Recent Advances in Bayesian Optimization

Payton Howell and Nate Whybra

Mathematics of Machine Learning JC, July 21, 2025

UNIVERSITY *of* WASHINGTON

Outline

- Set-Up
- Acquisition Functions
- Combinatorial Optimization
- Expensive Constrained Optimization
- High Dimensional Optimization

Set-Up

We've been working on the problem

$$x^* = \arg\max_{x \in \mathcal{D}} f(x)$$

Where $\mathcal{D} \subseteq \mathbb{R}^n$ and $f(x) : \mathbb{R}^n \to \mathbb{R}$ is costly to compute.

Set-Up

We've been working on the problem

$$x^* = \arg\max_{x \in \mathcal{D}} f(x)$$

Where $\mathcal{D} \subseteq \mathbb{R}^n$ and $f(x) : \mathbb{R}^n \to \mathbb{R}$ is costly to compute.
After getting N observations, our GP of the function has mean and variance

$$X = \left[x_1, \cdots, x_N\right], \quad y \in \mathbb{R}^N$$

$$\begin{aligned} \mu(x) =& K(x, X)\left(K(X, X) + \sigma^2 I\right)^{-1} y \\ \sigma(x) =& K(x, x) - K(X, x)^T \left(K(X, X) + \sigma^2 I\right)^{-1} K(X, x) \end{aligned}$$

Acquisition Functions

We've seen

- Expected Improvement

$$EI(x) = (f^* - \mu(x))\Phi\left(\frac{\mu(x) - f^*}{\sigma(x)}\right) + \sigma(x)\phi\left(\frac{\mu(x) - f^*}{\sigma(x)}\right)$$

where Φ and ϕ represent the CDF and PDF of a unit normal distribution.

Acquisition Functions

Probability of Improvement

$$PI(x) = \Phi\left(\frac{\mu(x) - f^*}{\sigma(x)}\right) \equiv \mathbb{P}\Big(f(x) \geq f^*\Big)$$

- Like EI, but PI does not account for the magnitude of improvement, increasing chances of exploration.

Acquisition Functions

Probability of Improvement

$$PI(x) = \Phi\left(\frac{\mu(x) - f^*}{\sigma(x)}\right) \equiv \mathbb{P}\Big(f(x) \geq f^*\Big)$$

- Like EI, but PI does not account for the magnitude of improvement, increasing chances of exploration.

To see the equivalence above, consider the following

$$\mathbb{P}(f(x) > f^*) = 1 - \Phi\left(\frac{f^* - \mu(x)}{\sigma(x)}\right) = \Phi\left(-\frac{f^* - \mu(x)}{\sigma(x)}\right)$$

Acquisition Functions

Upper Confidence Bound (UCB)

$$UCB(x) = \mu(x) + \beta\sigma(x), \quad \beta > 0$$

- β tunes the exploration vs. exploitation trade-off.
- $\beta < 0$ would be Lower Confidence Bound

Combinatorial Optimization

We want to find h^* according to:

$$h^* = \arg\max_h f(h)$$
$$f : \mathcal{H} \to \mathbb{R}$$

Combinatorial Optimization

We want to find h^* according to:

$$h^* = \arg\max_h f(h)$$
$$f : \mathcal{H} \to \mathbb{R}$$

- $\mathcal{H} = [\mathcal{C}, \mathcal{X}]$ where $\mathcal{X}$ is continuous and $\mathcal{C}$ is discrete/categorical.
- may suffer from combinatorial explosion
- need to make adjustments for the discrete domain

CO | Latent Representation: One-Hot Encoding

Suppose $\mathcal{C} \subset \mathbb{Z}^n$

What if we just pretend $\mathcal{C} \subset \mathbb{R}^n$?

CO | Latent Representation: One-Hot Encoding

Suppose $\mathcal{C} \subset \mathbb{Z}^n$
What if we just pretend $\mathcal{C} \subset \mathbb{R}^n$?

