High-Dimensional Bayesian Optimization with Sparse Axis-Aligned Subspaces

David Eriksson[*,1]   **Martin Jankowiak**[*,2]

[1]Facebook, Menlo Park, California, USA
[2]Broad Institute of Harvard and MIT, Cambridge, Massachusetts, USA

Abstract

Bayesian optimization (BO) is a powerful paradigm for efficient optimization of black-box objective functions. High-dimensional BO presents a particular challenge, in part because the curse of dimensionality makes it difficult to define—as well as do inference over—a suitable class of surrogate models. We argue that Gaussian process surrogate models defined on sparse axis-aligned subspaces offer an attractive compromise between flexibility and parsimony. We demonstrate that our approach, which relies on Hamiltonian Monte Carlo for inference, can rapidly identify sparse subspaces relevant to modeling the unknown objective function, enabling sample-efficient high-dimensional BO. In an extensive suite of experiments comparing to existing methods for high-dimensional BO we demonstrate that our algorithm, Sparse Axis-Aligned Subspace BO (`SAASBO`), achieves excellent performance on several synthetic and real-world problems without the need to set problem-specific hyperparameters.

1 INTRODUCTION

Optimization plays an essential role in many fields of science, engineering and beyond. From calibrating complex experimental systems to tuning hyperparameters of machine learning models, the need for scalable and efficient optimization methods is ubiquitous. Bayesian Optimization (BO) algorithms have proven particularly successful on a wide variety of domains including hyperparameter tuning [Snoek et al., 2012], A/B tests [Letham et al., 2019], chemical engineering [Hernández-Lobato et al., 2017], materials science [Ueno et al., 2016], control systems [Candelieri et al., 2018], and drug discovery [Negoescu et al., 2011].

These algorithms typically consist of two components. The first component employs Bayesian methods to construct a surrogate model of the (unknown) objective function. The second component uses this model together with an acquisition function to select the most promising query point(s) at which to evaluate the objective function. By leveraging the uncertainty quantification provided by the Bayesian model, a well-designed BO algorithm can provide an effective balance between exploration and exploitation, leading to highly sample-efficient optimization.

While BO has become a workhorse algorithm that is employed in a wide variety of settings, successful applications are often limited to low-dimensional problems, e.g. fewer than twenty dimensions [Frazier, 2018]. Applying BO to high-dimensional problems remains a significant challenge. The difficulty can be traced to both of the algorithm components mentioned above, although we postulate that suitable function priors are especially important for good performance. In particular, in order for BO to be sample-efficient in high-dimensional spaces, it is crucial to define surrogate models that are sufficiently parsimonious that they can be inferred from a small number of query points. An overly flexible class of models is likely to suffer from overfitting, which severely limits its effectiveness in decision-making. Likewise, an overly rigid class of models is unlikely to capture enough features of the objective function. A compromise between flexibility and parsimony is essential.

In this work we focus on the setting where we aim to optimize a black-box function with hundreds of variables and where we are limited to a few hundred queries of the objective function. We argue that in this low-sample regime Gaussian process surrogate models defined on sparse axis-aligned subspaces provide an attractive compromise between flexibility and parsimony. More specifically, our contributions are as follows:

- We propose the sparsity-inducing `SAAS` function prior
- We demonstrate that when combined with the No-Turn-U-Sampler (NUTS) for inference, our surrogate model

---

*Equal contribution

quickly identifies the most relevant low-dimensional subspace, which in turn leads to sample-efficient BO.

- We show that SAASBO outperforms a number of strong baselines on several problems, including three real-world problems with as many as 388 dimensions, all without setting problem-specific hyperparameters.

2 RELATED WORK

There is a large body of research on high-dimensional BO, and a wide variety of surrogate modelling and acquisition strategies have been proposed [Chen et al., 2012]. In the following we draw attention to a number of common themes.

A popular approach is to rely on low-dimensional structure, with several methods utilizing random projections [Wang et al., 2016, Qian et al., 2016, Binois et al., 2020, Letham et al., 2020]. REMBO uses a random projection to project low-dimensional points up to the original space [Wang et al., 2016]. ALEBO introduces several refinements to REMBO and demonstrates improved performance across a large number of problems [Letham et al., 2020]. Alternatively, the embedding can be learned jointly with the model, including both linear [Garnett et al., 2014] and non-linear [Lu et al., 2018] embeddings. Finally, Hashing-enhanced Subspace BO (HeSBO) [Nayebi et al., 2019] relies on hashing and sketching to reduce surrogate modeling and acquisition function optimization to a low-dimensional space.

Several methods rely on additive structure, where the function is assumed to be a sum of low-dimensional components [Kandasamy et al., 2015, Gardner et al., 2017, Mutny and Krause, 2018, Wang et al., 2018]. This approach allows separating the input space into independent domains, reducing the effective dimensionality of the model.

A common feature of many BO algorithms in high dimensions is that they tend to prefer highly uncertain query points near the domain boundary. As this is usually where the model is the most uncertain, this is often a poor choice that leads to over-exploration and poor optimization performance. Oh et al. [2018] address this issue by introducing a cylindrical kernel that promotes selection of query points in the interior of the domain. LineBO [Kirschner et al., 2019] optimizes the acquisition function along one-dimensional lines, which also helps to avoid highly uncertain points. The TuRBO algorithm uses several trust-regions centered around the current best solution [Eriksson et al., 2019]. These trust-regions are resized based on progress, allowing TuRBO to zoom-in on promising regions. Li et al. [2017] use dropout to select a subset of dimensions over which to optimize the acquisition function, with excluded dimensions fixed to the value of the best point found so far.

Most similar to our method is COMBO [Oh et al., 2019], which uses a sparsity-inducing prior in conjunction with a finite feature expansion to define a surrogate model that is suitable for BO on combinatorial search spaces. The finite feature expansion enables efficient inference via slice sampling. Unfortunately, a finite feature expansion is inappropriate in our setting with real-valued inputs, since the curse of dimensionality severely limits the flexibility of the resulting function prior.

It also important to note that there are many black-box optimization algorithms that do not rely on Bayesian methods, with evolutionary algorithms being especially common. While most methods require thousands of evaluations to find good minima [Yu and Gen, 2010], the popular covariance matrix adaptation evolution strategy (CMA-ES; [Hansen et al., 2003]) is competitive with BO on some problems [Letham et al., 2020].

3 BACKGROUND

We use this section to establish our notation and review necessary background material. Throughout this paper we work in the D-dimensional domain $\mathcal{D} = [0, 1]^D$. We consider the minimization problem $\mathbf{x}_{\min} \in \operatorname{argmin}_{\mathbf{x} \in \mathcal{D}} f_{\text{obj}}(\mathbf{x})$ for a noise-free objective function $f_{\text{obj}} : \mathcal{D} \to \mathbb{R}$. We assume that evaluations of f_{obj} are costly and that we are limited to at most a few hundred. Additionally, f_{obj} is a black-box function and gradient information is unavailable.

The rest of this section is organized as follows: in Sec. 3.1 we review Gaussian processes; and in Sec. 3.2 we review the expected improvement acquisition function.

3.1 GAUSSIAN PROCESSES

Gaussian processes (GPs) offer powerful non-parametric function priors that are the gold standard in BO due to their flexibility and excellent uncertainty quantification. A GP on the input space $\mathcal{D}$ is specified[1] by a covariance function or kernel $k : \mathcal{D} \times \mathcal{D} \to \mathbb{R}$ [Rasmussen, 2003]. A common choice is the RBF or squared exponential kernel, which is given by

$$k^\psi(\mathbf{x}, \mathbf{y}) = \sigma_k^2 \exp\{-\tfrac{1}{2} \sum_i \rho_i (x_i - y_i)^2\} \quad (1)$$

where ρ_i for $i = 1, ..., D$ are inverse squared length scales and where we use ψ to collectively denote all the hyperparameters, i.e. $\psi = \{\rho_{1:D}, \sigma_k^2\}$. For scalar regression $f : \mathcal{D} \to \mathbb{R}$ the joint density of a GP takes the form

$$p(\mathbf{y}, \mathbf{f}|\mathbf{X}) = \mathcal{N}(\mathbf{y}|\mathbf{f}, \sigma^2 \mathbb{1}_N) \mathcal{N}(\mathbf{f}|\mathbf{0}, K_{\mathbf{XX}}^\psi) \quad (2)$$

where $\mathbf{y}$ are the real-valued targets, $\mathbf{f}$ are the latent function values, $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^N$ are the N inputs with $\mathbf{x}_i \in \mathcal{D}$, σ^2 is the variance of the Normal likelihood $\mathcal{N}(\mathbf{y}|\cdot)$, and $K_{\mathbf{XX}}^\psi$ is

---

[1]Here and elsewhere we assume that the mean function is uniformly zero.

