Bayesian Optimization for Wide Landscapes (BOWL)

CIS 7000 / GRASP Lab / May 16, 2025

Milad Mesbahi
Robotics Department
University of Pennsylvania
mesbahi@seas.upenn.edu

I. Problem Overview

Bayesian Optimization (BayesOpt) is a global optimization strategy we can use to evaluate black-box functions that are expensive to sample and lack analytical gradients. It is particularly effective in scenarios where each function evaluation is costly, and thus the number of evaluations must be minimized.

We can further understand this mathematically by highlighting its main parts. Let $f : \mathcal{X} \to \mathbb{R}$ be the objective function we wish to maximize, where $\mathcal{X} \subset \mathbb{R}^d$ is a compact, bounded domain. We assume that f is unknown and expensive to evaluate, and that we can only observe noisy measurements of the form:

$$y = f(\mathbf{x}) + \varepsilon,$$

where $\varepsilon \sim \mathcal{N}(0, \sigma_n^2)$ represents Gaussian observation noise.

The goal is to find the global maximizer [3]:

$$\mathbf{x}^* \in \arg\max_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x}), \quad f^* = f(\mathbf{x}^*).$$

Since we don't have access to the true function value, BayesOpt constructs a probabilistic model $p(f \mid \mathcal{D})$ of the objective function using observed data $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$. A common choice is a Gaussian Process (GP) prior:

$$f \sim \mathcal{GP}(\mu(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')),$$

where $\mu(\mathbf{x})$ is the mean function and $k(\mathbf{x}, \mathbf{x}')$ is a positive-definite kernel function.
At each iteration, the model is updated with new data, and an acquisition function $\alpha(\mathbf{x}; \mathcal{D})$ is maximized to determine the next query point:

$$\mathbf{x}_{n+1} = \arg\max_{\mathbf{x}\in\mathcal{X}} \alpha(\mathbf{x}; \mathcal{D}).$$

The goal of these acquisition functions is really to balance *exploration* (sampling uncertain regions) and *exploitation* (sampling near high posterior mean) [3].

A. Two Bottlenecks in Modern BO Practice

Despite its success, two issues vanilla BO struggles with is when:

1) **Rugged landscapes.** In high dimensions the GP posterior becomes locally ill-conditioned; acquisition functions flatten out, leading to *stalling* in sharp or multi-modal regions.
2) **Poor generalisation of found optima.** Classic BO does not differentiate *wide, flat* basins (which generalise) from narrow spikes. The optimiser may converge to brittle solutions that deteriorate when evaluated under realistic perturbations of $\mathbf{x}$.

With these two core problems in mind, the goal of this paper was to attempt a new methodology for inducing current BO methods to search and explore these black-box spaces and return robust solutions. Below we will analyze how we measure and find solutions that are "robust" and the current methods we will be taking inspiration from.

II. Related Works

A. Entropy-SGD: Biasing Gradient Descent into Wide Valleys

The first algorithm we draw inspiration from, and is frankly the cornerstone of our custom algorithm *BOWL*, is the Entropy-SGD algorithm [2]. This optimization framework is designed for training deep neural networks by explicitly biasing search toward *wide valleys* in the loss landscape, and classically has access to the true function form and thus can do so with available gradient information. These regions of parameter space are empirically linked to better generalization, as they are more "robust" to perturbations in data and model parameters.

Empirical analysis of the Hessian at local minima in deep networks has shown that well-generalizing solutions often lie in flat regions characterized by many near-zero eigenvalues. In contrast, sharp minima, although possibly lower in training loss, tend to overfit and generalize poorly.

Entropy-SGD aims to modify the objective function to favor wider minima by introducing a new quantity called *local entropy*, which captures both the depth and the flatness of a valley around a candidate point.

Local Entropy Objective: Let $f : \mathbb{R}^n \to \mathbb{R}$ be the original loss function. For a current parameter vector $\mathbf{x}$, the local entropy is defined as:

$$F(\mathbf{x}, \gamma) = \log \int_{\mathbf{x}'} \exp\left(-f(\mathbf{x}') - \frac{\gamma}{2}\|\mathbf{x} - \mathbf{x}'\|^2\right) d\mathbf{x}',$$

where $\gamma > 0$ is a hyperparameter controlling the *scope*, i.e., how localized the entropy estimation is.

