Who is the Taste Master?

The project implements a competitive agent-based system where a user-configured optimizer challenges a pre-tuned "House Bot." Both agents utilize Gaussian Process Regression (GPR) as a surrogate model and the Upper Confidence Bound (UCB) acquisition function to navigate the search space. By exposing critical hyperparameters—specifically the exploration factor (κ), length scale (l), and noise variance (σ_n^2)—the simulation allows for empirical analysis of the exploration-exploitation trade-off. Post-hoc visualizations, including decision trajectory mapping and terrain heatmaps, provide a concrete validation of the algorithmic convergence behaviors. This project serves as both a rigorous implementation of probabilistic machine learning and an educational tool for visualizing high-dimensional optimization dynamics.

The story takes place in the year 2200, in a dimly lit, retro-futuristic underground bar on the edge of a colonized exoplanet. Every day, a diverse array of customers patronize the place; they are not necessarily ordinary humans, but more often travelers from distant space. In this simulation game, players take on the role of the head bartender. The task seems simple: concoct a drink using three main ingredients: sweet, sour, and bitter. However, the customers' taste preferences are a mystery. Mathematically, it's represented as a "black box" function: a complex, multimodal, three-dimensional terrain composed of a Gaussian distribution, filled with noise. You can't see that map, but fortunately, technology changes everything—you can design a robot based on a Bayesian optimization algorithm.

Players are not alone. At the other end of the bar, there's a robot claiming to be "state-of-the-art" in bartending. This artificial intelligence is also driven by a rigorous Bayesian optimization algorithm.

Each robot has ten chances to try different combinations, each one crucial. Trial and error is inefficient, and given the infinite number of ingredient combinations, exhaustive search is impossible. Given the dual constraints of limited resources and high evaluation costs, who will ultimately prevail?

1. Mathematical Modeling of the Culinary Space

To simulate the challenge of discovering a unique taste profile, we model the customer's preference as a high-dimensional, continuous, and stochastic objective function. The problem is framed as a "Black Box" optimization task, where the underlying function form is unknown to the agent, and evaluations are strictly limited to a budget of $T = 10$ iterations.

A. The Input Space. The domain $\mathcal{X}$ is defined as a three-dimensional hypercube representing the ratio of ingredients. Let $x \in \mathbb{R}^3$ be a recipe vector:

$$x = [s, a, b]^T \in [0, 100]^3 \quad [1]$$

where s, a, b represent Sweetness, Sourness (Acidity), and Bitterness, respectively. The search space is continuous, allowing for an infinite number of potential combinations.

B. The Objective Function (Customer Taste). The ground truth satisfaction score, $f(x)$, is modeled as a multimodal surface generated by a Sum of Gaussians (SoG). This creates a non-convex optimization landscape with multiple local maxima, simulating the complexity of sensory perception where specific "flavor accords" create peaks in enjoyment.

The function is defined as:

$$f(x) = \text{clamp}\left(\sum_{i=1}^{K} H_i \cdot \exp\left(-\frac{\|x - \mu_i\|^2}{2\sigma_i^2}\right) + \epsilon,\ 0,\ 100\right) \quad [2]$$

Where:

- K: The number of "flavor peaks," randomized between 2 and 4 for each game session.
- $H_i \sim \mathcal{U}(60, 100)$: The amplitude of the i-th peak, representing the maximum potential satisfaction of that specific flavor profile.
- $\mu_i \in \mathbb{R}^3$: The center of the i-th peak (the ideal ratio for that specific flavor profile).
- $\sigma_i \sim \mathcal{U}(8, 20)$: The characteristic width of the peak. A small σ implies a niche, precise taste requirement, while a large σ implies a broad tolerance.
- $\epsilon \sim \mathcal{N}(0, \sigma_{noise}^2)$: Additive Gaussian noise representing the inherent subjectivity and imperfection of the tasting process.

The agent observes $y_t = f(x_t)$ at each step t, but never observes the parameters H, μ, σ directly.

