Sample-Constrained Black Box Optimization for Audio Personalization

Rajalaxmi Rajagopalan, Yu-Lin Wei, Romit Roy Choudhury

Department of Electrical & Computer Engineering
University of Illinois at Urbana-Champaign
{rr30,yulinlw2,croy}@illinois.edu

Abstract

We consider the problem of personalizing audio to maximize user experience. Briefly, we aim to find a filter h^*, which applied to any music or speech, will maximize the user's satisfaction. This is a black-box optimization problem since the user's satisfaction function is unknown. Substantive work has been done on this topic where the key idea is to play audio samples to the user, each shaped by a different filter h_i, and *query* the user for their satisfaction scores $f(h_i)$. A family of "surrogate" functions is then designed to fit these scores and the optimization method gradually refines these functions to arrive at the filter $\hat{h}^*$ that maximizes satisfaction.

In certain applications, we observe that a second type of querying is possible where users can tell us the individual elements $h^*[j]$ of the optimal filter h^*. Consider an analogy from cooking where the goal is to cook a recipe that maximizes user satisfaction. A user can be asked to score various cooked recipes (e.g., tofu fried rice) or to score individual ingredients (say, salt, sugar, rice, chicken, etc.). Given a budget of B queries, where a query can be of either type, our goal is to find the recipe that will maximize this user's satisfaction.

Our proposal builds on Sparse Gaussian Process Regression (GPR) and shows how a hybrid approach can outperform any one type of querying. Our results are validated through simulations and real world experiments, where volunteers gave feedback on music/speech audio and were able to achieve high satisfaction levels. We believe this idea of hybrid querying opens new problems in black-box optimization and solutions can benefit other applications beyond audio personalization.

Introduction

Consider the problem of personalizing content to a user's taste. Content could be audio signals in a hearing aid, a salad cooked for the user, a personalized vacation package designed by an AI agent, etc. Given the content c, we intend to adjust the content with a linear filter h. Our goal is to find the optimal filter h^* that will maximize the user's personal satisfaction $f(h)$.

Finding h^* is difficult because the user's satisfaction function $f(h)$ is unknown; it is embedded somewhere inside the perceptual regions of the brain. Hence, gradient descent is not possible since gradients cannot be computed. Black box optimization (BBO) has been proposed for such settings, where one queries user-satisfaction scores for carefully chosen filters h_i. Using a budget of B such queries, BBO expects to estimate $\hat{h}^*$ that is close to the true h^*. Of course, reducing the query budget B is of interest, and the effects of lowering B have been studied extensively.

The above problem can be called "*filter querying*" because the user is queried using different filters $h_i \in \mathbf{R}^N$. In this paper, we discuss an extension to this problem where a second type of querying is possible, called "*dimension querying*". With dimension querying, it is possible to query the user for each dimension of the optimal filter, namely $h^*[1], h^*[2], \ldots, h^*[N]$. In audio personalization, for example, the optimal h^* is the hearing profile of a user in the frequency domain; if we accurately estimate the hearing profile, we can maximize their satisfaction. With dimension querying, a user can listen to sound at each individual frequency $j \in \{1, N\}$ and tell us the best score $h^*[j]$. The only problem is that N can be very large, say 8000 Hz, hence it is prohibitive to query the user thousands of times.

To summarize, a filter query offers the advantage of understanding user satisfaction for a complete filter, while a dimension query gives optimal information for only one dimension at a time. In our analogy from cooking, filter querying gives us the user satisfaction for a fully prepared recipe (e.g., tofu fried rice), while dimension querying gives us the user's optimal liking for individual dimensions, like salt, sugar, etc. This paper asks, for a given query budget B, can the combination of two types of querying lead to higher user satisfaction compared to a single type of querying? How much is the gain from the combination, and how does the gain vary against various application parameters?

Our solution builds on past work that uses Gaussian Process Regression (GPR). Conventional GPR begins with a family of surrogate functions for $f(h)$ and estimates a posterior on these functions, based on how well they satisfy the user's scores. Over time, GPR iterates through a two-step process, first picking an optimal filter h_i to query the user, and based on the score $f(h_i)$, updates the posterior distribution. The goal is to query and update the posterior until a desired criteria is met. Once met, the mean of this posterior $\hat{f}(h)$ is declared as the surrogate for $f(h)$ and $\hat{h}^* = \text{argmax}\, \hat{f}(h)$ is

announced as the final personalization filter.

When applied to our cooking analogy, conventional GPR will prescribe different recipes and based on scores from the user, will construct a posterior on candidate satisfaction functions. The next recipe will be chosen to be one that could return a higher score than all past scores (say, Japanese sushi). In essence, GPR tries to reason about the shape of the satisfaction function using only recipe-based querying.

Our contribution lies in *querying the user with better recipes* to expedite the process of estimating the satisfaction function. We modify the GPR framework to first estimate a batch of q filters (instead of just one) from the posterior; we then choose a single winner based on which has the strongest similarity to the dimension-scores. Finally, these operations are performed after GPR has been transformed into a sparse space, otherwise it becomes difficult to meet the query budget B.

Results show that spending some query budget on dimension queries (salt and sugar) as opposed to spending all the budget on filter queries (full recipes) offers benefits. We lack mathematical proof, instead show empirical results from extensive simulations and real-world experiments. With real volunteers who were asked to rate audio quality on a scale of $[0-10]$, our proposed method, ORACLEBO, achieves an average of 3.3 points higher satisfaction, within a budget of $B = 30$ queries. Using simulations with various synthetic satisfaction functions, we find that the break-down between the two types of queries exhibits a sweet spot. When the number of dimension queries increases (or decreases) beyond a fraction of B, the achieved maxima deviates from the global maxima. We present sensitivity analysis and ablation studies, and discuss a number of follow-up questions for future research.

