Exploration vs Exploitation in Bayesian Optimization

Ali Jalali
Turn Inc
ajalali@turn.com

Javad Azimi
InsightsOne Inc
azimi@insightsone.com

Xiaoli Fern
Oregon State University
xfern@eecs.oregon.edu

Abstract

The problem of optimizing unknown costly-to-evaluate functions has been studied for a long time in the context of Bayesian Optimization. Leveraging the Lipschitz property of the unknown function, we propose an algorithm with a distinct exploration phase followed by an exploitation phase. The exploration phase aims to select samples that shrink the search space as much as possible. The exploitation phase then focuses on the reduced search space and selects samples closest to the optimizer. Considering the Expected Improvement (EI) as a baseline, we empirically show that the proposed algorithm significantly outperforms EI.

1 Introduction

In many applications such as nanotechnology and finance, etc, we want to optimize an unknown function \( f(\cdot) \) that is costly to evaluate over a compact input space. Bayesian Optimization (BO) algorithms try to solve this problem with a small number of function evaluations. The typical BO algorithms have two key components: 1) A posterior model to predict the output value of the function at any arbitrary input point, and 2) A selection criterion to determine which point is going to be evaluated next.

In this paper, we focus on the design of the selection criterion for BO where the samples are chosen sequentially and a selection is made only after the function evaluations of the previous samples are revealed [5, 7]. We make a mild assumption that the unknown function is Lipschitz-continuous. Leveraging the Lipschitz property, we design a selection algorithm that operates in two distinct phases: the exploration phase and the exploitation phase.

The exploration phase of the proposed algorithm, at each step, selects a sample that eliminates the largest possible portion of the input space while guaranteeing with high probability that the eliminated part does not include the maximizer of the function. Hence, the exploration stage of the algorithm tries to shrink the search space of the function as much as possible. In contrast, the exploitation phase of our algorithm selects the point which is believed to be the closest sample to the optimal point with high probability. Experimental results over 8 real and synthetic benchmarks indicate that the proposed approach is able to outperform the Expected Improvement (EI) criterion, one of the current state-of-the-art BO selection methods. In particular, we show that our algorithm is better than EI both in terms of the mean and variance of the performance.

2 Finite Horizon Bayesian Optimization

Expected Improvement (EI) [6] is a popular method in Bayesian Optimization. Often, we use initial random samples as an exploration phase for EI. Fig 1 shows that as we increase the number of initial random samples, the performance of EI degrades; suggesting the use of better exploration phase. In this section, we introduce our optimal exploration algorithm (NBRS) that improves the performance of EI comparing to random initial sampling, and also, a new exploitation algorithm (NBIS) that can replace EI. The combination of NBRS + NBIS beats EI in most cases.
Figure 1: Plot of regret versus the number of random exploration for EI algorithm. For a fixed budget $n_b$, we run a number of experiments as follows: first we consider the case where there is 1 random sample followed by $n_b - 1$ EI samples, next we consider the case where there are 2 random samples followed by $n_b - 2$ EI samples and so on. This result shows that the best EI performance is when we do not do random exploration.

**Algorithm 1** Next Best exploRative Sample (NBRS)

**Input:** Maximum $M$, Lipschitz Constant $L$ and Set of observed samples $\{(x_1, f(x_1)), \ldots, (x_t, f(x_t))\}$

**Output:** Next best explorative sample $x$

Let $D_t = D - \bigcup_{i=1}^{t} S(x_i, r_{x_i})$ and $x \leftarrow \arg\max_{x \in D_t} \left\{ \text{Vol} \left( D_t \cap S \left( \frac{M - \mu_x}{L} - 1.5\sigma_x \right) \right) \right\}$

---

**Optimal Exploration Algorithm:** Generally, a good exploration algorithm should be able to shrink the search space, so that we are left with a small region to focus on during the exploiting stage. Let $D = \otimes_{i=1}^{d} [a_i, b_i] \subset \mathbb{R}^d$ be the Cartesian product of intervals $[a_i, b_i]$ for some $a_i < b_i$ and $i \in \{1, 2, \ldots, d\}$. Suppose the unknown function $f : D \mapsto [m, M]$ (with $f(x^*) = M$) is a Lipschitz function over $D$ with constant $L$, that is for all $x_1, x_2 \in D$, we have $|f(x_1) - f(x_2)| \leq L\|x_1 - x_2\|_2$.

For any point $x \in D$, let $r_x = \frac{M - f(x)}{L}$ be the associated radius to the point $x$. By Lipschitz continuity assumption, we know that $x^* \notin S(x, r_x)$, where, $S(x, r_x)$ is the set of all points inside the sphere (or circle) with radius $r_x$ centered at $x$ (and single point $x$ if $r_x \leq 0$); otherwise, the Lipschitz assumption is violated. This means if we have a sample at point $x$, then we do not need any more samples inside $S(x, r_x)$.

