f(\mathbf{x}^+)\}.$$

The new sample is then decided by maximizing the expected improvement,

$$\mathbf{x}_t = \arg\max_{\mathbf{x}} \mathbb{E}\left[\max\{0, f_t(\mathbf{x}) - f(\mathbf{x}^+)\}|\mathcal{D}_{t-1}\right].$$

This expression can be evaluated analytically Jones et al. 1998, and Lizotte 2008 gives the exploration-exploitation trade-off version of this analytic expression:

$$\mathrm{EI}(\mathbf{x}) = \begin{cases} (\mu(\mathbf{x}) - f(\mathbf{x}^+) - \xi)\Phi(Z) + \sigma(\mathbf{x})\phi(Z) & \text{if } \sigma(\mathbf{x}) > 0 \\ 0 & \text{if } \sigma(\mathbf{x}) = 0, \end{cases}$$

where

$$Z = \begin{cases} \frac{\mu(\mathbf{x}) - f(\mathbf{x}^+) - \xi}{\sigma(\mathbf{x})} & \text{if } \sigma(\mathbf{x}) > 0 \\ 0 & \text{if } \sigma(\mathbf{x}) = 0, \end{cases}$$

and ϕ and Φ denote the normal probability distribution function and the normal cumulative distribution function respectively.

3.4.3 GP-UCB

The GP-UCB algorithm Srinivas et al. 2009 is a somewhat different kind of acquisition function. One may think of querying the next point by maximizing the variance. However in practice, it is wasteful to maximize merely the variance as it only concentrates on reducing the global uncertainty, but not exploiting knowledge around the maxima. Another idea is to choose points by maximizing the expected rewards, i.e. $\mathbf{x}_t = \arg\max_{\mathbf{x}\in\mathcal{X}} \mu_{t-1}(\mathbf{x})$. However, this may soon lead the algorithm to a local optimum.

A combination of these two strategies is proposed to overcome this dilemma:

$$\mathbf{x}_t = \arg\max_{\mathbf{x}\in\mathcal{X}} \mu_{t-1}(\mathbf{x}) + \sqrt{\beta_t}\sigma_{t-1}(\mathbf{x}),$$

where β_t are some well chosen constant.

GP-UCB implicitly trades off between exploration and exploitation. It explores by sampling $\mathbf{x}$ with large $\sigma_n^2(\mathbf{x})$ and it exploits by sampling $\mathbf{x}$ with large $\mu_n(\mathbf{x})$. Indeed, this GP-UCB selection is mainly motivated by the UCB algorithm, as $\mu_{t-1}(\mathbf{x}) + \sqrt{\beta_t}\sigma_{t-1}(\mathbf{x})$ is an upper confidence bound on $f(\mathbf{x})$.

Some cumulative regret bounds have been given for GP-UCB, more precisely in the case when $\mathcal{X}$ is finite or compact and convex. Here, we show the result for finite $\mathcal{X}$.

For that purpose, we need a notion of information gain. In order to estimate globally the target function f as quickly as possible, we need to choose carefully a set of samples $A \subset \mathcal{X}$. The way we measure the informativeness of a set of samples is the information gain Cover and Thomas 2012 which is the mutual information between f and observations $\mathbf{y}_A$,

$$I(\mathbf{y}_A; f) = H(\mathbf{y}_A) - H(\mathbf{y}_A|f).$$

Finding a subset $A \subset \mathcal{X}$ that maximizes the information gain is an NP-hard problem, however, it can be approximated by a greedy algorithm. Indeed, if we denote $F(A) = I(\mathbf{y}_A; f)$, then at step t of the algorithm, we can choose

$$\mathbf{x}_t = \arg\max_{\mathbf{x} \in \mathcal{X}} F(A_{t-1} \cup \{\mathbf{x}\}),$$

which can be shown to be equivalent to pick

$$\mathbf{x}_t = \arg\max_{\mathbf{x} \in \mathcal{X}} \sigma_{t-1}(\mathbf{x})).$$

And we can show that this greedy heuristic is guaranteed to find a near-optimal solution after n rounds,

$$F(A_n) \geq (1 - 1/e) \max_{|A| \leq n} F(A).$$

This is true mainly due to the fact that F satisfies a good property called *submodularity* Krause and Guestrin 2012. Indeed, this greedy approximation guarantee holds for any submodular function Nemhauser et al. 1978.

Now We define the *maximum information gain* after n rounds as:

$$\gamma_n = \max_{A \subset \mathcal{X}, |A| = n} I(\mathbf{y}_A; f),$$

and we can obtain the following bound for finite $\mathcal{X}$.

Theorem 12. *Let $\delta \in (0, 1)$ and $\beta_t = 2\log(|D|t^2\pi^2/\delta)$. Running GP-UCB of a sample f of $GP(0, k(\boldsymbol{x}, \boldsymbol{x}'))$, we obtain a cumulative bound:*

$$R_n = \mathcal{O}^*(n\gamma_n \log |\mathcal{X}|)$$

with a high probability, i.e.,

$$P\{R_n \leq \sqrt{nC_1\beta_n\gamma_n}, \forall n \geq 1\} \geq 1 - \delta,$$

where $C_1 = 8/\log(1 + \sigma^{-2})$.

Remark 7. Here bounds on cumulative regret are given. This may seem to be somewhat weird since our objective is sequential global optimization. It may be more appropriate to focus on the simple regret.

3.4.4 Optimization of the Acquisition Function

One problem we encounter in the implementation of GP methods is how to optimize the acquisition function. It may seem a bit strange since we need to tackle with a secondary optimization problem in order to solve the original one. But this secondary problem is usually easier because the acquisition function is easy to evaluate.

A naive but efficient idea could be just optimizing the acquisition function on a fixed discrete grid. Other approaches can be considered as well like DIRECT Jones et al. 1993 and LBFGS Broyden 1970 (the last one requires the knowledge of gradients which may not be the case for all acquisition functions).

3.5 Discussion

Bayesian optimization is studied by a huge range of research work, and is more widely used in real applications than hierarchical optimization algorithms. This does not mean that Bayesian optimization has better performance than hierarchical optimization. Indeed, hierarchical-inspired methods are developed more recently and comparison between these two family of methods are rarely done before. In the next section, we show several numerical results on this task and we will actually see that it is hard to compare them directly.

4 Implementations and Experiments

4.1 Implementations

All codes for this work have been carried out in `Python`. One unique framework for both Bayesian optimization and hierarchical optimization is implemented so that we can compare them. For applications to hyper-parameter optimization, all machine learning models we used are directly imported from `scikit-learn`.

4.2 Synthetic Results

As one of the objectives of this internship, we want to compare the performance between hierarchical optimization methods and Bayesian optimization methods. Recall that we focus mostly on HOO for numerical simulations rather than HCT (see 2.4), thus in this part, we compare HOO, POO and BO-based methods on some benchmark functions. We run 10 experiments over 100 evaluations for each algorithm.

We first run experiments on a "difficult function" displayed in Fig. 6 Grill et al. 2015. It is a difficult function since it possesses upper and lower envelope around its global maximum that are equivalent to x^2 and $\sqrt{x}$ respectively, and therefore its near-optimality dimension is not equal to 0. It is thus a good benchmark function to compare POO and HOO. Fig. 7 shows that POO works almost as well as HOO on this function.

On the other hand, we can see that POO or HOO outperforms BO methods on this function. This is quite understandable since this function can clearly not be considered as a sample from some Gaussian process.

However, if we run experiments on a function like Fig. 5, we can see in Fig. 8 that this time BO methods outperforms HOO and POO, since we are trying to optimize for a sinusoidal function which can perfectly be considered as a sample from some Gaussian process.

Discussion It is hard to tell whether Bayesian optimization works better or hierarchical optimization does since they do not cover the same class of functions. More precisely, we do not know if we can find any relation between these two classes of functions. However, one major advantage of hierarchical optimization algorithms is that they are much faster in term of time complexity. Time complexity for hierarchical optimization is around $O(n\log(n)$, whereas Bayesian methods have a time complexity of $O(n^3)$ since we need to update the covariance matrix at every time step.

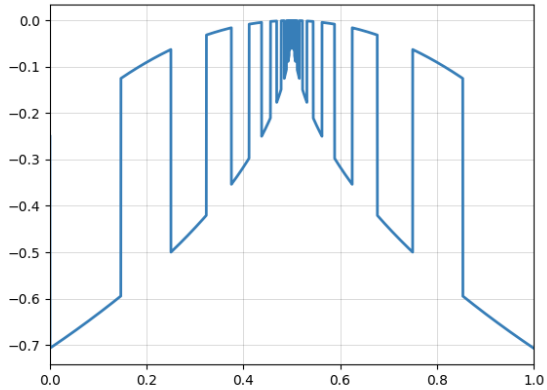


Figure 6: Difficult function $f : x \mapsto s(\log_2(|x-0.5|)(\sqrt{|x-0.5|} - (|x-0.5|)^2)$ where $s(x) = 1$ if the fractional part of x is less than 0.5, and $s(x) = 0$ otherwise.

4.3 Applications to Hyper-parameter Optimization for Machine Learning

One important application of black box optimization or multi-armed bandit model is hyper-parameter optimization in the context of machine learning. Hyper-parameter optimization is usually the most tedious part in fitting a machine learning algorithm. In practice, we are interested in models whose loss, evaluated on an independent part of data is as low as possible. Different hyper-parameter choices lead to different losses, therefore finding the optimal set of hyper-parameters is of importance. In reality, we never know that if the point found is the global optimum, but from a practical point of view, we are only interested in finding the model that works best in a production setting.

In this part, we first formalize our experimental setup by introducing benchmark strategies, rules used for benchmarking, as well as some explanation on the machine learning models used here.

4.3.1 General Setting

A hyper-parameter optimization problem can be considered as a fixed-budget pure-exploration problem. We recall that in a pure- exploration problem, the decision maker has to explore resources within a limited budget, e.g. within n rounds of exploration, then give a recommendation at the end. The quality of the decision is evaluated only on this final recommendation. This corresponds exactly to the hyper-parameter optimization situation where one hyper-parameter configuration can be considered as one arm. Indeed, if we consider a function $f : \mathcal{X} \to \mathbb{R}$ and take $\mathcal{X}$ as a set of parameter configurations, we can then map one parameter configuration to the performance of the machine learning algorithm using the corresponding parameters. If $|\mathcal{X}| < \infty$, we can refer to a bandit problem; otherwise, we can refer to a general sequential optimization problem.

We can cite numerous approaches that can be applied to solve this problem, such as grid search, Bayesian optimization, random search, gradient-based optimization, etc. Particularly, there exist several multi-armed bandit-inspired algorithms such as UCBE Audibert et al. 2010, UGapE Gabillon et al. 2012, BayesUCB Kaufmann et al. 2012a, GP-UCB, Thompson sampling Kaufmann et al. 2012b, BayesGap Hoffman et al. 2014, etc.

