High-dimensional Bayesian Optimization via Condensing-Expansion Projection

Anonymous authors
Paper under double-blind review

Abstract

In high-dimensional settings, Bayesian optimization (BO) can be expensive and infeasible. The random embedding Bayesian optimization algorithm is commonly used to address high-dimensional BO challenges. However, this method relies on the effective subspace assumption on the optimization problem's objective function, which limits its applicability. In this paper, we introduce Condensing-Expansion Projection Bayesian optimization (CEPBO), a novel random projection-based approach for high-dimensional BO that does not reply on the effective subspace assumption. The approach is both simple to implement and highly practical. We present two algorithms based on different random projection matrices: the Gaussian projection matrix and the hashing projection matrix. Experimental results demonstrate that both algorithms outperform existing random embedding-based algorithms in most cases, achieving superior performance on high-dimensional BO problems. The code is available in `https://anonymous.4open.science/r/CEPBO-14429`.

1 Introduction

For many optimization problems, the objective function f lacks a closed-form expression, and gradient information is often unavailable, leading to what we are generally referred to as black-box functions Jones et al. (1998); Snoek et al. (2012); Shahriari et al. (2015). Bayesian optimization (BO) is an efficient method for solving such optimization problems by modeling the unknown objective function through a probabilistic surrogate model, typically a Gaussian Process. The BO routine is a sequential search algorithm where each iteration involves estimating the surrogate model from available data and then maximizing an acquisition function to determine which point should be evaluated next. As the input space dimension D increases, typically $D \geq 10$, BO encounters the so-called 'curse of dimensionality' Bellman (1966). This phenomenon refers to the exponential increase in difficulty resulting from higher query complexity and the computational cost associated with calculating the acquisition function.

To address the issue, numerous studies have proposed high-dimensional BO algorithms (Wang et al., 2016; Chen et al., 2012; Binois et al., 2015; 2020; Nayebi et al., 2019; Letham et al., 2020b) that typically translate high-dimensional optimizations into low-dimensional ones by various techniques, and search the new point in the low-dimensional space. However, these approaches can become inefficient when the maximum over the high-dimensional space cannot be well approximated by the maximum over the low-dimensional space.

In this paper, we introduce a novel search strategy in high-dimensional BO problems called the Condense-Expansion Projection (CEP) technique, which is both simple to implement and highly practical. In each iteration of the sequential search, the CEP technique generates a random projection matrix $\mathbf{A} \in \mathbb{R}^{d\times D}$, where $d \ll D$, to project the available data from the high-dimensional space to the low-dimensional embedding space by multiplying them with $\mathbf{A}$. It estimates the surrogate model and searches for the next point to evaluation in the low-dimensional embedding space. Subsequently, it projects the searched data point back to the high-dimensional space by multiplying it with $\mathbf{A}^\top$ for evaluation in the original space.

We utilize two distinct random projection matrices: the Gaussian projection matrix Dasgupta & Gupta (2002) and the hashing projection matrix Rokhlin & Tygert (2008); Boutsidis & Gittens (2013), within the CEP technique, resulting in the development of two algorithms, CEP-REMBO and CEP-HeSBO. Our experimental results, comprising comprehensive simulation studies and analysis of four real-world datasets, demonstrate that both algorithms generally outperform existing random embedding-based algorithms, showcasting the superior performance of the CEP technique on high-dimensional BO problems.

2 Related Work

There is a substantial body of literature on high-dimensional BO.The most closely related approach is REMBO Wang et al. (2016) by fitting a Gaussian Process model in a low-dimensional embedding space obtained through a Gaussian random projection matrix. This approach has been further investigated by Binois et al. (2015; 2020); Binois (2015); Letham et al. (2020b) under various conditions. Nayebi et al. (2019) proposed HeSBO that utilizes a hashing projection matrix. However, these studies are based on the assumption of an effective subspace, where a small number of parameters have a significant impact on the objective function. Similar to these studies, our approach evaluates the acquisition function over the embedding space. However, unlike these studies, our approach select the new point in the original space, implying that our approach does not relies on the effective space assumption. The second distinguishing aspect of our approach is that it generates a new random projection matrix in each iteration, thereby increasing the flexibility to accommodate the possibility that the global optimum may not be located in a single embedding space.

Besides the embedding approach, two other techniques warrant consideration: one based on the additive form of the objective function, and the other based on the dropout approach. Kandasamy et al. (2015) tackled the challenges in high-dimensional BO by assuming an additive structure for the function. Other works along the line include GPs with an additive kernel M. & Krause (2018); Wang et al. (2017a) or cylindrical kernels Oh et al. (2018). However, this approach is limited by its reliance on the assumption of the additive form of the objective function. Li et al. (2017) applied the dropout technique into high-dimensional BO to alleviate reliance on assumptions regarding limited “active” features or the additive form of the objective function. This method randomly selects subset of dimensions and optimizes variables only from these chosen dimensions via Bayesian optimization. However, it necessitates “filling-in" the variables from the left-out dimensions. The proposed “fill-in" strategy, which involves copying the value of variables from the best function value, may lead to being trapped in a local optimum, although the strategy is enhanced by mixing random values.

3 Method

3.1 Bayesian Optimization

We consider the optimization problem

$$\mathbf{x}^* = \arg\max\nolimits_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x}),$$

where f is a black-box function and $\mathcal{X} \subset \mathbb{R}^D$ is some bounded set. BO is a form of sequential model-based optimization, where we fit a surrogate model, typically a Gaussian Process (GP) model, for f that is used to identify which parameters $\mathbf{x}$ should be evaluated next. The GP surrogate model is $f \sim \mathcal{GP}\left(m(\cdot), k(\cdot,\cdot)\right)$, with a mean function $m(\cdot)$ and a kernel $k(\cdot,\cdot)$. Under the GP prior, the posterior for the value of $f(\mathbf{x})$ at any point in the space is a normal distribution with closed-form mean and variance. Using that posterior, we construct an acquisition function $\alpha(\mathbf{x})$ that specifies the utility of evaluating f at $\mathbf{x}$, such as Expected Improvement Jones et al. (1998). We find $\mathbf{x}_{\text{new}} = \arg\max_{\mathbf{x}\in\mathcal{X}} \alpha(\mathbf{x})$, and in the next iteration evaluate $f(\mathbf{x}_{\text{new}})$. However, GPs are known to predict poorly for large dimension D Wang et al. (2016), which prevents the use of standard BO in high dimensions.

3.2 Condensing-Expansion Projection

The framework of Condensing-Expansion Projection (CEP) is delineated through two essential projection procedures.

- **Condensing Projection:** transpose points from the original space into a reduced-dimensional embedding subspace, where the surrogate model is fitted from available data and the acquisition function is maximized to determine which point should be evaluated next.
- **Expansion Projection:** revert these points in the embedding subspace back to the original space, where the searched point is evaluated.

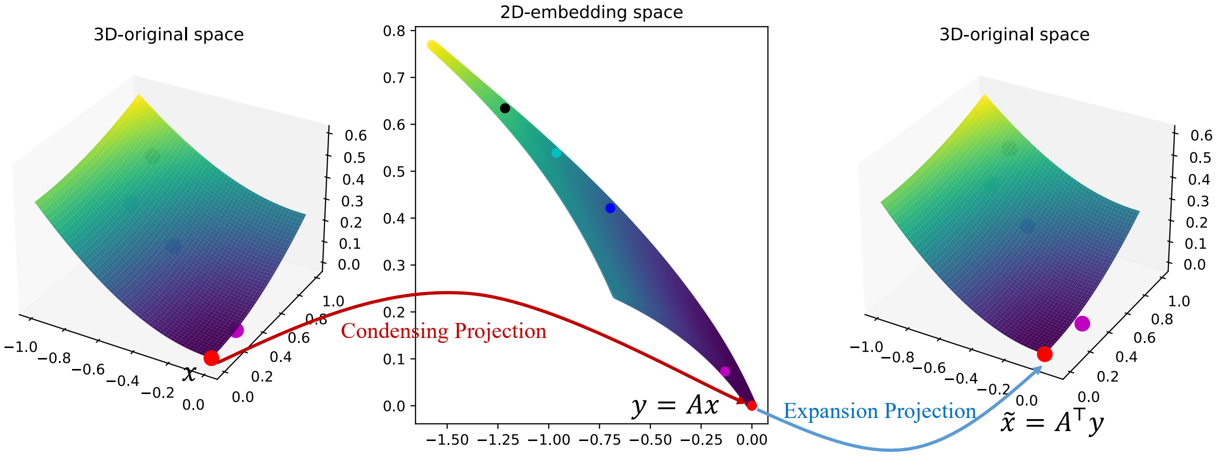

Figure 1: An illustration of CEP. The dimension of the original space is $D = 3$, and the dimension of the embedding subspace is $d = 2$. The five points in the orginal space is projected to the embedding subspace by Condensing Projection, then they are projected back to the original space by Expansion Projection. The optimal point (red dot) in the original space is still at the (approximately) optimal position after CEP.

Let us define an embedding subspace $\mathcal{Y} \subset \mathbb{R}^d$ of dimension d. We generate a random projection matrix $\mathbf{A} \in \mathbb{R}^{d \times D}$. Various methodologies exist for the construction of such a matrix, including the Gaussian random matrix, sparse random matrix Dasgupta (2000); Bingham & Mannila (2001), and the Subsampled Randomized Hadamard Transform TROPP (2011). In this paper, we utilize the Gaussian random matrix and the Hashing matrix.

Consider a point $\mathbf{x} \in \mathcal{X}$ within the original space $\mathcal{X}$. In the condensing projection, we project $\mathbf{x}$ from the original space $\mathcal{X}$ to the embedding subspace $\mathcal{Y}$ by multiplying $\mathbf{x}$ with the matrix $\mathbf{A}$, resulting in

$$\mathbf{y} = \mathbf{A}\mathbf{x} \in \mathcal{Y},$$

thereby reducing the dimension from D in the original space to d in the embedding subspace. In the expansion projection, we project $\mathbf{y}$ back to $\mathcal{X}$ by multiplying $\mathbf{y}$ with the transposed matrix $\mathbf{A}^\top$, expressed as as

$$\tilde{\mathbf{x}} = \mathbf{A}^\top \mathbf{y} = \mathbf{A}^\top \mathbf{A}\mathbf{x}.$$

This completes the Condensing-Expansion Projection, which can be outlined as follows: transforming a point from the original space to the embedding subspace and then restoring it back to the original space, represented as

$$\mathbf{x} \to \mathbf{y} \to \tilde{\mathbf{x}}.$$

We outline an illustration in Figure1.