the $N \times N$ kernel matrix. Throughout this paper we will be interested in modeling noise-free functions, in which case σ^2 is set to a small constant. The marginal likelihood of the observed data can be computed in closed form:

$$p(\mathbf{y}|\mathbf{X}, \psi) = \int d\mathbf{f}\, p(\mathbf{y}, \mathbf{f}|\mathbf{X}) = \mathcal{N}(\mathbf{y}, K_{\mathbf{XX}}^{\psi} + \sigma^2 \mathbb{1}_N). \tag{3}$$

The posterior distribution of the GP at a query point $\mathbf{x}^* \in \mathcal{D}$ is the Normal distribution $\mathcal{N}(\mu_{\mathbf{f}}(\mathbf{x}^*), \sigma_{\mathbf{f}}(\mathbf{x}^*)^2)$ where $\mu_{\mathbf{f}}(\cdot)$ and $\sigma_{\mathbf{f}}(\cdot)^2$ are given by

$$\mu_{\mathbf{f}}(\mathbf{x}^*) = k_{*\mathbf{X}}^{\psi\,\mathrm{T}} (K_{\mathbf{XX}}^{\psi} + \sigma^2 \mathbb{1}_N)^{-1} \mathbf{y} \tag{4}$$

$$\sigma_{\mathbf{f}}(\mathbf{x}^*)^2 = k_{**}^{\psi} - k_{*\mathbf{X}}^{\psi\,\mathrm{T}} (K_{\mathbf{XX}}^{\psi} + \sigma^2 \mathbb{1}_N)^{-1} k_{*\mathbf{X}}^{\psi} \tag{5}$$

Here $k_{**}^{\psi} = k^{\psi}(\mathbf{x}^*, \mathbf{x}^*)$ and $k_{*\mathbf{X}}^{\psi}$ is the column vector specified by $(k_{*\mathbf{X}}^{\psi})_n = k^{\psi}(\mathbf{x}^*, \mathbf{x}_n)$ for $n = 1, ..., N$.

3.2 EXPECTED IMPROVEMENT

Expected improvement (EI) is a popular acquisition function that is defined as follows [Mockus et al., 1978, Jones et al., 1998]. Suppose that in previous rounds of BO we have collected $\mathcal{H} = \{\mathbf{x}_{1:N}, y_{1:N}\}$. Then let $y_{\min} = \min_n y_n$ denote the best function evaluation we have seen so far. We define the *improvement* $u(\mathbf{x}|y_{\min})$ at query point $\mathbf{x} \in \mathcal{D}$ as $u(\mathbf{x}|y_{\min}) = \max(0, y_{\min} - f(\mathbf{x}))$. EI is defined as the expectation of the improvement over the posterior of $f(\mathbf{x})$:

$$\mathrm{EI}(\mathbf{x}|y_{\min}, \psi) = \mathbb{E}_{\mathrm{p}(\mathrm{f}(\mathbf{x})|\psi, \mathcal{H})}\left[\mathrm{u}(\mathbf{x}|y_{\min})\right] \tag{6}$$

where our notation makes explicit the dependence of Eqn. (6) on the kernel hyperparameters ψ. For a GP like in Sec. 3.1 this expectation can be evaluated in closed form:

$$\mathrm{EI}(\mathbf{x}|y_{\min}, \psi) = (y_{\min} - \mu_{\mathbf{f}}(\mathbf{x}))\Phi(Z) + \sigma_{\mathbf{f}}(\mathbf{x})\phi(Z) \tag{7}$$

where $Z \equiv (y_{\min} - \mu_{\mathbf{f}}(\mathbf{x}))/\sigma_{\mathbf{f}}(\mathbf{x})$ and where $\Phi(\cdot)$ and $\phi(\cdot)$ are the CDF and PDF of the unit Normal distribution, respectively. By maximizing Eqn. (7) over $\mathcal{D}$ we can find query points $\mathbf{x}$ that balance exploration and exploitation.

4 BAYESIAN OPTIMIZATION WITH SPARSE AXIS-ALIGNED SUBSPACES

We now introduce the surrogate model we use for high-dimensional BO. For a large number of dimensions, the space of functions mapping $\mathcal{D}$ to $\mathbb{R}$ is—to put it mildly— very large, even assuming a certain degree of smoothness. To facilitate sample-efficient BO it is necessary to make additional assumptions. Intuitively, we would like to assume that the dimensions of $\mathbf{x} \in \mathcal{D}$ exhibit a hierarchy of relevance. For example in a particular problem we might have that $\{\mathbf{x}_3, \mathbf{x}_{52}\}$ are crucial features for mapping the principal variation of f_{obj}, $\{\mathbf{x}_7, \mathbf{x}_{14}, \mathbf{x}_{31}, \mathbf{x}_{72}\}$ are of moderate

importance, while the remaining features are of marginal importance. This motivates the following desiderata for our function prior:

1. Assumes a hierarchy of feature relevances
2. Encompasses a flexible class of smooth non-linear functions
3. Admits tractable (approximate) inference

4.1 SAAS FUNCTION PRIOR

To satisfy our desiderata we introduce a GP model with a structured prior over the kernel hyperparameters, in particular one that induces sparse structure in the (inverse squared) length scales ρ_i. In detail we define the following model:

$$[\text{kernel variance}] \quad \sigma_k^2 \sim \mathcal{LN}(0, 10^2) \tag{8}$$
$$[\text{global shrinkage}] \quad \tau \sim \mathcal{HC}(\alpha)$$
$$[\text{length scales}] \quad \rho_i \sim \mathcal{HC}(\tau) \quad \text{for } i = 1, ..., D.$$
$$[\text{function values}] \quad \mathbf{f} \sim \mathcal{N}(\mathbf{0}, K_{\mathbf{XX}}^{\psi}) \quad \text{with } \psi = \{\rho_{1:d}, \sigma_k^2\}$$
$$[\text{observations}] \quad \mathbf{y} \sim \mathcal{N}(\mathbf{f}, \sigma^2 \mathbb{1}_N)$$

where $\mathcal{LN}$ denotes the log-Normal distribution and $\mathcal{HC}(\alpha)$ denotes the half-Cauchy distribution, i.e. $p(\tau|\alpha) \propto (\alpha^2 + \tau^2)^{-1}\mathbb{1}(\tau > 0)$, and $p(\rho_i|\tau) \propto (\tau^2 + \rho_i^2)^{-1}\mathbb{1}(\rho_i > 0)$. Here $\alpha > 0$ is a hyperparameter that controls the level of shrinkage (our default is $\alpha = 0.1$). We use an RBF kernel, although other choices like the Matérn-5/2 kernel are also possible. We also set $\sigma^2 \to 10^{-6}$, since we focus on noise-free objective functions f_{obj}. Noisy objective functions can be accommodated by placing a weak prior on σ^2, for example $\sigma^2 \sim \mathcal{LN}(0, 10^2)$.

The SAAS function prior defined in (12) has the following important properties. First, the prior on the kernel variance σ_k^2 is weak (i.e. non-informative). Second, the level of global shrinkage (i.e. sparsity) is controlled by the scalar $\tau > 0$, which tends to concentrate near zero due to the half-Cauchy prior. Third, the (inverse squared) length scales ρ_i are also governed by half-Cauchy priors, and thus they too tend to concentrate near zero (more precisely for most i we expect $\rho_i \lesssim \tau$). Consequently most of the dimensions are 'turned off' in accord with the principle of *automatic relevance determination* introduced by MacKay and Neal [1994]. Finally, while the half-Cauchy priors favor values near zero, they have heavy tails. This means that if there is sufficient evidence in the observations $\mathbf{y}$, the posterior over τ will be pushed to higher values, thus reducing the level of shrinkage and allowing more of the ρ_i to escape zero, effectively 'turning on' more dimensions. The parsimony inherent in our function prior is thus *adaptive*: as more data is accumulated, more of the ρ_i will escape zero, and posterior mass will give support to a richer class of functions. This is in contrast to a standard GP fit with maximum likelihood estimation (MLE), which will generally exhibit non-negligible ρ_i for

Algorithm 1: We outline the main steps in `SAASBO` when NUTS is used for inference. To instead use MAP we simply swap out line 4. For details on inference see Sec. 4.2; for details on EI maximization see Sec. 4.3.