In our experiments, the procedure can be described briefly as follow, at each round t:

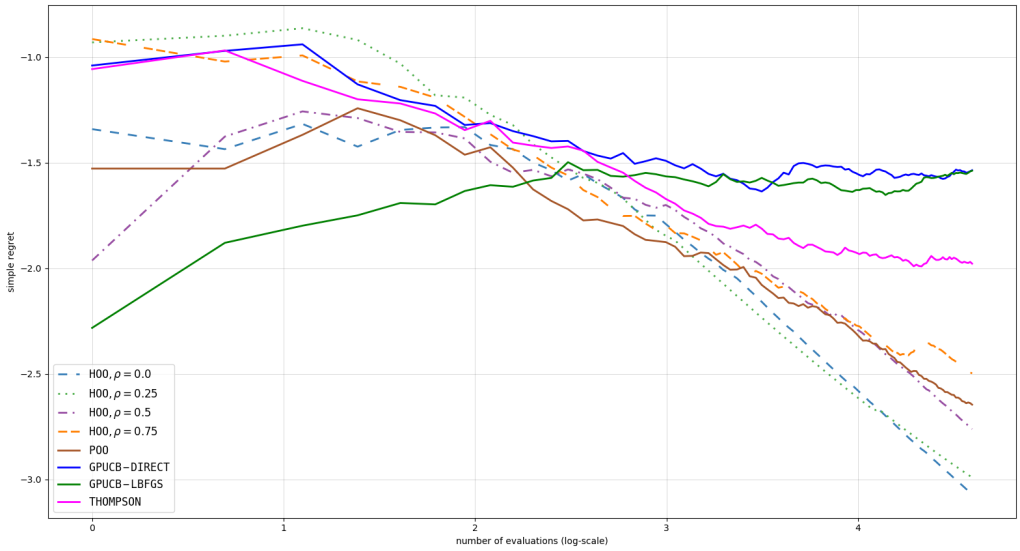


Figure 7: Log-scale results for HOO, POO and BO methods on a difficult function.

- the action of pulling an arm consists in choosing a vector of hyper-parameters $\mathbf{x}_t$ using some adaptive algorithm;
- $\mathbf{x}_t$ is then used in some machine learning to obtain a "reward" r_t (more details are given in 4.3.3);
- is recommended an arm $\mathbf{x}(t)$ (the same notation as Section 2).

Remark 8. Compared to experiments carried out in Hoffman et al. 2014, where only a finite grid of hyper-parameters are taken into account, in our setting, we consider a continuous arm space.

4.3.2 Benchmark Strategies

In the machine learning community, hyper-parameter optimization is often overlooked, and in fact, some of the most famous models (e.g. Random Forests) have been shown to be somewhat insensitive to hyper-parameter optimization Verikas et al. 2011. Two common strategies for finding *better* hyper-parameters are presented here: grid search and random search. These two methods are used in different occasions: in particular grid search is typically used when the search space is not too large and the model to fit is not expensive to evaluate. Random search is more popular when it becomes impractical to explore the search. More details of these two common strategies are presented below.

Grid Search Also known as parameter sweep. This is simply an exhaustive search in a user specified subset of parameter space. Search bounds have to be set manually. When multiple hyper-parameters have to be optimized over their sets, grid search considers the Cartesian product of

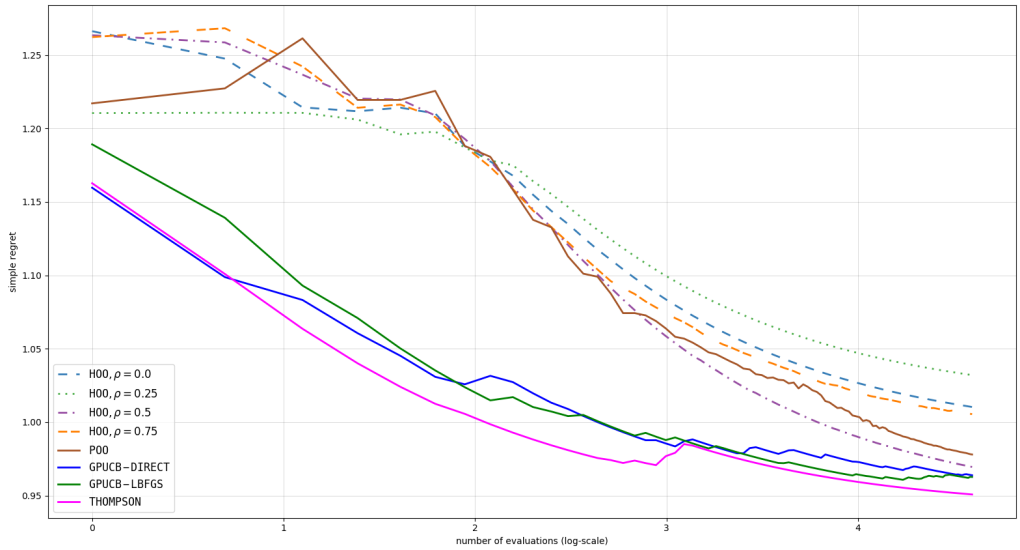


Figure 8: Log-scale results for HOO, POO and BO methods on $x \in [0, 2\pi] \mapsto -\cos(x) - \sin(3x)$.

all of them. This poses a problem when the number of hyper-parameters to optimize grows, also known as the curse of dimensionality. While suffering from this problem, grid search is still widely used for small to mid-sized problems, where function evaluations are not very expensive. Also, it is embarrassingly parallel: function evaluations can be distributed over in a simple way

Random Search Grid search is exhaustive since it considers all possible evaluations over a Cartesian product over parameter sets. Randomized search in hyper-parameter space is also a very popular method for the task at hand. In particular, this method has been shown to work considerably better in high-dimensional spaces than grid search Bergstra and Yoshua 2012. There is also evidence that often sometimes some hyper-parameters do not affect the loss significantly.

4.3.3 Evaluation Metrics

Throughout all the experiments, we will be minimizing loss functions, in some form or another. Every problem presented here is formulated either as a regression or a classification task, therefore it is of importance to define early on what type of loss to use in each problem. For binary classification problems, we use the logarithmic loss, also known as binary cross-entropy, defined by:

$$\mathcal{L}(\boldsymbol{y}, \hat{\boldsymbol{y}}) = -\frac{1}{n} \sum_i \left(y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i) \right)$$

The log-loss is very popular in the machine learning community Bishop 2006. This particular metric does not only take into account whether a classifier makes the right decision given a threshold c, (like the 0/1 loss would), but also the confidence in predictions $\hat{\boldsymbol{y}}$. It is also very natural since

it is just the negative log-likelihood of a Bernoulli random variable. The use of the logarithm both punishes erroneous extremely confident positive or negative predictions.

We generalize the previous metric for multi-class classification problems. The following expression is typically called categorical cross-entropy:

$$\mathcal{L}(\boldsymbol{y}, \hat{\boldsymbol{p}}) = -\frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{m} y_{ij} \log(\hat{p}_{ij})$$

where $\hat{p}_{ij}$ is the predicted probability of a sample i belonging to class j, and m is the number of classes considered. For continuous regression problems, the loss that we use is the typical mean squared error, defined by:

$$\mathcal{L}(\boldsymbol{y}, \hat{\boldsymbol{y}}) = \frac{1}{n} \sum_{i} (y_i - \hat{y_i})^2$$

notice that in our situation, it would be difficult to estimate the "probability of error", thus we report the mean squared root error instead, which is a typical approximation for the error rate to our interest

While we have discussed the metrics to evaluate in each predictive problem, we still need to define how these losses is evaluated in each step. To evaluate the performance of a given hyper-parameter optimization procedure, we have to balance both the performance in terms of loss and the number of evaluations necessary in order to get to a satisfactory solution. In practice, this is exactly how we benchmark the different strategies, through a plot where the x-axis represent the number of function evaluations, and the y-axis the best log-loss found.

Evaluating the loss function itself can lead to multiple different values depending on the test values. One could choose an approach where this loss is evaluated on a single holdout test. This would lead to noisy estimates, however. We choose the more stable approach of performing a shuffled $k = 5$ cross-validation scheme to obtain a more reliable loss estimate. In practice, this means that we fit 5 models with the same architecture to different train/test splits and average the loss results in each. More precisely, for every cross-validation split $\texttt{cv}_j, j = 1 \dots 5$, we get a loss $\mathcal{L}_j(\boldsymbol{y}_j, \hat{\boldsymbol{y}}_j) = \frac{1}{n} \sum_i (y_{ji} - \hat{y_{ji}})^2$, thus the loss at time t is

$$\frac{1}{5} \sum_{j=1}^{5} \mathcal{L}_j(\boldsymbol{y}_j, \hat{\boldsymbol{y}}_j) = \frac{1}{5n} \sum_{j=1}^{5} \sum_{i} (y_{ji} - \hat{y_{ji}})^2$$

where $\boldsymbol{y}_j$ stands for test values for cross-validation split $\texttt{cv}_j$, and $\hat{\boldsymbol{y}}_j$ stands for the corresponding predicted values.

Now that we have defined the loss function for both classification and regression problem, we can thus specify the meaning of function evaluations in our task. Indeed, the underlying task is to find some classifier (with a certain vector of paramters) $\texttt{clf}_{\mathbf{x}_t}$ which minimizes the expected loss $f(\texttt{clf}_{\mathbf{x}_t}) = \mathbb{E}\left[\mathcal{L}(\boldsymbol{y}, \texttt{clf}_{\mathbf{x}_t}(\boldsymbol{X})\right]$ where f corresponds to our underlying black box function, $\boldsymbol{X}$ corresponds to test points and $\boldsymbol{y}$ corresponds to test values. Notice that we only have access to estimate of this underlying function, it is difficult to report directly simple or cumulative regret in this situation, we thus report logistic loss or mean square error instead.

Remark 9. Since we do cross-validation at each step, and average the loss results, we can consider that the estimates of the mean reward are not very noisy. Thus at each round t, we choose to recommend the arm $\mathbf{x}(t)$ that gives us the best reward so far.

4.3.4 Machine Learning Models

Since the shape of our objective function depends on both the dataset and the predictor, we try to span as many different types of machine learning models as possible to provide the most extensive evaluation as possible. Except for rare occasions where we could not fit a model to a particular dataset for numerical conditions, all models are evaluated in all datasets with the same number of parameters and bounds to optimize over. We consider the following models: Support Vector Machines (SVM) with radial basis function kernel, K-nearest neighbors (KNN), neural networks with a single hidden layer (MLP) and Gradient Boosting Machines (GBM). We briefly detail how these work now.

Support Vector Machines A SVM model Cortes and Vapnik 1995 uses the concept of hyperplanes in high or infinite dimensional space in order for classification or regression purposes. In particular, a good classification model is the one that places an hyperplane that achieves maximum distance to training points of any class. Intuitively, the larger this margin, the lower the generalization error of the model. For classification, the model can be defined via optimization. Assume $\boldsymbol{x}_i \in \mathbb{R}^p$, and $y_i \in \{0, 1\}$, then the problem is to minimize:

$$\begin{aligned} \min_{\boldsymbol{w},b,\boldsymbol{\epsilon}} \quad & \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\sum_{i=1}^{n}\epsilon_i \\ s.t. \quad & y_i(\boldsymbol{w}^T\phi(\boldsymbol{x}_i) + b) \geq 1 - \epsilon_i \\ & \epsilon_i \geq 0, \quad i = 1, \ldots, n \end{aligned}$$

In practice it makes more sense to minimize its dual:

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & \frac{1}{2}\boldsymbol{\alpha}^T Q\boldsymbol{\alpha} - \boldsymbol{e}^T\boldsymbol{\alpha} \\ s.t. \quad & \boldsymbol{y}^T\boldsymbol{\alpha} = 0 \\ & 0 \leq \alpha_i \leq C, \quad , i = 1, \ldots, n \end{aligned} \tag{1}$$

where $\boldsymbol{e}$ is the unit vector, C is an hyper-parameter controlling an upper bound, Q is a $n \times n$ semidefinite positive matrix defined by $Q_{ij} = y_i y_j K(\boldsymbol{x}_i, \boldsymbol{x}_j)$ and K is our defined kernel function. Finally, the decision function is defined as:

$$d(\boldsymbol{x}) = \operatorname{sgn}\left(\sum_{i=1}^{n} y_i\alpha_i K(\boldsymbol{x_i}, \boldsymbol{x}) + \rho\right)$$

For regression problems we now consider $y_i \in \mathbb{R}$ and we try to minimize:

$$\begin{aligned} \min_{\boldsymbol{w},b,\gamma,\gamma^*} \quad & \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\sum_{i=1}^{n}(\gamma_i + \gamma_i^*) \\ s.t. \quad & y_i - \boldsymbol{w}^T\phi(\boldsymbol{x}_i) - b \leq \epsilon + \gamma_i \\ & \boldsymbol{w}^T\phi(\boldsymbol{x}_i) + b - y_i \leq \epsilon + \gamma_i^* \\ & \gamma_i, \gamma_i^* \geq 0 \quad i = 1, \ldots, n \end{aligned}$$

Parameter	Type	Bounds
C	$\mathbb{R}^+$	$\left[10^{-5}, 10^5\right]$ (log-scaled)
γ	$\mathbb{R}^+$	$\left[10^{-5}, 10^5\right]$ (log-scaled)

Table 1: Parameters to be optimized for SVM models.