2. Algorithmic Implementation: The Bayesian Bot

To navigate this landscape efficiently, the automated agents utilize Bayesian Optimization (BO). This is a sequential design strategy for global optimization of black-box functions. The implementation consists of two primary components: a Gaussian Process (GP) surrogate model and an Upper Confidence Bound (UCB) acquisition function.

A. Gaussian Process Regression (GPR). The bot maintains a probabilistic belief over the objective function $f(x)$. We assume a Gaussian Process prior:

$$f(x) \sim \mathcal{GP}(m(x), k(x, x')) \quad [3]$$

For the covariance function $k(x, x')$, I utilize the Radial Basis Function (RBF) kernel (also known as the Squared Exponential kernel). This kernel encodes the assumption that recipes with similar ingredient ratios yield similar scores:

$$k(x, x') = \sigma_f^2 \exp\left(-\frac{\|x - x'\|^2}{2l^2}\right) \quad [4]$$

Given a dataset of previous observations $\mathcal{D}_{1:t} = \{(x_1, y_1), \ldots, (x_t, y_t)\}$, the posterior distribution for a new candidate recipe x_* follows a Gaussian distribution $\mathcal{N}(\mu_*(x_*), \sigma_*^2(x_*))$. The predicted mean and variance are derived by inverting the covariance matrix K:

$$\mu_*(x_*) = k_*^T (K + \sigma_n^2 I)^{-1} \mathbf{y} \quad [5]$$

$$\sigma_*^2(x_*) = k(x_*, x_*) - k_*^T (K + \sigma_n^2 I)^{-1} k_* \quad [6]$$

B. Acquisition Function: Upper Confidence Bound (UCB). To select the next recipe to try, the bot must balance *exploration* (sampling areas with high uncertainty σ) and *exploitation* (sampling areas with high predicted mean μ). We employ the UCB acquisition function:

$$\alpha_{UCB}(x) = \mu(x) + \kappa \cdot \sigma(x) \quad [7]$$

The optimal next point is selected by maximizing this acquisition function:

$$x_{t+1} = \arg\max_{x \in \mathcal{X}} \alpha_{UCB}(x) \quad [8]$$

In my implementation, this maximization is approximated via Monte Carlo sampling ($N = 1000$) over the input domain.

C. Hyperparameter Configuration. The behavior of the Bayesian optimizer is governed by specific hyperparameters. In this simulation, the "House Bot" (the opponent) utilizes a specific tuning configuration designed to be robust yet aggressive:

- **Length Scale ($l = 15$):** Defines the "smoothness" of the function. The bot assumes that flavor correlations decay significantly if ingredients differ by more than 15 units.
- **Exploration Factor ($\kappa = 2.5$):** Controls the risk appetite. A value of 2.5 indicates an aggressive exploration strategy suitable for short-horizon tasks (10 turns), prioritizing high-uncertainty regions over safe, local optima.
- **Signal Variance ($\sigma_f = 50$):** Represents the expected vertical amplitude of the function (score range).
- **Observation Noise ($\sigma_n^2 = 25$):** A regularization term added to the diagonal of the covariance matrix to ensure numerical stability and prevent overfitting to noisy customer feedback.

3. Interactive Simulation and Analysis

To validate the theoretical models and allow for empirical observation of Bayesian Optimization in action, I developed "Taste Master," a web-based simulation environment. The application transforms the abstract mathematical problem into a visual narrative, allowing users to assume the role of an Algorithm Engineer.

A. Algorithm Configuration. Before the simulation commences, the user is tasked with defining the hyperparameters for their custom optimizer. Unlike standard black-box tools, our interface exposes the critical internal variables of the Gaussian Process.

As shown in Figure 1, the user adjusts:

- **Exploration (Kappa):** Determining the agent's risk tolerance.
- **Sensitivity (Length Scale):** Defining the assumed correlation between ingredient ratios.
- **Confidence (Noise Variance):** Establishing the agent's trust in the noisy feedback.