Input: Objective function f_{obj}; initial evaluation budget $m \geq 2$; total evaluation budget $T > m$; hyperparameter α; number of NUTS samples L; and initial query set $\mathbf{x}_{1:m}$ and evaluations $y_{1:m}$ (*optional*)

Output: Approximate minimizer and minimum $(\mathbf{x}_{\min}, y_{\min})$

1 If $\{\mathbf{x}_{1:m}, y_{1:m}\}$ is not provided, let $\mathbf{x}_{1:m}$ be a Sobol sequence in $\mathcal{D}$ and let $y_t = f_{\text{obj}}(\mathbf{x}_t)$ for $t = 1, ..., m$.
2 **for** $t = m + 1, ..., T$ **do**
3   Let $\mathcal{H}_t = \{\mathbf{x}_{1:t-1}, y_{1:t-1}\}$ and $y_{\min}^t = \min_{s < t} y_s$.
4   Fit `SAAS` GP in Eqn. (12) to $\mathcal{H}_t$ using NUTS to obtain L hyperparameter samples $\{\psi_\ell^t\}$.
5   Optimize the expected improvement in Eqn. (10) to obtain $\mathbf{x}_t = \arg\max_{\mathbf{x}} \text{EI}(\mathbf{x}|y_{\min}^t, \{\psi_\ell^t\})$.
6   Query f_{obj} and set $y_t = f_{\text{obj}}(\mathbf{x}_t)$.
7 **return** $(\mathbf{x}_{\min}, y_{\min})$ where $(\mathbf{x}_{\min}, y_{\min}) \equiv (\mathbf{x}_{t_{\min}}, y_{t_{\min}})$ and $t_{\min} = \arg\min_t y_t$.

most dimensions—since there is no mechanism regularizing the length scales—typically resulting in drastic overfitting in high-dimensional settings.

Conceptually, our function prior describes functions defined on sparse axis-aligned subspaces, thus the name of our prior (`SAAS`) and our method (`SAASBO`).

4.2 INFERENCE

Doing inference for the model defined in Sec. 4.1 is challenging because of the dimension of the latent space and the many non-linearities. Thankfully, the latent variables in our model are continuous (and the joint density is differentiable), so we can leverage efficient gradient-based inference techniques. In this section we describe the two inference strategies we pursue. The first relies on the No-U-Turn sampler (NUTS) [Hoffman and Gelman, 2014], an adaptive variant of Hamiltonian Monte Carlo that is the gold standard for inference in models like ours. The second is a maximum a posteriori (MAP) approach, which trades off fidelity of the posterior approximation for faster runtime. In both cases we make use of the marginal likelihood $p(\mathbf{y}|\mathbf{X}, \psi)$ in Eqn. (3), i.e. we integrate out the latent function $\mathbf{f}$ analytically.

4.2.1 No-U-Turn Sampler (NUTS)

We use the NUTS sampler implemented in NumPyro [Phan et al., 2019] to target the un-normalized joint density

$$p(\mathbf{y}|\mathbf{X}, \psi)p(\psi|\tau)p(\tau) \propto p(\tau, \psi|\mathbf{X}, \mathbf{y}). \qquad (9)$$

Here $p(\psi|\tau)p(\tau)$ denotes the density over the kernel hyperparameters ψ and shrinkage parameter τ given in Eqn. (12). After running NUTS we obtain L approximate posterior samples for the kernel hyperparameters, $\{\psi_\ell\}_{\ell=1}^L$. The cost of obtaining a posterior sample is $\mathcal{O}(N^3 D)$ where N is the total number of datapoints and D is the dimension of the input domain $\mathcal{D}$.[2] Thus our method inherits the scalability

bottleneck of all BO methods that rely on GPs and is most suitable for moderate numbers of datapoints, e.g. $N \lesssim 500$. The kernel hyperparameters can then be plugged into the closed form GP predictive formulae in Eqn. (4)-(5).

4.2.2 Maximum a posteriori (MAP)

In MAP we target the same un-normalized density as in Eqn. (13), with a few small differences. First, since MAP is formulated as an optimization problem w.r.t. the target density, the result of inference is a single point estimate and not a bag of samples as in NUTS. Second, we remove the prior over τ and instead learn separate models for a small number S of pre-selected[3] values of τ, e.g. $\tau_s \in \{10^{-1}, 10^{-2}, 10^{-3}\}$. Thus after convergence we obtain S point estimates $\{(\psi_s)\}_{s=1}^S$. Finally, to choose between these S point estimates we use a leave-one-out measure of the predictive log likelihood to select the best performing ψ_s. See Sec. **??** in the supplementary materials for details.

4.3 ACQUISITION STRATEGY

We use expected improvement (EI) as our acquisition function given its simplicity, favorable computational properties, and good empirical performance. We begin by noting that the expression for EI given in Eqn. (7) depends on the kernel hyperparameters ψ through $\mu_{\mathbf{f}}(\mathbf{x})$ and $\sigma_{\mathbf{f}}^2(\mathbf{x})$. Thus in our context where ψ is a latent variable, the expected improvement is defined by averaging Eqn. (7) over posterior samples $\{\psi_\ell\}_{\ell=1}^L \sim p(\psi|\mathcal{H})$

$$\text{EI}(\mathbf{x}|y_{\min}, \{\psi_\ell\}) \equiv \frac{1}{L} \sum_{\ell=1}^L \text{EI}(\mathbf{x}|y_{\min}, \psi_\ell) \qquad (10)$$

where in Eqn. (10) we assume we have obtained L samples from NUTS.

---

[2]The factor of D comes from computing terms that arise in the gradients of Eqn. (3).

[3]Note that this means that `SAASBO-MAP` does not require specifying the hyperparameter α.

An essential property of Eqn. (10) is that it is differentiable w.r.t. $\mathbf{x}$ and thus can be efficiently optimized with gradient methods. In practice we optimize Eqn. (10) by generating a Sobol sequence in $\mathcal{D}$ to find a small number K of promising starting points $\{\tilde{\mathbf{x}}_k\}_{k=1}^K$ and then use these to initialize K runs of L-BFGS-B to obtain the query point

$$\mathbf{x}_{\text{next}} = \operatorname{argmax}_{\mathbf{x}} \text{EI}(\mathbf{x}|y_{\min}, \{\psi_\ell\}) \qquad (11)$$

See the supplementary materials for further details and Alg. 1 for a complete outline of the SAASBO algorithm.

4.4 DISCUSSION

We note that the axis-aligned structure of our model need not be as restrictive as one might at first assume. For example, suppose that f_{obj} can be written as $f_{\text{obj}}(\mathbf{x}) = g(\mathbf{x}_3 - \mathbf{x}_7)$ for some $g : \mathbb{R} \to \mathbb{R}$. In order for our model to capture the structure of f_{obj}, both $\mathbf{x}_3$ and $\mathbf{x}_7$ need to be identified as relevant. In many cases we expect this to be possible with a relatively small number of samples. While it is true that identifying the direction $\mathbf{z} = \mathbf{x}_3 - \mathbf{x}_7$ could be even easier in a different coordinate system, inferring non-axis-aligned subspaces would come at the cost of substantially increased computational cost. More importantly, by searching over a much larger set of subspaces our surrogate model would likely be much more susceptible to overfitting. Given that for many problems we expect much of the function variation to be captured by axis-aligned blocks of input features, we view our axis-aligned assumption as a good compromise between flexibility and parsimony. Indeed in Sec. D.4 in the supplementary materials we describe an experiment in which we construct objective functions with significant non-axis-aligned structure by performing a random rotation on an objective function that is axis-aligned. We find that SAASBO performs well in this challenging setting; see Fig. 10 in the supplementary material. Importantly, our modeling approach does not sacrifice any of the many benefits of GPs (e.g. flexible non-linearity and non-parametric latent functions) nor do we need to make any unduly strong assumptions about f_{obj} (e.g. additive decomposition).

It is important to emphasize that it is by design that the model defined in Sec. 4.1 does not include any discrete latent variables. A natural alternative to our model would introduce D binary-valued variables that control whether or not a given dimension is relevant to modeling f_{obj}. However, inference in any such model is very challenging as it requires exploring a discrete space of size 2^D. Our model can be understood as a continuous relaxation of such an approach. Indeed, the structure of our sparsity-inducing prior closely mirrors the justly famous Horseshoe prior [Carvalho et al., 2009], which is a popular prior for Sparse Bayesian linear regression. We note that in contrast to the linear regression setting of the Horseshoe prior, our sparsity-inducing prior governs inverse squared length scales in a non-linear kernel

and *not* variances. We discuss this point in more detail in Sec. A.3 in the supplementary materials.

5 EXPERIMENTS

We present an empirical validation of our approach. In Sec. 5.1-5.2 we characterize the behavior of SAASBO in controlled settings. In Sec. 5.4-5.7 we benchmark SAASBO against a number of state-of-the-art methods for high-dimensional BO. An open source implementation of SAASBO that relies on Pyro [Bingham et al., 2019] will be made available in BoTorch [Balandat et al., 2019], while a NumPyro [Phan et al., 2019] version is available at https://github.com/martinjankowiak/saasbo.