The CEP offers flexibility in selecting random projection matrix $\mathbf{A}$. In this paper, we focus on two types: Gaussian random matrices and Hashing random matrices.

Definition 1. *(Gaussian Random Matrix) Let* $\mathbf{A} \in \mathbb{R}^{d \times D}$ *be a random matrix with independent Gaussian entries. For any* $1 \leq i \leq d$ *and* $1 \leq j \leq D$*, the element* a_{ij} *defined as*

$$a_{ij} \sim \mathcal{N}\left(0, \frac{1}{d}\right).$$

Definition 2. *(Hashing Random Matrix) Let* $\mathbf{A} \in \mathbb{R}^{d \times D}$ *be a hashing random matrix. Specifically, (1) Each column of* $\mathbf{A}$ *has a single non-zero element that is selected at random. (2) This non-zero element has an equal probability* $p = 0.5$ *of being either* $+1$ *or* -1.

Assume a matrix $\mathbf{A}$ satisfying Definition 1 or Definition 2, we have

$$\mathbb{E}\left[\mathbf{A}^\top \mathbf{A}\right] = \mathbf{I}_D. \tag{1}$$

The proof of (1) is provided in the appendix 6.1. (1) represents the isometry in expectation, suggesting that the process of two projections, on average, preserves both the magnitude and direction of $\mathbf{x}$, thereby making the gap from $\tilde{\mathbf{x}}$ to $\mathbf{x}$ in the original space is zero on average.

Besides of the isometry in expectation, we also concern the concentration of $\tilde{\mathbf{x}}$ around the original point $\mathbf{x}$ in terms of the function f. We measure the concentration by the difference between $\mathbf{x}^\top \tilde{\mathbf{x}}$ round $\mathbf{x}^\top \mathbf{x}$, which presents how much $\tilde{\mathbf{x}} - \mathbf{x}$ projects onto $\mathbf{x}$. Assume a matrix $\mathbf{A}$ satisfying Definition 1, we have

$$\mathbb{E}\left[(\mathbf{x}^\top \mathbf{A}^\top \mathbf{A}\mathbf{x} - \mathbf{x}^\top \mathbf{x})^2\right] \leq \frac{2}{d}\|\mathbf{x}\|^4. \tag{2}$$

Assume a matrix $\mathbf{A}$ satisfying Definition 2, we have

$$\mathbb{E}\left[(\mathbf{x}^\top \mathbf{A}^\top \mathbf{A}\mathbf{x} - \mathbf{x}^\top \mathbf{x})^2\right] \leq \frac{1}{d}\left(\|\mathbf{x}\|^4 - \sum_{i=1}^{D} x_i^4\right). \tag{3}$$

The proofs of (2) & (3) are provided in the appendix 6.2 & 6.3, respectively. They state that for the variance of $\mathbf{x}^\top \tilde{\mathbf{x}}$ with respect to $\mathbf{x}^\top \mathbf{x}$ diminishes as the dimension d of the embedding subspace increases. Hence, CEP can maintain the function in the original space well when d is not small.

3.3 The CEPBO Algorithm

We employ Condensing-Expansion Projection in Bayesian Optimization, leading to the development of the Condensing-Expansion Projection Bayesian Optimization (CEPBO) algorithm. In contrast to Random Embedding algorithms, sucha as REMBOWang et al. (2016), HeSBONayebi et al. (2019) and ALEBOLetham et al. (2020b), where a fixed projection matrix is employed, the CEPBO algorithm dynamically generates a new projection matrix $\mathbf{A}_t$ during each iteration t. Through Condensing Projection, which condenses available points from the original space to a new embedding subspace via multiplication with $\mathbf{A}_t$, CEPBO leverages past information to conduct BO within the embedding subspace. It then determines which point to evaluate next within this subspace. Afterward, the selected point in the embedding subspace undergoes Expansion Projection, where it is projected back to the original space via multiplication with $\mathbf{A}_t^\top$. Subsequently, the objective function is evaluated at that point. This approach bypasses the stringent assumption associated with effective dimension by facilitating the projection of a new embedding subspace with each search iteration. This adaptability acknowledges the possibility that the global optimum may not be confined to a single search.

The detailed procedural flow of the algorithm is outlined in Algorithm1. By using different random projection matrices at line 4 of the Algorithm1, we derive two algorithms: CEP-REMBO and CEP-HeSBO. These can be regarded as enhanced versions of REMBO and HeSBO, respectively.

Condense original space into the embedding subspace. The core concept of employing Condensing Projection entails the creation of a new subspace $\mathcal{Y}$ with each iteration, where BO is subsequently performed. However, as the historical trajectories remains preserved within the original space $\mathcal{X}$, the newly formed subspace must be equipped with the requisite information to enable effective BO. To tackle this issue, the primary objective of Condensing Projection is to transfer the historical trajectories from the original space $\mathcal{X}$ into an embedding subspace $\mathcal{Y}$, thereby furnishing the embedding subspace $\mathcal{Y}$ with the necessary information to facilitate BO. At the current iteration t, let $\mathcal{D}_{t-1}$ represent the trajectories in the original space $\mathcal{X}$, given by: $\mathcal{D}_{t-1} = \{(\mathbf{x}_1, f(\mathbf{x}_1)), (\mathbf{x}_2, f(\mathbf{x}_2)), \ldots, (\mathbf{x}_{t-1}, f(\mathbf{x}_{t-1}))\}$. During this iteration, a new projection matrix $\mathbf{A}_t$ of

Algorithm 1: CEPBO Algorithm

```
Input: Objective $f : \mathcal{X} \to \mathbb{R}$; Acquisition criterion $\alpha$; Original dimension $D$; Embedding dimension $d$;
       Initial points $t_0$; Evaluation trials $t_N$
Output: Best point $\mathbf{x} \in \arg\max_{\mathcal{X}} f(\mathbf{x})$
Uniformly sample $t_0$ points $\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_{t_0}\}$ in the original space;
Define $\mathcal{D}_0 = \{(\mathbf{x}_1, f(\mathbf{x}_1)), (\mathbf{x}_2, f(\mathbf{x}_2)), \ldots, (\mathbf{x}_{t_0}, f(\mathbf{x}_{t_0}))\}$;
while $t_0 + 1 \leq t \leq t_N$ do
    Construct the projection matrix $\mathbf{A}_t \in \mathbb{R}^{d \times D}$ according to Gaussian projection matrix in the Definition
      1 or hashing projection matrix in the Definition 2;
    Project the points in $\mathcal{D}_{t-1}$ onto the embedding subspace $\mathcal{Y}_t$ via $\mathbf{A}_t$, obtaining the set of points in the
      embedding subspace $\mathcal{D}^y_{t-1} = \{(\mathbf{A}_t\mathbf{x}_1, f(\mathbf{x}_1)), (\mathbf{A}_t\mathbf{x}_2, f(\mathbf{x}_2)), \cdots, (\mathbf{A}_t\mathbf{x}_{t-1}, f(\mathbf{x}_{t-1}))\}$;
    Estimate the hyperparameters $\theta_t$ of the Gaussian Process prior for the given $\mathcal{D}^y_{t-1}$;
    Calculate the posterior probability of the Gaussian Process based on $\mathcal{D}^y_{t-1}$ and the estimated
      hyperparameters $\theta_t$.
    Compute the maximum of the acquisition criterion $\alpha$, $\mathbf{y_t} \in \arg\max_{\mathbf{y} \in \mathcal{Y}} \alpha(\mathbf{y} \mid \mathcal{D}^y_{t-1})$;
    Project $\mathbf{y}_t$ back to the original space via $\mathbf{A}_t^\top$, obtaining $\mathbf{x}_t = \mathbf{A}_t^\top \mathbf{y_t}$;
    Update the dataset $\mathcal{D}_t = \mathcal{D}_{t-1} \cup \{(\mathbf{x}_t, f(\mathbf{x}_t))\}$, and $t = t + 1$.
end
```

dimensions $\mathbb{R}^{d \times D}$ is sampled. This matrix serves as projecting the historical point from the original space $\mathcal{X}$ into a newly formed embedding subspace $\mathcal{Y}_t$: $\mathcal{D}^y_{t-1} = \{(\mathbf{A}_t\mathbf{x}_1, f(\mathbf{x}_1)), (\mathbf{A}_t\mathbf{x}_2, f(\mathbf{x}_2)), \cdots, (\mathbf{A}_t\mathbf{x}_{t-1}, f(\mathbf{x}_{t-1}))\}$.

Optimize over the embedding subspace. The objective f is fitted by a Gaussian Process model over the embedding subspace $\mathcal{Y}_t$:

$$f(\mathbf{x}) \approx \mathcal{GP}\left(m(\mathbf{A}_t\mathbf{x}), k(\mathbf{A}_t\mathbf{x}, \mathbf{A}_t\mathbf{x})\right).$$

We demonstrate that the optimization over the embedding subspace closely approximates that over the original space. Since the mean in the Gaussian process is typically constant, we primarily need to consider the covariance matrix. We show that $k(\mathbf{A}_t\mathbf{x}_1, \mathbf{A}_t\mathbf{x}_2) = (1 + O_p(d^{-1/2}))k(\mathbf{x}_1, \mathbf{x}_2)$. Details are provided in the appendix 6.4. Hence, the Gaussian process fit in the embedding space converges to the fit in the original space. Within the embedding subspace $\mathcal{Y}_t$, the dataset $\mathcal{D}^y_{t-1}$ informs the estimation of the hyperparameters θ_t for the Gaussian process, and the posterior probability of the Gaussian process is computed. The acquisition function α (such as Expected Improvement) identifies the embedding subspace's optimal point $\mathbf{y}_t^*$ within the embedding subspace, which is represented by the equation $\mathbf{y}_t^* = \arg\max_{\mathbf{y} \in \mathcal{Y}_t} \alpha(\mathbf{y} \mid \mathcal{D}^y_{t-1})$.