The expected value of $r_x$ satisfies $E[r_x] = \frac{|M - \mu_x|}{L}$. Since $f(x)$ is $N(\mu_x, \sigma_x^2)$, using Hoeffding inequality for all $\epsilon > 0$, we have $\mathbb{P} \left[ r_x < \frac{|M - \mu_x|}{L} - \epsilon \right] \leq \exp \left( -\frac{2\epsilon^2}{\sigma_x^2} \right)$. Replacing $\epsilon$ with $1.5\frac{\sigma_x}{L}$, the above inequality entails that with high probability (99%), $r_x \geq \frac{|M - \mu_x|}{L} - 1.5\sigma_x$. Hence, a “good” algorithm for exploration should try to find $x$ that maximizes the lower bound on $r_x$. This choice of $x$ will remove a large volume of points from the search space.

Note that if $x$ is close to the boundaries of $D$, then it might be the case that most of the volume of the sphere lies outside $D$. Also, the sphere associated with $x$ might have significant overlap with spheres of other points that are already selected. The pseudo code of this method is described in Algorithm 1, which we refer to as the Next Best exploRative Sample (NBRS) algorithm. NBRS achieves the optimal exploration in the sense that it maximizes the expected explored volume.

**Optimal Exploitation Algorithm:** Suppose we have explored the search space with $t$ samples and we want to find $x^* \in D_t$. In order to exploit, we would like to find points $x$ whose sphere is small. The reason is that if $r_x = \frac{M - f(x)}{L} \leq \gamma$ is small enough, then by local strong convexity of $f(\cdot)$ around $x^*$, for some constant $\kappa$ we have $\frac{\kappa}{L} \|x - x^*\|_2^2 \leq M - f(x) \leq L\gamma$.

We estimate $r_x$ by its mean $E[r_x] = \frac{|M - \mu_x|}{L}$. By Hoeffding inequality, for all $\epsilon > 0$, we have $\mathbb{P}\left[ r_x > \frac{|M - \mu_x|}{L} + \epsilon \right] \leq \exp \left( -\frac{2\epsilon^2}{\sigma_x^2} \right)$. Replacing $\epsilon$ with $1.5\frac{\sigma_x}{L}$, the above inequality entails that
Algorithm 2 Next Best Exploitative Sample (NBIS)

**Input:** Maximum $M$, Lipschitz Constant $L$ and Set of observed samples \{$(x_1, f(x_1)), \ldots, (x_q, f(x_q))$\}

**Output:** Next best exploitative sample $x$

Let $D = D - \bigcup_{i=1}^{q} S(x_i, r_x)$ and $x \leftarrow \arg\min_{x \in D} \text{Vol}\left( S(x, \frac{|M-M_{x,\mathcal{C}}|+1.5\sigma_{x,\mathcal{C}}}{L}) \right)$

<table>
<thead>
<tr>
<th>Benchmark Functions</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cosines(2)</td>
<td>$1 - (u^2 + v^2 - 0.3 \cos(3\pi u) - 0.3 \cos(3\pi v))$</td>
</tr>
<tr>
<td></td>
<td>$u = 1.6\pi - 0.5, v = 1.6\pi - 0.5$</td>
</tr>
<tr>
<td>Rosenbrock(2)</td>
<td>$10 - 100(y - x^2)^2 - (1 - x)^2$</td>
</tr>
<tr>
<td>Hartman(3,6)</td>
<td>$\sum_{i=1}^{4} \Omega_i \exp\left( -\sum_{j=1}^{d} A_{ij}(x_j - P_{ij})^2 \right)$</td>
</tr>
<tr>
<td></td>
<td>$\Omega_{i \times k}, A_{i \times d}, P_{i \times d}$ are constants</td>
</tr>
<tr>
<td>Michalewicz(5)</td>
<td>$- \sum_{i=1}^{5} \sin(x_i) \sin\left( \frac{i^2 x_i^2}{\pi} \right)^{20}$</td>
</tr>
<tr>
<td>Shekel(4)</td>
<td>$\sum_{i=1}^{10} \omega_{10} + \sum_{j=1}^{4} (x_j - B_{j,10})^2$</td>
</tr>
<tr>
<td></td>
<td>$\omega_{1 \times 10}, B_{4 \times 10}$ are constants</td>
</tr>
</tbody>
</table>

Table 1: Benchmark Functions

with high probability (99%), $r_x \leq \frac{|M-M_{x,\mathcal{C}}|+1.5\sigma_{x}}{L}$. Hence, a “good” algorithm for exploitation should try to find the point $x$ that minimizes the upper bound on $r_x$. This choice of $x$ introduces the expected closest point to $x^\ast$. We present the pseudo code of this method in Algorithm 2.