This is a form of a smoothed objective that integrates the loss over a local neighborhood, weighting nearby parameters more strongly.

We can then calculate the gradient of the local entropy objective by:

$$\nabla F(\mathbf{x}, \gamma) = \gamma \left(\langle \mathbf{x}' \rangle - \mathbf{x} \right),$$

where the expectation $\langle \mathbf{x}' \rangle$ is taken over the local Gibbs distribution:

$$P(\mathbf{x}'|\mathbf{x}) \propto \exp\left(-f(\mathbf{x}') - \frac{\gamma}{2} \|\mathbf{x} - \mathbf{x}'\|^2 \right).$$

Since this expectation is intractable to compute exactly in high dimensions, the algorithm approximates it using *Stochastic Gradient Langevin Dynamics (SGLD)*, a Markov Chain Monte Carlo method that introduces noise into gradient updates to sample from the posterior.

Putting this all together, we can describe the Entropy-SGD method effectively as a two-loop algorithm:

- The **inner loop** runs L steps of SGLD to estimate $\langle \mathbf{x}' \rangle$ given the current parameter vector $\mathbf{x}$.
- The **outer loop** performs a standard SGD update using the entropy-adjusted gradient:

$$\mathbf{x} \leftarrow \mathbf{x} - \eta \cdot \gamma(\mathbf{x} - \langle \mathbf{x}' \rangle),$$

  where η is the learning rate.

This effectively pulls the weights toward regions where a large volume of nearby parameters yield low loss — i.e., flat valleys. This is the cornerstone two-loop optimization pattern we will use in BOWL to bias our gradient toward robust solutions.

Lastly, to balance exploration and exploitation during training, Entropy-SGD employs a *scoping schedule* that gradually increases γ. A typical schedule is:

$$\gamma(t) = \gamma_0 (1 + \gamma_1)^t,$$

which narrows the entropy region over time, allowing the optimizer to zoom into the most robust valleys.

The main issue you may notice with bringing this to BayesOpt is that it relies on access to gradient information in order to run the SGLD loop. This is where the second method we will be drawing from comes into play.

B. Maximum Probability of Descent (MPD)

Given a GP belief over the black-box function $f(\mathbf{x})$, we can form a belief over its gradient at a location $\mathbf{x}$:

$$\nabla f(\mathbf{x}) \sim \mathcal{N}(\boldsymbol{\mu}_\mathbf{x}, \boldsymbol{\Sigma}_\mathbf{x}).$$

For any unit direction vector $\mathbf{v}$, the directional derivative $\nabla_\mathbf{v} f(\mathbf{x})$ is a scalar random variable:

$$\nabla_\mathbf{v} f(\mathbf{x}) = \mathbf{v}^\top \nabla f(\mathbf{x}) \sim \mathcal{N}(\mathbf{v}^\top \boldsymbol{\mu}_\mathbf{x}, \mathbf{v}^\top \boldsymbol{\Sigma}_\mathbf{x} \mathbf{v}).$$

The probability that a step in direction $\mathbf{v}$ leads to descent is:

$$\Pr(\nabla_\mathbf{v} f(\mathbf{x}) < 0) = \Phi\left(\frac{-\mathbf{v}^\top \boldsymbol{\mu}_\mathbf{x}}{\sqrt{\mathbf{v}^\top \boldsymbol{\Sigma}_\mathbf{x} \mathbf{v}}} \right),$$

where $\Phi(\cdot)$ is the standard normal CDF.