Parameter	Type	Bounds
k	Integer	$\{10, \ldots, 50\}$

Table 2: Parameters to be optimized for KNN models.

Likewise, we normally minimize its dual:

$$\begin{aligned} \min_{\boldsymbol{\alpha},\boldsymbol{\alpha}^*} \quad & \frac{1}{2}(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*)Q(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*) + \epsilon \boldsymbol{e}^T(\boldsymbol{\alpha}+\boldsymbol{\alpha}^*) - \boldsymbol{y}^T(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*) \\ s.t. \quad & \boldsymbol{e}^T(\boldsymbol{\alpha}-\boldsymbol{\alpha}^*) = 0 \\ & 0 \leq \alpha_i, \alpha_i^* \leq C, \quad i = 1, \ldots, n \end{aligned} \tag{2}$$

Our decision function now becomes:

$$g(\boldsymbol{x}) = \sum_{i=1}^{n} (\alpha_i - \alpha_i^*) K(\boldsymbol{x_i}, \boldsymbol{x}) + \rho$$

For all the experiments involved in the following sections, we use `scikit-learn` implementation of Support Vector Machines, which is in turn based on `LibSVM` Chang and Lin 2011. We optimize over two hyper-parameters, C, the penalty parameter in the error term of Equations 1 and 2, and γ, an radial basis function hyper-parameter controlling the smoothness of the decision function. They are optimized on the range defined by Table 1.

K-Nearest Neighbors In contrast to other strategies presented here, K-nearest neighbors does not approach learning by constructing a generalizable internal model, but simply stores training instances. Classification for an example is then performed using a majority vote of its closest points in distance. For the case of regression, we take the average of mentioned points target instead. Since computing a whole distance matrix for all examples is computationally expensive ($\mathcal{O}(dn^2)$ for n samples and d dimensions), several alternatives have been proposed. KDTree Kennel 2004 is arguably one of the most popular ones. Intuitively, it works the following way: if we know points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are far in space, and we know point $\boldsymbol{x}_k$ is close to $\boldsymbol{x}_j$, then we know $\boldsymbol{x}_i$ and $\boldsymbol{x}_k$ must be far in space without *explicitly* having to compute their distance. It can be proven that this can reduce the computational complexity to $\mathcal{O}(dn \log n)$. For all benchmarking run in this work, we optimize only parameter k, the number of neighbors to consider, over a range specified in Table 2.

Gradient Boosting Machines Boosting Schapire 1999 is a machine learning technique for simultaneously reducing the bias and variance of a classifier or regressor. It is based on the concept of ensembles, that is, a set of weak models, such as trees that are combined in a smart way to produce

a strong model. In particular, Gradient Boosting Machines Friedman 2001 are a particular instance of models using this boosting principle. It builds its internal model considering tree models in a stage-wise fashion and generalizes them by optimizing a given differentiable loss function. Assuming training data $\{\boldsymbol{x}_i, y_i\} \quad i = 1, \ldots, n$, the algorithm works by approximating a function $\hat{F}(\boldsymbol{x})$ to an original $F(\boldsymbol{x})$, which minimizes the expected value of some loss function $\mathcal{L}(\boldsymbol{y}, F(\boldsymbol{x}))$, that is:

$$\hat{F}(\boldsymbol{x}) = \arg\min_{F} \mathbb{E}_{\boldsymbol{x},y} \left[\mathcal{L}(\boldsymbol{y}, F(\boldsymbol{x})\right]$$

Gradient boosting machine defines F to be a weighted sum of weak learners h_i from some class $\mathcal{H}$:

$$F(\boldsymbol{x}) = \sum_{i=1}^{n} \gamma_i h_i(\boldsymbol{x}) + c$$

We start by some constant approximation F_0 and expand it iteratively:

$$\begin{aligned} F_0(\boldsymbol{x}) &= \arg\min_{\gamma} \sum_{i=1}^{n} \mathcal{L}(y_i, \gamma) \\ F_m(\boldsymbol{x}) &= F_{m-1}(\boldsymbol{x}) + \arg\min_{f \in \mathcal{H}} \sum_{i=1}^{n} \mathcal{L}(y_i, F_{m-1} + f(\boldsymbol{x}_i)) \end{aligned}$$

The problem comes when trying to optimize and arbitrary f for any loss $\mathcal{L}$, so instead, we use gradient descent to minimize:

$$\begin{aligned} F_m(\boldsymbol{x}) &= F_{m-1}(x) - \gamma_m \sum_{i=1}^{n} \nabla_{F_{m-1}} \mathcal{L}(y_i, F_{m-1}(x_i)) \\ \gamma_m &= \arg\min_{\gamma} \sum_{i=1}^{n} \mathcal{L}\left(y_i, F_{m-1}(\boldsymbol{x}_i) - \gamma \frac{\partial \mathcal{L}\left(y_i, F_{m-1}(\boldsymbol{x}_i)\right)}{\partial F_{m-1}(\boldsymbol{x}_i)}\right) \end{aligned}$$

γ is then chosen by some univariate optimization algorithm, such as line search. For the benchmarks considered in this work, we use `scikit-learn`'s GBM implementation, and try to optimize the hyper-parameters defined by Table 3. The `learning_rate` parameter shrinks the contribution of each tree, `n_estimators` is the number of weak trees to fit, `max_depth` is the maximum depth of each weak tree, and `min_samples_split` is the minimum required number of samples to split a node in each tree.

Multilayer Perceptron Neural networks are a popular model now, specially with the rise of Deep Learning LeCun et al. 2015 over the last years. In this work, we only consider neural network models with a single hidden layer, or Multilayer Perceptron (MLP) models. The output of one layer, given some input $\boldsymbol{x} \in \mathbb{R}^p$ is a function $f : \mathbb{R}^p \to \mathbb{R}^q$. Since a MLP contains only one hidden layer, the output of the whole model for either regression or binary classification can be written as:

$$f(\boldsymbol{x}) = G\left(b^{(2)} + W^{(2)}\left(s\left(b^{(1)} + W^{(1)}\boldsymbol{x}\right)\right)\right),$$

Parameter	Type	Bounds
`learning_rate`	$\mathbb{R}^+$	$[10^{-5}, 10^{-2}]$
`n_estimators`	Integer	$\{10, \ldots, 100\}$
`max_depth`	Integer	$\{2, \ldots, 100\}$
`min_samples_split`	Integer	$\{2, \ldots, 100\}$

Table 3: Parameters to be optimized for GBM models.

Parameter	Type	Bounds
`hidden_layer_size`	Integer	$[5, 50]$
`alpha`	$\mathbb{R}^+$	$[0, 0.9]$

Table 4: Parameters to be optimized for MLP models.

where $W^{(1)}$ and $W^{(2)}$ is a matrix of learned weights of size $p \times q$ and $q \times 1$ respectively, $b^{(1)}$ and $b^{(2)}$ are bias vectors and both G and s are non-linear differentiable functions. In practice, s is a rectified linear unit function, that is:

$$s(\boldsymbol{x}) = \max(0, \boldsymbol{x}),$$

and $G(\boldsymbol{x})$ is the logistic function for binary classification:

$$G(\boldsymbol{x}) = \frac{1}{1 + \exp(-\boldsymbol{x})}$$

For regression, no non-linearity is applied. For c multiple classes, $W^{(2)}$ changes size to $q \times c$, and typically chooses G to be the softmax function:

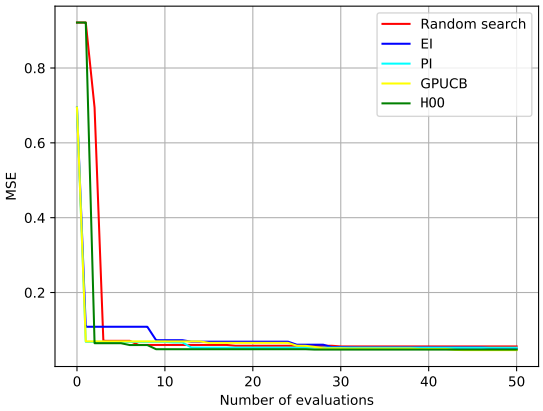
$$G(\boldsymbol{x})_j = \frac{\exp(\boldsymbol{x}_j)}{\sum_{i=1}^{c} \exp(\boldsymbol{x}_i)}, \quad j = 1, \ldots, c$$

Matrices $W^{(1)}$, $W^{(2)}$ and biases $b^{(1)}$, $b^{(2)}$ are trained using backpropagation Rumelhart et al. 1986 through the use of stochastic gradient descent (SGD) Bottou 2004. For this particular piece of work, we use the Multilayer Perceptron implementation of `scikit-learn`, and optimize over hyper-parameters detailed in Table 4. In particular, we consider the number of hidden units in the hidden layer and α, a parameter controlling L_2 regularization on the learned weights.

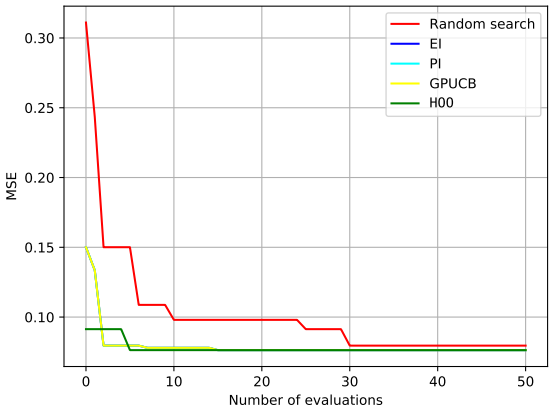
4.3.5 Results

In this part, we report some numerical results on different datasets and we run experiments for both classification and regression tasks. The budget for each task is fixed at $n = 50$ steps.

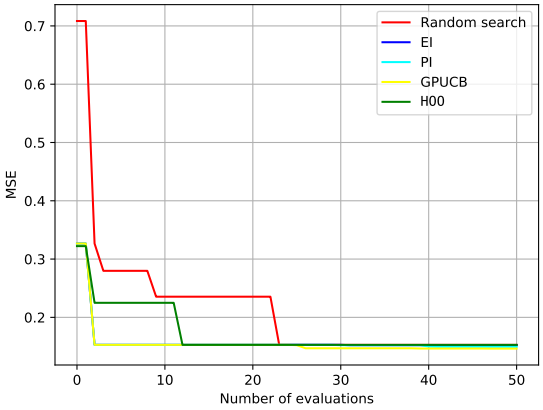
UCI Wine Dataset The UCI Wine Dataset consists of results of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivators. It includes 13 continuous attributes and 178 instances. It is a multi-classification problem for which the task is to determine the type of wines. Fig. 9 shows the results for this dataset.