Project back and evaluate in the original space. After searching the optimal point with the acquisition function, we need to project this point back to the original space $\mathcal{X}$ for objective function evaluation. Subsequently, we add this point to the historical trajectories. To be more specific, we use the transpose projection matrix $\mathbf{A}_t$ to map the optimal point $\mathbf{y}_t^*$ from the embedding subspace back to $\mathcal{X}$ by applying its transpose $\mathbf{A}_t^\top$, expressed as: $\tilde{\mathbf{x}}_t^* = \mathbf{A}_t^\top \mathbf{y}_t^*$. As demonstrated in Section 3.2, $\tilde{\mathbf{x}}_t^*$ concentrates to $\mathbf{x}_t^*$. Subsequently, we evaluate the objective function at $\tilde{\mathbf{x}}_t^*$ within $\mathcal{X}$ to obtain $f(\tilde{\mathbf{x}}_t^*)$. This data, denoted as $(\tilde{\mathbf{x}}_t^*, f(\tilde{\mathbf{x}}_t^*))$, is then added to the historical trajectories $\mathcal{D}_{t-1}$, resulting in: $\mathcal{D}_t = \mathcal{D}_{t-1} \cup \{(\tilde{\mathbf{x}}_t^*, f(\tilde{\mathbf{x}}_t^*))\}$. This completes a full iteration cycle of the CEPBO algorithm.

3.4 Address the boundary issue

Our approach, akin to REMBO, encounters the issue of excessive exploration along the boundary of $\mathcal{X}$. To ensure the effective tuning of the acquisition function and to facilitate BO, it is crucial for the embedding subspace $\mathcal{Y}$ to have a bounded domain. However, random projections between the original space of dimension D and the embedding subspace of dimension d can lead to points exceeding domain boundaries after CEP. These exceedances occur in two scenarios: $\mathbf{y} = \mathbf{A}\mathbf{x} \notin \mathcal{Y}$, $\tilde{\mathbf{x}} = \mathbf{A}^\top \mathbf{y} \notin \mathcal{X}$.

To mitigate the issue, we employ the convex projection of the original space, $P_{\mathcal{X}}$, and that of the embedding subspace, $P_{\mathcal{Y}}$, to rectify boundary transgressions. Specifically, the convex projection within the original space $\mathcal{X}$ is defined as follows:

$$P_{\mathcal{X}} : \mathbb{R}^D \to \mathbb{R}^D,\ P_{\mathcal{X}}(\tilde{\mathbf{x}}) = \underset{\mathbf{z}\in\mathcal{X}}{\arg\min}\|\mathbf{z} - \tilde{\mathbf{x}}\|_2.$$

Similarly, the convex projection within the embedding subspace $\mathcal{Y}$ is expressed as:

$$P_{\mathcal{Y}} : \mathbb{R}^d \to \mathbb{R}^d,\ P_{\mathcal{Y}}(\mathbf{y}) = \underset{\mathbf{z}\in\mathcal{Y}}{\arg\min}\|\mathbf{z} - \mathbf{y}\|_2.$$

Convex projection will lead to an issue where multiple distinctive values in the original space are mapped to identical boundary points within the embedding subspace, i.e., for $\mathbf{x}_1 \neq \mathbf{x}_2$ such that $f(\mathbf{x}_1) \neq f(\mathbf{x}_2)$, the equality $P_{\mathcal{Y}}(\mathbf{A}\mathbf{x}_1) = P_{\mathcal{Y}}(\mathbf{A}\mathbf{x}_2)$ holds. Moreover, the substantial disparity between the dimensions d and D exacerbates the likelihood of such instances. This issue can undermine the precision of Gaussian process models and consequently, diminish the efficacy of optimization. To mitigate this risk, a scaling strategy is implemented within the Condensing Projection and Expansion Projection phases to diminish the probability of such occurrences. This involves scaling the projected points $\mathbf{Ax}$ using a reduction factor before applying convex projection, as follows:

$$\mathbf{y} = P_{\mathcal{Y}}\left(\frac{1}{\sqrt{D}}\mathbf{A}\mathbf{x}\right).$$

In a parallel procedure, the optimal points of the acquisition function in the embedding subspace $\mathbf{y}$ undergo an inverse scaling:

$$\tilde{\mathbf{x}} = P_{\mathcal{X}}(\sqrt{D}\mathbf{A}^\top\mathbf{y}).$$

In this context, the scaling factors $\frac{1}{\sqrt{D}}$ and $\sqrt{D}$ confine the scope of projection within the viable domain. These factors are verified through empirical experimentation.

4 Experiments

We conduct experiments to demonstrate the performance of the proposed method across various functions and real-world scenarios. In Section 4.1, we evaluate its performance on three benchmark functions. In Section 4.2, we assess it across four real-world problems. These experimental results indicate that our algorithms, CEP-REMBO and CEP-HeSBO, achieve superior results.

Because our approach, CEPBO, represents an advancement in the domain of the linear embeddings, our experiments focus on comparing it with other linear embeddings. We aim to assess the improvement achieved by applying CEP compared to REMBOWang et al. (2016) and HeSBONayebi et al. (2019), respectively. ALEBO is considered to achieve state-of-the-art performance on this class of optimization problems with a true linear subspace. Therefore, we chose REMBOWang et al. (2016), HeSBONayebi et al. (2019), and ALEBOLetham et al. (2020b) as the benchmark algorithms for our comparisons.

4.1 Numerical Results

We evaluated the performance of the algorithms using the following benchmark functions: (1) the Holder Table function, (2) the Schwefel function, and (3) the Griewank function. Each function's input space was extended to a dimensionality of $D = 100$. The Holder Table function has an effective dimension of 2, whereas the Schwefel and Griewank functions are fully defined over the entire D-dimensional space. The goal is to find the minimum value of these functions. The number of initialization trials for each algorithm was kept the same as the dimensionality of its embedding subspace. Each experiment is independently repeated 50 times, with 50 evaluations per experiment. To assess the performance of the CEPBO algorithm under various embedding space dimensions, we take $d = 2, 5, 10$ in the Holder Table function and $d = 2, 5, 20$ in the Schwefel and Griewank functions. Since an effective dimension for the Schwefel and Griewank functions is 100, we prioritize the larger d for assessment. In these experiments, we utilize expected improvement as the acquisition function.

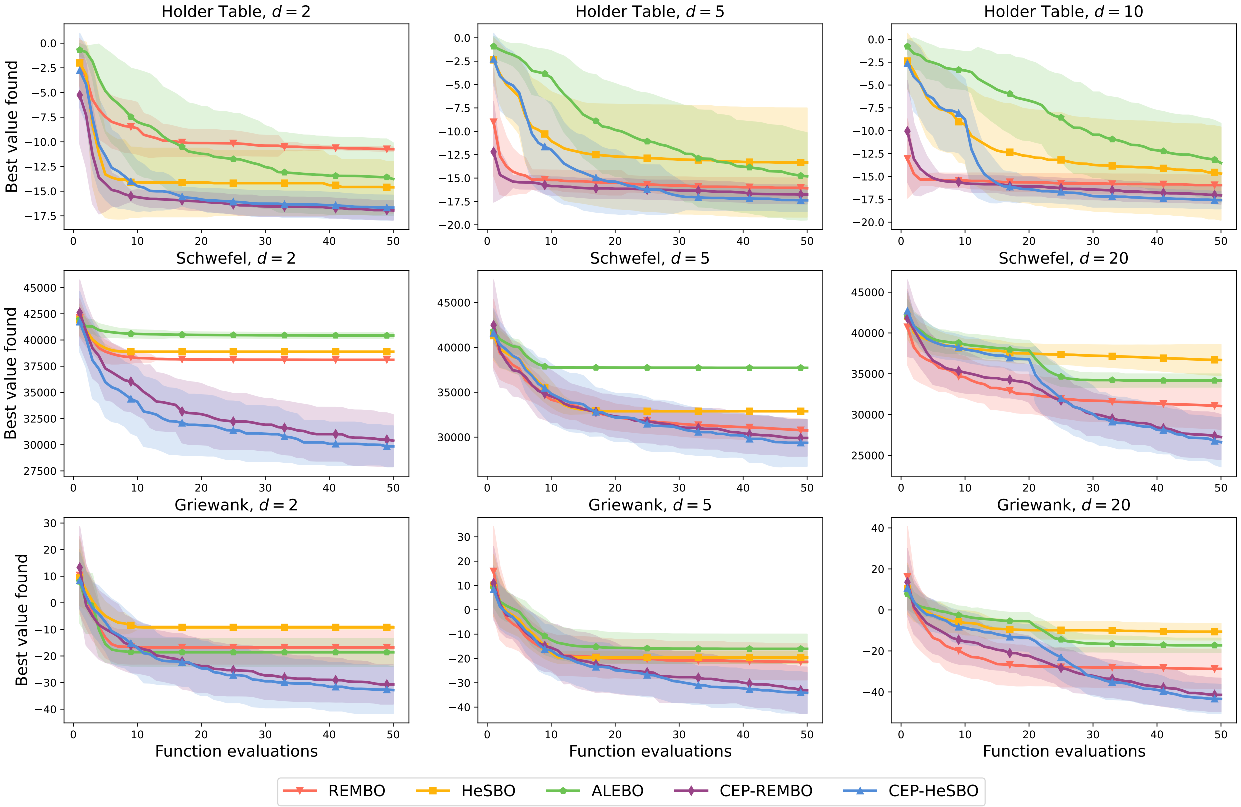

Figure 2: The results of the optimization experiments for three functions across various embedding dimensions. From top to bottom: Holder Table, Schwefel, and Griewank functions.

We report the results in Figure 2. For the Schwefel and Griewank functions, where embedding dimensions are smaller than the effective dimension of 100, the baseline algorithms nearly ceased functioning, settling in local optima, which is visually depicted as a flat horizontal line on the corresponding graphs. Interestingly, even in the instance of the Holder Table function, where the embedding dimension met or exceeded the effective dimension, a circumstance where the baseline algorithms typically perform well, the approached algorithms continued to show superior performance over all baselines. Comparative the performance across a range of embedding dimensions d, the performance are similar and CEP-REMBO and CEP-HeSBO consistently surpassed all baseline algorithms. Therefore, the REMBO and HeSBO algorithms experienced substantial improvement with the integration of the Condense-Expansion Projection mechanism.

Impact of higher dimension D.

To assess the performance of the CEPBO algorithm in higher dimensions, we conducted simulations using the well-known Hartmann function. Specifically, we utilized the Hartmann function with an original space dimension of $D = 6$ and set the embedded space dimension to $d = 6$ as well. To simulate a high-dimensional environment, we expanded the original space from $D = 6$ to $D = 1000$, but in practice, only the 6-dimensional data is valid.