- We're ignoring the nature of the categorical space.
- Encoding during AF evaluation adds computation
- By virtue of how BO works we may sample the same discrete point multiple times in the continuous space

CO | Latent Representation: One-Hot Encoding

Suppose $\mathcal{C} \subset \mathbb{Z}^n$
What if we just pretend $\mathcal{C} \subset \mathbb{R}^n$?

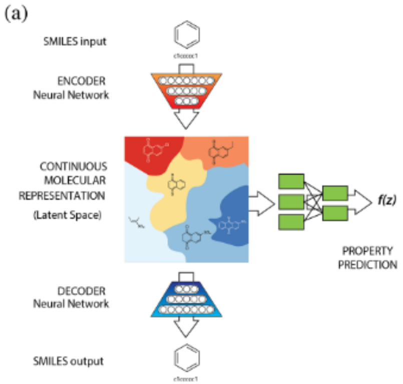
- We're ignoring the nature of the categorical space.
- Encoding during AF evaluation adds computation
- By virtue of how BO works we may sample the same discrete point multiple times in the continuous space

One - Hot encoding consists of taking a categorical variable, e.g. $h_j = \{$ red, green, blue$\}$, and changing it into a n-dimensional subspace of the reals.

$$h^j \in [0,1]^3, \qquad x_j = [0,0,1] \mapsto \text{blue}$$

CO | Latent Representation: VAEs

Change from a discrete space to a continuous one and then project solution onto the discrete space.
Most use VAEs (variational auto-encoders) to do this.[2]



CO | Latent Representation: VAEs

Allows us to do Bayesian Optimization but has pitfalls

- May not capture actual max on $\mathcal{C}$
- Only allows BO to use information from the VAE
- Decoding process may cause impossible solutions
- Stochasticity in VAEs can cause different final outputs from the decoder

CO | Combinatorial Graphs - COMBO

We model everything in a combinatorial graph, $\mathcal{G}$

$$\mathcal{G} = (\mathcal{V}, \mathcal{E})$$

Edges exists between nodes where there is a single ordered change in a category.

- Distance measure. Hamming Distance
- Degree Matrix, $D_{i,i} = \delta(v_i)$
- Adjacency Matrix $A_{i,j} = |\{e_{s,t} \in \mathcal{E} : e_{i,j}\}|$
- Laplacian $\mathcal{L}(\mathcal{G}) = D_{\mathcal{G}} - A_{\mathcal{G}}$

CO | Combinatorial Graphs - COMBO

We'll need a kernel to compute the mean and variance for our GP.

$$K(\mathcal{V}, \mathcal{V}) = U \exp(-\beta \Lambda) U^T$$

where $\Lambda = \mathrm{diag}(\lambda_1, \cdots, \lambda_n)$ and λ, U are the graph Fourier frequencies and bases (eigenvals and vects of $\mathcal{L}(\mathcal{G})$).

Can define subgraphs $\mathcal{G} = \cup_i \mathcal{G}_i$ and add scaling of the laplacians of these subgraphs.

$$K(\mathcal{V}) = \bigotimes_i \exp(\beta_i \mathcal{L}(\mathcal{G}_i)),\ \beta_i \geq 0$$

CO | Combinatorial Graphs - COMBO

Some things to note:

- The eigendecomposition has complexity $\mathcal{O}\Big(\sum_i |\mathcal{V}_i|^3\Big)$
- COMBO takes advantage of the categorical nature of the space
- COMBO assumes all variable are categorical, is not able to deal with mixed search spaces

Expensive Constrained Optimization | ECOP

Here we have a minimization problem

$$\begin{aligned} \min_x \; & f(x) = [f_1(x), \cdots, f_m(x)] \\ \text{s.t.} \quad & c_j(x) \leq a_j, \qquad j = 1, \cdots, q \\ & x \in \mathcal{X} \end{aligned}$$

- Both f and constraint equations, $c_j(x)$, are computationally expensive
- $x = [x_1, \cdots, x_d]^T$ is the d-dimensional decision vector
- Decision space, $\mathcal{X}$
- $m > 1$ when dealing with multi-objective problems

ECOP | Augmented Lagrangian Relaxation

We convert the problem into an unconstrained problem

$$L_A(x; \lambda, \rho) = f(x) + \lambda^T c(x) + \frac{1}{2\rho} \sum_{j=1}^{q} \max(0, c_j(x))^2$$

$\rho > 0$ penalty param $\lambda \in \mathbb{R}_+^q$ Lagrange Multiplier.