Problem Formulation

Consider an *unknown* real-valued function $f : \mathcal{H} \rightarrow \mathbf{R}$ where the domain is a compact subspace $\mathcal{H} \subseteq \mathbf{R}^N$, $N \geq 500$. Let h^* be the minimizer of $f(h)$. We want to estimate h^* using a budget of B queries, where a query can be one of two types:

1. The unknown function f can be sampled at a given h_i. This query yields $f(h_i)$. We call these *filter queries*, Q_f.
2. An Oracle is assumed to know information about the minimizer h^*. The Oracle when queried can give us one dimension of the vector h^*, i.e., we can obtain $h^*[j]$, for any given $j \in [1, 2, \ldots, N]$. We call these *dimension queries*, Q_d.

Thus, the optimization problem is shown in Eqn. 1.

$$\begin{aligned} \underset{\hat{h} \in \mathcal{H}}{\text{argmin}} \quad & ||f(\hat{h}) - f(h^*)||_2 \\ \text{s.t.} \quad & Q_f + Q_d \leq B \end{aligned} \tag{1}$$

where Q_f, Q_d are the number of filter and dimension queries described above, and the query budget $B \ll N$.

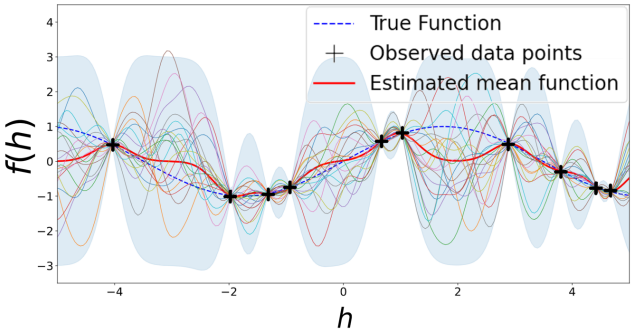

Figure 1: A GPR Posterior: The black "pluses" denote observed points. The dashed-blue line is the true f, and the red line is the estimated $\hat{f}$. The light blue shading is the variance.

The unknown function f may be non-convex, may not have a closed-form expression, and its gradient is unavailable. However, since it can be queried for a given h_i, it is called a black-box optimization problem. We approach this through Bayesian Optimization that builds on Gaussian Process Regression (GPR), Expected Improvement (EI) acquisition function, and sparsity transformations (to cope with the large gap between N and B). We review the relevant background on Bayesian optimization next, followed by our proposed algorithm, ORACLEBO.

Bayesian Optimization

Bayesian optimization (Frazier 2018) broadly consists of the following two modules:

(1) **Surrogate model**: A family of functions that serve as candidates for the unknown objective function. The functions are commonly drawn from a Gaussian process generated by **Gaussian Process Regression (GPR)**. This essentially means that GPR generates a Gaussian posterior distribution of the likely values function f can take at any point of interest h.

(2) **Acquisition function**: A sampling strategy that prescribes the point at which f should be observed next. The GPR posterior model is used to evaluate the function at new points h', and one is picked that maximizes a desired metric. This new point h' when observed will maximally improve the GPR posterior.

We review GPR next, followed by a popular acquisition function called "Expected Improvement".

Gaussian Process Regression (GPR)

Non-parametric Model: Gaussian processes (Wang 2020) are helpful for black-box optimization because they provide a non-parametric mechanism to generate a surrogate for the unknown function f. Given a set of samples $\mathcal{X} = \{h_1, h_2, \ldots, h_K\}$ at which the function f has been observed, i.e., we know $\mathcal{F} = \{f(h_1), f(h_2), \ldots, f(h_K)\}$, we can identify an infinite number of candidate functions that match the observed function values. Figure 1 shows an example function f in 1-dimensional space.

Function Distribution & Kernel: GPR generates the surrogate model by defining a Gaussian distribution over these infinite candidate functions. Given the set of observations $(\mathcal{X}, \mathcal{F})$, the mean $\boldsymbol{\mu}$ of the distribution is the most likely surrogate of the function f. The covariance $\mathbf{K}$ is a kernel that dictates the smoothness (or shape) of the candidate functions and must be chosen based on domain knowledge of f. One commonly used kernel is the ARD (Automatic Relevance Determination) *power exponential kernel*:

$$k(h, h') = a_0 \exp -\frac{1}{2}(h - h')^T \Sigma^{-1}(h - h') \quad (2)$$

where, a_0 and $\Sigma = \text{diag}_i(\sigma_i)$ are the kernel parameters.

Prior & Posterior: Before any observations, the distribution defined by $\mathbf{K}$ and $\boldsymbol{\mu} = \mathbf{0}$ forms the prior distribution. Given a set of observations $(\mathcal{X}, \mathcal{F})$, the prior is updated to form the posterior distribution over the candidate functions. Figure 1 shows the distribution of candidates and the mean surrogate model of an example f. With more observations, the current posterior serves as the prior, and the new posterior updates from the new observations. Eqn. 3 models the function f with the posterior generated by GPR.

$$P(\mathcal{F}|\mathcal{X}) \sim \mathcal{N}(\mathcal{F}|\boldsymbol{\mu}, \mathbf{K}) \quad (3)$$

where, $\boldsymbol{\mu} = \{\mu(h_1), \mu(h_2), \ldots, \mu(h_K)\}$ and $\mathbf{K}_{ij}$=$k(h_i, h_j)$, k represents a kernel function.

Predictions: To make predictions $\hat{\mathcal{F}} = f(\hat{\mathcal{X}})$ at new points $\hat{\mathcal{X}}$, GPR uses the current posterior $P(\mathcal{F}|\mathcal{X})$ to define the joint distribution of $\mathcal{F}$ and $\hat{\mathcal{F}}$, $P(\mathcal{F}, \hat{\mathcal{F}}|\mathcal{X}, \hat{\mathcal{X}})$ in Eqn. 4.

$$\begin{bmatrix} \mathcal{F} \\ \hat{\mathcal{F}} \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu(\mathcal{X}) \\ \mu(\hat{\mathcal{X}}) \end{bmatrix}, \begin{bmatrix} \mathbf{K} & \hat{\mathbf{K}} \\ \hat{\mathbf{K}}^T & \hat{\hat{\mathbf{K}}} \end{bmatrix} \right) \quad (4)$$

where, $\mathbf{K} = k(\mathcal{X}, \mathcal{X})$, $\hat{\mathbf{K}} = k(\mathcal{X}, \hat{\mathcal{X}})$, $\hat{\hat{\mathbf{K}}} = k(\hat{\mathcal{X}}, \hat{\mathcal{X}})$ and $(\mu(\mathcal{X}), \mu(\hat{\mathcal{X}})) = \mathbf{0}$.