The results are reported on the left in Figure 3. In this setup, ALEBO, REMBO, and HeSBO were identified as the best-performing configurations. The experimental results demonstrate that our proposed algorithm still maintains a certain level of superiority, with the CEP-REMBO algorithm being the optimal one. Additionally, the results indicate that incorporating the CEP projection mechanism can significantly improve the performance of both REMBO and HeSBO algorithms.

To realistically simulate the optimization performance of the CEPBO algorithm on an actual 1000-dimensional function, we still use the settings from Section 4.1. However, we employ an Griewank function with an effective dimension of 1000, where all 1000 dimensions have a tangible impact on the function results.

As shown on the middle in Figure 3, other algorithms, apart from CEPBO, quickly fail and become trapped in local optima. This indicates that even in extremely high dimensions, once the effective dimension of the space exceeds the embedding dimension, non-CEPBO algorithms struggle to perform. However, our projection can effectively alleviate this issue, allowing for continuous searching for optimal solutions even when using a very small embedding space.

Robustness to noisy rewards.

To assess the performance of the CEPBO algorithm in a noisy setting, we conducted simulations using the well-known Holder Table function. Specifically, the function settings were the same as those outlined in Section 4.1. Furthermore, during the iterations of the Bayesian algorithm, we introduced a random normal distribution noise disturbance to the reward function, where $\epsilon \sim N(0, 1)$, to simulate noise in real-world environments.

The results are illustrated on the right in Figure 3. The experimental findings indicate that our proposed algorithm retains a significant advantage, with the CEP-HeSBO algorithm demonstrating optimal performance. Additionally, the results suggest that incorporating the CEP projection mechanism can substantially enhance the performance of the REMBO and HeSBO algorithms.

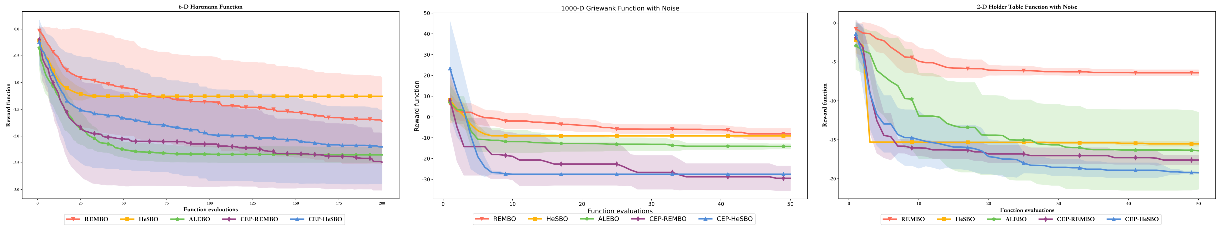

Figure 3: From left to right: the 1000-D Hartmann function, 1000-D Griewank function, and the Noisy Holder Table function.

4.2 Real-World Problems

In this section, we evaluate the CEPBO algorithm on real-world optimization problems. The test cases consist of lunar landing task in the realm of reinforcement learning with $D = 12$ Eriksson et al. (2019), a robot pushing problem with $D = 14$ Wang et al. (2017b), a problem in neural architecture search with $D = 36$ Letham et al. (2020a), and a rover trajectory planning problem with $D = 60$ Wang et al. (2018). Algorithmic configurations and acquisition function selections strictly adhere to the settings outlined in the original papers. For additional details, please refer to the appendix. The optimization goal is to maximize the reward function, and each experiment is independently repeated 10 times, with 500 evaluations per experiment.

Lunar Landing. This experiment entails the task of devising a reinforcement learning strategy for the lunar lander's control mechanism, aiming to minimize fuel consumption and proximity to the landing site while preventing a crash. The original space dimension is $D = 12$. In the first column of Figure 4, REMBO, HeSBO, and ALEBO algorithms become trapped in local optima to different extents. As the dimensionality of the embedding subspace d increases from $d = 2$ to $d = 5$, notable performance improvements are observed for most algorithms, except REMBO. The introduced CEP-REMBO and CEP-HeSBO approaches consistently demonstrate the ability to enhance and identify novel optimal resolutions.

Robot Pushing. This scenario involves a robotics dual-arm manipulation task where the robot's arms are controlled by adjusting 14 modifiable parameters to push two objects while tracking their movement trajectories. The original space dimension is $D = 14$. The second column of Figure 4 demonstrates that the proposed CEP-REMBO and CEP-HeSBO methods significantly outperform others when $d = 2$. When increasing to $d = 5$, all methods exhibit varying degrees of performance enhancement. This suggests that

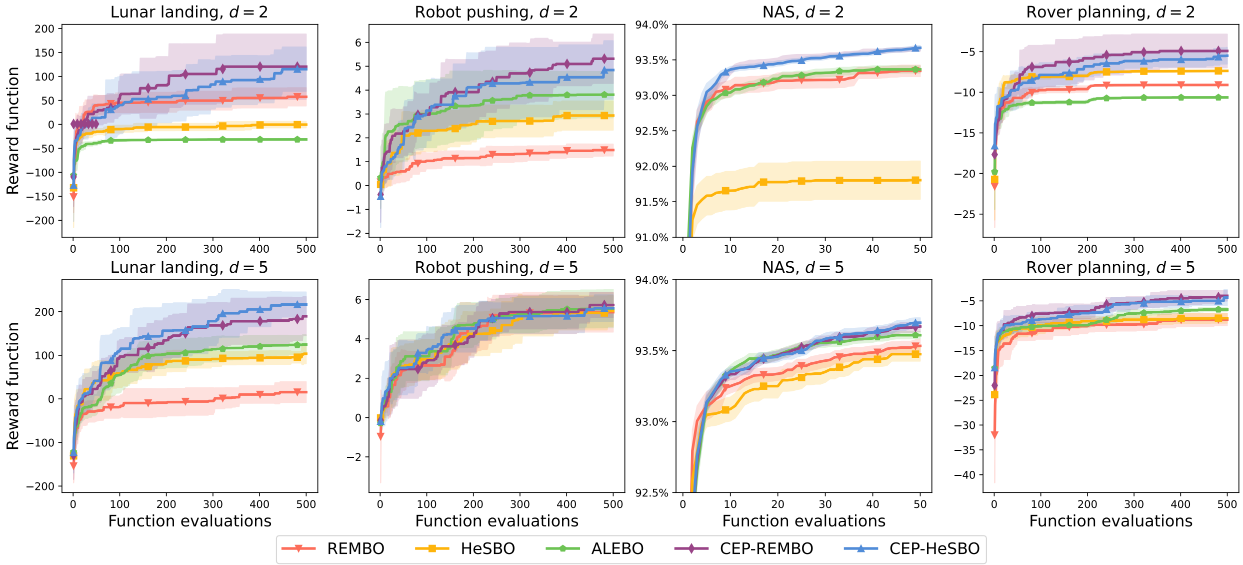

Figure 4: The results of the optimization experiments for four real-world scenarios. From left to right: Lunar landing, Robot pushing, NAS, and Rover planning.

for optimization issues with moderate to low dimensionalities, escalating the dimensions of the embedding subspaces can notably bolster the algorithms' efficacy. Notwithstanding these improvements, the CEP-REMBO and CEP-HeSBO methods consistently maintain their leading positions.

Neural Architecture Search (NAS). The objective of this experiment is to identify an optimal architecture for neural networks, paralleling the methodology utilized by Letham et al. (2020b). Leveraging data from the NAS-Bench-101 benchmark Ying et al. (2019), we have developed an optimization problem focused on searching for a convolutional neural network architecture characterized by 36 dimensions. The original space dimension is $D = 36$. In the third column of Figure 4, at an embedding subspace dimension of $d = 2$, the REMBO, HeSBO, and ALEBO algorithms rapidly converge to less than ideal solutions, hindering the exploration of superior neural network structures. On the contrary, the CEP-REMBO and CEP-HeSBO methods maintain the capability to persistently optimize, discovering architectures with improved accuracy. Increasing the subspace dimension to $d = 5$ reveals the ALEBO's enhanced ability to perform on par with CEP-REMBO and CEP-HeSBO methods; however, CEP-HeSBO consistently exhibits the highest performance across all conditions.

Rover Trajectory Planning. This task involves optimizing a 2D trajectory comprising of 30 pivotal points that collectively define a navigational path. The original space dimension is $D = 60$. The fourth column of Figure 4 indicates that for an embedding subspace dimension of $d = 2$, the REMBO, HeSBO, and ALEBO algorithms can not successfully converge to an advantageous reward. In contrast, the CEP-REMBO and CEPHeSBO algorithms exhibit a capacity to consistently identify superior solutions. This pattern is similarly observed when the subspace dimension is increased to $d = 5$.

5 Conclusion

This paper proposes a Bayesian optimization framework utilizing the Condensing-Expansion Projection technique, free from reliance on the assumption of effective dimension. The primary concept involves employing projection twice within each iteration: first, projecting to an embedding subspace, and then projecting back to retain optimization information in the original high-dimensional space. The approach does not impose additional requirements on the projection matrix used, thereby significantly enhancing the applicability of the embedding-based Bayesian optimization algorithms. This flexibility enables the selection of a suitable projection matrix according to the problem's characteristics beforehand. Two new Bayesian optimization algorithms based on Condensing-Expansion Projection are proposed: CEP-REMBO and CEP-HeSBO based on the Gaussian projection matrix and the hash-enhanced projection matrix, respectively. Empirically, this paper conducts comprehensive experiments to assess the performance of the proposed algorithms across diverse optimization scenarios. The experimental results demonstrate that the Bayesian optimization algorithms based on Condensing-Expansion Projection achieved promising performance across these optimization functions, overcoming the reliance on effective dimension.

For previous embedding-based Bayesian algorithms, achieving an optimal solution requires d to be greater than or equal to the true effective dimension of the optimization problem. In contrast, our algorithm does not have this requirement and performs robustly across different choices of d. In practice, d can be viewed as a hyperparameter to be set. When selecting d, it is crucial to balance the choice of a value that is not too small with considerations of computational costs. Empirically, setting $d = 5$ is typically effective.

Our work has several limitations that can be addressed by future works. For instance, while our analysis supports the concentration of the transformed point around the original point after employing CEP, we acknowledge the absence of ϵ-subspace embedding to preserve the variance function of a Gaussian process. Although reference to Nayebi et al. (2019) suggests a potential avenue for such analysis, a more thorough examination is warranted. Another constraint of our study is the absence of an evaluation of CEP-based algorithms for optimization problems with billions of dimensions. Despite this potential, our approach lacks empirical validation, whereas REMBO has been shown to effectively address such challenges.