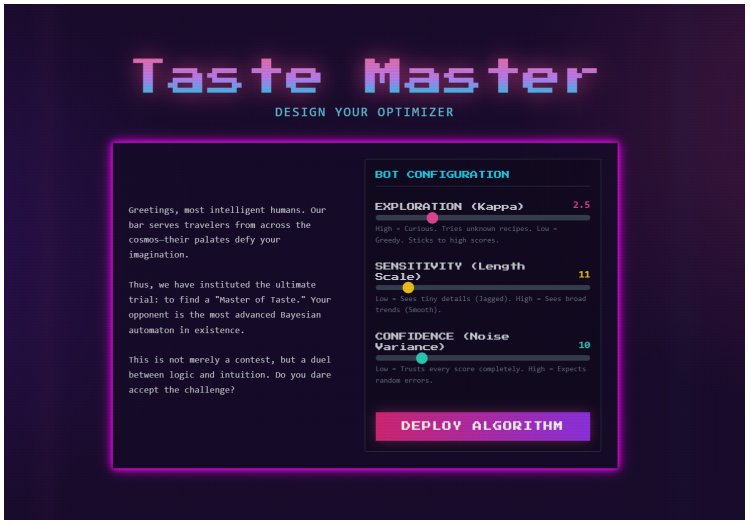

Fig. 1. The Algorithm Configuration Interface. The user defines the hyperparameters κ, l, and σ_n^2 before the simulation begins.

B. Real-Time Competitive Simulation. Once the simulation initializes, the interface shifts to the "Battle Arena" (Figure 2). The screen displays two dynamic liquid containers representing the distinct ingredient vectors selected by the opposing algorithms.

The simulation proceeds in discrete turns ($t = 1 \ldots 10$). In each iteration:

1. The **Custom Bot** (User) and the **House Bot** (Rival) independently compute the maximization of their respective Acquisition Functions.
2. The selected recipes are submitted to the hidden Objective Function $f(x)$.
3. The system maps the continuous numerical score y_t to qualitative performance tiers. A critique is then stochastically selected from a pre-defined corpus of personality-driven dialogue. This textual feedback is paired with a procedurally generated pixel-art avatar, adding narrative immersion to the numerical optimization process without external dependencies.

C. Visualizing the Decision Landscape. Upon the conclusion of the simulation, the "Black Box" is revealed. Because the objective function exists in a 3D domain (Sweet, Sour, Bitter), visualizing the full terrain is distinctively challenging.

I employ a "Slicing Strategy" to render the results, as seen in Figure 3. The system identifies the global maximum (the hidden "God Particle" recipe) and generates three 2D heatmaps. Each heatmap represents a cross-section of the hypercube passing through this optimal point.

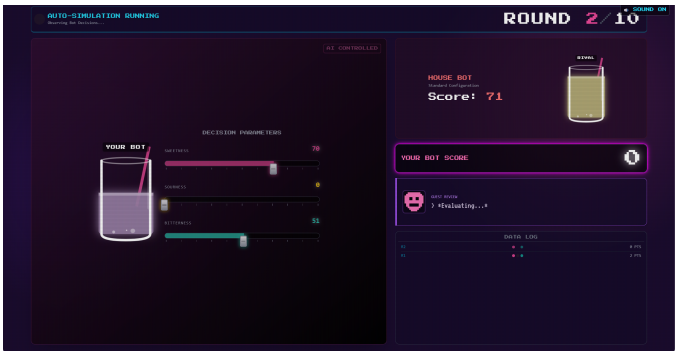

Fig. 2. The Simulation Interface. The left panel displays the User's Bot decisions, the right panel shows the House Bot, and the bottom section displays the customer feedback.

- The **Cyan Path** traces the trajectory of the User's Custom Bot.
- The **Red Path** traces the trajectory of the House Bot.
- The **White Crosshair** indicates the true global optimum.

This visualization effectively demonstrates the "Exploration vs. Exploitation" behavior, showing how the agents spiraled toward (or missed) the peak.

Fig. 3. Post-Hoc Analysis. The heatmaps reveal the hidden Sum-of-Gaussians terrain. The arrows indicate the sequential decision-making path of the algorithms.

4. Appendix

Project URL: https://statscolin.github.io/taste-master/