Now, a key insight of MPD is that the direction maximizing the descent probability is not necessarily $-\boldsymbol{\mu}_\mathbf{x}$ [5]. Instead, the optimal direction is derived from solving:

$$\mathbf{v}^* = \arg\max_{\|\mathbf{v}\|=1} \Phi\left(\frac{-\mathbf{v}^\top \boldsymbol{\mu}_\mathbf{x}}{\sqrt{\mathbf{v}^\top \boldsymbol{\Sigma}_\mathbf{x} \mathbf{v}}} \right).$$

This is equivalent to minimizing:

$$\frac{\mathbf{v}^\top \boldsymbol{\mu}_\mathbf{x}}{\sqrt{\mathbf{v}^\top \boldsymbol{\Sigma}_\mathbf{x} \mathbf{v}}},$$

which is solved analytically by:

$$\mathbf{v}^* \propto -\boldsymbol{\Sigma}_\mathbf{x}^{-1} \boldsymbol{\mu}_\mathbf{x}.$$

The corresponding descent probability is then:

$$\Pr(\nabla_{\mathbf{v}^*} f(\mathbf{x}) < 0) = \Phi\left(\sqrt{\boldsymbol{\mu}_\mathbf{x}^\top \boldsymbol{\Sigma}_\mathbf{x}^{-1} \boldsymbol{\mu}_\mathbf{x}} \right).$$

Again, we can put this all together by describing that at each iteration, MPD alternates between:

i) **Learning Phase:** Collecting observations to improve the posterior belief over $\nabla f(\mathbf{x})$ such that the expected probability of descent increases.
ii) **Movement Phase:** Iteratively stepping in direction $\mathbf{v}^*$ as long as $\Pr(\nabla_{\mathbf{v}^*} f(\mathbf{x}) < 0) > p^*$.

To then efficiently choose where to sample next, MPD defines an acquisition function that approximates the expected maximum descent probability after gathering data at a batch of query points Z:

$$\alpha(Z) = \mathbb{E}\left[\boldsymbol{\mu}_{\mathbf{x}|Z}^\top \boldsymbol{\Sigma}_{\mathbf{x}|Z}^{-1} \boldsymbol{\mu}_{\mathbf{x}|Z} \right],$$

which is tractable in closed form, given Gaussian updates to the GP posterior.

So, how do we combine these to get the best of both worlds? *BOWL* is what we propose.

PROPOSED METHOD - BOWL

Our proposed algorithm, BOWL (Bayesian Optimization for Wide Landscapes), attempts to be a novel hybrid that leverages the strengths of Entropy-SGD and MPD, adapted into the context of Bayesian Optimization. Its core motivation is to construct a principled descent mechanism over black-box functions that achieves robust convergence, with an emphasis on identifying *wide, flat optima*.

BOWL solves the problem of optimizing an expensive black-box function $f : \mathcal{X} \to \mathbb{R}$ by replacing true gradient information (unavailable in black-box settings) with posterior gradient estimates from a Gaussian Process surrogate. These estimates are then used to drive a modified Entropy-SGD process that behaves as if it were descending the true landscape, but with uncertainty-aware guidance.

In essence, BOWL reframes Entropy-SGD as a stochastic policy in parameter space, but guided by a Bayesian model rather than by actual gradients. Simultaneously, the MPD formulation allows us to estimate the *most robust direction of*

descent under GP uncertainty, providing a natural plug-in for Entropy-SGD's inner dynamics.

We do this by using the MPD GP (*DerivativeExactGPSEModel*), which is a modification of the standard exact GP from BoTorch and GPyTorch. This model does not only provides posterior estimates for function values, but is also capable of computing the posterior distribution over gradients:

$$\nabla f(\mathbf{x}) \sim \mathcal{N}(\boldsymbol{\mu}_{\mathbf{x}}, \boldsymbol{\Sigma}_{\mathbf{x}}).$$

This posterior is updated online with new function evaluations, and supports:

- Exact computation of GP gradient mean and variance;
- Analytical expressions for the derivative and second derivative of the RBF kernel;
- Streaming updates, which enable flexible asynchronous sampling.