References

Eytan Bakshy, Lili Dworkin, Brian Karrer, Konstantin Kashin, Benjamin Letham, Ashwin Murthy, and Shaun Singh. Ae: A domain-agnostic platform for adaptive experimentation. In *Advances in neural information processing systems*, pp. 1–8, 2018.

Maximilian Balandat, Brian Karrer, Daniel R. Jiang, Samuel Daulton, Benjamin Letham, Andrew Gordon Wilson, and Eytan Bakshy. Botorch: A framework for efficient monte-carlo bayesian optimization. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

Richard Bellman. Dynamic programming. *Science*, 153(3731):34–37, 1966.

Ella Bingham and Heikki Mannila. Random projection in dimensionality reduction: applications to image and text data. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 245–250, 2001.

Mickaël Binois. *Uncertainty quantification on Pareto fronts and high-dimensional strategies in Bayesian optimization, with applications in multi-objective automotive design.* PhD thesis, Ecole Nationale Supérieure des Mines de Saint-Etienne, 2015.

Mickaël Binois, David Ginsbourger, and Olivier Roustant. A warped kernel improving robustness in bayesian optimization via random embeddings. In *Learning and Intelligent Optimization: 9th International Conference, LION 9, Lille, France, January 12-15, 2015. Revised Selected Papers 9*, pp. 281–286. Springer, 2015.

Mickaël Binois, David Ginsbourger, and Olivier Roustant. On the choice of the low-dimensional domain for global optimization via random embeddings. *Journal of Global Optimization*, 76:69–90, 2020.

Christos Boutsidis and Alex Gittens. Improved matrix algorithms via the subsampled randomized hadamard transform. *SIAM Journal on Matrix Analysis and Applications*, 34(3):1301–1340, 2013.

Erin Catto. Box2d: a 2d physics engine for games, 2011. URL `http://box2d.org`.

Bo Chen, Rui M. Castro, and Andreas Krause. Joint optimization and variable selection of high-dimensional gaussian processes. In *Proceedings of the 29th International Coference on International Conference on Machine Learning*, pp. 1379–1386, Madison, WI, USA, 2012. Omnipress. ISBN 9781450312851.

Sanjoy Dasgupta. Experiments with random projection. In *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence*, pp. 143–151, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc. ISBN 1558607099.

Sanjoy Dasgupta and Anupam Gupta. An elementary proof of a theorem of johnson and lindenstrauss. *Random Structures & Algorithms*, 22(1):60–65, 2002.

David Eriksson, Michael Pearce, Jacob Gardner, Ryan D Turner, and Matthias Poloczek. Scalable global optimization via local bayesian optimization. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Donald R Jones, Matthias Schonlau, and William J Welch. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13:455–492, 1998.

Kirthevasan Kandasamy, Jeff Schneider, and Barnabás Póczos. High dimensional bayesian optimisation and bandits via additive models. In *International Conference on Machine Learning*, pp. 295–304. PMLR, 2015.

Ben Letham, Roberto Calandra, Akshara Rai, and Eytan Bakshy. Re-examining linear embeddings for high-dimensional bayesian optimization. In *Advances in Neural Information Processing Systems*, volume 33, pp. 1546–1558, 2020a.

Benjamin Letham, Roberto Calandra, Akshara Rai, and Eytan Bakshy. Re-examining linear embeddings for high-dimensional Bayesian optimization. In *Advances in Neural Information Processing Systems*, volume 33, 2020b.

Cheng Li, Sunil Gupta, Santu Rana, Vu Nguyen, Svetha Venkatesh, and Alistair Shilton. High dimensional bayesian optimization using dropout. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pp. 2096–2102. AAAI Press, 2017. ISBN 9780999241103.

Mutný M. and A. Krause. Efficient high dimensional bayesian optimization with additivity and quadrature fourier features. In *Advances in Neural Information Processing Systems*, pp. 9005–9016, 2018.

Amin Nayebi, Alexander Munteanu, and Matthias Poloczek. A framework for bayesian optimization in embedded subspaces. In *International Conference on Machine Learning*, pp. 4752–4761. PMLR, 2019.

C. Oh, E. Gavves, and M. M. Welling. Bock : Bayesian optimization with cylindrical kernels. In *Proceedings of the 35th International Conference on Machine Learning*, pp. 3868–3877, 2018.

Vladimir Rokhlin and Mark Tygert. A fast randomized algorithm for overdetermined linear least-squares regression. *PNAS*, 105(36):13212–13217, 2008.

Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando De Freitas. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2015.

Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems*, volume 25, 2012.

JOEL A. TROPP. Improved analysis of the subsampled randomized hadamard transform. *Advances in Adaptive Data Analysis*, 3(1n02):115–126, 2011. ISSN 1793-5369.

Z. Wang, C. Li, S. Jegelka, and P. Kohli. Batched high-dimensional bayesian optimization via structural kernel learning. In *Proceedings of the 34th International Conference on Machine Learning*, pp. 3656–3664, 2017a.

Zi Wang, Chengtao Li, Stefanie Jegelka, and Pushmeet Kohli. Batched high-dimensional bayesian optimization via structural kernel learning. In *International Conference on Machine Learning*, pp. 3656–3664. PMLR, 2017b.

Zi Wang, Clement Gehring, Pushmeet Kohli, and Stefanie Jegelka. Batched large-scale bayesian optimization in high-dimensional spaces. In *International Conference on Artificial Intelligence and Statistics*, pp. 745–754. PMLR, 2018.

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

Chris Ying, Aaron Klein, Eric Christiansen, Esteban Real, Kevin Murphy, and Frank Hutter. Nas-bench-101: Towards reproducible neural architecture search. In *International Conference on Machine Learning*, pp. 7105–7114. PMLR, 2019.

6 Appendix

6.1 Proof of (1)

Proof. (1) First, we consider the Gaussian Projection. Let $\mathbf{A} = (\boldsymbol{l}_1, \cdots, \boldsymbol{l}_d)^\top$, such that $\boldsymbol{l}_i = (\alpha_{i1}, \cdots, \alpha_{iD})^\top$ is a $D \times 1$ vector where each element is from $\mathcal{N}\left(0, \frac{1}{d}\right)$ distribution. It is easy to show that, given any vector $\mathbf{x} \in \mathcal{X}$ and any vector $\mathbf{g} \in \mathbb{R}^D$,

$$\mathbb{E}(\mathbf{A}^\top \mathbf{A}) = \sum_{i=1}^{d} \mathbb{E}\left(\boldsymbol{l}_i \boldsymbol{l}_i^\top\right) = \mathbf{I}.$$

(2) Second, we consider the Hashing Random Projection. For the Hashing Random Matrix, we rewrite $\mathbf{A} = \mathbf{SD}$, where $\mathbf{S} \in \mathbb{R}^{d\times D}$ has each column chosen independently and uniformly from the r standard basis vectors of $\mathbb{R}^d$ and $\mathbf{D} \in \mathbb{R}^{D\times D}$ is a diagonal matrix with diagonal entries chosen independently and uniformly on $b_i \in \{\pm 1\}$.

Let $\mathbf{S} = (\boldsymbol{s}_1, \cdots, \boldsymbol{s}_D)$, such that $\boldsymbol{s}_i$ is a random vector taking the vector e_j for equal probability, where e_k is the kth standard unit vector of $\mathbb{R}^d$ for $k = 1, \cdots, d$. Then $\mathbb{E}(\boldsymbol{s}_i) = \frac{1}{d}\mathbf{1}_d$ and $\mathbb{E}(\boldsymbol{s}_i^\top \boldsymbol{s}_i) = 1$, which follow that $\mathbb{E}[(\mathbf{S}^\top\mathbf{S})_{ij}] = \mathbb{E}(\boldsymbol{s}_i^\top \boldsymbol{s}_j) = \mathbb{E}(\boldsymbol{s}_i)^\top \mathbb{E}(\boldsymbol{s}_j) = \frac{1}{d^2}$ for $i \neq j$; and $\mathbb{E}[(\mathbf{S}^\top\mathbf{S})_{ii}] = \mathbb{E}(\boldsymbol{s}_i^\top \boldsymbol{s}_i) = 1$. Obviously,

$$\mathbb{E}(\mathbf{A}^\top \mathbf{A}) = \mathbf{I}.$$

□

6.2 Proof of (2)

Proof. Let $\mathbf{A} = (\boldsymbol{l}_1, \cdots, \boldsymbol{l}_d)^\top$, such that $\boldsymbol{l}_i = (\alpha_{i1}, \cdots, \alpha_{iD})^\top$ is a $D \times 1$ vector where each element is from $\mathcal{N}\left(0, \frac{1}{d}\right)$ distribution. (1) follows that

$$\mathbb{E}[(\mathbf{x}^\top\mathbf{A}^\top\mathbf{A}\mathbf{x} - \mathbf{x}^\top\mathbf{x})^2] = \mathbb{E}[(\mathbf{x}^\top\mathbf{A}^\top\mathbf{A}\mathbf{x})^2] - (\mathbf{x}^\top\mathbf{x})^2. \tag{4}$$

In (4), we have,

$$\begin{aligned}\mathbb{E}\left[(\mathbf{x}^\top\mathbf{A}^\top\mathbf{A}\mathbf{x})^2\right] &= \mathbb{E}\Big[(\sum_{i=1}^{d}\mathbf{x}^\top\boldsymbol{l}_i\boldsymbol{l}_i^\top\mathbf{x})^2\Big] \\ =&\mathbb{E}\Big[\sum_{i=1}^{d}(\mathbf{x}^\top\boldsymbol{l}_i\boldsymbol{l}_i^\top\mathbf{x})^2 + \sum_{i\neq j}(\mathbf{x}^\top\boldsymbol{l}_i\boldsymbol{l}_i^\top\mathbf{x})(\mathbf{x}^\top\boldsymbol{l}_j\boldsymbol{l}_j^\top\mathbf{x})\Big] \\ =&d\mathbb{E}[(\mathbf{x}^\top\boldsymbol{l}_i\boldsymbol{l}_i^\top\mathbf{x})^2] + (d^2-d)\Big[\mathbb{E}(\mathbf{x}^\top\boldsymbol{l}_i\boldsymbol{l}_i^\top\mathbf{x})\Big]^2.\end{aligned} \tag{5}$$

By (8) in Lemma 1,

$$\mathbb{E}[(\mathbf{x}^\top\boldsymbol{l}_i\boldsymbol{l}_i^\top\mathbf{x})^2] = \frac{3}{d^2}\|\mathbf{x}\|^4.$$

Then we have

$$\mathbb{E}\left[(\mathbf{x}^\top\mathbf{A}^\top\mathbf{A}\mathbf{x})^2\right] = (\mathbf{x}\top\mathbf{x})^2 + \frac{2}{d}\|\mathbf{x}\|^4. \tag{6}$$

Therefore, we have

$$\mathbb{E}\left[(\mathbf{x}^\top\mathbf{A}^\top\mathbf{A}\mathbf{x} - \mathbf{x}^\top\mathbf{x})^2\right] = \frac{2}{d}\|\mathbf{x}\|^4.$$

6.3 Proof of (3)

For the Hashing Random Matrix, we rewrite $\mathbf{A} = \mathbf{SD}$, where $\mathbf{S} \in \mathbb{R}^{d\times D}$ has each column chosen independently and uniformly from the r standard basis vectors of $\mathbb{R}^d$ and $\mathbf{D} \in \mathbb{R}^{D\times D}$ is a diagonal matrix with diagonal entries chosen independently and uniformly on $b_i \in \{\pm 1\}$.