The conditional distribution and hence prediction of $\hat{\mathcal{F}}$ is derived from the joint distribution shown in Eqn. 5. The proof and explanations of all the above are clearly presented in (Wang 2020)).

$$P(\hat{\mathcal{F}}|\mathcal{F}, \mathcal{X}, \hat{\mathcal{X}}) \sim \mathcal{N}(\hat{\mathbf{K}}^T \mathbf{K}^{-1} \mathcal{F}, \hat{\hat{\mathbf{K}}} - \hat{\mathbf{K}}^T \mathbf{K}^{-1} \hat{\mathbf{K}}) \quad (5)$$

Acquisition Function

In each GPR iteration, a new h must be acquired such that the observed $f(h)$ maximally improves the posterior from the previous iteration. This requires a **judicious sampling** strategy that optimizes an improvement metric. "Expected Improvement" (EI) is one such popular metric that we will build on in this paper.

Expected Improvement: Given previous observations $\mathcal{D} = (\mathcal{X}, \mathcal{F})$, let $f^* = \min_{x \in \mathcal{X}} f(x)$ be the current function minimum (i.e., the minimum observed till now). If a new observation $f(h)$ is made at h, then the minimum now will be either $f(h)$ if $f(h) \leq f^*$ or f^* if $f(h) \geq f^*$.
Hence, the improvement from observing f at h is $[f^* - f(h)]^+$, where, $a^+ = \max(a, 0)$.

We want to choose h that maximizes this improvement. However, $f(h)$ is unknown until the observation is made, so we choose h that maximizes the expectation of this improvement. *Expected Improvement* is thus defined as:

$$\mathbf{EI}(h|\mathcal{X}, \mathcal{F}) = \mathbf{E}[[f^* - f(h)]^+|\mathcal{X}, \mathcal{F}] \quad (6)$$

where, $\mathbf{E}[\cdot|\mathcal{X}, \mathcal{F}]$ is the expectation taken on the GPR posterior distribution given observations $(\mathcal{X}, \mathcal{F})$. This posterior is as specified in Eqn. 3. Thus, the next sample to make an observation at is:

$$h = \underset{h_i \in \mathbf{R}^N}{\text{argmax}}\, \mathbf{EI}(h_i|\mathcal{X}, \mathcal{F}) \quad (7)$$

High Dimensional Bayesian Optimization

Curse of Dimensionality: The objective function in Eqn. 1 typically lies in a high dimensional space (i.e., $h \in \mathcal{H} \subseteq \mathbf{R}^N, N \geq 500$). Bayesian optimization works well for functions of < 20 dimensions (Frazier 2018); with more dimensions, the search space $\mathcal{H}$ increases exponentially, and finding the minimum with *few* evaluations becomes untenable. One approach to reducing the number of queries is to exploit the sparsity inherent in most real-world functions.

We assume our function in Eqn. 1 is sparse, i.e., there is a low-dimensional space that compactly describes f, so f has "low effective dimensions". We review ALEBO (Letham et al. 2020), a class of methods that exploit sparsity to create a low-dimensional embedding space using random projections. Our proposed idea builds on top of ALEBO, but we are actually agnostic of any specific sparsity method.

Linear Embedding Using Random Projections

Random Projections: Given a function $f : \mathbf{R}^N \to \mathbf{R}$ with effective dimension d_f, ALEBO's linear embedding algorithm uses random projections to transform f to a lower dimensional embedding space. This transformation must guarantee that the minimum h^* from high dimensional space $\mathcal{H}$ gets transformed to its corresponding minimum y^* in low dimensional embedding space. The right side of Figure 2 aims to visualize this transformation. Without satisfying this property, optimization in low-dimensions is not possible.

The random embedding is defined by an embedding matrix $\mathbf{B} \in \mathbf{R}^{d \times N}$ that transforms f into its lower dimensional equivalent $f_B(y) = f(h) = f(\mathbf{B}^\dagger y)$, where $\mathbf{B}^\dagger$ is the pseudo-inverse of $\mathbf{B}$. Bayesian optimization of $f_B(y)$ is performed in the lower dimensional space $\mathbf{R}^d$.

Clipping to $\mathcal{H}$: When f is optimized over a compact subset $\mathcal{H} \subseteq \mathbf{R}^N$, we cannot evaluate f outside $\mathcal{H}$. One approach to prevent any embedding point y from being projected outside of $\mathcal{H}$ is to "clip" such points to $\mathcal{H}$. This is done by projecting the points back into $\mathcal{H}$, i.e., $f_B(y) = f(p_{\mathcal{H}}(\mathbf{B}^\dagger y))$ where, $p_{\mathcal{H}} : \mathbf{R}^N \to \mathbf{R}^N$ is the clipping projection. However, this clipping to $\mathcal{H}$ causes nonlinear distortions.

Instead, constraining the optimization to only points in $\mathcal{Y}$ that do not project outside $\mathcal{H}$, i.e., $\mathbf{B}^\dagger y \in \mathcal{H}$, prevents distortions; however, it also reduces the probability of the embedding containing the optimum h^*. ALEBO remedies this by choosing $d > d_f$ (an embedding space larger than f's

effective dimensions). Then, the acquisition function evaluated in the constrained embedding space is given as:

$$\begin{aligned} &\underset{y\in\mathbf{R}^d}{\operatorname{argmax}}\ \mathbf{EI}(y) \\ \text{s.t.}\quad &-1 \le \mathbf{B}^\dagger y \le 1 \end{aligned} \tag{8}$$

where the constraint $-1 \le \mathbf{B}^\dagger y \le 1$ are linear and form a polytope.