We modify the standard Entropy-SGD loop by replacing true gradients with those derived from the GP posterior described above. Each Entropy-SGD update proceeds in two nested loops:

- In the **inner loop**, we sample local perturbations around the current point $\mathbf{x}$, following Langevin dynamics. However, the gradient signal used in this loop is obtained via our MPD oracle:

  $$\mathbf{v}^* \propto -\boldsymbol{\Sigma}_{\mathbf{x}}^{-1}\boldsymbol{\mu}_{\mathbf{x}},$$

  which captures the direction with the highest probability of decreasing the objective, considering both the mean gradient and its uncertainty.
- In the **outer loop**, we move the iterate toward the weighted average of Langevin samples μ, simulating the effect of minimizing the local entropy objective:

  $$\mathbf{x}_{t+1} = \mathbf{x}_t - \eta \cdot \gamma(\mathbf{x}_t - \mu).$$

Noise is injected during the inner loop via Gaussian perturbations to ensure that wide valleys are explored. The scoping parameter γ is annealed over time, gradually narrowing the focus of the search. Importantly, this stochastic dynamic is governed by the GP's beliefs about the gradient — no true gradients are ever computed.

Hybrid Acquisition Policy

To retain global exploration capabilities, BOWL augments the EntropySGD inner descent procedure with a simplified Bayesian Optimization acquisition mechanism. Every k iterations, we:

1) Compute a custom acquisition function (Downhill Quadratic from the MPD Github) based on the current GP posterior;
2) Optimize the acquisition function using multi-start sampling over the bounds;
3) Evaluate the objective at the acquired point and update the GP with this new observation;

Algorithm 1: BOWL

Input: Black-box objective f, domain bounds $\mathcal{X} \subset \mathbb{R}^d$, GP hyperparameters, Langevin loop length L, step size η, entropy decay $\gamma(t)$
Output: Best solution $\mathbf{x}^*$ found

Initialize:
Draw n_0 initial Sobol samples $\{\mathbf{x}_i\} \sim \text{Sobol}(\mathcal{X})$
Evaluate $f(\mathbf{x}_i)$ and fit initial GP surrogate model
Select best point $\mathbf{x}_0 \leftarrow \arg\max f(\mathbf{x}_i)$
for $t = 1, \ldots, T$ **do**
  Entropy-SGD Descent Step:
  for $\ell = 1, \ldots, L$ **do**
    Estimate GP gradient $\nabla f(\mathbf{x}_t) \sim \mathcal{N}(\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t)$
    Compute MPD direction: $\mathbf{v}_t \propto -\boldsymbol{\Sigma}_t^{-1}\boldsymbol{\mu}_t$
    Inject Gaussian noise and perform Langevin update
  Outer Update:
  Compute local entropy mean $\mu_t \leftarrow \mathbb{E}[\mathbf{x}']$ over Langevin samples
  Update current point: $\mathbf{x}_{t+1} \leftarrow \mathbf{x}_t - \eta \cdot \gamma(t) \cdot (\mathbf{x}_t - \mu_t)$
  Evaluation and Tracking:
  Evaluate $f(\mathbf{x}_{t+1})$, update GP with new observation
  Track best $f^* \leftarrow \max(f^*, f(\mathbf{x}_{t+1}))$
  if $t \mod k = 0$ **then**
    Acquisition Jump:
    Optimize MPD-inspired acquisition function $\alpha(\mathbf{x})$
    if *new point improves objective:* **then**
      Jump to new point: $\mathbf{x}_{t+1} \leftarrow \arg\max \alpha(\mathbf{x})$
  if *refitting condition met* **then**
    Refit GP hyperparameters using MLL optimization
return best seen point $\mathbf{x}^*$

4) Optionally jump to the acquired point if it significantly outperforms the current iterate.

This acquisition loop ensures that BOWL does not overcommit to local descent early on, and maintains a degree of exploration over the global search space.

Experimental Results

While the proposed BOWL framework has not yet achieved stable performance on high-dimensional black-box functions, early results on classic 2D and 3D synthetic benchmarks suggest its strong potential in low-dimensional settings. We evaluate BOWL on three commonly used global optimization functions: **Levy**, **Branin**, and **Hartmann 3D**. These are widely accepted as standard test cases for evaluating exploration-exploitation performance in Bayesian optimization.

Hartmann-3D Benchmark

To evaluate the practical effectiveness of our BOWL framework, we test it on the well-established **Hartmann-3D** function. This benchmark is a smooth, three-dimensional, multi-modal function defined on the unit cube $[0,1]^3$.