- Redefine $c_j(x)$ such that $a_j = 0, \forall j = 1, \cdots, q$
- Run regular BO with L_A instead of f, but we still observe $y = f(x) + \eta$
- Replace f^* in AF with best value of L_A.

ECOP | Probability of Feasibility

Instead of evaluating EI, we use EIC (EI with Constraints)

$$cEI(x) = EI(x)\prod_{j=1}^{q} Pr(c_j(x) \leq a_j)$$

We approximate the $c_j(\cdot)$ by GPs as well as f to make the probability calculate feasible.

ECOP | Probability of Feasibility

Instead of evaluating EI, we use EIC (EI with Constraints)

$$cEI(x) = EI(x) \prod_{j=1}^{q} Pr(c_j(x) \leq a_j)$$

We approximate the $c_j(\cdot)$ by GPs as well as f to make the probability calculate feasible.

- Can be brittle for highly constrained problems

ECOP | Multi-step look ahead Methods

Instead of the usual greedy AFs we can try to maximize longterm rewards.
We use the current GPs to simulate what the next step's selection step would be and from there model

$$g_{k+1} \sim \mathcal{N}\left(\bar{\mu}(x_{k+1}; g_k), \bar{\sigma}(x_{k+1}; g_k)\right)$$

We can approximate multiple steps ahead by repeating this

$$w_k = \left[f_k^f, c_1^k, \cdots, c_q^k\right]$$

$$AF(x_{k+s} w_{k+s})$$

ECOP | Multi-step look ahead Methods

Instead of the usual greedy AFs we can try to maximize longterm rewards.
We use the current GPs to simulate what the next step's selection step would be and from there model

$$g_{k+1} \sim \mathcal{N}\left(\bar{\mu}(x_{k+1}; g_k), \bar{\sigma}(x_{k+1}; g_k)\right)$$

We can approximate multiple steps ahead by repeating this

$$w_k = \left[f_k^f, c_1^k, \cdots, c_q^k\right]$$

$$AF(x_{k+s} w_{k+s})$$

Note: This introduces more and more uncertainty for each step forward you do!

High Dimensional BO (HDBO)

BO is attempting to approximate a function $f : \mathbb{R}^n \to \mathbb{R}$. However, in practice, the "straightforward" approach to BO can work well for input dimensions $n \leq 20$ [Page 6], but typically doesn't perform well when $n > 20$.

Several issues for higher dimensions:

1. Number of hyper-parameters often grows with input dimension
2. Inherent difficulty in high dimensional space (curse of dimensionality). The volume of search space grows exponentially, and the number of function evaluations needed to get a good approximation grows too large. (Remember, f is costly to evaluate so we want to sample it the least amount of times possible...)
3. AFs become harder to optimize as the dimensions increase (Why?)

HDBO | Addressing the Difficulty

Assumption 1: The function $f : \mathbb{R}^n \to \mathbb{R}$ can be represented well by another function $\hat{f} : \mathbb{R}^d \to \mathbb{R}$ where $d << n$.

- Although our data is really n dimensional, only d dimensions are needed to capture f well. (Dimensionality reduction)
- Leads to "variable selection methods" and "embedding methods"

Assumption 2: The function f has additive substructure, that is:

$$f(x) = f^{(1)}(x^{(1)}) + f^{(2)}(x^{(2)}) + \cdots + f^{M}(x^{(M)})$$

Where each $x^{(j)}$ are disjoint subsets of the n input variables.

- There are methods that exploit this structure to tackle HDBO, but I haven't dived into them yet, so I figured I would just mention this.

HDBO | Variable Selection

Idea: If only d input variables matter, which ones are they?