Modifications to the Kernel: ARD kernels in $\mathcal{H}$ (shown in Eqn. 2) do not translate to a product kernel in embedding $\mathcal{Y}$, since each dimension in $\mathcal{H}$ is independent (diagonal matrix Σ in Eqn. 2). However, moving along one dimension in embedding is similar to moving across all dimensions of $\mathcal{H}$. To combat this, a *Mahalanobis Kernel* is used in the embedding. Any two points in the embedding are projected up to $\mathbf{R}^N$ ($\mathbf{B}^\dagger$) and then projected down to $\mathcal{H}$ ($\mathbf{A}$), $f_B(y) = f(\mathbf{B}^\dagger y) = f(\mathbf{A}\mathbf{B}^\dagger y)$ and $\text{Cov}[f_B(y), f_B(y')] = \exp\{-(y-y')^T\mathbf{\Gamma}(y-y')\}$ where $\mathbf{\Gamma} = (\mathbf{A}^T\mathbf{B}^\dagger)^T\Sigma(\mathbf{A}^T\mathbf{B}^\dagger)$ is a symmetric positive definite matrix. This finally ensures correctness in sparsity-based Bayesian Optimization (BO).

With this review of Bayesian Optimization and sparsity-based ALEBO, we discuss our algorithm, ORACLEBO.

ORACLEBO

ORACLEBO's main contribution is in modifying ALEBO's acquisition function (Eqn. 8) to incorporate queries of type Q_d, namely *dimension queries*. ALEBO and related sparsity-based algorithms are designed to use filter queries of type Q_f; modifying these algorithms to also incorporate Q_d needs cautious design. This is because filter samples h_i are N-dimensional vectors while dimension queries $h^*[j]$ are scalar values. Combining these modalities correctly, especially through the sparsity transformations in ALEBO, requires reworking at the heart of sparsity-based BO methods.

Figure 2(a) illustrates the design of ORACLEBO – the modules in gray are the proposed extensions over the literature. The two key modules are (1) **Batch Acquisition Function (BAF)**, and (2) **Dimension Matched Sampler (DMS)**.

Conventional SparseBO obtains a *single* sample h' from the acquisition function and makes a function observation $f(h')$. This observation is then used to update the GPR posterior. In contrast, the **BAF** module in ORACLEBO intends to pick a batch of q (jointly optimal) samples $\{h'_1, h'_2, \ldots, h'_q\}$ and the **DMS** module orders them preferentially by matching them against the dimensional information of the minimizer $h^*[j], j \in \mathcal{L}$. Through this method, we are selecting the next filter sample h' by essentially combining the "wisdom" of both types of querying. Said differently, we first sample a batch of candidates which are all "good" choices as per the EI metric (e.g., each sample is a different sushi recipe), and then, dimension matching makes the final selection in favor of one $h' \in \{h'_1, h'_2, \ldots, h'_q\}$ that aligns with the optimal $h^*[j], j \in \mathcal{L}$. This means if the user has a high preference for the sweet dimension, then the next Q_f query h' becomes a "sweet sushi recipe". The details are presented next.

Batch Acquisition Function (BAF)

Instead of picking one sample h', **BAF** proposes to pick a batch of q samples $\{h'_1, h'_2, \ldots, h'_q\} = \mathbf{B}^\dagger\{y'_1, y'_2, \ldots, y'_q\}$ that *jointly* maximize the acquisition function.

We assign a joint metric, *q-ExpectedImprovement* (qEI), to a set of q candidate points $\mathcal{Q}' = \{y'_1, y'_2, \ldots, y'_q\}$ in the search space $\mathcal{Y}$. This is realized through two steps:

(1) Sampling the joint distribution of q points under the current posterior using MCMC sampling (Neal 2003) to obtain $\mathcal{Q} = \{y_1, y_2, \ldots, y_q\}$

(2) Evaluating joint metrics q$\mathbf{EI}(y_i)$ for $y_{1:q}$ over the current minimum $f^* = \min_{y\in\mathcal{Y}} f(\mathbf{B}^\dagger y)$ as follows:

$$\begin{aligned} \text{q}\mathbf{EI}(y_i|\mathcal{D}_n, \mathcal{Q}) &= \mathbf{E}[[f^* - f(y_i)]^+|\mathcal{D}_n, \mathcal{Q}] \\ \text{q}\mathbf{EI}(\mathcal{Q}) &\triangleq \{\text{q}\mathbf{EI}(y_i|\mathcal{D}_n, \mathcal{Q})\} \end{aligned} \tag{9}$$

where $\mathcal{D}_n$ denotes current set of observations. Compared to $\mathbf{EI}(y_i|\mathcal{D}_n)$ in Eqn. 6, q$\mathbf{EI}(y_i)$ marginalizes the expectation over the candidate set $\mathcal{Q}$.

We select the candidate set $\mathcal{Q}'$ of highest expected improvement as follows:

$$\begin{aligned} \mathcal{Q}' = \{y'_1, y'_2, \ldots, y'_q\} = &\underset{\mathcal{Q}=\{y_1,y_2,\ldots,y_q\}}{\operatorname{argmax}}\ max(\text{q}\mathbf{EI}(\mathcal{Q})) \\ &\text{s.t. } -1 \le \mathbf{B}^\dagger y'_i \le 1 \quad \forall y'_i \in \mathcal{Q}' \end{aligned} \tag{10}$$

where each $\mathcal{Q}$ is ranked with the highest q$\mathbf{EI}(y_i)$ for $y_i \in \mathcal{Q}$. Note that $\mathcal{Q}$ is subject to the constraint from Eqn. 8; this ensures **BAF** operates within the bounds of $\mathcal{H}$ under ALEBO's random projections to lower-dimensional space, $\mathbf{R}^d$.

These points and their corresponding q**EI** metric values $(\mathcal{Q}', \text{q}\mathbf{EI}(\mathcal{Q}'))$ are then passed as input to **DMS**.