Let $\mathbf{S} = (\boldsymbol{s}_1, \cdots, \boldsymbol{s}_D)$, such that $\boldsymbol{s}_i$ is a random vector taking the vector e_j for equal probability, where e_k is the kth standard unit vector of $\mathbb{R}^d$ for $k = 1, \cdots, d$. Then $\mathbb{E}(\boldsymbol{s}_i) = \frac{1}{d}\mathbf{1}_d$ and $\mathbb{E}(\boldsymbol{s}_i^\top \boldsymbol{s}_i) = 1$, which follow that $\mathbb{E}[(\mathbf{S}^\top\mathbf{S})_{ij}] = \mathbb{E}(\boldsymbol{s}_i^\top \boldsymbol{s}_j) = \mathbb{E}(\boldsymbol{s}_i)^\top\mathbb{E}(\boldsymbol{s}_j) = \frac{1}{d^2}$ for $i \neq j$; and $\mathbb{E}[(\mathbf{S}^\top\mathbf{S})_{ii}] = \mathbb{E}(\boldsymbol{s}_i^\top \boldsymbol{s}_i) = 1$.

Applying Lemma 2,

$$\mathbb{E}\left[(\mathbf{x}^\top\mathbf{A}^\top\mathbf{A}\mathbf{x})^2\right] = (\mathbf{x}^\top\mathbf{x})^2 + \frac{1}{d}\left(\|\mathbf{x}\|^4 - \sum_{i=1}^{D} x_i^4\right).$$

Thus, we have,

$$\mathbb{E}\left[(\mathbf{x}^\top\mathbf{A}^\top\mathbf{A}\mathbf{x} - \mathbf{x}^\top\mathbf{x})^2\right] = \frac{1}{d}\left(\|\mathbf{x}\|^4 - \sum_{i=1}^{D} x_i^4\right).$$

□

6.4 Consistency of Gaussian Process Fit in the embedding space

For simplifying, we focus on the squared exponential kernel

$$K_{SE}(\mathbf{x}_1, \mathbf{x}_2) = \theta_0 \exp\{-2^{-1} r^2(\mathbf{x}_1, \mathbf{x}_2)\},$$

where

$$r^2(\mathbf{x}_1, \mathbf{x}_2) = \sum_{j=1}^{D} (x_{1j} - x_{2j})^2/\theta_j^2.$$

Since θ_j can be absorbed into $x_{1j} - x_{2j}$, without loss of generality, we simplify to consider

$$r^2(\mathbf{x}_1, \mathbf{x}_2) = (\mathbf{x}_1 - \mathbf{x}_2)^\top(\mathbf{x}_1 - \mathbf{x}_2).$$

In the embedding space, the corresponding kernel is given by

$$r^2(\tilde{\mathbf{x}}_1, \tilde{\mathbf{x}}_2) = (\mathbf{x}_1 - \mathbf{x}_2)^\top\mathbf{A}^\top\mathbf{A}(\mathbf{x}_1 - \mathbf{x}_2).$$

According to (2) and (3), we have

$$\mathbb{E}[(r^2(\tilde{\mathbf{x}}_1, \tilde{\mathbf{x}}_2) - r^2(\mathbf{x}_1, \mathbf{x}_2))^2] \leq \frac{2}{d}[r^2(\mathbf{x}_1, \mathbf{x}_2)]^2.$$

By applying the Markov inequality, we obtain that there exists $\epsilon > \sqrt{2}$ such that

$$\begin{aligned} P\left(|r^2(\tilde{\mathbf{x}}_1, \tilde{\mathbf{x}}_2) - r^2(\mathbf{x}_1, \mathbf{x}_2)| \leq \epsilon d^{-1/2} r^2(\mathbf{x}_1, \mathbf{x}_2)\right) \geq& 1 - \frac{2}{d}\frac{[r^2(\mathbf{x}_1, \mathbf{x}_2)]^2}{[\epsilon d^{-1/2} r^2(\mathbf{x}_1, \mathbf{x}_2)]^2} \\ =& 1 - \frac{2}{\epsilon^2}. \end{aligned}$$

Therefore, $r^2(\tilde{\mathbf{x}}_1, \tilde{\mathbf{x}}_2) = (1 + O_p(d^{-1/2}))r^2(\mathbf{x}_1, \mathbf{x}_2)$. It follows

$$\kappa(\mathbf{A}\mathbf{x}_1, \mathbf{A}\mathbf{x}_2) = (1 + O_p(d^{-1/2}))\kappa(\mathbf{x}_1, \mathbf{x}_2).$$

6.4.1 Two Lemmas

Lemma 1. *Let $\mathbf{A} = (\boldsymbol{l}_1, \cdots, \boldsymbol{l}_d)^\top$, such that $\boldsymbol{l}_i = (\alpha_{i1}, \cdots, \alpha_{iD})^\top$ is an $D \times 1$ vector where each element is from zero mean distribution with $E(\alpha_{ij}^2) = 1$ and $E(\alpha_{ij}^4) = \gamma$, we have that, for any matrices $\mathbf{M}_1 \in \mathbb{R}^{D \times d_1}$ and $\mathbf{M}_2 \in \mathbb{R}^{D \times d_2}$, where $\boldsymbol{m}_{1i}$ and $\boldsymbol{m}_{2i}$ are their i-th row, respectively.*

$$E\left[(\mathbf{M}_1^\top \boldsymbol{l}_i \boldsymbol{l}_i^\top \mathbf{M}_2)(\mathbf{M}_2^\top \boldsymbol{l}_i \boldsymbol{l}_i^\top \mathbf{M}_1)\right] = \mathbf{M}_1^\top[(\gamma - 3)\mathbf{W}_2 + 2\mathbf{M}_2\mathbf{M}_2^\top + tr(\mathbf{M}_2\mathbf{M}_2^T)\mathbf{I}]\mathbf{M}_1, \tag{7}$$

where $\mathbf{W}_2 = diag\{\boldsymbol{m}_{21}^\top \boldsymbol{m}_{21}, \cdots, \boldsymbol{m}_{2D}^\top \boldsymbol{m}_{2D}\}$. *Particularly, for Gaussian projection,*

$$E\left[(\mathbf{M}_1^\top \boldsymbol{l}_i \boldsymbol{l}_i^\top \mathbf{M}_2)(\mathbf{M}_2^\top \boldsymbol{l}_i \boldsymbol{l}_i^\top \mathbf{M}_1)\right] = 2\mathbf{M}_1^\top[\mathbf{M}_2\mathbf{M}_2^\top]\mathbf{M}_1 + tr(\mathbf{M}_2\mathbf{M}_2^T)\mathbf{M}_1^T\mathbf{M}_1, \tag{8}$$

Proof. Since $\mathbf{M}_1^\top \boldsymbol{l}_i = \sum_{j=1}^{D} \alpha_{ij}\boldsymbol{m}_{1j}$ and $\mathbf{M}_2^\top \boldsymbol{l}_i = \sum_{j=1}^{D} \alpha_{ij}\boldsymbol{m}_{2j}$, where $\boldsymbol{m}_{1j}^\top$ and $\boldsymbol{m}_{2j}^\top$ are the jth row of $\mathbf{M}_1$ and $\mathbf{M}_2$ respectively, it follows that,

$$\mathbf{M}_1^\top \boldsymbol{l}_i \boldsymbol{l}_i^\top \mathbf{M}_2 = (\sum_{j=1}^{D} \alpha_{ij}\boldsymbol{m}_{1j})(\sum_{j=1}^{D} \alpha_{ij}\boldsymbol{m}_{2j})^\top = \sum_{j=1}^{n} \alpha_{ij}^2 \boldsymbol{m}_{1j}\boldsymbol{m}_{2j}^\top + \sum_{j_1 \neq j_2} \alpha_{ij_1}\alpha_{ij_2}\boldsymbol{m}_{1j_1}\boldsymbol{m}_{2j_2}^\top.$$

$$\mathbf{M}_2^\top \boldsymbol{l}_i \boldsymbol{l}_i^\top \mathbf{M}_1 = (\sum_{j=1}^{D} \alpha_{ij}\boldsymbol{m}_{2j})(\sum_{j=1}^{D} \alpha_{ij}\boldsymbol{m}_{1j})^\top = \sum_{j=1}^{n} \alpha_{ij}^2 \boldsymbol{m}_{2j}\boldsymbol{m}_{1j}^\top + \sum_{j_1 \neq j_2} \alpha_{ij_1}\alpha_{ij_2}\boldsymbol{m}_{2j_1}\boldsymbol{m}_{1j_2}^\top,$$