1. "Evaluate the relative importance of each variable with respect to some quantity of interest..." [Citation]
2. We will describe a method discussed in [Citation]

HDBO | Variable Selection

Algorithm (Find $A \subset [n]$ active variables):

1. Use Hierarchical Diagonal Sampling (HDS) to make a tree as follows. First, start with the n input variables $\{x_0, \cdots, x_{n-1}\}$ that all begin labeled as **underdetermined**. This grouping represents the root of the tree.
2. Use some method M (more on that later) to infer if any of the variables in the root node "matter". If any of the variables "matter", label the node as **active**, then split the node into 2 relatively equally sized child-nodes. If none of the variables "matter" label the node **inactive** (if the root node is inactive, we are done).
3. Using M again, check if the child nodes are active, if so split them again. If any of the nodes are inactive, ignore it.
4. Continue until the nodes are split into singletons or until no more nodes are active. The signletons labeled active at the end of the day will be our subset A of active variables.

HDBO | Variable Selection

So what is M? M is some process for determining if a node in the tree is active or not. We will describe something called the GP Sequential Likelihood Ratio Test (GPT), but first some set-up. Assume the input data is normalized to the unit hypercube $Q = [-1, 1]^n$ in $\mathbb{R}^n$, and denote a node in the tree as I (I contains the indices of the selected variables). Fix a random vector $v \in Q$, then an auxiliary function $X_I : [-1, 1] \to Q$ is defined as:

$$X_{I,i}(z) = \begin{cases} z & \text{if } i \in I \\ v_i & \text{else} \end{cases}$$

Then a $1D$ projection function $f_I : [-1, 1] \to \mathbb{R}$ is defined:

$$f_I(z) = f(X_I(z))$$

HDBO | Variable Selection

For this variable selection scheme, the kernel function for our GP prior (for BO) is assumed to take the form, with A being the indices of our unknown active variables:

$$K(x,x') = \sigma^2 \exp\left(-\sum_{i \in A} \frac{(x_i - x_i')^2}{b^2}\right)$$

As $f \sim GP(0,K)$ prior to any function evaluations, the function $f_I \sim GP(0,K_I)$, where $K_1(x,x') = K(x_I(x), x_I(x'))$, so each function f_I (after a small calculation) can be considered a sample of a $1D$ GP with corresponding kernel function:

$$K_I(x,x') = \sigma^2 \exp\left(-a_I \frac{(x-x')^2}{b^2}\right)$$

Where $x, x' \in [-1,1]$ and $a_I = |A \cap I| =$ (the number of active variables in node I). So to figure out if I contains any active variables, we can test various kernels on observations of f_I, and roughly deduce the number of active variables from the kernel.

HDBO | Variable Selection | GPT

Gaussian Process Sequential Likelihood Ratio Test: Focusing on a single node l, given past input/output pairs $x_{1:t-1}, y_{1:t-1}$, and the newest input value x_t, we can consider for the node l (the y values here come from f_l, not f, also the subscript a is to reflect that the kernel function depends on $a := a_l$):

$$y_{l_t} | x_t, x_{1:t-1}, y_{1:t-1} \sim \mathcal{N}(m_a^t(x_t), var_a^t(x_t))$$

This is exactly the same as normal Bayesian optimization. So the posterior mean and standard deviation of the above distribution are given by [CITATION] (as an aside I think we typically look at the variance equation, but I'm just gonna put what they have in the paper).

$$m_a^t(x) = K_a(X, x)^T (K_a + \sigma^2 I)^{-1} y_{1:t-1}$$

$$\sigma_a^t(x) = \sigma^2 + K_a(x, x) - K_a(X, x)^T (K_a + \sigma^2 I)^{-1} K_a(X, x)$$