Dimension Matched Sampler (DMS)

DMS's task is to output one sample $h' = \mathbf{B}^\dagger y'$ that best matches the dimensional information of the minimizer $h^*[j], j \in \mathcal{L}$ obtained from Q_d type queries. This h' will be the sample at which the Q_f filter query is made to update the GPR posterior. Clearly, the **DMS** algorithm must operate in high-dimensional space $\mathcal{H}$ as both h' and $h^* \in \mathcal{H}$. In contrast, the **BAF** module operates in the $d-$dimensional embedding space $\mathbf{R}^d$, hence the q candidates lie in this embedding space. To remedy this, the **BAF**'s outputs in the embedding space are projected up to $\mathcal{H}$, i.e., $\mathcal{Q}' = \{y'_1, y'_2, \ldots, y'_q\} \to \mathcal{T}' = \{h'_1, h'_2, \ldots, h'_q\}$ as shown in Figure 2(a) (the green box labeled $\mathbf{B}^\dagger$). Figure 2(b) illustrates the translation of any point from high-dimensional space $\mathcal{H}$ to the low-dimensional embedding space $\mathcal{Y}$ and vice versa. Since **DMS** utilizes **BAF**'s q**EI** metric, the q chosen candidate samples $\mathcal{Q}'$ adhere to the box constraints in Eqn. 8. This ensures that the q candidates in high-dimensional space $\mathcal{T}'$ lie inside $\mathcal{H}$, thereby avoiding any non-linearity due to clipping.

Once $\mathcal{Q}'$ has been translated to higher dimensional $\mathcal{T}'$, **DMS** uses a joint *likelihood* measure to preferentially order the q samples based on their degree of similarity to the L dimension queries $h^*[j], j \in \mathcal{L}$ as shown in Eqn. 11. Given

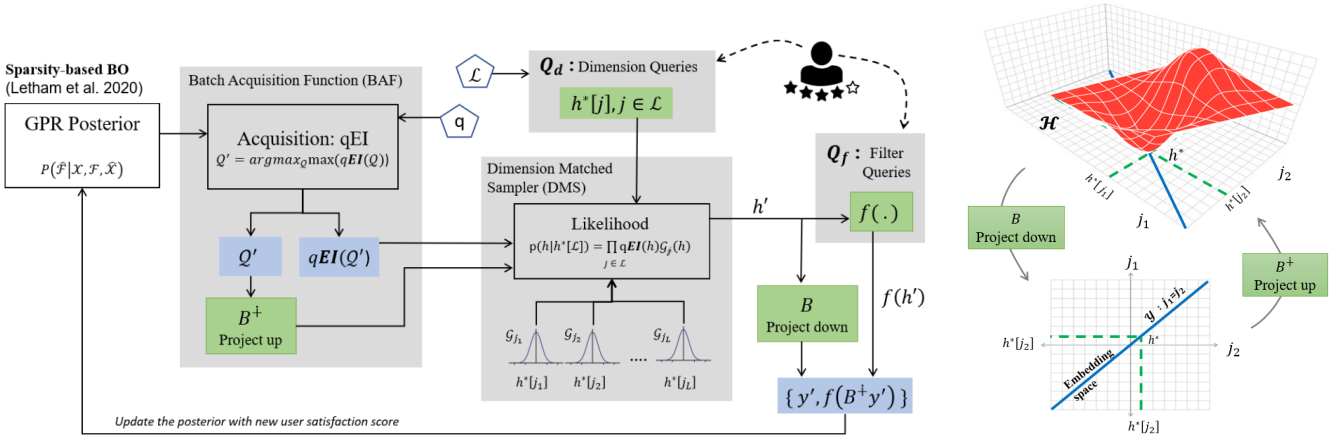

Figure 2: System flow: ORACLEBO consists of three modules: **BAF**, **DMS**, and GPR posterior. Green boxes denote the system inputs and hyper-parameters, and blue marks the module outputs. The right figure shows the transformation between the high and low dimensional spaces, made feasible by the random embedding matrix in ALEBO.

$h^*[j], j \in \mathcal{L}$ queries, **DMS**' joint likelihood measure computes the likelihood of a filter sample h maximally improving the GPR posterior when observed.

$$P(h|h^*[\mathcal{L}]) = \prod_{j \in \mathcal{L}} \text{qEI}(h)\mathcal{G}_j(h)$$
$$h' = \underset{h \in \mathcal{T}'}{\text{argmax}}\, P(h|h^*[\mathcal{L}]) \tag{11}$$

where, $\mathcal{G}_j = \mathcal{N}(\mu = h^*[j], \sigma)$ is a Gaussian distribution with mean $h^*[j]$ and variance σ for each Q_d dimension available $j \in \mathcal{L}$, and qEI(h'_i) is the **BAF** acquisition metric of a candidate sample h'_i.

The maximizer in Eqn. 11, h', is the sample at which the user is queried. This h' and the user's satisfaction score $f(h')$ are then projected down to the embedding as $(y', f(\mathbf{B}^\dagger y'))$. Finally, this tuple is used to update the GPR posterior for the next iteration of ORACLEBO.

Experiment: Synthetic BlackBox Functions

We first present experiments on various synthetic functions $f(h)$. We pretend the function is a black-box, but filter and dimension queries are feasible. Obviously, because we actually know the function, we will evaluate how close ORACLEBO can get to the global minima.

We evaluate two sets of objective functions:
(1) Staircase Satisfaction Functions that are shaped like a staircase (see Figure 3) and roughly mimic how humans rate their experiences in discrete steps (Al-Roomi 2015).
(2) Benchmark Functions commonly used in Bayesian optimization research (Sonja Surjanovic 2013), such as `BRANIN`, `HARTMANN6`, and `ROSENBROCK`.

Baseline and Metrics: We consider a baseline that extends ALEBO with the additional information from L dimensional queries. This implies that ALEBO's search space can be reduced from $\mathcal{R}^N$ to $\mathcal{R}^{N-L}$. Our evaluation metric is *Regret*, which is the difference between the predicted minimum and true global minimum $(f(\hat{h}^*) - f(h^*))$. All reported results are an average of 10 different runs. More details on the objective functions and evaluation parameters are included in the Appendix.