5.1 THE SAAS PRIOR PROVIDES GOOD MODEL FIT IN HIGH DIMENSIONS

In Fig. 1 we demonstrate the importance of using a sparsity-inducing prior like SAAS when fitting a GP in a high-
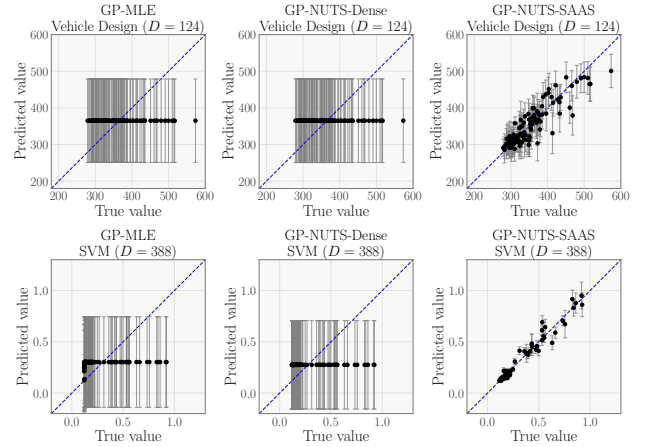

Figure 1: We compare model fit for three models on the $D = 124$ vehicle design problem and the $D = 388$ SVM problem (see Sec. 5.6-5.7 for details). We compare: (left) a GP fit with MLE; (middle) a GP with weak priors fit with NUTS; and (right) a GP with a SAAS prior (this paper; see Eqn. (12)) fit with NUTS. For the vehicle design problem we use 100 training points and for the SVM problem we use 50 training points. We use 100 test points for both problems. Only SAAS provides a good fit. In each figure mean predictions are depicted with dots and bars denote 95% confidence intervals.

dimensional domain. For these high-dimensional problems, both maximum likelihood estimation and full Bayesian inference for a GP with weak log-Normal priors on the squared length scales ρ_i^{-1} concentrate on solutions in which the vast majority of the ρ_i are $\mathcal{O}(1)$. Consequently with high probability the kernel similarity between a randomly chosen test point and any of the $N = 100$ training data points is

$\mathcal{O}(\exp(-D)) \approx 0$, with the result that both these models revert to a trivial mean prediction across most of the domain. By contrast, the SAAS prior only allows a few ρ_i to escape zero, resulting in a model that is much more useful for exploration and exploitation of the most important design variables.

5.2 SAASBO CAN QUICKLY IDENTIFY THE MOST RELEVANT DIMENSIONS

We characterize the behavior of SAASBO in a controlled setting where we embed the two-dimensional Branin function in $D = 100$ dimensions. First, we explore the degree to which SAASBO's performance depends on the approximate inference algorithm used, in particular comparing NUTS to MAP (see Sec. 4.2 for details on inference). In Fig. 2 (left) we see that NUTS outperforms MAP by a considerable margin. In Fig. 2 (middle and right) we demonstrate that both inference methods are able to reliably identify the two relevant dimensions after $\sim 20 - 30$ evaluations.

Why does NUTS outperform MAP even though MAP is able to identify the relevant subspace? We hypothesize that the primary reason for the superior performance of NUTS is that the EI objective in Eqn. (10) is considerably more robust when averaged over multiple samples of the GP kernel hyperparameters. In particular, averaging over multiple samples—potentially from distinct modes of the posterior—appears to mitigate EI's tendency to seek out the boundary of the domain $\mathcal{D}$. For this reason we use NUTS for the experiments in this work, noting that while we obtain good performance with MAP in some problem settings we find that NUTS is significantly more robust.

Next, we explore the dependence of SAASBO-NUTS on the hyperparameter α. In Fig. 2 (left) we see that there is minimal dependence on α, with the three values leading to similar optimization performance. In Fig. 2 (middle and right) we see that, as expected, smaller values of α are more conservative (i.e., prefer smaller subspaces), while larger values of α are less conservative (i.e., prefer larger subspaces). We note, however, that this effect is most pronounced when only a small number of datapoints have been collected. After ~ 20 function evaluations the observations overwhelm the prior $p(\tau)$ and the posterior quickly concentrates on the two relevant dimensions.

Given the good performance of all three values of α, for the remainder of our experiments we choose the intermediate value $\alpha = 0.1$. While performance can perhaps be improved in some cases by tuning α, we find it encouraging that we can get good performance with a single α. We emphasize that α is the only hyperparameter that governs the function prior, and that all remaining hyperparameters control the computational budget (e.g. the number of NUTS samples L). This is in contrast to the many methods for high-dimensional BO that rely on several (potentially sensitive) hyperparameters such as the dimension d_e of a random embedding.

5.3 BASELINES

We compare SAASBO to a comprehensive selection of baselines: ALEBO, CMA-ES, EBO, HeSBO, SMAC, Sobol, and TuRBO. ALEBO [Letham et al., 2020] is chosen as a representative random embedding method, as it improves upon the original REMBO method [Wang et al., 2016]. Additionally, we compare to HeSBO, which uses hashing and sketching to project low-dimensional points up to the original space [Nayebi et al., 2019]. The EBO method by Wang et al. [2018] exploits additive structure to scale to high-dimensional spaces. We also compare to CMA-ES [Hansen et al., 2003], which is a popular evolutionary method that is often competitive with BO methods on high-dimensional problems, see e.g., [Letham et al., 2020]. TuRBO [Eriksson et al., 2019] uses a trust region centered at the best solution to avoid exploring highly uncertain parts of the search space. We also include an additional BO method that does not rely on GPs, namely SMAC [Hutter et al., 2011]. Finally, we also compare to scrambled Sobol sequences [Owen, 2003].

We use the default settings for all baselines. For ALEBO and HeSBO we evaluate both $d_e = 5$ and $d_e = 10$ on the three synthetic problems in Sec. 5.4. As $d_e = 5$ does not perform well on the three real-world applications in Sec. 5.5-5.7, we instead evaluate $d_e = 10$ and $d_e = 20$ on these problems.

We also mention a baseline method for which we do not report results, since it underperforms random search. Namely for our surrogate model we use a quadratic polynomial over $\mathcal{D}$ with $\mathcal{O}(D^2)$ coefficients governed by a sparsity-inducing Horseshoe prior [Carvalho et al., 2009]. As in Baptista and Poloczek [2018], this finite feature expansion admits efficient inference with a Gibbs sampler. Unfortunately, in our setting, where $\mathcal{D}$ is continuous and not discrete, this leads to pathological behavior when combined with EI, since the minima of simple parametric models are very likely to be found at the boundary of $\mathcal{D}$. This is in contrast to the mean-reverting behavior of a GP with a RBF or Matérn kernel, which is a much more appropriate modeling assumption in high dimensions.

5.4 SYNTHETIC PROBLEMS

In this section we consider the Branin ($d = 2$), Hartmann ($d = 6$), and Rosenbrock ($d = 3$) test functions embedded in a $D = 100$ space.[4] These are problems with unambiguous low-dimensional structure where we expect both random embedding methods and SAASBO to perform well.