The Hartmann-3D function is given by [4]:

$$f(\mathbf{x}) = -\sum_{i=1}^{4} \alpha_i \exp\left(-\sum_{j=1}^{3} A_{ij}(x_j - P_{ij})^2\right),$$

where:

$$\alpha = [1.0, 1.2, 3.0, 3.2]^\top,$$

$$A = \begin{bmatrix} 3.0 & 10 & 30 \\ 0.1 & 10 & 35 \\ 3.0 & 10 & 30 \\ 0.1 & 10 & 35 \end{bmatrix}, \quad P = 10^{-4} \cdot \begin{bmatrix} 3689 & 1170 & 2673 \\ 4699 & 4387 & 7470 \\ 1091 & 8732 & 5547 \\ 381 & 5743 & 8828 \end{bmatrix}.$$

The global maximum of this function is approximately $f^* \approx 3.86278$, occurring near the point $\mathbf{x}^* = [0.1146, 0.5556, 0.8525]$. Note that the function is typically posed as a minimization problem, but in this experiment we reformulate it for maximization to be consistent with the goal of maximizing $f(\mathbf{x})$ to make sure it pairs with how BOWL is setup.

Figure 2 shows the convergence of BOWL on the Hartmann-3D benchmark. The y-axis reports the function value $f(\mathbf{x})$ obtained at each outer iteration, while the x-axis denotes the progression of optimization.

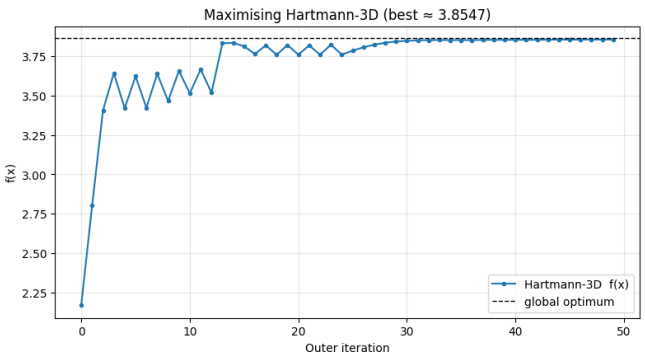

Fig. 1: Convergence of BOWL on the Hartmann-3D function. The dashed line indicates the global maximum $f^* \approx 3.8547$.

Levy Function

The **Levy function** is a highly multimodal, non-convex test function commonly used in global optimization benchmarks. The Levy function is designed to be deceptive: it contains many local minima and steep ridges, making it difficult for local optimizers to reach the global optimum without getting trapped [4].

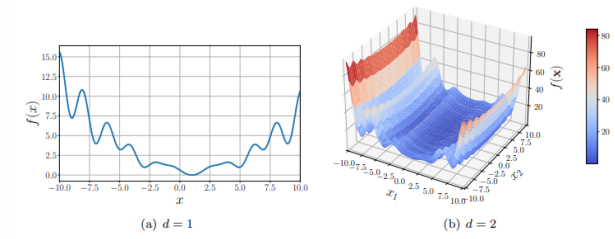

Fig. 2: The Levy Function

For a d-dimensional input $\mathbf{x} = [x_1, \ldots, x_d] \in [-10, 10]^d$, the Levy function is defined as:

$$f(\mathbf{x}) = \sin^2\left(\frac{\pi(x_1+3)}{4}\right) + \sum_{i=1}^{d-1}\left(\frac{x_i-1}{4}\right)^2\left(1+10\sin^2\left(\frac{\pi(x_i+3)}{4}+1\right)\right) + \left(\frac{x_d-1}{4}\right)^2\left(1+\sin^2\left(\frac{\pi(x_d+3)}{2}\right)\right).$$

The global minimum is known and occurs at:

$$\mathbf{x}^* = [1, \ldots, 1], \quad \text{with} \quad f(\mathbf{x}^*) = 0.$$

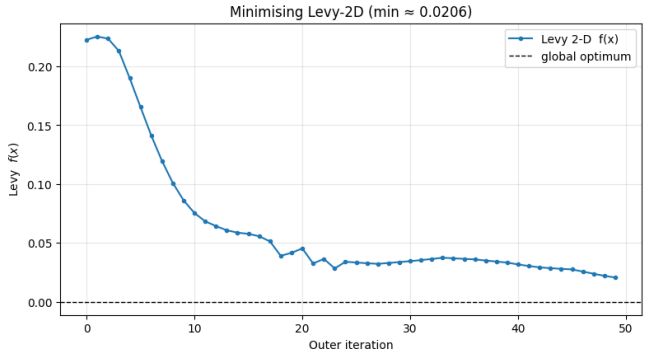

Fig. 3: Convergence of BOWL on the Levy function. Dashed line shows the global minimum $f^* \approx 0$.