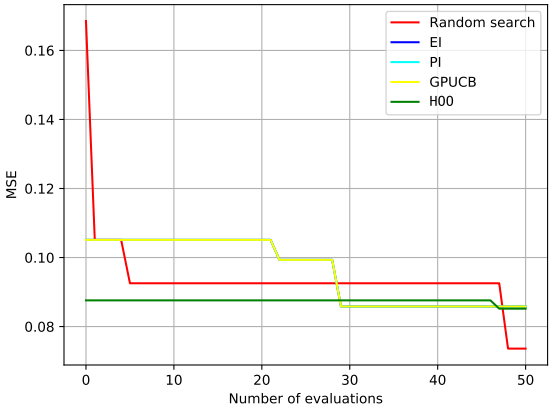
(a) SVM



(b) KNN



(c) MLP



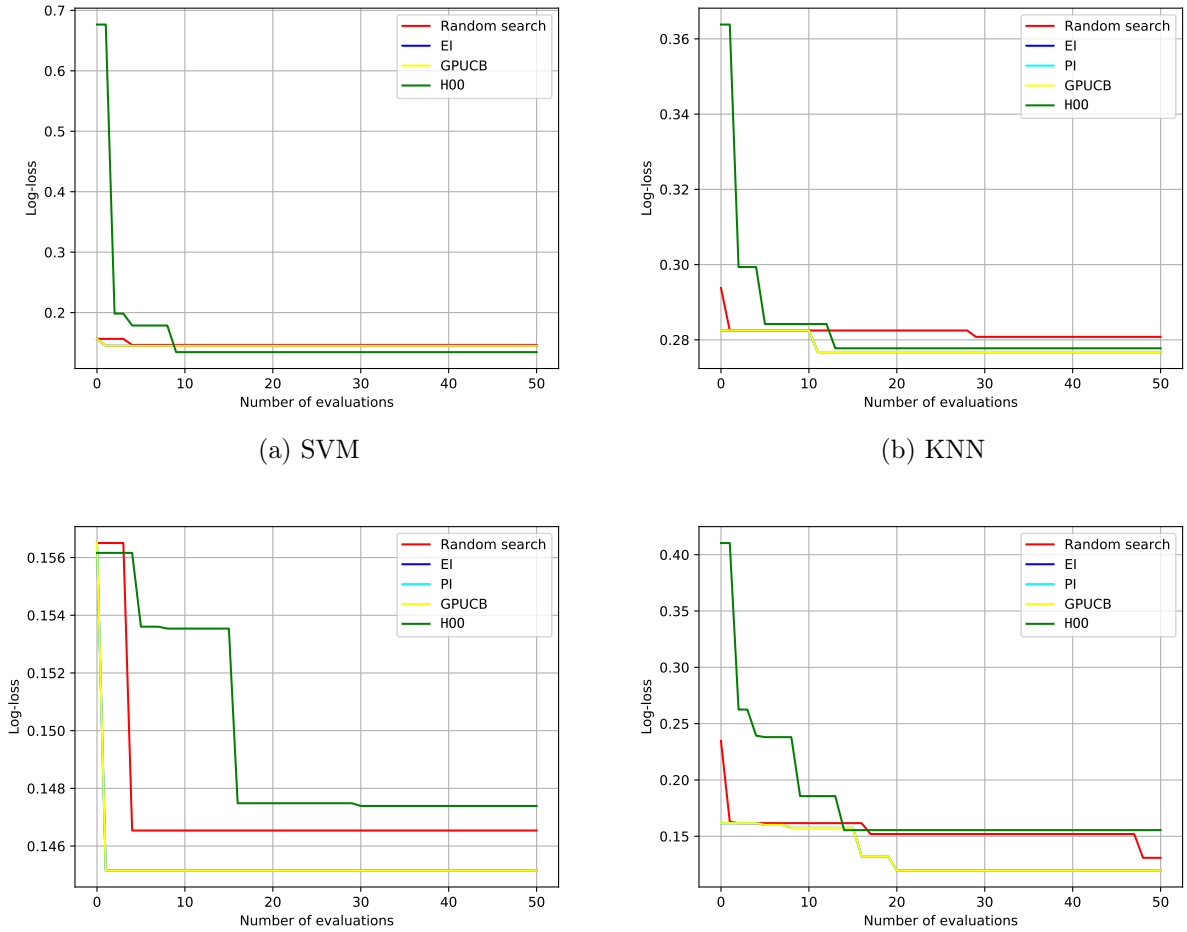
(d) GBM

Figure 9: Results for the Wine Dataset.

UCI Breast Cancer (Diagnostic) Dataset The UCI Breast Cancer (Diagnostic) Dataset is a binary classification task which contains 32 continuous attributes and 569 instances. Features are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. Fig. 10 shows the results for this dataset.

UCI Yacht Hydrodynamics Dataset The UCI Yacht Hydrodynamics Dataset is a regression problem, whose objective is to predict residuary resistance of sailing yachts which is crucial for evaluating the ship's performance. It contains 308 instances and 7 real attributes. Fig. 11 reports the results for this dataset.

Discussion From these plots we can see that our black box optimization algorithms achieve better results with less evaluations than random search which shows that they can be a good default strategy for optimizing machine learning hyper-parameters. However, random search can



(a) SVM

(b) KNN

(c) MLP

(d) GBM

Figure 10: Results for the Breast Cancer (Diagnostic) Dataset.

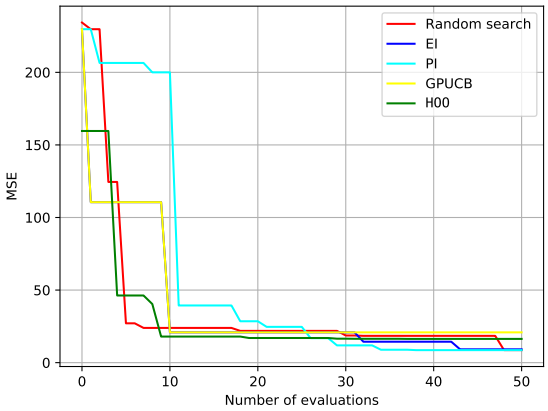
be parallelized over threads, and therefore can potentially outperforms techniques presented here in case we have powerful computing ability.

Remark 10. On some figures, we can only see the yellow curve for GP-UCB, indeed, curves for PI and EI just overlay with the yellow curve, which means the choice of acquisition function in the our context does not have a great influence on the final performance.

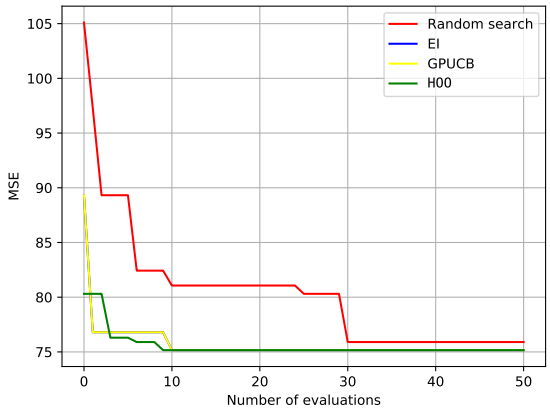
Conclusion

In this piece of work, we studied the global optimization problem, or sometimes called black box optimization, and its link to multi-armed bandit models. We concentrated especially on two dominant approaches for this problem which are hierarchical optimization and Bayesian optimization and we compared their performance.

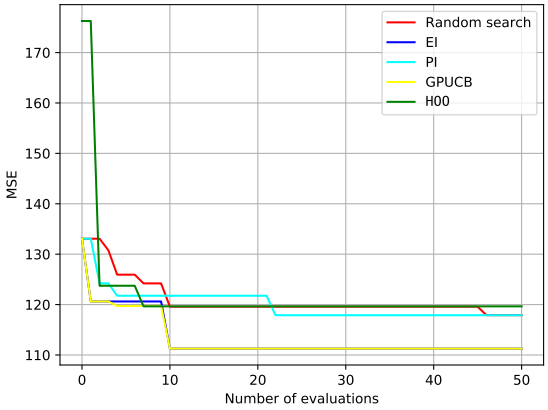
We proved that milder assumptions of POO can be adapted to HCT, which guarantees the



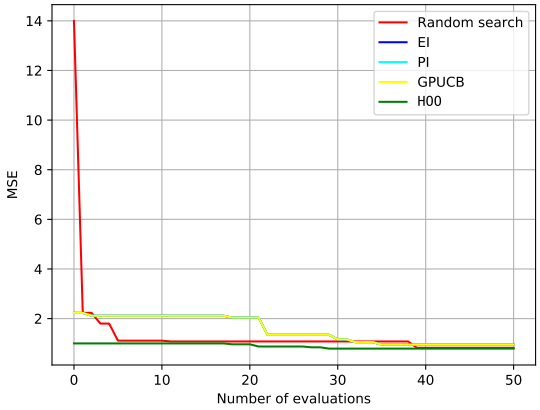
(a) SVM



(b) KNN



(c) MLP



(d) GBM

Figure 11: Results for the Yacht Hydrodynamics Dataset.

validity of POO analysis.

We also applied black box optimization algorithms that we have studied to the task of hyper-parameter optimization in the context of machine learning. It suggested that they can be seriously taken into account when we do not have enough budget to explore large number of different hyper-parameters.

Future Work For the future work, it would be better if we can adapt POO assumptions to HOO, since in our experience, HOO outperforms HCT in most situations as shown in Section 2.4.

As we already mentioned that using the optimism in the face of uncertainty principle for the exploration phase is not optimal for the optimization purpose, and that methods based on best arm identification are preferred. We thus intend to propose new algorithms where the exploration phase will be using best arm identification, and we are particularly interested in hybrid approaches of hierarchical optimization and Bayesian optimization, see for example Contal et al. 2015.

Acknowledgement

The internship opportunity I had with team SequeL at Inria Lille was a great chance for learning and career development. I feel very lucky to be part of this team for 5 months, and I am also grateful for having a chance to meet so many wonderful people who led me through this internship period.

Bearing in mind, I would like to express my special thanks to Prof. Philippe, without whom I would not have the chance to have contact with the team.

I express my deepest gratitude to my supervisors Emilie and Michal, for taking many time in useful advice and discussion for my internship. I choose this moment to acknowledge their guidance gratefully.

It is my radiant sentiment to place on record my best regards to Alessandro, Jérémie, Odalric, Daniil, Romaric, Olivier, James, Ralph, Matteo, Merwan, Lilian, Daniele, Nicolas, Ronan, Jean-Bastien, Julien, Florian, Juliia and Georgios for their welcome and precious guidance which were extremely valuable for my study.

References

J.-Y. Audibert, S. Bubeck, and R. Munos. Best arm identification in multi-armed bandits. *In Proceedings of the 23th Annual Conference on Computational Learning Theory (COLT)*, 2010.

P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning Research*, 47(2):235–256, 2002.

P. Auer, R. Ortner, and C. SzepesvÃąri. Improved rates for the stochastic continuum-armed bandit problem. *In Proceedings of the 20th Annual Conference on Computational Learning Theory (COLT)*, pages 454–468, 2007.

M. G. Azar, A. Lazaric, and E. Brunskill. Online stochastic optimization under correlated bandit feedback. *In International Conference on Machine Learning (ICML)*, pages 1557–1565, 2014.

J. Bergstra and B Yoshua. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13:281–305, 2012.

C. M. Bishop. *Pattern Recognition and Machine Learning*, volume 4. Springer Science+ Business Media, LLC, 2006.

L Bottou. Stochastic learning. *Advanced Lectures on Machine Learning*, pages 146–168, 2004.

E. Brochu, V. M. Cora, and N. de Freitas. A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *Technical report, University of Bristish Columbia*, 2010.

C. G. Broyden. The convergence of a class of double-rank minimization algorithms 1. general considerations. *IMA Journal of Applied Mathematics*, 6(1):76–90, 1970.

S. Bubeck, R. Munos, G. Stoltz, and Szepesvári C. X-armed bandits. *Machine Learning Research*, 12:1587–1627, 2011.

C. Chang and C. Lin. LIBSVM : A Library for Support Vector Machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2:1–39, 2011.