3 Experimental Results

In this section, we compare our algorithm with EI under different scenarios for different functions. We consider six well-known synthetic benchmark functions and two benchmarks derived from real-world applications, Hydrogen [3] and Fuel Cell [1]. We spend 20% of the budget on exploration and 80% on exploitation. The optimal allocation is to be considered as the future work.

Comparison to EI: In the first set of experiments, we would like to compare our algorithm with the best possible performance of EI. For each benchmark, we search over different values of the kernel width and find the one that optimizes EI’s performance. Fig. 1 is plotted using these optimal kernel widths and shows that the best performance of EI happens when we take only one random sample from a given budget. This performance is then used as the baseline for comparison in Table 2.

In light of the results of Fig. 1, we are also interested in whether our exploration algorithm can be used to improve the performance of EI. Table 2 summarizes “Regret” $= M - \max f(x_{\mathcal{C}})$ for different benchmarks estimated over 1000 random runs. Our algorithm (NBRS+NBIS) outperforms EI consistently except for the Shekel benchmark where EI and NBRS+EI have slightly better performances. We suspect that this is due to the fact that we have not optimized our kernel widths, where as the EI kernel width is optimized. We also note that NBRS+EI does not lead to any consistent improvement over EI. This is possibly due to the fact that EI does not take advantage of the reduced search space produced by NBRS during selection.

Exploration Analysis: In the second set of experiments, we would like to compare our exploration algorithm NBRS with random exploration when using NBIS for exploitation. As discussed previously, both random exploration and NBRS fail to produce better performance when used with EI. Thus, it is interesting to see whether they can help NBIS in terms of the overall regret, and if so which one is more effective. Figure 2 summarizes this result for all benchmarks. For a fixed budget $n_b$, we start with 1 explorative sample (either using NBRS or random) followed by $n_b - 1$ NBIS samples; next, we start with 2 explorative samples followed by $n_b - 2$ NBIS samples and so on. In each case, we average the regret over 1000 runs. The black line corresponds to the NBRS exploration and the green line corresponds to the random exploration. We will discuss each function in more details later, but in general, this result shows that our exploration algorithm is a) better than random exploration and b) necessary. To see why it is necessary, notice that the minimum regret on all curves is achieved for a non-zero number of NBRS samples. This means unlike EI, our exploitation algorithm benefits from NBRS.

NBRS starts from an initial point and explores the input space step by step. Imagine you are in a dark room with a torch in your hand and you want to explore the room. You start from an initial point and little by little walk through the space until you explore the whole space. This is exactly
Table 2: Comparison of the best results of EI, NBRS+EI and NBRS+NBIS. This result shows that our algorithm outperforms the other two counterparts significantly in most cases.

<table>
<thead>
<tr>
<th>Function</th>
<th>EI</th>
<th>NBRS+EI</th>
<th>NBRS+NBIS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cosines</td>
<td>.073 ± .016</td>
<td>.105 ± .029</td>
<td>.027 ± .009</td>
</tr>
<tr>
<td>Hydrogen</td>
<td>.090 ± .004</td>
<td>.114 ± .004</td>
<td>.047 ± .006</td>
</tr>
<tr>
<td>Hart(3)</td>
<td>.061 ± .006</td>
<td>.045 ± .003</td>
<td>.038 ± .003</td>
</tr>
<tr>
<td>Michalewicz</td>
<td>.517 ± .010</td>
<td>.501 ± .010</td>
<td>.455 ± .019</td>
</tr>
<tr>
<td>Fuel Cell</td>
<td>.136 ± .006</td>
<td>.1357 ± .004</td>
<td>.096 ± .004</td>
</tr>
</tbody>
</table>

Figure 2: Plot of regret versus the number of explorations for NBIS algorithm. This result shows that in most cases, our exploration is a) better than random, and b) necessary, since the regret achieves its minimum somewhere apart from zero.

how NBRS does the exploration. Roughly speaking, NBRS minimizes \( \mu_{x|O} + 1.5\sigma_{x|O} \) and hence, if a point is far from previous observations, i.e., \( \sigma_{x|O} \) is large, it is unlikely to be chosen. We see this effect in all functions, but most clearly in the Michalewicz benchmark. When the number of explorative samples is smaller than 10, the step-by-step explore procedure cannot explore the whole space and the exploitation can be trapped in local minima. For 10 – 15 explorative samples, NBRS can walk through the entire space fairly well and hence we get a minimum regret.

References