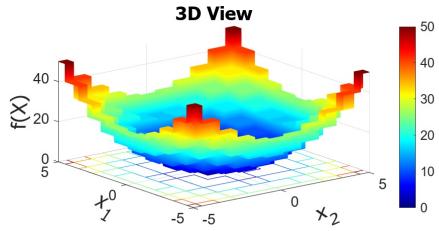

Figure 3: Satisfaction Function $P1$: Discontinuous staircase structure, containing infinite zero gradient regions.

In the following figures, X-axis label "function evaluations" indicates the number of filter queries (Q_f), each of which produces a GPR iteration. Also, L denotes the number of dimension queries (Q_d). For comparison, we mark points on the graph that use the same query budget, $B = Q_f + Q_d$[1].

Comparison to ALEBO(L): Figure 4 shows the performance of ORACLEBO against ALEBO(L). ALEBO($L = 0$) shows the weakest performance because it does not benefit from dimensional queries. With 5 dimensional queries, ALEBO($L = 5$) shows immediate gain since it has to only search $\mathbf{R}^{N-5}$. ORACLEBO shows further improvement with $L = 5$, implying that the combination of Q_f and Q_d queries are beneficial, even though the search space is $\mathcal{R}^N$. Observe that points marked with stars all have the same query budget $B = 90$, thus, ORACLEBO achieves high satisfaction (low regret) for a given B. Of course, if B is too small, say < 30, then the gains reduce. This is understand-

[1]For readability, we abuse the notation Q_d, which is equal to the number of dimension queries, L.

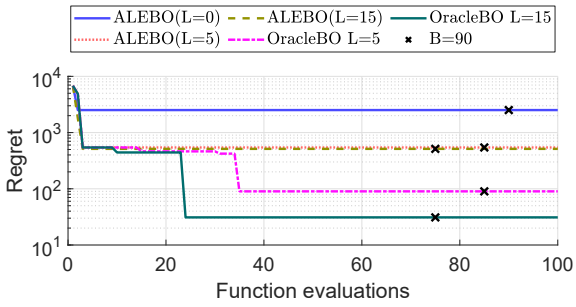

Figure 4: Performance on ALEBO(L) and ORACLEBO.

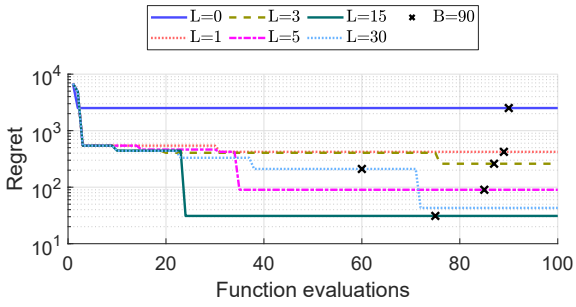

Figure 5: Different number of Q_d queries on ORACLEBO.

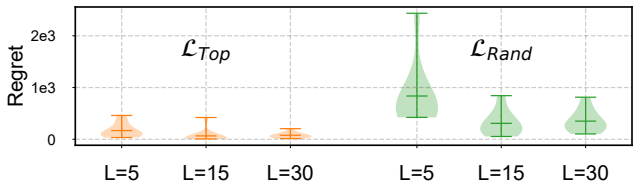

Figure 6: Distribution of ORACLEBO's regret on different Q_d subsets $\mathcal{L}_{Top}$ and $\mathcal{L}_{Rand}$.

N	d	(q, σ)				
		(5,1)	(2,0.2)	(2,10)	(7,0.2)	(7,10)
500	4	**83**	445	316	1459	1459
	10	148	166	237	760	883
	20	477	551	609	1201	1255
2000	4	**90**	242	514	543	628
	10	2166	2331	2753	2331	2753
	20	2677	2764	3125	2764	3125
4000	4	513	606	1268	1268	1268
	10	1532	1627	2154	4006	5278
	20	1749	1993	3332	10213	10213

Table 1: Hyperparameter analysis on regret for $P1$.

able because the GPR posterior has not yet converged. Finally, when L=15, the regret is even lower.

Effect of Varying L: Figure 5 reports the impact of increasing dimension queries, L, on regret. Observe that for a fixed query budget $B = 90$, increasing L is beneficial but only up to $L = 15$. Increasing L further offers more information about the optimal h^* but at the expense of lowering the number of Q_f queries. Evidently, for the staircase function, the empirical optimal for L is in the neighborhood of 15.

Which L out of N queries? Given $L = 15$ queries, say, different subsets of N dimensions can be chosen. Let us denote this subset as $\mathcal{L}$. If $f(h)$ hardly varies along the dimensions included in $\mathcal{L}$, then $\mathcal{L}$ contributes little to estimating the satisfaction function. Figure 6 shows ORACLEBO's regret on two different $\mathcal{L}$. Note that because the objective function f is synthesized, the variation of f against any dimension y is known. In $\mathcal{L}_{Top}$, we select the L dimensions of largest variances; $\mathcal{L}_{Rand}$ denotes the randomly selected dimensions from $\{1, N\}$. Results show that $\mathcal{L}_{Top}$ achieves lower regret (median and variance) compared to $\mathcal{L}_{Rand}$. Thus, in real applications, it helps to choose L dimensions that are likely to influence the user's satisfaction

Hyperparameter Selection: Parameters in **BAF** and **DMS** modules include: N, dimension of the filter; d, dimension of the embedding; q, number of candidates BAF outputs, and σ, the dimensional variance in **DMS**.

Experiments: Audio Personalization

This section reports experiments with real volunteers in the context of personalizing hearing aids. Today's hearing aids aim to filter the audio with h so that the user's hearing loss is compensated, and their satisfaction $f(h)$ is maximized. Hearing aids prescriptions exactly perform the process of dimension querying where different frequency tones j are played to the user, and their optimal audibility is recorded as $h^*[j]$. To minimize user burden, audio clinics play around L=7 frequency tones (from different octaves) and interpolate through them to generate the user's personalized filter. This filter is called the *audiogram*.