Branin Function

Lastly, the **Branin function** is a well-known two-dimensional optimization benchmark that presents a multimodal landscape with three global optima. It is often used to evaluate the balance between exploration and exploitation in global optimization algorithms.

Given a two-dimensional input $\mathbf{x} = [x_1, x_2] \in [-5, 10] \times [0, 15]$, the Branin function is defined as:

$$f(\mathbf{x}) = \left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos(x_1) + 10.$$

The function has three identical global minima with value $f^* \approx 0.3979$ located at:

$$\mathbf{x}^* \in \{[-\pi, 12.275],\ [\pi, 2.275],\ [9.42478, 2.475]\}.$$

These optima are well-separated and embedded in broad, curved valleys. The presence of multiple basins also makes

this function an effective test for whether an optimizer can consistently locate diverse good regions rather than converging prematurely [1].

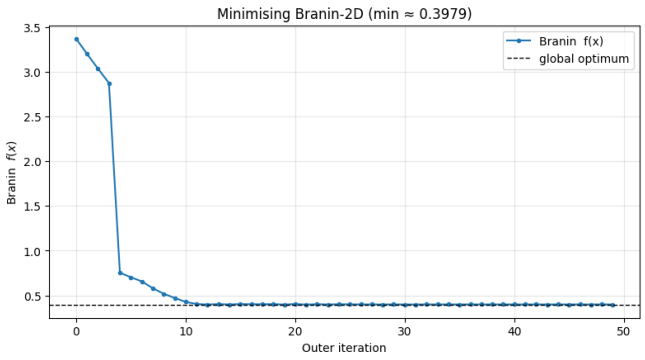

Fig. 4: Convergence of BOWL on the Branin function. Dashed line shows the global minimum $f^* \approx 0.3979$.

Discussion

Across all three benchmark problems, BOWL demonstrates clear signs of promise, especially in low-dimensional, rugged landscapes. The consistent use of MPD-informed Langevin steps, guided by GP posterior gradients, appears to help the method escape narrow basins and avoid local traps.

Convergence Stability. On the Hartmann-3D problem, BOWL shows rapid initial gains and stabilizes just below the known global maximum of $f^* \approx 3.8547$, even when started from random Sobol points. The presence of occasional dips followed by strong recoveries in the convergence curve (Figure 2) supports the idea that the optimizer retains flexibility to reorient and course-correct, rather than getting stuck, a seemingly direct benefit of the inner-outer structure.

Robust Descent. For Levy-2D, a notoriously deceptive and rugged landscape with many local minima, BOWL steadily converges toward the global optimum at $\mathbf{x}^* = [1, 1]$ despite noise and irregularities. While it does not immediately snap to the lowest region, the progressive improvement seen across 50 steps indicates that the optimizer is capable of navigating complex multimodal terrain.

Generalization over Local Traps. On Branin, BOWL reaches a final score close to the global minimum $f^* \approx 0.3979$ and avoids getting locked into local optima and converged quite fast.

Effect of Hyperparameters. All tests used the same optimizer configuration:

- `n_initial` $= 20$ (i.e., $5 \times d$) initial Sobol points
- `n_iters` $= 50$ outer loop steps
- Outer step size $\eta = 0.05$
- $L = 20$ inner SGLD updates
- Scope parameters $\gamma_0 = 10^{-2}, \gamma_1 = 10^{-4}$
- GP Refit frequency = Every 3 outer loop steps

The fact that these fixed settings yielded near-optimal solutions across function classes without heavy tuning shows promise for what could come.