E. Contal, C. Malherbe, and N. Vayatis. Optimization for gaussian processes via chaining. *In NIPS workshop on Bayesian Optimization*, 2015.

C. Cortes and V. Vapnik. Support vector networks. *Machine Learning*, 20(3):273–297, 1995.

T. M. Cover and J. A. Thomas. Elements of information theory. *John Wiley & Sons*, 2012.

C. A. Floudas and P. M. Pardalos. Optimization in computational chemistry and molecular biology: Local and global approaches. *Nonconvex Optimization and Its Applications. Springer*, 2000.

H. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29(5):1189–1232, 2001.

V. Gabillon, M. Ghavamzadeh, and A. Lazaric. Best arm identification: A unified approach to fixed budget and fixed confidence. *In Neural Information Processing Systems (NIPS)*, 2012.

J.-B. Grill, M. Valko, and R. Munos. Black-box optimization of noisy functions with unknown smoothness. *In Neural Information Processing Systems (NIPS)*, 2015.

W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.

M. Hoffman, B. Shahriari, and N. de Freitas. On correlation and budget constraints in model-basedbandit optimization with application to automatic machine learning. *In Proceedings of the 17th International Conference on Artificial Intelligence and Statistics (AIStats)*, 2014.

D. R. Jones, C. D. Perttunen, and B. E. Stuckman. Lipschitzian optimization without the lipschitz constant. *Journal of Optimzation Theory and Applications*, 79(1):157–181, 1993.

D. R. Jones, M. Schonlau, and W. J. Welch. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13(4):455–492, 1998.

S. Kalyanakrishnan, A. Tewari, P. Auer, and P. Stone. Pac subset selection in stochastic multi-armed bandits. *In International Conference on Machine Learning (ICML)*, 2012.

E. Kaufmann, O. Cappé, and A. Garivier. On bayesian upper conf. bounds for bandit problems. *In Proceedings of the 15th International Conference on Artificial Intelligence and Statistics (AIStats)*, 2012a.

E. Kaufmann, N. Korda, and R. Munos. Thompson sampling: an asymptotically optimal finite-time analysis. *In International Conference on Algorithmic Learning Theory*, 2012b.

B. Kennel. KDTREE 2: Fortran 95 and C++ software to efficiently search for near neighbors in a multi-dimensional Euclidean space. *arXiv preprint arXiv:0408067*, 2004.

L. Kocsis and C. Szepesvári. Bandit based monte-carlo planning. *In Proceedings of the 17th European Conference on Machine Learning (ECML)*, pages 282–293, 2006.

A. Krause and C. E. Guestrin. Near-optimal nonmyopic value of information in graphical models. *arXiv:1207.1394*, 2012.

H. Kushner. A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise. *Journal of Basic Engineering*, 86:97–106, 1964.

Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

L. Li, K. Jamieson, G. A. Rostamizadeh, and A. Talwalkar. Efficient hyperparameter optimization and infinitely many armed bandits. *arXiv:1603.06560v1*, 2016.

D. J. Lizotte. *Practical bayesian optimization.* PhD thesis, University of Alberta, 2008.

B. Matérn. *Spatial variation.* Springer-Verlag, 1960.

C. G. Moles, P. Mendes, and J. R. Banga. Parameter estimation in biochemical pathways: a comparison of global optimization methods. *Genome Research*, 13(11):2467–2474, 2003.

J. Močkus, V Tiesis, and A Žilinskas. The application of bayesian methods for seeking the extremum. In L. C. W. Dixon and G. P. Szegö, editors, *Towards Global Optimisation*, volume 2, pages 117–128. Elsevier, 1978.

G. Nemhauser, L. Wolsey, and M. Fisher. An analysis of the approximations for maximizing submodular set functions. *Math. Prog.*, 14:265–294, 1978.

C. E. Rasmussen and C. Williams. *Gaussian processes for machine learning.* The MIT Press, 2006.

E. Rumelhart, E. Hinton, and J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

E. Schapire. A brief introduction to boosting. In *IJCAI International Joint Conference on Artificial Intelligence*, volume 2, pages 1401–1406, 1999.

J. Sherman and W. J. Morrison. Adjustment of an inverse matrix corresponding to a change in one element of a given matrix. *The Annals of Mathematical Statistics*, 21(1):124–127, 1950.

D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T Graepel, and D Hassabis. Mastering the game of go with deep neuralnetworks and tree search. *Nature*, 529:484–489, 2016.

N. Srinivas, A. Krause, S. M. Kakade, and M. Seeger. aussian process optimization in the bandit setting: No regret and experimental design. *arXiv:0912.3995*, 2009.

A. Verikas, A. Gelzinis, and M. Bacauskiene. Mining data with random forests: A survey and results of new tests. *Pattern Recognition*, 44(2):330–349, 2011.

G. Wang and S. Shan. Review of metamodeling techniques in support of engineering design optimization. *Journal of Mechanical Design*, 129(4):370–380, 2007.

W. T. Ziemba and R. G. Vickson. Stochastic optimization models in finance. *World Scientific Singapore*, 2006.

Appendices

A Proof of Lemma 5

Proof. For all $\delta > 0$, let $\mathbf{x}_{h,i}(\delta)$ be an arm of $\mathcal{P}_{h,i}$ s.t.,

$$f(\mathbf{x}_{h,i}(\delta)) \geq f^*_{h,i} - \delta = f^* - \Delta_{h,i} - \delta. \tag{3}$$

It follows that for all $\mathbf{y} \in \mathcal{P}_{h,i}$ and all $\delta > 0$,

$$\begin{aligned}
f^* - f(\mathbf{y}) &\leq f^* - f(\mathbf{x}_{h,i}(\delta)) + \max\{f^* - f(\mathbf{x}_{h,i}(\delta)), l(\mathbf{x}_{h,i}(\delta), \mathbf{y})\} && \text{by Assumption 2} \\
&\leq \Delta_{h,i} + \delta + \max\{\Delta_{h,i} + \delta, l(\mathbf{x}_{h,i}(\delta), \mathbf{y})\} && \text{by Eq. 3} \\
&\leq \Delta_{h,i} + \delta + \max\{\Delta_{h,i} + \delta, \operatorname{diam}(\mathcal{P}_{h,i})\} && \text{by definition of the diameter}
\end{aligned}$$

Now letting δ tends toward 0, we have for all $\mathbf{y} \in \mathcal{P}_{h,i}$,

$$\begin{aligned}
f^* - f(\mathbf{y}) &\leq \Delta_{h,i} + \max\{\Delta_{h,i}, \operatorname{diam}(\mathcal{P}_{h,i})\} \\
&\leq c\nu_1\rho^h + \max\{c\nu_1\rho^h, \nu_1\rho^h\} && \text{by Assumption 2} \\
&= \max\{2c, c+1\}\nu_1\rho^h.
\end{aligned}$$

□

B Regret Analysis for HCT

Before starting the proof, we first fix some constants and introduce some additional notations as required for the analysis.

- $c_1 \coloneqq (\rho/(3\nu))^{1/8}$, $c \coloneqq 2\sqrt{1/(1-\rho)}$;
- $\forall 1 \leq h \leq H(t)$ and $t > 0$, $\mathcal{I}_h(t)$ denotes the set of all nodes created by HCT at level h up to step t;
- $\forall 1 \leq h \leq H(t)$ and $t > 0$, $\mathcal{I}_h^+(t)$ denotes the subset of $\mathcal{I}_h(t)$ which contains only the internal nodes;
- At each step t, (h_t, i_t) denotes the node selected by the algorithm;
- $\mathcal{C}_{h,i} \coloneqq \{t = 1, \cdots, n : (h_t, i_t) = (h, i)\}$;
- $\mathcal{C}^+_{h,i} \coloneqq \mathcal{C}_{h+1,2i-1} \cup \mathcal{C}_{h+1,2i}$;
- $\bar{t}_{h,i} \coloneqq \max_{t \in \mathcal{C}_{h,i}} t$ denotes the last time (h, i) has been selected;
- $\tilde{t}_{h,i} \coloneqq \max_{t \in \mathcal{C}^+_{h,i}} t$ denotes the last time when one of its children has been selected;
- $t_{h,i} \coloneqq \min\{t : T_{h,i}(t) \geq \tau_h(t)\}$ is the time when (h, i) is expanded.

Another important notion in the HCT algorithm is the threshold τ_h on the number of pulls needed before a node at level h can be expanded. τ_h is chosed such that the two confidence terms in $U_{h,i}$ are roughly equivalent, that is,

$$\nu\rho^h = c\sqrt{\frac{\log(1/\tilde{\delta}(t^+))}{\tau_h(t)}},$$

thus we choose,

$$\tau_h(t) = \lceil \frac{c^2 \log(1/\tilde{\delta}(t^+))}{\nu^2}\rho^{-2h} \rceil.$$

Since t^+ is defined as $2^{\lceil \log(t) \rceil}$, we have $t \leq t^+ \leq 2t$. In addition, log is an increasing function, thus we have,

$$\frac{c^2}{\nu^2}\rho^{-2h} \leq \frac{c^2 \log(1/\tilde{\delta}(t))}{\nu^2}\rho^{-2h} \leq \tau_h(t) \leq \frac{c^2 \log(2/\tilde{\delta}(t))}{\nu^2}\rho^{-2h}, \tag{4}$$

where the first inequality follows the fact that $0 < \tilde{\delta}(t) \leq \frac{1}{2}$. Now we begin our analysis by bounding the maximum depth of the trees constructed by HCT-*iid*.

Lemma 13. *Given $\tau_h(t)$ for the expansion of nodes at depth h, the maximum depth $H_{\max}(n)$ of the tree $\mathcal{T}_n$ will be*

$$H_{\max}(n) = \frac{1}{2(1-\rho)} \log\left(\frac{n\nu^2}{c^2\rho^2}\right).$$

Proof. The deepest tree that can be constructed by HCT-*iid* is a linear tree, where at each level one unique node is expanded which means $|\mathcal{I}_h^+(n)| = 1$ and $|\mathcal{I}_h(n)| = 2$ for all $h < H(n)$. Thus we have,

$$\begin{aligned}
n &= \sum_{h=0}^{H(n)} \sum_{i \in \mathcal{I}_h(n)} T_{h,i}(n) \\
&\geq \sum_{h=0}^{H(n)-1} \sum_{i \in \mathcal{I}_h^+(n)} T_{h,i}(n) \\
&\geq \sum_{h=0}^{H(n)-1} \sum_{i \in \mathcal{I}_h^+(n)} T_{h,i}(t_{h,i}) \\
&\geq \sum_{h=0}^{H(n)-1} \sum_{i \in \mathcal{I}_h^+(n)} \tau_h(t_{h,i}) && \text{by definition of } t_{h,i} \\
&\geq \sum_{h=0}^{H(n)-1} \frac{c^2}{\nu^2}\rho^{-2h} && \text{by Eq. 4} \\
&\geq \frac{(c\rho)^2}{\nu^2}\rho^{-2H(n)} H(n) && \text{since } h \leq H(n)-1 \\
&\geq \frac{(c\rho)^2}{\nu^2}\rho^{-2H(n)}.
\end{aligned}$$

By solving this expression, we obtain

$$\begin{aligned}
H(n) &\leq \frac{1}{2}\log\left(\frac{n\nu^2}{c^2\rho^2}\right)/\log(1/\rho) \\
&\leq \frac{1}{2(1-\rho)}\log\left(\frac{n\nu^2}{c^2\rho^2}\right) && \text{follows the fact that } \log(1/\rho) \geq 1-\rho \\
&= H_{\max}(n).
\end{aligned}$$

□

We now introduce a high probability event $\mathcal{E}_t$ under which the mean reward of all expanded nodes is with a confidence interval of the empirical mean. For that purpose, we define at first the set of all possible nodes in trees of maximum depth $H_{\max}(t)$ as

$$\mathcal{L}_t = \bigcup_{\mathcal{T}:\text{depth}(\mathcal{T})\leq H_{\max}(t)} \text{Nodes}(\mathcal{T}),$$

then we define the event as

$$\mathcal{E}_t = \left\{\forall (h,i) \in \mathcal{L}_t, \forall T_{h,i}(t) = 1\ldots t : |\hat{\mu}_{h,i}(t) - \mu_{h,i}| \leq c\sqrt{\frac{\log(1/\tilde{\delta}(t))}{T_{h,i}(t)}}\right\},$$

and we have the following lemma.