---

[4] That is to say each synthetic function depends on exactly d variables and is independent of the remaining $D - d$ variables.
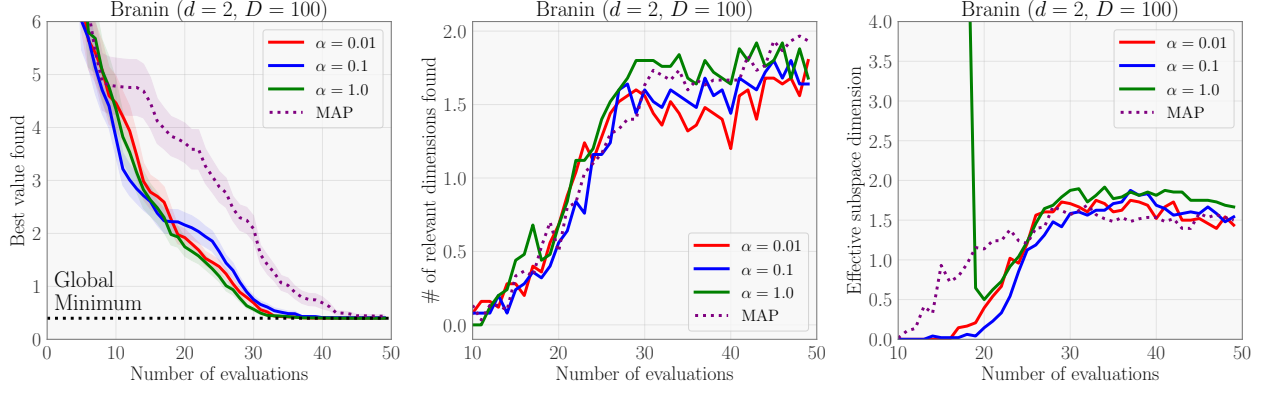
Figure 2: We explore how SAASBO performs on Branin ($D = 100$), comparing SAASBO-NUTS for three values of the sparsity controlling hyperparameter α to SAASBO-MAP. Each curve corresponds to 60 independent replications of Algorithm 1. **Left**: We compare performance w.r.t. the best minimum found (the mean is depicted by a thick line and shaded bands denote standard errors). **Middle**: We depict the mean number of *relevant* dimensions found, where a relevant dimension is declared 'found' if its corresponding PosteriorMedian(ρ_k) is among the two largest $\{\text{PosteriorMedian}(\rho_i)\}_{i=1}^{D}$. **Right**: We depict the mean effective subspace dimension, defined to be the number of dimensions for which PosteriorMedian(ρ_k) > 0.5.
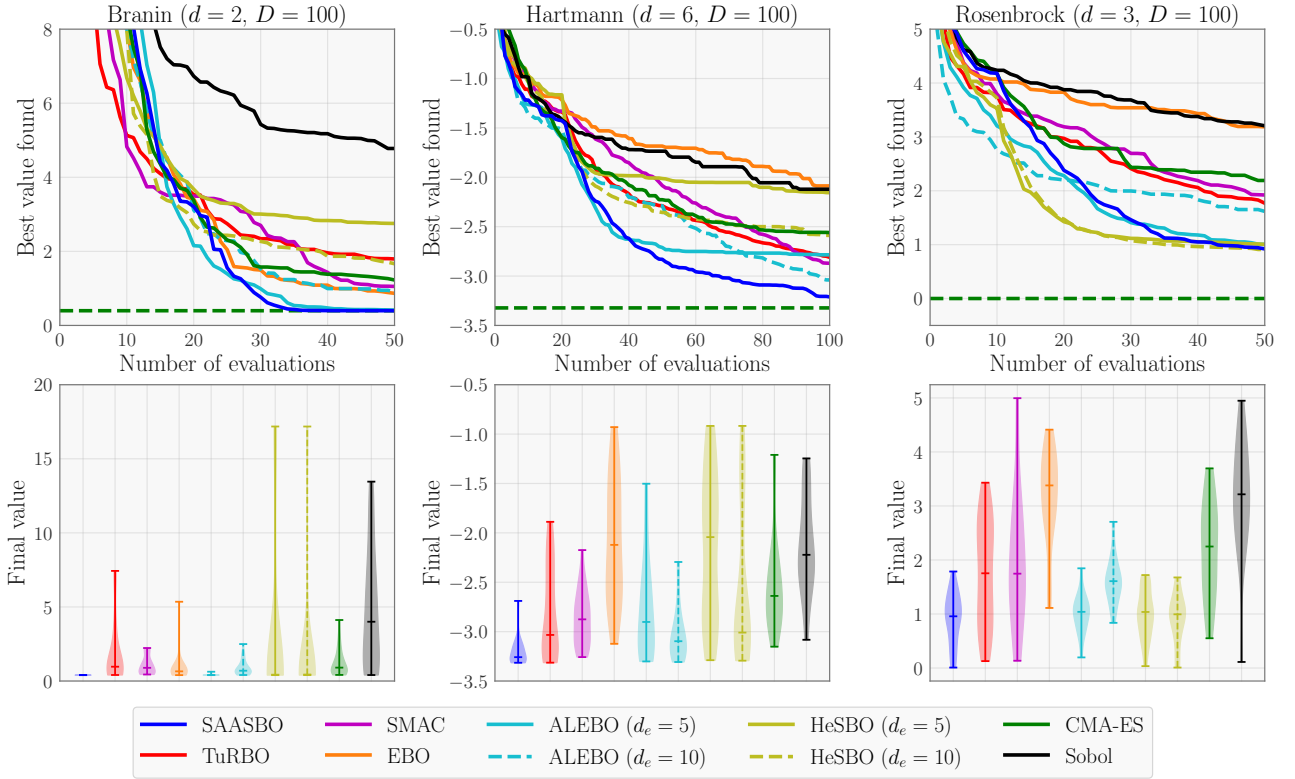

Figure 3: We compare SAASBO to seven baseline methods on three $d-$dimensional functions embedded in $D = 100$ dimensions. In each case we do 30 independent replications. **Top row:** For each method we depict the mean value of the best minimum found at a given iteration. **Bottom row:** For each method we depict the distribution over the final approximate minimum $y_{\min}$ encoded as a violin plot, with horizontal bars corresponding to 5%, 50%, and 95% quantiles.

Fig. 3 shows that SAASBO and ALEBO-5 perform the best on Branin. SAASBO performs the best on Hartmann followed by ALEBO-10. HeSBO performs well on Rosenbrock

and the final performance of SAASBO, HeSBO-5, HeSBO-10, and ALEBO-5 are similar. However, both ALEBO and HeSBO show significant sensitivity to the embedded sub-
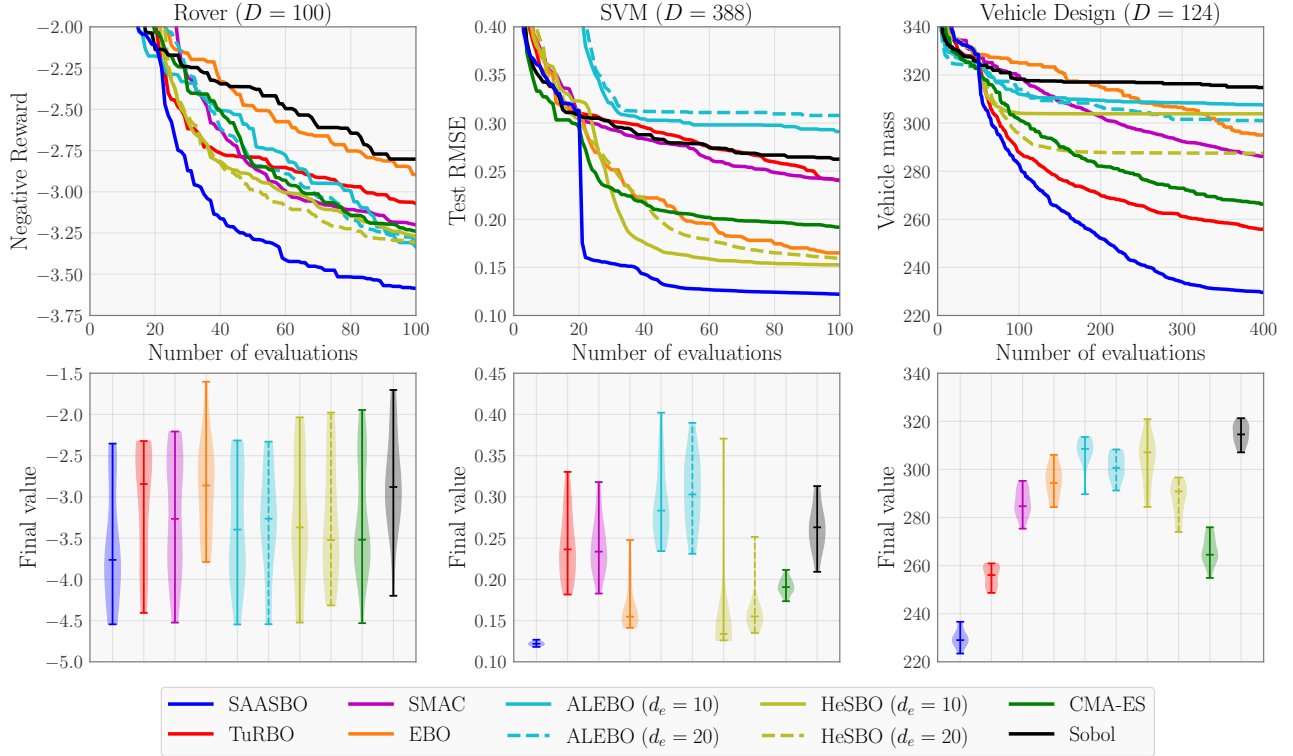
Figure 4: We compare SAASBO to baseline methods on rover trajectory planning ($D = 100$), SVM hyperparameter tuning ($D = 388$), and MOPTA vehicle design ($D = 124$). We do 30 independent replications for Rover and SVM and 15 replications for MOPTA. **Top row:** For each method we depict the mean value of the best minimum found at a given iteration. **Bottom row:** For each method we depict the distribution over the final approximate minimum $y_{\min}$ encoded as a violin plot, with horizontal bars corresponding to 5%, 50%, and 95% quantiles.

space dimension on at least two of the three problems, highlighting a serious downside of random embedding methods. Crucially this important hyperparameter needs to be chosen before the start of optimization and is not learned.