Limitations

While BOWL performs well on structured low-dimensional tasks, it faces significant challenges when scaling to higher-dimensional black-box problems. The core issue stems from the naive way high-dimensionality is currently handled, as we treat all input dimensions equally and attempt to model full gradient posteriors using relatively few observations. As dimensionality increases, the quality of GP gradient estimates deteriorates rapidly, leading to noisy, unstable inner-loop dynamics. Moreover, because BOWL currently relies on a single-point update strategy, it lacks the parallel information gain that could come from batch evaluations. Incorporating a mini-batching scheme — where multiple points are proposed and queried per iteration — could dramatically improve sample efficiency and help stabilize learning in wider, under-explored regions of the search space.

Another shortcoming is the relatively loose integration between Entropy-SGD and MPD-style guidance. While BOWL uses the MPD gradient posterior to steer inner Langevin steps, it does not yet adopt the full trajectory-based search or probabilistic descent analysis that MPD supports. In higher dimensions, such MPD features like evaluating descent probabilities across multiple local directions or performing directional projections may become essential for preserving signal in a sea of noise. A more principled fusion, where entropy-scoped updates are explicitly weighted by directional descent probability, could offer a stronger inductive bias toward robustness. Addressing both of these fronts remains critical for generalizing the method to real-world, large-scale applications.

Compelling Idea

Beyond the core implementation of BOWL, a compelling direction for future work is to build a more tightly integrated version of MPD that directly incorporates entropy-based updates. We could reverse the relationship: start from the original MPD framework and embed a local entropy-driven optimizer within it. This would mean using the MPD loop to determine robust movement directions and descent probabilities which perform well in high-dimensions, but instead of taking deterministic steps or posterior mean updates, use an inner Entropy-SGD optimizer to refine the local search, leveraging stochasticity to bias the solution toward flat optima.

This could be added as a *move* method in the MPDOptimizer class (which includes "step", "iter", and "mu" already). In this version, each MPD update includes an "entropy move" phase, where a local Langevin-style optimizer is run using GP-derived posterior gradients. Crucially, this optimizer operates over the GP model and never requires access to the true function's gradients. This introduces a more expressive and uncertainty-aware descent step, where sampling noise and posterior variance naturally steer the optimization away from

sharp, brittle basins. By combining MPD's global view with Entropy-SGD's local robustness, this could be an interesting way to regularize the MPD gradient ascent.

Conclusion

In this work, we introduced BOWL, a framework that fuses the robustness-seeking dynamics of Entropy-SGD with the uncertainty-aware guidance of Maximum Probability of Descent (MPD), adapted for black-box optimization. Our goal was to build a principled optimizer that not only finds high-performing solutions, but favors those lying in wide, flat regions of the landscape, solutions that are empirically more stable and generalizable.

Through experiments on classic synthetic benchmarks including Hartmann-3D, Levy, and Branin, we demonstrated that BOWL is capable of consistently reaching near-optimal values with relatively few queries. Its performance, while still limited in high-dimensional settings, shows strong promise in low-dimensional tasks.

Although suffering from certain limitations, future work would include scaling BOWL to higher dimensions via smarter batch selection and sparse modeling, as well as exploring more expressive acquisition policies directly inspired by MPD.

References

[1] Eric Brochu, Vlad M Cora, and Nando De Freitas. *A Tutorial on Bayesian Optimization of Expensive Cost Functions, with Application to Active User Modeling and Hierarchical Reinforcement Learning*. arXiv preprint arXiv:1012.2599, 2010.

[2] Pratik Chaudhari et al. "Entropy-SGD: Biasing Gradient Descent Into Wide Valleys". In: *Journal of Statistical Mechanics: Theory and Experiment* 2019.12 (2019).

[3] Roman Garnett. *Bayesian Optimization*. Available online. Cambridge University Press, 2023.

[4] Jungtaek Kim. *Benchmark Functions for Bayesian Optimization*. https://github.com/jungtaekkim/bayeso-benchmarks. Version updated February 2023. 2023.

[5] Lam Si Nguyen et al. "Maximum Probability of Descent for Bayesian Optimization". In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 11841–11853.