Lemma 14. *With c_1 and c defined as in the beginning of this section, we have for any fixed time step t,*

$$\mathbb{P}\left[\mathcal{E}_t\right] \geq 1 - \frac{\delta}{t^6}.$$

Proof. We upper bound the probability that the complementary event $\mathcal{E}_t^c$ holds.

$$\begin{aligned}
\mathbb{P}\left[\mathcal{E}_t^c\right] &\leq \sum_{(h,i)\in\mathcal{L}_t}\sum_{T_{h,i}(t)=1}^{t} \mathbb{P}\left[|\hat{\mu}_{h,i}(t) - \mu_{h,i}| \geq c\sqrt{\frac{\log(1/\tilde{\delta}(t))}{T_{h,i}(t)}}\right] && \text{union bound} \\
&\leq \sum_{(h,i)\in\mathcal{L}_t}\sum_{T_{h,i}(t)=1}^{t} 2\exp\left(-2c^2\log(1/\tilde{\delta}(t))\right) && \text{Chernoff-Hoeffding inequality} \\
&= 2\exp\left(-2c^2\log(1/\tilde{\delta}(t))\right) t|\mathcal{L}_t| \\
&= 2(\tilde{\delta}(t))^{2c^2} t|\mathcal{L}_t| \\
&\leq 2(\tilde{\delta}(t))^{2c^2} t 2^{H_{max}(t)+1} \\
&= 4t(\tilde{\delta}(t))^{2c^2}\left(\frac{t\nu^2}{c^2\rho^2}\right)^{\frac{1}{2(1-\rho)}} && \text{by Lemma 13} \\
&\leq 4t\left(\frac{\delta}{t}(\rho/(3\nu))^{1/8})^{\frac{8}{1-\rho}}\right)\left(\frac{t\nu^2(1-\rho)}{4\rho^2}\right)^{\frac{1}{2(1-\rho)}} && \text{plug in the expression of } c \text{ and } c_1
\end{aligned}$$

$$
\begin{aligned}
&= 4t\left(\frac{\delta}{t}\right)^{\frac{8}{1-\rho}}\left(\frac{\rho}{3\nu}\right)^{\frac{1}{1-\rho}} t^{\frac{1}{2(1-\rho)}}\left(\frac{\nu\sqrt{1-\rho}}{2\rho}\right)^{\frac{1}{1-\rho}} \\
&\leq \frac{2}{3}\delta t^{\frac{-2\rho-13}{2(1-\rho)}} \\
&\leq \frac{\delta}{t^6}
\end{aligned}
$$

□

Now that we have a high probability event, we consider of decomposing the regret of HCT-*iid* into two terms depending on whether $\mathcal{E}_t$ holds or not. Let us denote $\Delta_t = f^* - r_t$, then we decompose the regret as

$$
R_n = \sum_{t=1}^{n} \Delta_t = \sum_{t=1}^{n} \Delta_t \mathbb{1}_{\mathcal{E}_t} + \sum_{t=1}^{n} \Delta_t \mathbb{1}_{\mathcal{E}_t^c} = R_n^{\mathcal{E}} + R_n^{\mathcal{E}^c}
$$

We first bound the failing confidence term $R_n^{\mathcal{E}^c}$.

Lemma 15. *With the same parameters c and c_1, the regret of HCT-iid when confidence intervals fail to hold is bounded as*

$$
R_n^{\mathcal{E}^c} \leq \sqrt{n}
$$

with probability $1 - \delta/5n^2$.

Proof. We split the term into time steps from 1 to $\sqrt{n}$ and the rest,

$$
R_n^{\mathcal{E}^c} = \sum_{t=1}^{n} \Delta_t \mathbb{1}_{\mathcal{E}_t^c} = \sum_{t=1}^{\sqrt{n}} \Delta_t \mathbb{1}_{\mathcal{E}_t^c} + \sum_{t=\sqrt{n}+1}^{n} \Delta_t \mathbb{1}_{\mathcal{E}_t^c}
$$

The first term can be bounded trivially by $\sqrt{n}$ since $|\Delta_t| \leq 1$, now we demonstrate that the probability that the second term is non zero is bounded by $\frac{\delta}{5n^2}$.

$$
\begin{aligned}
\mathbb{P}\left[\sum_{t=\sqrt{n}+1}^{n} \Delta_t \mathbb{1}_{\mathcal{E}_t^c} > 0\right] &= \mathbb{P}\left[\bigcup_{t=\sqrt{n}+1}^{n} \mathcal{E}_t^c\right] \\
&\leq \sum_{t=\sqrt{n}+1}^{n} \mathbb{P}\left[\mathcal{E}_t^c\right] && \text{union bound} \\
&\leq \sum_{t=\sqrt{n}+1}^{n} \frac{\delta}{t^6} && \text{by Lemma 14} \\
&\leq \int_{\sqrt{n}}^{\infty} \frac{\delta}{t^6}\,\mathrm{d}t \\
&= \frac{\delta}{5n^{5/2}} \\
&\leq \frac{\delta}{5n^2}
\end{aligned}
$$

□

Now we come back to the Theorem 11. We study the regret of HCT under events $\{\mathcal{E}_t\}$ and prove the following result,

$$R_n \leq 2\sqrt{2n\log(\frac{4n^2}{\delta})} + 3\left(\frac{2^{2d+6}\nu^d C\rho^d}{(1-\rho)^{1-d/2}}\right)^{\frac{1}{d+2}}\left(\log\left(\frac{2n}{\delta}\sqrt[8]{\frac{3\nu}{\rho}}\right)\right)^{\frac{1}{d+2}} n^{\frac{d+1}{d+2}}$$

with probability $1-\delta$.

Proof. The proof is constructed in 3 steps.

Step 1: Decomposition of the regret. We start by further decomposing the instantaneous regret into two terms,

$$\Delta_t = f^* - r_t = f^* - f(\mathbf{x}_{h_t,i_t}) + f(\mathbf{x}_{h_t,i_t}) - r_t = \Delta_{h_t,i_t} + \hat{\Delta}_t,$$

the regret of HCT-*iid* when confidence intervals hold can thus be rewritten as

$$R_n^{\mathcal{E}} = \sum_{t=1}^{n} \Delta_{h_t,i_t}\mathbb{1}_{\mathcal{E}_t} + \sum_{t=1}^{n}\hat{\Delta}_t\mathbb{1}_{\mathcal{E}_t} \leq \sum_{t=1}^{n}\Delta_{h_t,i_t}\mathbb{1}_{\mathcal{E}_t} + \sum_{t=1}^{n}\hat{\Delta}_t = \tilde{R}_n^{\mathcal{E}} + \hat{R}_n^{\mathcal{E}}. \tag{5}$$

We notice that the sequence $\{\hat{\Delta}_t\}_{t=1}^n$ is a bounded martingale difference sequence since $\mathbb{E}\left[\hat{\Delta}_t|\mathcal{F}_{t-1}\right] = 0$ and $|\hat{\Delta}_t| \leq 1$, we thus apply the Azuma's inequality on this sequence,

$$\hat{R}_n^{\mathcal{E}} \leq \sqrt{2n\log(4n^2/\delta)} \tag{6}$$

with probability $1-\frac{\delta}{4n^2}$.

Step 2: Preliminary bound on the regret of selected nodes and their parents. Now we proceed with the bound of the first term $\tilde{R}_n^{\mathcal{E}}$. Let $(h',i') \in P_t$ and (h'',i'') be the node which immediately follow (h',i') in P_t. By definition of B-value and U-value, we have

$$B_{h',i'}(t) \leq \max(B_{h'+1,2i'-1}(t), B_{h'+1,2i'}(t)) = B_{h'',i''}(t) \tag{7}$$

where the last equality follows the fact that the subroutine *OptTraverse* selects the node with the largest B-value. By iterating the previous inequality along with the path P_t until the selected node (h_t,i_t) and its parent (h_t^p,i_t^p), we obtain

$$\forall (h',i') \in P_t, B_{h',i'}(t) \leq B_{h_t,i_t}(t) \leq U_{h_t,i_t}(t),$$

$$\forall (h',i') \in P_t - (h_t,i_t), B_{h',i'}(t) \leq B_{h_t^p,i_t^p}(t) \leq U_{h_t^p,i_t^p}(t).$$

Since the root, which is an optimal node, is in P_t, thus there exists at least one optimal node (h^*,i^*) in P_t, thus we have

$$B_{h^*,i^*}(t) \leq U_{h_t,i_t}(t), \tag{8}$$

$$B_{h^*,i^*}(t) \leq U_{h_t^p,i_t^p}(t). \tag{9}$$

Now we expand the Eq. 8 on both sides under $\mathcal{E}_t$. First we have

$$U_{h_t,i_t}(t) = \hat{\mu}_{h_t,i_t}(t) + \nu\rho^{h_t} + c\sqrt{\frac{\log(1/\tilde{\delta}(t^+))}{T_{h_t,i_t}(t)}} \tag{10}$$

$$\leq f(\mathbf{x}_{h_t,i_t}) + \nu\rho^{h_t} + 2c\sqrt{\frac{\log(1/\tilde{\delta}(t^+))}{T_{h_t,i_t}(t)}}. \tag{11}$$

And the same result holds for the parent of the selected node:

$$U_{h_t^p,i_t^p}(t) \leq f(\mathbf{x}_{h_t^p,i_t^p}) + \nu\rho^{h_t^p} + 2c\sqrt{\frac{\log(1/\tilde{\delta}(t^+))}{T_{h_t^p,i_t^p}(t)}}. \tag{12}$$

We now show that for any optimal node (h^*, i^*), $U_{h^*,i^*}(t)$ is a valid upper bound on f^*:

$$\begin{aligned}
U_{h^*,i^*}(t) &= \hat{\mu}_{h^*,i^*}(t) + \nu\rho^{h^*} + c\sqrt{\frac{\log(1/\tilde{\delta}(t^+))}{T_{h^*,i^*}(t)}} \\
&\geq \hat{\mu}_{h^*,i^*}(t) + \nu\rho^{h^*} + c\sqrt{\frac{\log(1/\tilde{\delta}(t))}{T_{h^*,i^*}(t)}} && \text{follows the fact that } t^+ \geq t \\
&\geq f(\mathbf{x}_{h^*,i^*}) + \nu\rho^{h^*} && \text{since we are under } \mathcal{E}_t \\
&\geq f^*. && \text{by Assumption 2}
\end{aligned}$$

If an optimal node (h^*, i^*) is a leaf, then $B_{h^*,i^*}(t) = U_{h^*,i^*}(t)$ is also a valid upper bound on f^*. Otherwise, there always exists a leaf which contains the optimum for which (h^*, i^*) is its ancestor. Now if we propagate the bound backward from this leaf to (h^*, i^*) through Eq. 7, we have that $B_{h^*,i^*}(t)$ is still a valid upper bound on f^*. Thus for any optimal node (h^*, i^*), at time t under $\mathcal{E}_t$, we have

$$B_{h^*,i^*}(t) \geq f^*. \tag{13}$$

Then if we combine this Eq. 13 with Eq. 8 and Eq. 11, we obtain

$$\Delta_{h_t,i_t} = f^* - f(\mathbf{x}_{h_t,i_t}) \leq \nu\rho^{h_t} + 2c\sqrt{\frac{\log(1/\tilde{\delta}(t^+))}{T_{h_t,i_t}(t)}}. \tag{14}$$

The same result holds for its parent,

$$\Delta_{h_t^p,i_t^p} = f^* - f(\mathbf{x}_{h_t^p,i_t^p}) \leq \nu\rho^{h_t^p} + 2c\sqrt{\frac{\log(1/\tilde{\delta}(t^+))}{T_{h_t^p,i_t^p}(t)}}. \tag{15}$$

Now we can still refine a bit these two expressions. The subroutine *OptTraverse* tells us that HCT-*iid* only selects a node when $T_{h,i}(t) < \tau_h(t)$. Thus by the definition of $\tau_{h_t}(t)$, we have

$$\Delta_{h_t,i_t} \leq 3c\sqrt{\frac{\log(2/\tilde{\delta}(t))}{T_{h_t,i_t}(t)}}. \tag{16}$$

On the other side, the *OptTraverse* function tells us that $T_{h_t^p,i_t^p}(t) \geq \tau_{h_t^p}(t)$, thus

$$\Delta_{h_t^p,i_t^p} \leq 3\nu\rho^{h_t^p}, \tag{17}$$

which means that every selected node has a parent which is $3\nu\rho^{h_t-1}$-optimal.