Since $E(\alpha_{ij}^4) = \gamma$, so, we have that

$$\begin{aligned} & E\left(\sum_{j=1}^{D} \alpha_{ij}^2 \boldsymbol{m}_{1j}\boldsymbol{m}_{2j}^\top\right)\left(\sum_{j=1}^{n} \alpha_{ij}^2 \boldsymbol{m}_{2j}\boldsymbol{m}_{1j}^\top\right) \\ =& \gamma \sum_{j=1}^{D} \boldsymbol{m}_{1j}\boldsymbol{m}_{2j}^\top \boldsymbol{m}_{2j}\boldsymbol{m}_{1j}^\top + \left(\sum_{j_1 \neq j_2} \boldsymbol{m}_{1j_1}\boldsymbol{m}_{2j_1}^\top \boldsymbol{m}_{2j_2}\boldsymbol{m}_{1j_2}^\top\right), \end{aligned}$$

$$\begin{aligned} & E\left(\sum_{j_1 \neq j_2} \alpha_{ij_1}\alpha_{ij_2}\boldsymbol{m}_{1j_1}\boldsymbol{m}_{2j_2}^\top\right)\left(\sum_{j_1 \neq j_2} \alpha_{ij_1}\alpha_{ij_2}\boldsymbol{m}_{2j_1}\boldsymbol{m}_{1j_2}^\top\right) \\ =& \left(\sum_{j_1 \neq j_2} \boldsymbol{m}_{1j_1}\boldsymbol{m}_{2j_2}^\top \boldsymbol{m}_{2j_1}\boldsymbol{m}_{1j_2}^\top\right) + \sum_{j_1 \neq j_2} \boldsymbol{m}_{1j_1}\boldsymbol{m}_{2j_2}^T \boldsymbol{m}_{2j_2}\boldsymbol{m}_{1j_1}^T, \\ & E\left(\sum_{j=1}^{D} \alpha_{ij}^2 \boldsymbol{m}_{1j}\boldsymbol{m}_{2j}^\top\right)\left(\sum_{j_1 \neq j_2} \alpha_{ij_1}\alpha_{ij_2}\boldsymbol{m}_{2j_1}\boldsymbol{m}_{1j_2}^\top\right) = 0, \\ & E\left(\sum_{j_1 \neq j_2} \alpha_{ij_1}\alpha_{ij_2}\boldsymbol{m}_{1j_1}\boldsymbol{m}_{2j_2}^\top\right)\left(\sum_{j=1}^{D} \alpha_{ij}^2 \boldsymbol{m}_{2j}\boldsymbol{m}_{1j}^\top\right) = 0. \end{aligned}$$

Combing the four equations above, it is easy to verify that,

$$\begin{aligned} & E\left[(\mathbf{M}_1^\top \boldsymbol{l}_i \boldsymbol{l}_i^\top \mathbf{M}_2)\mathbf{M}_2^\top \boldsymbol{l}_i \boldsymbol{l}_i^\top \mathbf{M}_1\right] \\ =& (\gamma - 3)\sum_{j=1}^{n} \boldsymbol{m}_{1j}\boldsymbol{m}_{2j}^\top \boldsymbol{m}_{2j}\boldsymbol{m}_{1j}^\top + 2\mathbf{M}_1^\top \mathbf{M}_2\mathbf{M}_2^\top \mathbf{M}_1 + \mathrm{tr}(\mathbf{M}_2\mathbf{M}_2^T)\mathbf{M}_1^T\mathbf{M}_1. \end{aligned}$$

□

Lemma 2. *For the Hashing random projection*

$$E\left[(\mathbf{M}_1^\top \mathbf{A}^\top \mathbf{A}\mathbf{M}_2)(\mathbf{M}_1^\top \mathbf{A}^\top \mathbf{A}\mathbf{M}_2)^\top\right] = \mathbf{M}_1^\top \mathbf{M}_2\mathbf{M}_2^\top \mathbf{M}_1 + \frac{1}{d}\mathbf{M}_1^\top \mathbf{M}_1 tr(\mathbf{M}_2\mathbf{M}_2^\top) - \frac{1}{d}\mathbf{M}_1^\top \mathbf{W}_2\mathbf{M}_1.$$

Proof. For the Hashing Random Matrix projection, we rewrite $\mathbf{A} = \mathbf{SD}$, where $\mathbf{S} \in \mathbb{R}^{d \times D}$ has each column chosen independently and uniformly from the r standard basis vectors of $\mathbb{R}^d$ and $\mathbf{D} \in \mathbb{R}^{D \times D}$ is a diagonal matrix with diagonal entries chosen independently and uniformly on $b_i \in \{\pm 1\}$.

Let $\mathbf{S} = (\boldsymbol{s}_1, \cdots, \boldsymbol{s}_n)$, such that $\boldsymbol{s}_i$ is a random vector taking the vector e_j for equal probability, where e_j is the jth standard unit vector of $\mathbb{R}^d$ for $j = 1, \cdots, r$. Then $E(\boldsymbol{s}_i) = \frac{1}{d}\mathbf{1}_d$ and $E(\boldsymbol{s}_i^\top \boldsymbol{s}_i) = 1$, which follow that $E[(\mathbf{S}^\top\mathbf{S})_{ij}] = E(\boldsymbol{s}_i^\top \boldsymbol{s}_j) = E(\boldsymbol{s}_i)^\top E(\boldsymbol{s}_j) = \frac{1}{d^2}$ for $i \neq j$; and $E[(\mathbf{S}^\top\mathbf{S})_{ii}] = E(\boldsymbol{s}_i^\top \boldsymbol{s}_i) = 1$. We have that,

$$\begin{aligned}
\mathbf{M}_1^\top \mathbf{A}^\top \mathbf{A} \mathbf{M}_2 &= (\sum_{i=1}^{D} b_i \boldsymbol{s}_i \boldsymbol{m}_{1i}^\top)^\top (\sum_{i=1}^{D} b_i \boldsymbol{s}_i \boldsymbol{m}_{2i}^\top) \\
&= \sum_{i=1}^{D} b_i^2 \boldsymbol{m}_{1i} \boldsymbol{s}_i^\top \boldsymbol{s}_i \boldsymbol{m}_{2i}^\top + \sum_{i \neq j} b_i b_j \boldsymbol{m}_{1i} \boldsymbol{s}_i^\top \boldsymbol{s}_j \boldsymbol{m}_{2j}^\top \qquad (9)
\end{aligned}$$

From (9), we have,

$$\begin{aligned}
& E\left[\mathbf{M}_1^\top \mathbf{A}^\top \mathbf{A} \mathbf{M}_2\right]\left[\mathbf{M}_1^\top \mathbf{A}^\top \mathbf{A} \mathbf{M}_2\right]^\top \\
=& E\left(\sum_{i=1}^{D} b_i^2 \boldsymbol{m}_{1i} \boldsymbol{s}_i^\top \boldsymbol{s}_i \boldsymbol{m}_{2i}^\top + \sum_{i \neq j} b_i b_j \boldsymbol{m}_{1i} \boldsymbol{s}_i^\top \boldsymbol{s}_j \boldsymbol{m}_{2j}^\top\right)\left(\sum_{i=1}^{D} b_i^2 \boldsymbol{m}_{1i} \boldsymbol{s}_i^\top \boldsymbol{s}_i \boldsymbol{m}_{2i}^\top + \sum_{i \neq j} b_i b_j \boldsymbol{m}_{1i} \boldsymbol{s}_i^\top \boldsymbol{s}_j \boldsymbol{m}_{2j}^\top\right)^\top \\
=& \sum_{i=1}^{D} E(\boldsymbol{m}_{1i} \boldsymbol{s}_i^\top \boldsymbol{s}_i \boldsymbol{m}_{2i}^\top)(\boldsymbol{m}_{1i} \boldsymbol{s}_i^\top \boldsymbol{s}_i \boldsymbol{m}_{2i}^\top)^\top + \sum_{i \neq j} E(\boldsymbol{m}_{1i} \boldsymbol{s}_i^\top \boldsymbol{s}_i \boldsymbol{m}_{2i}^\top)(\boldsymbol{m}_{1j} \boldsymbol{s}_j^\top \boldsymbol{s}_j \boldsymbol{m}_{2j}^\top)^\top \\
& + \sum_{i \neq j} E(\boldsymbol{m}_{1i} \boldsymbol{s}_i^\top \boldsymbol{s}_j \boldsymbol{m}_{2j}^\top)(\boldsymbol{m}_{1i} \boldsymbol{s}_i^\top \boldsymbol{s}_j \boldsymbol{m}_{2j}^\top)^\top =: E_1 + E_2 + E_3.
\end{aligned}$$

Specifically, we have that

$$\begin{aligned}
E_1 &= \sum_{i=1}^{D} E(\boldsymbol{m}_{1i} \boldsymbol{s}_i^\top \boldsymbol{s}_i \boldsymbol{m}_{2i}^\top)(\boldsymbol{m}_{1i} \boldsymbol{s}_i^\top \boldsymbol{s}_i \boldsymbol{m}_{2i}^\top)^\top = \sum_{i=1}^{D} \sum_{k=1}^{d} \frac{1}{d} (\boldsymbol{m}_{1i} \mathbf{e}_k^\top \mathbf{e}_k \boldsymbol{m}_{2i}^\top)(\boldsymbol{m}_{1i} \mathbf{e}_k^\top \mathbf{e}_k \boldsymbol{m}_{2i}^\top)^\top \\
&= \sum_{i=1}^{D} (\boldsymbol{m}_{1i} \boldsymbol{m}_{2i}^\top)(\boldsymbol{m}_{1i} \boldsymbol{m}_{2i}^\top)^\top. \\
E_2 &= \sum_{i \neq j} E(\boldsymbol{m}_{1i} \boldsymbol{s}_i^\top \boldsymbol{s}_i \boldsymbol{m}_{2i}^\top)(\boldsymbol{m}_{1j} \boldsymbol{s}_j^\top \boldsymbol{s}_j \boldsymbol{m}_{2j}^\top)^\top = \sum_{i \neq j} E(\boldsymbol{m}_{1i} \boldsymbol{s}_i^\top \boldsymbol{s}_i \boldsymbol{m}_{2i}^\top) E(\boldsymbol{m}_{1j} \boldsymbol{s}_j^\top \boldsymbol{s}_j \boldsymbol{m}_{2j}^\top)^\top \\
&= \sum_{i \neq j} (\boldsymbol{m}_{1i} \boldsymbol{m}_{2i}^\top)(\boldsymbol{m}_{1j} \boldsymbol{m}_{2j}^\top)^\top \\
E_3 &= \sum_{i \neq j} E(\boldsymbol{m}_{1i} \boldsymbol{s}_i^\top \boldsymbol{s}_j \boldsymbol{m}_{2j}^\top)(\boldsymbol{m}_{1i} \boldsymbol{s}_i^\top \boldsymbol{s}_j \boldsymbol{m}_{2j}^\top)^\top = \sum_{i \neq j} \boldsymbol{m}_{1i} E(\boldsymbol{s}_i^\top \boldsymbol{s}_j \boldsymbol{m}_{2j}^\top \boldsymbol{m}_{2j} \boldsymbol{s}_j^T \boldsymbol{s}_i) \boldsymbol{m}_{1i}^T \\
&= \sum_{i \neq j} \boldsymbol{m}_{1i} \boldsymbol{m}_{2j}^\top \boldsymbol{m}_{2j} E(\boldsymbol{s}_i^\top \boldsymbol{s}_j \boldsymbol{s}_j^T \boldsymbol{s}_i) \boldsymbol{m}_{1i}^T = \sum_{i \neq j} \boldsymbol{m}_{1i} \boldsymbol{m}_{2j}^\top \boldsymbol{m}_{2j} E(\boldsymbol{s}_i^\top \boldsymbol{s}_j)^2 \boldsymbol{m}_{1i}^T \\
&= \frac{1}{d} \sum_{i \neq j} \boldsymbol{m}_{1i} \boldsymbol{m}_{2j}^\top \boldsymbol{m}_{2j} \boldsymbol{m}_{1i}^T \\
&= \frac{1}{d} \mathbf{M}_1^\top \mathbf{M}_1 \mathrm{tr}(\mathbf{M}_2 \mathbf{M}_2^\top) - \frac{1}{d} \mathbf{M}_1^\top \mathbf{W}_2 \mathbf{M}_1 \qquad (10)
\end{aligned}$$