5.5 ROVER TRAJECTORY PLANNING

We consider a variation of the rover trajectory planning problem from [Wang et al., 2018] where the task is to find an optimal trajectory through a 2d-environment. In the original problem, the trajectory is determined by fitting a B-spline to 30 waypoints and the goal is to optimize the locations of these waypoints. This is a challenging problem that requires thousands of evaluations to find good solutions, see e.g. [Eriksson et al., 2019]. To make the problem more suitable for small evaluation budgets, we require that the B-spline starts and ends at the pre-determined starting position and destination. We also increase the dimensionality to $D = 100$ by using 50 waypoints. Fig. 4 shows that SAASBO performs the best on this problem. This problem is challenging for all methods, each of which had at least one replication where the final reward was below 2.5.

5.6 HYPERPARAMETER TUNING OF AN SVM

We define a hyperparameter tuning problem using a kernel support vector machine (SVM) trained on a 385-dimensional regression dataset. This results in a $D = 388$ problem, with 3 regularization parameters and 385 kernel length scales. We expect this problem to have some amount of low-dimensional structure, as we expect the regularization parameters to be most relevant, with a number of length scales of secondary, but non-negligible importance. This intuition is confirmed in Fig. 8 in the supplementary materials, which demonstrates that SAASBO quickly focuses on the regularization parameters, explaining the superior performance of SAASBO seen in Fig. 4. ALEBO makes little progress after iteration 30, indicating that there may not be any good solutions within the random embeddings. HeSBO and EBO do better than the other methods, but fail to match the final performance of SAASBO.

5.7 VEHICLE DESIGN

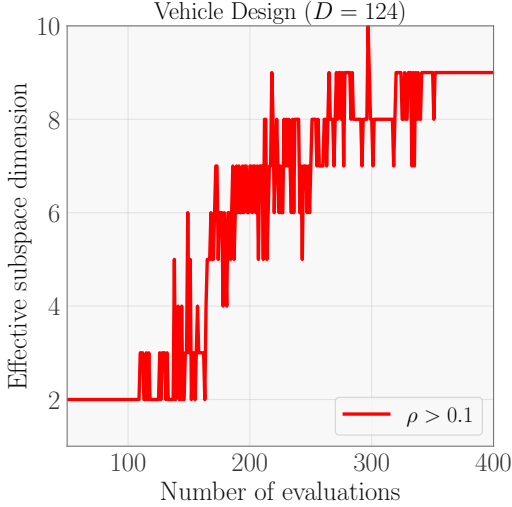We consider the vehicle design problem MOPTA08, a challenging real-world high-dimensional BO problem [Jones,

Figure 5: We depict the effective subspace dimension during a single run of Algorithm 1 on the MOPTA vehicle design problem. Here the effective subspace dimension is the number of dimensions for which $\mathrm{PosteriorMedian}(\rho_k) > \xi$, with $\xi = 0.1$ an arbitrary cutoff.

2008]. The goal is to minimize the mass of a vehicle subject to 68 performance constraints. The $D = 124$ design variables describe materials, gauges, and vehicle shape. To accommodate our baseline methods, While some methods such as Scalable Constrained Bayesian Optimization (SCBO) [Eriksson and Poloczek, 2021] can handle this constrained problem with thousands of evaluations, we convert the hard constraints into a soft penalty, yielding a scalar objective function. Fig. 4 shows that SAASBO outperforms other methods by a large margin. TuRBO and CMA-ES perform better than the remaining methods, which fail to identify good solutions. While this problem does not have obvious low-dimensional structure, our flexible SAAS prior still results in superior optimization performance.

In Fig. 5 we see that during the course of a single run of SAASBO on this problem, the effective dimension of the identified subspace steadily increases from about 2 to about 10 as more evaluations are collected. Using an increasingly flexible surrogate model over the course of optimization is key to the excellent optimization performance of SAASBO.

6 DISCUSSION

Black-box optimization in hundreds of dimensions presents a number of challenges, many of which can be traced to the many degrees of freedom that characterize high-dimensional spaces. The majority of approaches to Bayesian optimization try to circumvent this potential hazard by reducing the effective dimensionality of the problem. For example ran-

dom projection methods like ALEBO and HeSBO work directly in a low-dimensional space, while methods like TuRBO or LineBO constrain the domain over which the acquisition function is optimized. We take the view that it is much more natural to work directly in the full space and instead rely on a sparsity-inducing function prior to mitigate the curse of dimensionality.

As we have shown in a comprehensive set of experiments, SAASBO outperforms state-of-the-art BO methods on several synthetic and real-world problems. Our approach provides several distinct advantages: we highlight three. First, it preserves—and therefore can exploit—structure in the input domain, in contrast to methods like ALEBO or HeSBO which risk scrambling it. Second, it is adaptive and exhibits little sensitivity to its hyperparameters. Third, it can naturally accommodate both input and output constraints, in contrast to methods that rely on random projections, for which input constraints are particularly challenging.

While we have obtained strikingly good performance using a simple acquisition strategy, it is likely that making the most of our SAAS function prior will require a decision-theoretic framework that is better suited to high-dimensional settings. This is an interesting direction for future elaborations of SAASBO.

Acknowledgements

References

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. *arXiv preprint arXiv:1910.06403*, 2019.

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 33*, 2020.

Ricardo Baptista and Matthias Poloczek. Bayesian optimization of combinatorial structures. volume 80 of *Proceedings of Machine Learning Research*, pages 471–480. PMLR, 2018.

Eli Bingham, Jonathan P Chen, Martin Jankowiak, Fritz Obermeyer, Neeraj Pradhan, Theofanis Karaletsos, Rohit Singh, Paul Szerlip, Paul Horsfall, and Noah D Goodman. Pyro: Deep universal probabilistic programming. *The*

Journal of Machine Learning Research, 20(1):973–978, 2019.

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(1):69–90, 2020.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, and Skye Wanderman-Milne. JAX: Composable transformations of Python+NumPy programs, 2018. *URL http://github.com/google/jax*, 4:16, 2020.

Antonio Candelieri, Raffaele Perego, and Francesco Archetti. Bayesian optimization of pump operations in water distribution systems. *Journal of Global Optimization*, 71(1):213–235, 2018.

Carlos M Carvalho, Nicholas G Polson, and James G Scott. Handling sparsity via the horseshoe. In *Artificial Intelligence and Statistics*, pages 73–80. PMLR, 2009.

Bo Chen, Rui Castro, and Andreas Krause. Joint optimization and variable selection of high-dimensional gaussian processes. *arXiv preprint arXiv:1206.6396*, 2012.

Dheeru Dua and Casey Graff. Uci machine learning repository, 2017. *URL: http://archive.ics.uci.edu/ml*, 7(1), 2019.

David Eriksson and Matthias Poloczek. Scalable constrained Bayesian optimization. In *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pages 730–738. PMLR, 2021.

David Eriksson, Michael Pearce, Jacob R. Gardner, Ryan Turner, and Matthias Poloczek. Scalable global optimization via local Bayesian optimization. In *Advances in Neural Information Processing Systems 32*, pages 5497–5508, 2019.

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

Jacob R. Gardner, Chuan Guo, Kilian Q. Weinberger, Roman Garnett, and Roger B. Grosse. Discovering and exploiting additive structure for Bayesian optimization. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pages 1311–1319. PMLR, 2017.

Roman Garnett, Michael A. Osborne, and Philipp Hennig. Active learning of linear embeddings for Gaussian processes. In *Proceedings of the Thirtieth Conference on Uncertainty in Artificial Intelligence*, pages 230–239. AUAI Press, 2014.

Nikolaus Hansen, Sibylle D Müller, and Petros Koumoutsakos. Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES). *Evolutionary computation*, 11(1):1–18, 2003.

José Miguel Hernández-Lobato, James Requeima, Edward O. Pyzer-Knapp, and Alán Aspuru-Guzik. Parallel and distributed Thompson sampling for large-scale accelerated exploration of chemical space. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1470–1479. PMLR, 2017.

Matthew D Hoffman and Andrew Gelman. The No-U-Turn sampler: Adaptively setting path lengths in Hamiltonian Monte Carlo. *J. Mach. Learn. Res.*, 15(1):1593–1623, 2014.

Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *International conference on learning and intelligent optimization*, pages 507–523. Springer, 2011.

Donald R Jones. Large-scale multi-disciplinary mass optimization in the auto industry. In *MOPTA 2008 Conference (20 August 2008)*, 2008.

Donald R Jones, Matthias Schonlau, and William J Welch. Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, 13(4):455–492, 1998.

Kirthevasan Kandasamy, Jeff G. Schneider, and Barnabás Póczos. High dimensional Bayesian optimisation and bandits via additive models. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 295–304. JMLR.org, 2015.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations*, 2015.

Johannes Kirschner, Mojmir Mutny, Nicole Hiller, Rasmus Ischebeck, and Andreas Krause. Adaptive and safe bayesian optimization in high dimensions via one-dimensional subspaces. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 3429–3438. PMLR, 2019.

Benjamin Letham, Brian Karrer, Guilherme Ottoni, Eytan Bakshy, et al. Constrained Bayesian optimization with noisy experiments. *Bayesian Analysis*, 14(2):495–519, 2019.