Step 3: Bound on the cumulative regret. Now we come back to the term $\tilde{R}_n^{\mathcal{E}}$, and we split it into different depths. Let $1 \leq \bar{H} \leq H(n)$ be a constant to be fixed later. We have

$$
\begin{aligned}
\tilde{R}_n^{\mathcal{E}} &= \sum_{t=1}^{n} \Delta_{h_t,i_t} \mathbb{1}_{\mathcal{E}_t} \\
&\leq \sum_{h=0}^{H(n)} \sum_{i \in \mathcal{I}_h(n)} \sum_{t=1}^{n} \Delta_{h,i} \mathbb{1}_{(h_t,i_t)=(h,i)} \mathbb{1}_{\mathcal{E}_t} \\
&\leq \sum_{h=0}^{H(n)} \sum_{i \in \mathcal{I}_h(n)} \sum_{t=1}^{n} 3c \sqrt{\frac{\log(2/\tilde{\delta}(t))}{T_{h,i}(t)}} \mathbb{1}_{(h_t,i_t)=(h,i)} && \text{by Eq. 16} \\
&= \sum_{h=0}^{\bar{H}} \sum_{i \in \mathcal{I}_h(n)} \sum_{t=1}^{n} 3c \sqrt{\frac{\log(2/\tilde{\delta}(t))}{T_{h,i}(t)}} \mathbb{1}_{(h_t,i_t)=(h,i)} + \sum_{h=\bar{H}+1}^{H(n)} \sum_{i \in \mathcal{I}_h(n)} \sum_{t=1}^{n} 3c \sqrt{\frac{\log(2/\tilde{\delta}(t))}{T_{h,i}(t)}} \mathbb{1}_{(h_t,i_t)=(h,i)} \\
&\leq \sum_{h=0}^{\bar{H}} \sum_{i \in \mathcal{I}_h(n)} \sum_{s=1}^{\tau_h(\bar{t}_{h,i})} 3c \sqrt{\frac{\log(2/\tilde{\delta}(\bar{t}_{h,i}))}{s}} + \sum_{h=\bar{H}+1}^{H(n)} \sum_{i \in \mathcal{I}_h(n)} \sum_{s=1}^{T_{h,i}(n)} 3c \sqrt{\frac{\log(2/\tilde{\delta}(\bar{t}_{h,i}))}{s}} \\
&\leq \sum_{h=0}^{\bar{H}} \sum_{i \in \mathcal{I}_h(n)} \int_1^{\tau_h(\bar{t}_{h,i})} 3c \sqrt{\frac{\log(2/\tilde{\delta}(\bar{t}_{h,i}))}{s}} \, \mathrm{d}s + \sum_{h=\bar{H}+1}^{H(n)} \sum_{i \in \mathcal{I}_h(n)} \int_1^{T_{h,i}(n)} 3c \sqrt{\frac{\log(2/\tilde{\delta}(\bar{t}_{h,i}))}{s}} \, \mathrm{d}s \\
&\leq \sum_{h=0}^{\bar{H}} \sum_{i \in \mathcal{I}_h(n)} 6c \sqrt{\tau_h(\bar{t}_{h,i}) \log(2/\tilde{\delta}(\bar{t}_{h,i}))} + \sum_{h=\bar{H}+1}^{H(n)} \sum_{i \in \mathcal{I}_h(n)} 6c \sqrt{T_{h,i}(n) \log(2/\tilde{\delta}(\bar{t}_{h,i}))} \\
&= 6c \left(\underbrace{\sum_{h=0}^{\bar{H}} \sum_{i \in \mathcal{I}_h(n)} \sqrt{\tau_h(\bar{t}_{h,i}) \log(2/\tilde{\delta}(\bar{t}_{h,i}))}}_{\text{(a)}} + \underbrace{\sum_{h=\bar{H}+1}^{H(n)} \sum_{i \in \mathcal{I}_h(n)} \sqrt{T_{h,i}(n) \log(2/\tilde{\delta}(\bar{t}_{h,i}))}}_{\text{(b)}} \right).
\end{aligned}
$$

We now bound separately these two terms (a) and (b). Since $\bar{t}_{h,i} \leq n$, we have

$$
\text{(a)} \leq \sum_{h=0}^{\bar{H}} \sum_{i \in \mathcal{I}_h(n)} \sqrt{\tau_h(n) \log(2/\tilde{\delta}(n))} \leq \sum_{h=0}^{\bar{H}} |\mathcal{I}_h(n)| \sqrt{\tau_h(n) \log(2/\tilde{\delta}(n))}. \tag{18}
$$

We notice that the covering tree is binary tree, thus we have $|\mathcal{I}_h(n)| \leq 2|\mathcal{I}_{h-1}(n)|$, and we recall that HCT-*iid* only selects a node (h_t, i_t) when its parent is $3\nu\rho^{h_t-1}$-optimal, thus by definition of the near-optimality dimension for HCT, we have

$$
|\mathcal{I}_h(n)| \leq |2\mathcal{I}_{h-1}(n)| \leq 2C\rho^{-d(h-1)} \tag{19}
$$

where d is the near-optimality dimension. Now we come back to the term (a) and we obtain that

$$
\text{(a)} \leq \sum_{h=0}^{\bar{H}} 2C\rho^{-d(h-1)} \sqrt{\tau_h(n) \log(2/\tilde{\delta}(n))}
$$

$$= \sum_{h=0}^{\bar{H}} 2C\rho^{-d(h-1)} \sqrt{\frac{c^2 \log(2/\tilde{\delta}(n))}{\nu^2} \rho^{-2h} \log(2/\tilde{\delta}(n))} \qquad \text{by Eq. 4}$$

$$= 2C\rho^d \frac{c \log(2/\tilde{\delta}(n))}{\nu} \sum_{h=0}^{\bar{H}} \rho^{-h(d+1)}.$$

Therefore we can bound (a) as

$$\text{(a)} \leq 2C\rho^d \frac{c \log(2/\tilde{\delta}(n))}{\nu} \frac{\rho^{-\bar{H}(d+1)}}{1-\rho}. \tag{20}$$

Now we proceed to bound the second term (b). Using the Cauchy-Schwartz inequality, we have

$$\text{(b)} \leq \sqrt{\sum_{h=\bar{H}+1}^{H(n)} \sum_{i \in \mathcal{I}_h(n)} \log(2/\tilde{\delta}(\bar{t}_{h,i}))} \sqrt{\sum_{h=\bar{H}+1}^{H(n)} \sum_{i \in \mathcal{I}_h(n)} T_{h,i}(n)} \leq \sqrt{n \sum_{h=\bar{H}+1}^{H(n)} \sum_{i \in \mathcal{I}_h(n)} \log(2/\tilde{\delta}(\bar{t}_{h,i}))}, \tag{21}$$

where we trivially bound the second square root by the totol number of pulls. Now we focus on the first square root. Recall that HCT-*iid* only selects a node when $T_{h,i}(t) \geq \tau_h(t)$ for its parent, thus we have $T_{h,i}(\tilde{t}_{h,i}) \geq \tau_h(\tilde{t}_{h,i})$ and the following sequence of inequalities.

$$\begin{aligned}
n &= \sum_{h=0}^{H(n)} \sum_{i \in \mathcal{I}_h(n)} T_{h,i}(n) \\
&\geq \sum_{h=0}^{H(n)-1} \sum_{i \in \mathcal{I}_h^+(n)} T_{h,i}(n) \\
&\geq \sum_{h=0}^{H(n)-1} \sum_{i \in \mathcal{I}_h^+(n)} T_{h,i}(\tilde{t}_{h,i}) && \tilde{t}_{h,i} \text{ well defined for } i \in \mathcal{I}_h^+(n) \\
&\geq \sum_{h=0}^{H(n)-1} \sum_{i \in \mathcal{I}_h^+(n)} \tau_h(\tilde{t}_{h,i}) \\
&\geq \sum_{h=\bar{H}}^{H(n)-1} \sum_{i \in \mathcal{I}_h^+(n)} \tau_h(\tilde{t}_{h,i}) \\
&= \sum_{h=\bar{H}}^{H(n)-1} \sum_{i \in \mathcal{I}_h^+(n)} \frac{c^2 \log(1/\tilde{\delta}(\tilde{t}_{h,i}^+)))}{\nu^2} \rho^{-2h} \\
&\geq \sum_{h=\bar{H}}^{H(n)-1} \sum_{i \in \mathcal{I}_h^+(n)} \frac{c^2 \log(1/\tilde{\delta}(\tilde{t}_{h,i}^+)))}{\nu^2} \rho^{-2\bar{H}} \\
&= \frac{c^2 \rho^{-2\bar{H}}}{\nu^2} \sum_{h=\bar{H}}^{H(n)-1} \sum_{i \in \mathcal{I}_h^+(n)} \log(1/\tilde{\delta}(\tilde{t}_{h,i}^+)))
\end{aligned}$$

$$
\begin{aligned}
&= \frac{c^2\rho^{-2\bar{H}}}{\nu^2} \sum_{h=\bar{H}}^{H(n)-1} \sum_{i\in\mathcal{I}_h^+(n)} \log(1/\tilde{\delta}([\max(\bar{t}_{h+1,2i-1}, \bar{t}_{h+1,2i})]^+)) && \text{since } \tilde{t}_{h,i} = \max(\bar{t}_{h+1,2i-1}, \bar{t}_{h+1,2i}) \\
&= \frac{c^2\rho^{-2\bar{H}}}{\nu^2} \sum_{h=\bar{H}}^{H(n)-1} \sum_{i\in\mathcal{I}_h^+(n)} \log(1/\tilde{\delta}(\max(\bar{t}^+_{h+1,2i-1}, \bar{t}^+_{h+1,2i}))) && \forall t_1, t_2, [\max(t_1, t_2)]^+ = \max(t_1^+, t_2^+) \\
&= \frac{c^2\rho^{-2\bar{H}}}{\nu^2} \sum_{h=\bar{H}}^{H(n)-1} \sum_{i\in\mathcal{I}_h^+(n)} \max(\log(1/\tilde{\delta}(\bar{t}^+_{h+1,2i-1})), \log(1/\tilde{\delta}(\bar{t}^+_{h+1,2i}))) \\
&\geq \frac{c^2\rho^{-2\bar{H}}}{\nu^2} \sum_{h=\bar{H}}^{H(n)-1} \sum_{i\in\mathcal{I}_h^+(n)} \frac{\log(1/\tilde{\delta}(\bar{t}^+_{h+1,2i-1})) + \log(1/\tilde{\delta}(\bar{t}^+_{h+1,2i}))}{2} \\
&= \frac{c^2\rho^{-2\bar{H}}}{\nu^2} \sum_{h=\bar{H}+1}^{H(n)} \sum_{i\in\mathcal{I}_{h-1}^+(n)} \frac{\log(1/\tilde{\delta}(\bar{t}^+_{h,2i-1})) + \log(1/\tilde{\delta}(\bar{t}^+_{h,2i}))}{2} && \text{change of variables} \\
&= \frac{c^2\rho^{-2\bar{H}}}{2\nu^2} \sum_{h=\bar{H}+1}^{H(n)} \sum_{i\in\mathcal{I}_h^+(n)} \log(1/\tilde{\delta}(\bar{t}^+_{h,i})).
\end{aligned}
$$