HDBO | Variable Selection | GPT

Hypothesis Testing: Say H_0 is the hypothesis that I contains no active variables (the kernel function K_I has $a_I = 0$), and H_1 is the hypothesis that I contains at least one active variable (the kernel function K_I has $a_I \geq 1$). Then the log-likelihood ratio for sample t is:

$$LLR_t(y_t) = \log\left(\frac{\text{Likelihood under } H_1}{\text{Liklihood under } H_0}\right) = \log\left(\frac{\mathcal{N}(y_t | m_1^t(x_t), var_1^t(x_t))}{\mathcal{N}(y_t | m_0^t(x_t), var_0^t(x_t))}\right)$$

Then the LLR for all observed samples so far is:

$$LLR_{1:t} = \sum_{i=1}^{t} LLR_i(y_i)$$

HDBO | Variable Selection | GPT

If the quotient inside the logarithm is greater than 1, then the logarithm will be greater than 0, and the LRR will be positive for that sample, so a positve LRR means that the hypothesis H_1 is supported, and otherwise if the LRR is negative. To that effect, you can choose 2 thresholds T_1, T_2, where if $LRR > T_1$ that means you label I as active, and if $LRR < T_2$ you label I inactive, and if $T_2 < LRR < L_1$, then you sample another input and update the posterior again.

HDBO | Random Embeddings

Here we present another approach to HDBO, but this time before presenting the algorithm, we will first talk about math. I initially spoke about how a function $f : \mathbb{R}^n \to \mathbb{R}$ might be well approximated by a function from $\hat{f} : \mathbb{R}^d \to \mathbb{R}$, we will formalize this language here. From [Citation], we have the following definition:

Definition 1: A function $f : \mathbb{R}^n \to \mathbb{R}$ is said to have **effective dimensionality** d, with $d \leq N$ if:

- There exists a linear subspace $V \subset \mathbb{R}^n$ of dimension d such that for all $x \in V$ and $x_\perp \in V^\perp$, we have $f(x + x_\perp) = f(x)$
- d is the smallest integer with the above property
- Call V the **effective subspace** of f and $V^\perp$ the **constant subspace** of f

HDBO | Random Embeddings

That is, the function does not change along any coordinates $x_\perp$. The next Therorem is about random embeddings:

Theorem 1: Assume we are given a function $f : \mathbb{R}^n \to \mathbb{R}$ with effective dimensionality d, and also a random matrix $A \in \mathbb{R}^{n \times d}$ where each $a_{i,j} \sim \mathcal{N}(0, 1)$, then for any $x \in \mathbb{R}^n$, the probability there exists a $y \in \mathbb{R}^d$ such that $f(x) = f(Ay)$ is 1, that is:

$$P(\exists y \in \mathbb{R}^d : f(x) = f(Ay)) = 1$$

HDBO | Random Embeddings

Proof. Since f has effective dimensionality d, there exists an effective subspace $V \subset \mathbb{R}^d$ such that $rank(V) = d$. Furthermore, for any $x \in \mathbb{R}^n$, we can write $x = v + v_\perp$, where $v \in V$ and $v_\perp \in V^\perp$, so that $f(x) = f(v + v_\perp) = f(v)$. Therefore, without loss of generality, it suffices to show that for all $v \in V$, there exists a $y \in \mathbb{R}^d$ such that $f(v) = f(Ay)$.

Now let $B \in \mathbb{R}^{n \times d}$ be a matrix whose columns form an orthonormal basis for V. Then for each $v \in V$, there exists some $c \in \mathbb{R}^d$ such that $v = Bc$. Now for the sake of argument, just pretend for now that the matrix $B^T A \in \mathbb{R}^{d \times d}$ has rank d. Then there exists a $y \in \mathbb{R}^d$ such that $(B^T A)y = c$. The orthogonal projection of Ay in V is $BB^T Ay = Bc = v$.

Therefore, $Ay = v + v_\perp$ for some $v_\perp \in V^\perp$ since v is the projection of Ay in V. Thus, $f(Ay) = f(v + v_\perp) = f(v)$ as desired. To finish the proof, we must show that $P(B^T A \text{ has rank d}) = 1$, but for this part of the proof, see the Appendix in [Citation].