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 33*, 2020.

Cheng Li, Sunil Gupta, Santu Rana, Vu Nguyen, Svetha Venkatesh, and Alistair Shilton. High dimensional Bayesian optimization using dropout. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*, pages 2096–2102. ijcai.org, 2017.

Xiaoyu Lu, Javier Gonzalez, Zhenwen Dai, and Neil D. Lawrence. Structured variationally auto-encoded optimization. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 3273–3281. PMLR, 2018.

David JC MacKay and Radford M Neal. Automatic relevance determination for neural networks. In *Technical Report in preparation*. Cambridge University, 1994.

Jonas Mockus, Vytautas Tiesis, and Antanas Zilinskas. Toward global optimization, volume 2, chapter Bayesian methods for seeking the extremum, 1978.

Mojmir Mutny and Andreas Krause. Efficient high dimensional Bayesian optimization with additivity and quadrature Fourier features. In *Advances in Neural Information Processing Systems 31*, pages 9019–9030, 2018.

Amin Nayebi, Alexander Munteanu, and Matthias Poloczek. A framework for Bayesian optimization in embedded subspaces. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 4752–4761. PMLR, 2019.

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

ChangYong Oh, Efstratios Gavves, and Max Welling. BOCK: Bayesian optimization with cylindrical kernels. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 3865–3874. PMLR, 2018.

ChangYong Oh, Jakub M. Tomczak, Efstratios Gavves, and Max Welling. Combinatorial Bayesian optimization using the graph cartesian product. In *Advances in Neural Information Processing Systems 32*, pages 2910–2920, 2019.

Art B Owen. Quasi-Monte Carlo sampling. *Monte Carlo Ray Tracing: Siggraph*, 1:69–88, 2003.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in Python. *The Journal of machine Learning research*, 12:2825–2830, 2011.

Du Phan, Neeraj Pradhan, and Martin Jankowiak. Composable effects for flexible and accelerated probabilistic programming in NumPyro. *arXiv preprint arXiv:1912.11554*, 2019.

Hong Qian, Yi-Qi Hu, and Yang Yu. Derivative-free optimization of high-dimensional non-convex functions by sequential random embeddings. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, pages 1946–1952. IJCAI/AAAI Press, 2016.

Carl Edward Rasmussen. Gaussian processes in machine learning. In *Summer School on Machine Learning*, pages 63–71. Springer, 2003.

Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. Practical Bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems 25*, pages 2960–2968, 2012.

Tsuyoshi Ueno, Trevor David Rhone, Zhufeng Hou, Teruyasu Mizoguchi, and Koji Tsuda. COMBO: An efficient Bayesian optimization library for materials science. *Materials discovery*, 4:18–21, 2016.

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*, volume 84 of *Proceedings of Machine Learning Research*, pages 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.

Xinjie Yu and Mitsuo Gen. *Introduction to evolutionary algorithms*. Springer Science & Business Media, 2010.

Ciyou Zhu, Richard H Byrd, Peihuang Lu, and Jorge Nocedal. Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization. *ACM Transactions on Mathematical Software (TOMS)*, 23(4): 550–560, 1997.

A INFERENCE

A.1 NUTS

We use the NUTS sampler implemented in NumPyro [Phan et al., 2019, Bingham et al., 2019], which leverages JAX for efficient hardware acceleration [Bradbury et al., 2020]. In most of our experiments (see Sec. E for exceptions) we run NUTS for $768 = 512 + 256$ steps where the first $N_{\text{warmup}} = 512$ samples are for burn-in and (diagonal) mass matrix adaptation (and thus discarded), and where we retain every 16th sample among the final $N_{\text{post}} = 256$ samples (i.e. sample thinning), yielding a total of $L = 16$ approximate posterior samples. It is these L samples that are then used to compute Eqns. (4), (5), (10). We also limit the maximum tree depth in NUTS to 6.

We note that these choices are somewhat conservative, and in many settings we would expect good results with fewer samples. Indeed on the Branin test function, see Fig. 6, we find a relatively marginal drop in performance when we reduce the NUTS sampling budget as follows: i) reduce the number of warmup samples from 512 to 128; ii) reduce the number of post-warmup samples from 256 to 128; and iii) reduce the total number of retained samples from 16 to 8. We expect broadly similar results for many other problems. See Sec. C for corresponding runtime results.

It is worth emphasizing that while SAASBO requires specifying a few hyperparameters that control NUTS, these hyperparameters are purely computational in nature, i.e. they have no effect on the SAAS function prior. Users simply choose a value of L that meets their computational budget. This is in contrast to e.g. the embedding dimension d_e that is required by ALEBO and HeSBO: the value of d_e often has significant effects on optimization performance.

To improve the geometry of the joint density defined by the model—and thus make NUTS more efficient—we reparameterize the prior in Eqn. 12 as follows

[global shrinkage]	$\tau \sim \mathcal{HC}(\alpha)$	(12)
[reparameterized length scales]	$\tilde{\rho}_i \sim \mathcal{HC}(1)$	
[effective length scales]	$\rho_i = \tau \times \tilde{\rho}_i$	

where we note that the final equation is a deterministic equality and HMC is performed in the coordinate system defined by $\tilde{\rho}_i$. Note that this sort of reparameterization can be implemented in NumPyro using the `deterministic` primitive.

We also note that it is possible to make SAASBO-NUTS faster by means of the following modifications:

1. Warm-start mass adaptation with mass matrices from previous iterations.

2. Instead of fitting a new SAAS GP at each iteration, only fit every M iterations (say $M = 5$), and reuse

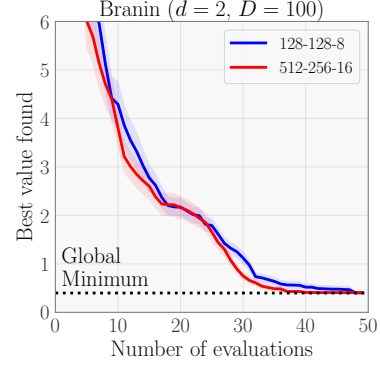hyperparameter samples $\{\psi_\ell\}$ across M iterations of SAASBO.

Figure 6: We depict how SAASBO-NUTS performs on Branin as we reduce the sampling budget $(N_{\text{warmup}}, N_{\text{post}}, L) = (512, 256, 16)$ to $(N_{\text{warmup}}, N_{\text{post}}, L) = (128, 128, 8)$. We compare performance w.r.t. the best minimum found (the mean is depicted by a thick line and shaded bands denote standard errors). Each curve corresponds to 60 independent replications of Algorithm 1.

A.2 MAP

We run the Adam optimizer [Kingma and Ba, 2015] for 1500 steps and with a learning rate of 0.02 and $\beta_1 = 0.50$ to maximize the log density

$$U_s(\psi_s|\tau_s) = \log p(\mathbf{y}|\mathbf{X}, \psi_s) + \log p(\psi_s|\tau_s) \qquad (13)$$

w.r.t. ψ_s for $S = 4$ pre-selected values of τ_s: $\tau_s \in \{1, 10^{-1}, 10^{-2}, 10^{-3}\}$. This optimization is trivially optimized across S.

For each $s = 1, ..., S$ we then compute the leave-one-out predictive log likelihood using the mean and variance functions given in Eqns. (4)-(5). We then choose the value of s that maximizes this predictive log likelihood and use the corresponding kernel hyperparameter ψ_s to compute the expected improvement in Eqn. (10).

A.3 NO DISCRETE LATENT VARIABLES

As discussed briefly in the main text, it is important that the SAAS prior defined in Sec. 4.1 does not include any discrete latent variables. Indeed a natural alternative to our model would introduce D binary-valued latent variables that control whether or not a given dimension is relevant to modeling f_{obj}. However, inference in any such model can be very challenging, as it requires exploring an extremely large discrete space of size 2^D. Our model can be understood as a continuous relaxation of such an approach. This

is a significant advantage since it means we can leverage gradient information to efficiently explore the posterior. Indeed, the structure of our sparsity-inducing prior closely mirrors the justly famous Horseshoe prior [Carvalho et al., 2009], which is a popular prior for Sparse Bayesian linear regression. We note that in contrast to the linear regression setting of the Horseshoe prior, our sparsity-inducing prior governs inverse squared length scales in a non-linear kernel and *not* variances. While we expect that any prior that concentrates ρ_i at zero can exhibit good empirical performance in the setting of high-dimensional BO, this raises the important question whether distributional assumptions other than those in Eqn. (12) may be better suited to governing our prior expectations about ρ_i. Making a careful investigation of this point is an interesting direction for future work.