Interpolation is obviously a coarse approximation of the user's true personal filter, h^*. We expect to improve the user's satisfaction over their *audiogram*, using a modest number of Q_f queries prescribed by ORACLEBO. In other words, the user can attain higher satisfaction $\hat{f}^*$ if they are willing to listen and rate some audio clips (Q_f) at home.

For experimentation, we invited 3 volunteers with no hearing loss. To emulate hearing loss, we played audio that was deliberately "`corrupted`" with hearing loss profiles from the public hearing-loss dataset in NHANES (Salmon et al. 2022). This corrupted audio obviously yields a poor satisfaction score from our volunteers. We compute the coarse-grained audiogram for the volunteers and compare ORACLEBO against this "`baseline`".

Results: Figure 7(a) plots the final satisfaction score from the 3 volunteers (U1, U2, U3), first for the "Audiogram" experiment, and then for a "Random" filter experiment (to be described soon). The `Corrupted` signal obviously receives a low score, but the interpolated audiogram, labeled `Baseline`, considerably improves the score. ORACLEBO is still able to match/improve user satisfaction with $Q_d = 5$ and $Q_f = 25$ queries. With fewer Q_f of 5 and 15, ORACLEBO could not outperform `Baseline` as the function

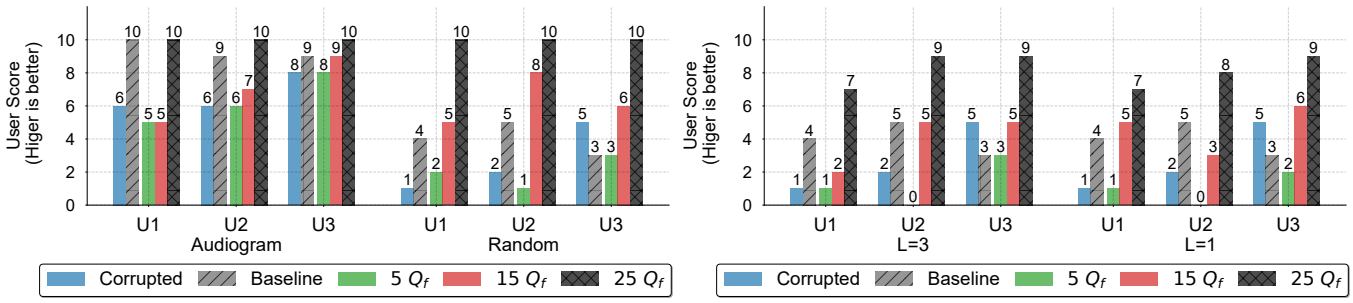

Figure 7: User score comparison on (a) $L = 5$ on hearing-loss profile and random profile. (b) $L = [1, 3]$ on random profile.

f's search space $\mathcal{R}^{4000}$ had not been sufficiently sampled.

The audiogram `Baseline` performs quite well primarily because human hearing loss is reasonably flat within octaves, hence, interpolation is adequate. We thus explore another application that injects more complex audio distortions, e.g., a cheap music speaker.

We again emulate this distortion by deliberately corrupting the audio with a random filter h, each $h[j]$ selected independently from $[-30, 30]$dB. Similar interpolation as an audiogram, using $L = 5$, will give us a new `Baseline`. Fig 7(a)-Random plots the results for the same 3 users. ORACLEBO improves the satisfaction scores even with $Q_f = 15$ queries, and achieves the maximum with $Q_f = 20$ queries.

We also investigate the degradation of user satisfaction with fewer dimensional queries $L = 1, 3$. Figure 7(b) reports the results for only the Random distortion filter. Evidently, the degradation is graceful, i.e., as L reduces, more Q_f filter queries are needed to achieve the same level of personal satisfaction. The audio demos at various stages of the optimization are made available[2].

Related Work

To the best of our knowledge, ORACLEBO is the first work that combines two different types of queries, Q_f and Q_d, for Bayesian Optimization. We also believe such hybrid querying has not been applied in audio personalization. The closest work in the application context is (Solnik et al. 2017) where authors used conventional BO to search for the best-rated cookie recipe at Google. We believe more applications can benefit, and the burden of querying users can become practical with dimension querying (Q_d). Of course, BO has been extensively used to solve complex problems that do not have humans in the loop. These include material science (Ueno et al. 2016), medicine (Negoescu, Frazier, and Powell 2011), hyperparameter tuning in neural networks (Snoek, Larochelle, and Adams 2012), etc.

ORACLEBO inherits sparsity-based BO frameworks based on low-dimensional embeddings. These papers use linear random projections to map from high to low-dimensional spaces (Qian, Hu, and Yu 2016)(Wang et al. 2016)(Binois, Ginsbourger, and Roustant 2020)(Letham et al. 2020). Authors of (Garnett, Osborne, and Hennig 2013)(Lu et al. 2018) use a Gaussian Process to simultaneously learn the model and the embedding. Non-linear embeddings are learnt using Variation Autoencoders in (Gómez-Bombarelli et al. 2018)(Moriconi, Deisenroth, and Sesh Kumar 2020)(Lu et al. 2018). ORACLEBO is agnostic to these algorithms and we expect their advantages to reflect in our performance as well. On a similar note, various papers modify the kernel (Kandasamy, Schneider, and Póczos 2015)(Gardner et al. 2017)(Mutny and Krause 2018)(Wang et al. 2018) to restrict the candidate function choices (Oh, Gavves, and Welling 2018), reflecting additional structure in the objective function. LineBO (Kirschner et al. 2019) optimizes the acquisition function along one-dimensional lines. TuRBO (Eriksson et al. 2019) employs trust regions around the current minimizer. (Oh et al. 2019), (Eriksson and Jankowiak 2021) use sparsity creating prior. The performance of ORACLEBO can be boosted with such kernel manipulations.