HDBO | Random Embeddings

This is really cool, but how does it help us with HDBO? The above theorem says that given any point $x \in \mathbb{R}^n$ and any random matrix $A \in \mathbb{R}^{n \times d}$, there is almost surely a point $y \in \mathbb{R}^d$ (the lower dimensional space) such that $f(x) = f(Ay)$. This means that for any optimizer $x^* \in \mathbb{R}^n$, there is a point $y^* \in \mathbb{R}^d$ with $f(x^*) = f(Ay^*)$. The utility in this is that instead of optimizing f in $\mathbb{R}^n$, we can instead choose to search over $\mathbb{R}^d$!

In practical applications, the search space for f is often restricted to a box X in $\mathbb{R}^n$, therefore it is possible that some point $y \in \mathbb{R}^d$ is selected such that Ay is outside of X. This is no good, and to protect against that, Ay can be projected back down onto the box before being evaluated by f. That is $g(y) = f(p_X(Ay))$ where $p_X(y) = argmin_{z \in X} ||z - y||_2$ (the point closest to the box). The next theorem tells us roughly where to search for y given a bounding box.

HDBO | Random Embeddings

Theorem 2: Suppose we want to optimize a function $f : \mathbb{R}^n \to \mathbb{R}$ with effective dimension d subject to the box constraint $X \subset \mathbb{R}^n$ where X is centered at 0. Suppose further that the effective subspace V of f is such that V is dimension d, and let $x^* \in V \cap X$ be an optimizer of f inside V. If $A \in \mathbb{R}^{n \times d}$ is a random matrix with each $a_{ij} \sim \mathcal{N}(0, 1)$, then there exists an optimizer $y^* \in \mathbb{R}^d$ such that $f(Ay^*) = f(x^*)$ and $||y^*||_2 \leq (\sqrt{d}/\epsilon)||x^*||_2$ with probability at least $1 - \epsilon$.

Suppose the box constraint is $X = [-1, 1]^D$ (which is always possible with scaling), then with probability $1 - \epsilon$, $||y||_2 \leq (\sqrt{d}/\epsilon)\sqrt{d} = d/\epsilon$. This tells you the size of the ball around the origin you should search for $y \in \mathbb{R}^d$ depending on how strong the guarantee you would like.

HDBO | Random Embeddings | REMBO

For formality, the most simplest form of BO with Random Embeddings (REMBO), is summarized by the following algorithm:

Algorithm (REMBO)

1. Generate a random matrix $A \in \mathbb{R}^{n \times d}$ with $a_{ij} \sim \mathcal{N}(0, 1)$.
2. Choose the bounded ball set $Y(\epsilon, d) \subset \mathbb{R}^d$ corresponding to Theorem 2.
3. Define the function $g : \mathbb{R}^d \to \mathbb{R}$ such that $g(y) = f(p_X(Ay))$.
4. Perform Bayesian optimization as normal on g.

Bibliography I

Bo Chen, Rui M. Castro, and Andreas Krause.

Joint optimization and variable selection of high-dimensional gaussian processes.

In *Proceedings of the 29th International Conference on Machine Learning (ICML)*, pages 137–144, 2012.

Gómez-Bombarelli Rafael et al.

Automatic chemical design using a data-driven continuous representation of molecules.

ACS Central Science, 4(2):268–276, 2018.

PMID: 29532027.

Bibliography II

Robert B. Gramacy, Genetha A. Gray, Sebastien Le Digabel, Herbert K. H. Lee, Pritam Ranjan, Garth Wells, and Stefan M. Wild.

Modeling an augmented lagrangian for blackbox constrained optimization, 2015.

Remi Lam and Karen Willcox.

Lookahead bayesian optimization with inequality constraints.

In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

Bibliography III

Xilu Wang, Yaochu Jin, Sebastian Schmitt, and Markus Olhofer.
Recent advances in bayesian optimization.
ACM Comput. Surv., 55(13s), July 2023.

Ziyu Wang, Masrour Zoghi, Frank Hutter, David Matheson, and Nando de Freitas.
Bayesian optimization in a billion dimensions via random embeddings.
Journal of Artificial Intelligence Research, 55:361–387, 2016.