where $\mathbf{W}_2 = \mathrm{diag}\{\boldsymbol{m}_{21}^\top \boldsymbol{m}_{21}, \cdots, \boldsymbol{m}_{2n}^\top \boldsymbol{m}_{2n}\}$. Thus, we have,

$$E\left[(\mathbf{M}_1^\top \mathbf{A}^\top \mathbf{A} \mathbf{M}_2)(\mathbf{M}_1^\top \mathbf{A}^\top \mathbf{A} \mathbf{M}_2)^\top\right] = \mathbf{M}_1^\top \mathbf{M}_2 \mathbf{M}_2^\top \mathbf{M}_1 + \frac{1}{d} \mathbf{M}_1^\top \mathbf{M}_1 \mathrm{tr}(\mathbf{M}_2 \mathbf{M}_2^\top) - \frac{1}{d} \mathbf{M}_1^\top \mathbf{W}_2 \mathbf{M}_1.$$

□

6.5 Details of machine information

The entire experiment in this paper is programmed in Python and run under the Linux system, using open-source Bayesian optimization libraries such as AxBakshy et al. (2018) and BoTorchBalandat et al. (2020) for assistance. The experimental equipment is equipped with 2 AMD EPYC 7601 processors, each with 32 cores and a base clock frequency of 2.2GHz. The experiment is conducted in the form of parallel processing, with different independent repeating experiments distributed to different CPU cores for acceleration. In addition, the system has a memory capacity of 768GB, providing sufficient memory space for large-scale data processing and complex algorithm operation.

6.6 Modified Schwefel function and Griewank function

To investigate the efficacy of the proposed method within the complex context of high-dimensional optimization problems that entail numerous local minima, we applied a Schwefel function of $D = 100$ and a Griewank function of $D = 100$. In the context of the Schwefel function and Griewank function, every dimension qualifies as an effective dimension, hence $d_e = D = 100$. We increased the optimization challenge by altering the Schwefel function and Griewank function. This alteration involved the adjustment of positions within different dimensions where minimum values are attained, thereby making the optimization of the Schwefel function and Griewank function more challenging. The modified Schwefel function is:

$$f(\mathbf{x}) = \sum_{i=1}^{D} \frac{x_i^2}{4000} - \prod_{i=1}^{d} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1, \tag{11}$$

Where $b_i \sim \mathcal{N}(0, 1)$. We maintain the constancy of b_i values across different independent repeated experiments, while allowing b_i values to vary across different dimensions.

The modified Griewank function can be expressed as:

$$f(\mathbf{x}) = \sum_{i=1}^{D} w_i(x_i - b_i)^2 - \prod_{i=1}^{d} \cos\left(\frac{x_i}{\sqrt{i}}\right), \tag{12}$$

where $w_i \sim \mathcal{N}(0, 1)$ and $b_i \sim \mathcal{N}(0, 1)$. We ensure that the values of w_i and b_i remain consistent within various independent repeat experiments, while differing across the several dimensions.

6.7 Details about Real-World Problems

6.7.1 Lunar landing

In this experiment, our goal is to learn a strategy that controls the lunar lander, so that the lunar lander can minimize fuel consumption and distance from the landing target, while avoiding crashes. This optimization task was proposed by ErikssonEriksson et al. (2019). The simulation environment of the control task is implemented through OpenAI gym [1]. The state space of the lunar lander includes its coordinates x and y, linear velocities x_v and y_v, its angle, its angular velocity, and two boolean values indicating whether each leg is in contact with the ground. At any moment, the current controller state can be represented with an 8-dimensional vector. After obtaining the state vector, the controller can choose one of four possible actions, corresponding to pushing the thrusters left, right, up or none. In the experiment, it can be considered as a $D = 12$ optimization problem. Once the parameters are determined, the corresponding rewards can be obtained through in-game feedback. If the lander deviates from the landing pad, it loses rewards. If the lander crashes, it gets an extra -100 points. If it successfully controls the lander to stop, it will get an extra $+100$ points. Each leg touching the ground gets $+10$ points. Igniting the main engine gets -0.3 points per frame. Each frame starts side engine for -0.03 points. The goal of the control task optimization is to maximize the average final reward on a fixed set of 50 randomly generated terrains, initial positions, and speed combinations. We observe that even minor perturbations can have an impact on the simulation.

---

[1] www.gymlibrary.dev/environments/box2d/lunar_lander/

6.7.2 Robot pushing

This paper follows the experimental setup of WangWang et al. (2017b), ErikssonEriksson et al. (2019) et al., and also realizes the simulation of using two robot arms to push two objects in the Box 2DCatto (2011) physics engine. In the simulation environment, the parameters of the robot arms are simulated to push two objects, and the trajectories of the object movements are recorded at the same time. A total of 14 parameters are used by the two robot arms, which respectively specify the position and rotation of the robot hands, the pushing speed, the moving direction, and the pushing time. The lower bounds for these parameters are

$$[-5, -5, -10, -10, 2, 0, -5, -5, -10, -10, 2, 0, -5, -5],$$

and the upper bounds are

$$[5, 5, 10, 10, 30, 2\pi, 5, 5, 10, 10, 30, 2\pi, 5, 5].$$

The initial positions of the objects are designated as s_{i0} and s_{i1}, and the end positions as s_{e0} and s_{e1}. The target positions of the two objects are indicated by s_{g0} and s_{g1}. The reward is defined as

$$r = |s_{g0} - s_{i0}| + |s_{g1} - s_{i1}| - |s_{g0} - s_{e0}| - |s_{g1} - s_{e1}|,$$

namely the distance by which the objects move towards their target positions.

6.7.3 NAS

In this paper, referring to the settings of LethamLetham et al. (2020a) and others, by parameterizing operations and edges respectively, the optimal architecture search problem in NASBench-101 is set as a continuous high-dimensional Bayesian optimization problem. Specifically, L different operations are represented by one-hot encoding.

Since two of the seven nodes are fixed as input and output nodes, the remaining five optional nodes, each node corresponds to three different operations, which generate a total of 15 different parameters. We optimize these parameters in the continuous $[0, 1]$ space. For each node, we take the "operation" corresponding to the maximum value of the three operations under that node as the "operation" adopted by that node, and use one-hot encoding to represent the specific "operation" used under that node.

Since NASBench-101 uses a 7×7 upper triangular adjacency matrix to represent edges, it generates a total of $\frac{7 \cdot 6}{2} = 21$ possible edges. And the five optional vertices can have three different operations, so under this encoding there are about $2^{21} \cdot 3^5 \approx 510M$ unique models, After removing a large number of unreasonable input and output models and models with more than 9 edges, the search space still has about $423k$ unique models. We convert these 21 possible edges into 21 binary parameters that are similarly optimized in a continuous $[0, 1]$ space. We rank the continuous values corresponding to these 21 binary parameters and create an empty adjacency matrix. Then, we add edges to the adjacency matrix in the percentile order of the 21 binary parameters iteratively, while pruning parts that are not connected to the input or output nodes, until reaching the limit of 9 edges. Finally, the combination of adjacency matrix parameters (21) and one-hot encoded "operation" parameters (15) constitutes a 36-dimensional optimization space. The Bayesian optimization algorithm only needs to be optimized in a high-dimensional space with $D = 36$, and the boundary constraint is $[-1, 1]^{36}$. Each vector $\mathbf{x} \in \mathbb{R}^{36}$ can be decoded into a DAG and lookup evaluated in NASBench-101.

6.7.4 Rover planning

To explore the performance of the proposed method in complex high-dimensional optimization scenarios, we considered a two-dimensional trajectory optimization task aimed at simulating detector navigation missions. This optimization task was proposed by WangWang et al. (2018), and the experimental setup by WangWang et al. (2018) was continued to be used here, with the optimization objective being to maximize the reward function. The problem instance is described by defining the starting position s, the target position g, and a cost function on the state space. The goal of the problem is to optimize the detector's trajectory on rugged terrain. The trajectory consists of a set of points on a two-dimensional plane, and there are 30 points in this instance, which can be fitted into a B-spline curve, so it is considered a high-dimensional optimization problem with $D = 60$. Through a set of trajectories, $\boldsymbol{x} \in [0,1]^{60}$, and a specific cost function, we can calculate the cost of a trajectory $c(\boldsymbol{x})$.

The reward for this problem is defined as

$$f(\boldsymbol{x}) = c(\boldsymbol{x}) + \lambda \left(|\boldsymbol{x}_{0,1} - s|\, 1 + |\boldsymbol{x}_{59,60} - g|_1 \right) + b.$$

The reward function is non-smooth, discontinuous, and concave. The four input dimensions involved in the reward function respectively represent the starting and target positions of the trajectory. Set $\lambda = -10, b = 5$, any collision with objects along the trajectory will incur a penalty of -20, which is the collision cost of the trajectory. Thus, in addition to penalties in the reward function caused by collisions, adverse deviations from the trajectory's starting point will also incur additional penalties.