Follow-up Work and Conclusion

This paper adds a new type of querying to black-box optimization where an Oracle can reveal information about the minimizer, h^*. This type of dimension querying maps to practical applications and solutions that achieve a low query budget B can be useful. This paper is a first step in this direction, starting with an empirical treatment of the problem. We believe the findings are promising and open doors to follow-up work. For instance, (1) an analytical treatment on convergence is needed for the hybrid $Q_f + Q_d$ querying. (2) If j can be freely chosen in $h^*[j]$, how many and which j's are optimal, given the query budget B? (3) What other applications lend themselves to the notion of hybrid querying? (4) Dimension queries may not independent, i.e., a user may like sugar for a certain recipe and may not like sugar for another recipe. How can such conditionals be incorporated in the formulation? We hope ORACLEBO serves as a stepping stone to solving important problems along these directions.

Acknowledgements

We thank R. Srikant for valuable guidance at the start of the project, and the reviewers for their insightful feedback. We are also grateful to Foxconn and NSF (grant 2008338, 1909568, and MRI-2018966) for partially funding this research.

[2]https://oraclebo.github.io/

References

Al-Roomi, A. R. 2015. Unconstrained Single-Objective Benchmark Functions Repository.

Binois, M.; Ginsbourger, D.; and Roustant, O. 2020. On the choice of the low-dimensional domain for global optimization via random embeddings. *Journal of global optimization*, 76: 69–90.

Eriksson, D.; and Jankowiak, M. 2021. High-dimensional Bayesian optimization with sparse axis-aligned subspaces. In *Uncertainty in Artificial Intelligence*, 493–503. PMLR.

Eriksson, D.; Pearce, M.; Gardner, J.; Turner, R. D.; and Poloczek, M. 2019. Scalable global optimization via local Bayesian optimization. *Advances in neural information processing systems*, 32.

Frazier, P. I. 2018. A tutorial on Bayesian optimization. *arXiv preprint arXiv:1807.02811*.

Gardner, J.; Guo, C.; Weinberger, K.; Garnett, R.; and Grosse, R. 2017. Discovering and exploiting additive structure for Bayesian optimization. In *Artificial Intelligence and Statistics*, 1311–1319. PMLR.

Garnett, R.; Osborne, M. A.; and Hennig, P. 2013. Active learning of linear embeddings for Gaussian processes. *arXiv preprint arXiv:1310.6740*.

Gómez-Bombarelli, R.; Wei, J. N.; Duvenaud, D.; Hernández-Lobato, J. M.; Sánchez-Lengeling, B.; Sheberla, D.; Aguilera-Iparraguirre, J.; Hirzel, T. D.; Adams, R. P.; and Aspuru-Guzik, A. 2018. Automatic chemical design using a data-driven continuous representation of molecules. *ACS central science*, 4(2): 268–276.

Kandasamy, K.; Schneider, J.; and Póczos, B. 2015. High dimensional Bayesian optimisation and bandits via additive models. In *International conference on machine learning*, 295–304. PMLR.

Kirschner, J.; Mutny, M.; Hiller, N.; Ischebeck, R.; and Krause, A. 2019. Adaptive and safe Bayesian optimization in high dimensions via one-dimensional subspaces. In *International Conference on Machine Learning*, 3429–3438. PMLR.

Letham, B.; Calandra, R.; Rai, A.; and Bakshy, E. 2020. Re-examining linear embeddings for high-dimensional Bayesian optimization. *Advances in neural information processing systems*, 33: 1546–1558.

Lu, X.; Gonzalez, J.; Dai, Z.; and Lawrence, N. D. 2018. Structured variationally auto-encoded optimization. In *International conference on machine learning*, 3267–3275. PMLR.

Moriconi, R.; Deisenroth, M. P.; and Sesh Kumar, K. 2020. High-dimensional Bayesian optimization using low-dimensional feature spaces. *Machine Learning*, 109: 1925–1943.

Mutny, M.; and Krause, A. 2018. Efficient high dimensional bayesian optimization with additivity and quadrature fourier features. *Advances in Neural Information Processing Systems*, 31.

Neal, R. M. 2003. Slice sampling. *The annals of statistics*, 31(3): 705–767.

Negoescu, D. M.; Frazier, P. I.; and Powell, W. B. 2011. The knowledge-gradient algorithm for sequencing experiments in drug discovery. *INFORMS Journal on Computing*, 23(3): 346–363.

Oh, C.; Gavves, E.; and Welling, M. 2018. BOCK: Bayesian optimization with cylindrical kernels. In *International Conference on Machine Learning*, 3868–3877. PMLR.

Oh, C.; Tomczak, J.; Gavves, E.; and Welling, M. 2019. Combinatorial bayesian optimization using the graph cartesian product. *Advances in Neural Information Processing Systems*, 32.

Qian, H.; Hu, Y.-Q.; and Yu, Y. 2016. Derivative-Free Optimization of High-Dimensional Non-Convex Functions by Sequential Random Embeddings. In *IJCAI*, 1946–1952.

Salmon, M. K.; Brant, J.; Hohman, M. H.; and Leibowitz, D. 2022. Audiogram Interpretation.

Snoek, J.; Larochelle, H.; and Adams, R. P. 2012. Practical bayesian optimization of machine learning algorithms. *Advances in neural information processing systems*, 25.

Solnik, B.; Golovin, D.; Kochanski, G.; Karro, J. E.; Moitra, S.; and Sculley, D. 2017. Bayesian optimization for a better dessert.

Sonja Surjanovic, D. B. 2013. Virtual Library of Optimization Functions.

Ueno, T.; Rhone, T. D.; Hou, Z.; Mizoguchi, T.; and Tsuda, K. 2016. COMBO: An efficient Bayesian optimization library for materials science. *Materials discovery*, 4: 18–21.

Wang, J. 2020. An intuitive tutorial to Gaussian processes regression. *arXiv preprint arXiv:2009.10862*.

Wang, Z.; Gehring, C.; Kohli, P.; and Jegelka, S. 2018. Batched large-scale Bayesian optimization in high-dimensional spaces. In *International Conference on Artificial Intelligence and Statistics*, 745–754. PMLR.

Wang, Z.; Hutter, F.; Zoghi, M.; Matheson, D.; and De Feitas, N. 2016. Bayesian optimization in a billion dimensions via random embeddings. *Journal of Artificial Intelligence Research*, 55: 361–387.