Here the last equality relies on the fact that for any $h > 0$, $\mathcal{I}_h^+(n)$ covers all the internal nodes at level h, thus its children cover $\mathcal{I}_{h+1}(n)$. Thus we obtain

$$
\sum_{h=\bar{H}+1}^{H(n)} \sum_{i\in\mathcal{I}_h^+(n)} \log(1/\tilde{\delta}(\bar{t}^+_{h,i})) \leq \frac{2\nu^2\rho^{2\bar{H}}n}{c^2}. \tag{22}
$$

On the other hand, we have

$$
\begin{aligned}
\text{(b)} &\leq \sqrt{n \sum_{h=\bar{H}+1}^{H(n)} \sum_{i\in\mathcal{I}_h(n)} \log(2/\tilde{\delta}(\bar{t}_{h,i}))} \\
&\leq \sqrt{n \sum_{h=\bar{H}+1}^{H(n)} \sum_{i\in\mathcal{I}_h(n)} 2\log(1/\tilde{\delta}(\bar{t}_{h,i}))} \\
&\leq \sqrt{n \sum_{h=\bar{H}+1}^{H(n)} \sum_{i\in\mathcal{I}_h(n)} 2\log(1/\tilde{\delta}(\bar{t}^+_{h,i}))}. && \text{since } \bar{t}_{h,i} \leq \bar{t}^+_{h,i}
\end{aligned}
$$

Now by plugging Eq. 22 into this expression, we obtain

$$
\text{(b)} \leq \frac{2\nu\rho^{\bar{H}}n}{c}. \tag{23}
$$

Now if we combine this Eq. 23 with Eq. 20, we have the following bound for $\tilde{R}_n^{\mathcal{E}}$:

$$\tilde{R}_n^{\mathcal{E}} \leq 12\nu \left[C\rho^d \frac{c^2 \log(2/\tilde{\delta}(n))}{\nu^2} \frac{\rho^{-\bar{H}(d+1)}}{1-\rho} + \rho^{\bar{H}} n \right]. \tag{24}$$

We now choose $\bar{H}$ to minimize this bound by equalizing the two terms in the sum, and we obtain

$$\rho^{\bar{H}} = \left(\frac{C\rho^d c^2 \log(2/\tilde{\delta}(n))}{n(1-\rho)\nu^2} \right)^{\frac{1}{d+2}}, \tag{25}$$

which after being plugged into Eq. 24 gives us

$$\tilde{R}_n^{\mathcal{E}} \leq \frac{24\nu}{c} \left(\frac{C\rho^d c^2 \log(2/\tilde{\delta}(n))}{(1-\rho)\nu^2} \right)^{\frac{1}{d+2}} n^{\frac{d+1}{d+2}}. \tag{26}$$

Finally, combining this Eq. 26 with Eq. 6 and Lemma 15, we obtain

$$\begin{aligned} R_n &\leq \sqrt{n} + \sqrt{2n\log(\frac{4n^2}{\delta})} + \frac{12\nu}{\sqrt{1/(1-\rho)}} \left(\frac{4C\rho^d}{(1-\rho)^2\nu^2} \right)^{\frac{1}{d+2}} \left(\log\left(\frac{2n}{\delta} \sqrt[8]{\frac{3\nu}{\rho}} \right) \right)^{\frac{1}{d+2}} n^{\frac{d+1}{d+2}} \\ &= \sqrt{n} + \sqrt{2n\log(\frac{4n^2}{\delta})} + 3 \left(\frac{2^{2d+6}\nu^d C\rho^d}{(1-\rho)^{1-d/2}} \right)^{\frac{1}{d+2}} \left(\log\left(\frac{2n}{\delta} \sqrt[8]{\frac{3\nu}{\rho}} \right) \right)^{\frac{1}{d+2}} n^{\frac{d+1}{d+2}} \\ &\leq 2\sqrt{2n\log(\frac{4n^2}{\delta})} + 3 \left(\frac{2^{2d+6}\nu^d C\rho^d}{(1-\rho)^{1-d/2}} \right)^{\frac{1}{d+2}} \left(\log\left(\frac{2n}{\delta} \sqrt[8]{\frac{3\nu}{\rho}} \right) \right)^{\frac{1}{d+2}} n^{\frac{d+1}{d+2}} \end{aligned}$$

with probability $1-\delta$.

□

C Conditional Distributions of A Multivariate Gaussian Distribution

In this section we show how to compute the analytic formulae for mean and covariance of the posterior distribution which is not trivial. Here, instead of calculating the conditional density manually, we use the following trick.

Given a set of observations $\mathbf{r} = [r_1, r_2, \ldots, r_n]^T$ evaluated over $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n]^T$ and a set of new samples $\mathbf{r}_* = \left[r_1^{'}, r_2^{'}, \ldots, r_m^{'} \right]^T$ evaluated over $\mathbf{X}_* = [\mathbf{x}_1^{'}, \mathbf{x}_2^{'}, \ldots, \mathbf{x}_m^{'}]^T$ instead of one new sample (we thus obtain more general formulae), the joint distribution is given by

$$\begin{bmatrix} \mathbf{r} \\ \mathbf{r}_* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \mathbf{K}(\mathbf{X},\mathbf{X}) & \mathbf{K}(\mathbf{X},\mathbf{X}_*) \\ \mathbf{K}(\mathbf{X}_*,\mathbf{X}) & \mathbf{K}(\mathbf{X}_*,\mathbf{X}_*) \end{bmatrix} \right).$$

Notice that for the sake of simplicity, we assume that we are in a zero-mean and noise-free setting. To make the notations more readable, we denote $\mathbf{\Sigma}_{1,1} = \mathbf{K}(\mathbf{X},\mathbf{X})$, $\mathbf{\Sigma}_{1,2} = \mathbf{K}(\mathbf{X},\mathbf{X}_*)$, $\mathbf{\Sigma}_{2,1} = \mathbf{K}(\mathbf{X}_*,\mathbf{X})$ and $\mathbf{\Sigma}_{2,2} = \mathbf{K}(\mathbf{X}_*,\mathbf{X}_*)$. Now we need to demonstrate the following expressions:

$$\mathbb{E}\left[\mathbf{r}_* | \mathbf{r}\right] = \mathbf{\Sigma}_{1,2}^T \mathbf{\Sigma}_{1,1}^{-1} \mathbf{r},$$

$$\mathrm{Var}\left(\mathbf{r}_* | \mathbf{r}\right) = \mathbf{\Sigma}_{2,2} - \mathbf{\Sigma}_{1,2}^T \mathbf{\Sigma}_{1,1}^{-1} \mathbf{\Sigma}_{1,2}.$$

Proof. Let $\mathbf{z} = \mathbf{r}_* + \mathbf{A}\mathbf{r}$ where $\mathbf{A} = -\mathbf{\Sigma}_{1,2}^T \mathbf{\Sigma}_{1,1}^{-1}$. Then we have

$$\begin{aligned}
\mathrm{Cov}(\mathbf{z}, \mathbf{r}) &= \mathrm{Cov}(\mathbf{r}_* + \mathbf{A}\mathbf{r}, \mathbf{r}) \\
&= \mathrm{Cov}(\mathbf{r}_*, \mathbf{r}) + \mathrm{Cov}(\mathbf{A}\mathbf{r}, \mathbf{r}) \\
&= \mathbf{\Sigma}_{1,2}^T - \mathbf{\Sigma}_{1,2}^T \mathbf{\Sigma}_{1,1}^{-1} \mathbf{\Sigma}_{1,1}.
\end{aligned}$$

$\mathbf{z}$ and $\mathbf{r}$ are uncorrelated, and since they are jointly normal, thus they are independent, and we have

$$\begin{aligned}
\mathbb{E}\left[\mathbf{r}_* | \mathbf{r}\right] &= \mathbb{E}\left[\mathbf{z} - \mathbf{A}\mathbf{r} | \mathbf{r}\right] \\
&= \mathbb{E}\left[\mathbf{z} | \mathbf{r}\right] - \mathbb{E}\left[\mathbf{A}\mathbf{r} | \mathbf{r}\right] \\
&= \mathbb{E}\left[\mathbf{z}\right] - \mathbb{E}\left[\mathbf{A}\mathbf{r} | \mathbf{r}\right] \\
&= \mathbb{E}\left[\mathbf{r}_* + \mathbf{A}\mathbf{r}\right] - \mathbf{A}\mathbf{r} \\
&= \mathbb{E}\left[\mathbf{r}_*\right] + \mathbf{A}\mathbb{E}\left[\mathbf{r}\right] - \mathbf{A}\mathbf{r} \\
&= \mathbf{\Sigma}_{1,2}^T \mathbf{\Sigma}_{1,1}^{-1} \mathbf{r}.
\end{aligned}$$

On the other hand, for the covariance matrix, we have,

$$\begin{aligned}
\mathrm{Var}\left(\mathbf{r}_* | \mathbf{r}\right) &= \mathrm{Var}\left(\mathbf{z} - \mathbf{A}\mathbf{r} | \mathbf{r}\right) \\
&= \mathrm{Var}\left(\mathbf{z} | \mathbf{r}\right) + \mathrm{Var}\left(\mathbf{A}\mathbf{r} | \mathbf{r}\right) - \mathbf{A}\,\mathrm{Cov}(\mathbf{z}, -\mathbf{r}) - \mathrm{Cov}(\mathbf{z}, -\mathbf{r})\mathbf{A}^T \\
&= \mathrm{Var}\left(\mathbf{z} | \mathbf{r}\right) \\
&= \mathrm{Var}(\mathbf{z}) \\
&= \mathrm{Var}(\mathbf{r}_* + \mathbf{A}\mathbf{r}) \\
&= \mathrm{Var}(\mathbf{r}_*) + \mathbf{A}\,\mathrm{Var}(\mathbf{r})\mathbf{A}^T + \mathbf{A}\,\mathrm{Cov}(\mathbf{r}, \mathbf{r}_*) + \mathrm{Cov}(\mathbf{r}_*, \mathbf{r})\mathbf{A}^T \\
&= \mathbf{\Sigma}_{2,2} + \mathbf{A}\mathbf{\Sigma}_{1,1}\mathbf{A}^T + \mathbf{A}\mathbf{\Sigma}_{1,2} + \mathbf{\Sigma}_{1,2}^T \mathbf{A}^T \\
&= \mathbf{\Sigma}_{2,2} + \mathbf{\Sigma}_{1,2}^T \mathbf{\Sigma}_{1,1}^{-1} \mathbf{\Sigma}_{1,1} \mathbf{\Sigma}_{1,1}^{-1} \mathbf{\Sigma}_{1,2} - 2\mathbf{\Sigma}_{1,2}^T \mathbf{\Sigma}_{1,1}^{-1} \mathbf{\Sigma}_{1,2} \\
&= \mathbf{\Sigma}_{2,2} - \mathbf{\Sigma}_{1,2}^T \mathbf{\Sigma}_{1,1}^{-1} \mathbf{\Sigma}_{1,2}.
\end{aligned}$$

□