B   EXPECTED IMPROVEMENT MAXIMIZATION

We first form a scrambled Sobol sequence $\mathbf{x}_{1:Q}$ (see e.g. [Owen, 2003]) of length $Q = 5000$ in the D-dimensional domain $\mathcal{D}$. We then compute the expected improvement in Eqn. (10) in parallel for each point in the Sobol sequence. We then choose the top $K = 3$ points in $\mathbf{x}_{1:Q}$, that yield the largest EIs. For each of these K approximate maximizers we run L-BFGS [Zhu et al., 1997] initialized with the approximate maximizer and using the implementation provided by Scipy (in particular fmin_l_bfgs_b) to obtain the final query point $\mathbf{x}_{\text{next}}$, which (approximately) maximizes Eqn. (10). We limit fmin_l_bfgs_b to use a maximum of 100 function evaluations.

C   RUNTIME EXPERIMENT

We measure the runtime of SAASBO as well as each baseline method on the Branin test problem. See Table 1 for the results. We record runtimes for both the default SAASBO-NUTS settings described in Sec. A.1 as well as one with a reduced NUTS sampling budget. While SAASBO requires more time per iteration than other methods such as TuRBO and HeSBO, the overhead is relatively moderate in the setting where the black-box function f_{obj} is very expensive to evaluate. We note that after reducing the NUTS sampling budget to $(N_{\text{warmup}}, N_{\text{post}}, L) = (128, 128, 8)$ about 75% of the runtime is devoted to EI optimization. Since our current implementation executes $K = 3$ runs of L-BFGS serially, this runtime could be reduced further by executing L-BFGS in parallel.

D   ADDITIONAL FIGURES AND EXPERIMENTS

Table 1: Average runtime per iteration on the Branin test function embedded in a 100-dimensional space. Each method uses $m = 10$ initial points and a total of 50 function evaluations. Runtimes are obtained using a 2.4 GHz 8-Core Intel Core i9 CPU outfitted with 32 GB of RAM.

Method	Time / iteration
SAASBO (default)	26.51 seconds
SAASBO (128-128-8)	19.21 seconds
TuRBO	1.52 seconds
SMAC	12.12 seconds
EBO	128.10 seconds
ALEBO ($d_e = 5$)	4.34 seconds
ALEBO ($d_e = 10$)	11.91 seconds
HeSBO ($d_e = 5$)	0.70 seconds
HeSBO ($d_e = 10$)	1.51 seconds
CMA-ES	< 0.1 seconds
Sobol	< 0.01 seconds

D.1   MODEL FITTING

In Fig. 7 we reproduce the experiment described in Sec. 5.1, with the difference that we replace the RBF kernel with a Matérn-5/2 kernel.
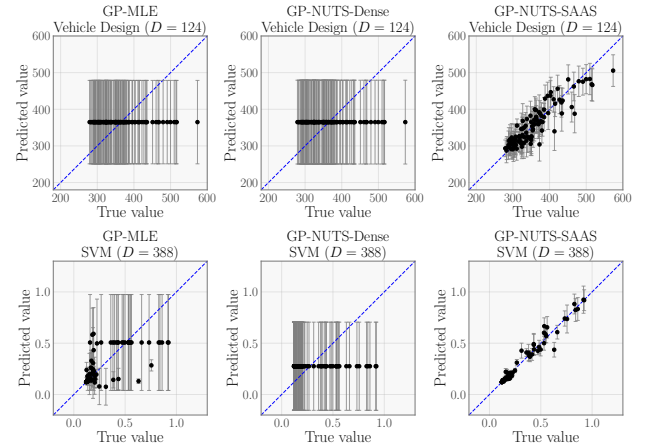

Figure 7: This figure is an exact reproduction of Fig. 1 in the main text apart from the use of a Matérn-5/2 kernel instead of a RBF kernel. We compare: (left) a GP fit with MLE; (middle) a GP with weak priors fit with NUTS; and (right) a GP with a SAAS prior (this paper; see Eqn. (12)) fit with NUTS. For the vehicle design problem we use 100 training points and for the SVM problem we use 50 training points. We use 100 test points for both problems. Only SAAS provides a good fit. In each figure mean predictions are depicted with dots and bars denote 95% confidence intervals.

We note that the qualitative behavior in Fig. 7 matches the behavior in Fig. 1. In particular, only the sparsity-inducing

SAAS function prior provides a good fit. This emphasizes that the potential for drastic overfitting that arises when fitting a non-sparse GP in high dimensions is fundamental and is not ameliorated by using a different kernel. In particular the fact that the Matérn-5/2 kernel decays less rapidly at large distances as compared to the RBF kernel (quadratically instead of exponentially) does not prevent the non-sparse models from yielding essentially trivial predictions across most of the domain $\mathcal{D}$.

D.2 SVM RELEVANCE PLOTS

In Fig. 8 we explore the relevant subspace identified by SAASBO during the course of optimization of the SVM problem discussed in Sec. 5.6. We see that the three most important hyperparameters, namely the regularization hyperparameters, are consistently found more or less immediately once the initial Sobol phase of Algorithm 1 is over. This explains the rapid early progress that SAASBO makes in Fig. 4 during optimization. We note that the 4th most relevant dimension turns out to be a length scale for a patient ID feature, which makes sense given the importance of this feature to the regression problem.
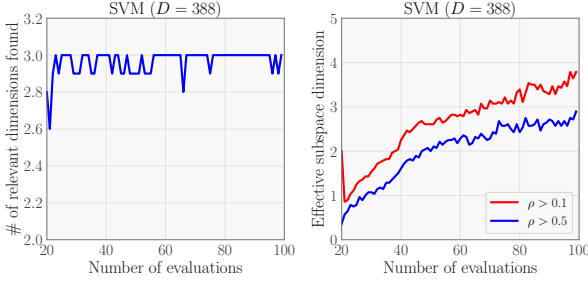

Figure 8: **Left**: We depict the mean number of regularization hyperparameters that have been 'found' in the SVM problem, where a regularization hyperparameter is 'found' if its corresponding $\text{PosteriorMedian}(\rho_{\text{k}})$ is among the three largest $\{\text{PosteriorMedian}(\rho_{\text{i}})\}_{\text{i}=1}^{\text{D}}$. Note that there are three regularization hyperparameters in total. **Right**: We depict the mean effective subspace dimension, defined to be the number of dimensions for which $\text{PosteriorMedian}(\rho_{\text{k}}) > \xi$ where $\xi \in \{0.1, 0.5\}$ is an arbitrary cutoff. Means are averages across 30 independent replications.

D.3 SVM ABLATION STUDY

In Fig. 9 we depict results from an ablation study of SAASBO in the context of the SVM problem. First, as a companion to Fig. 1 and Fig. 7, we compare the BO performance of the SAAS function prior to a non-sparse function prior that places weak priors on the length scales. As we

would expect from Fig. 1 and Fig. 7, the resulting BO performance is very poor for the non-sparse prior. Second, we also compare the default RBF kernel to a Matérn-5/2 kernel. We find that, at least on this problem, both kernels lead to similar BO performance.
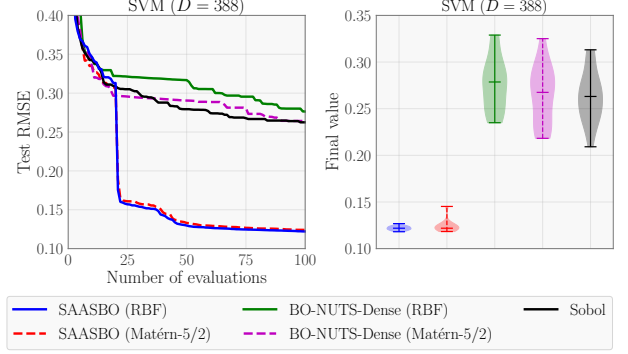

Figure 9: We compare the BO performance of the SAAS function prior to a non-sparse function prior on the SVM hyperparameter tuning problem ($D = 388$). In addition we compare the RBF kernel to the Matérn-5/2 kernel. We do 15 independent replications for each method, except for SAASBO-RBF and Sobol, for which we reproduce the same 30 replications from the main text. **Left:** For each method we depict the mean value of the best minimimum found at a given iteration. **Right:** For each method we depict the distribution over the final approximate minimum y_{min} encoded as a violin plot, with horizontal bars corresponding to 5%, 50%, and 95% quantiles.

D.4 ROTATED HARTMANN

In this experiment we study whether the axis-aligned assumption in SAAS leads to degraded performance on non-axis-aligned objective functions. In particular, we consider the Hartmann function f_{hart} for $d = 6$ embedded in $D = 100$ dimensions. Given a projection dimensionality $d_p \geq d$, we generate a random linear projection $P_{d_p} \in \mathbb{R}^{d \times d_p}$ where $[P_{d_p}]_{ij} \sim \mathcal{N}(0, 1/d_p)$. The task is to optimize $\tilde{f}(\mathbf{x}) = f_{\text{hart}}(P_{d_p}\mathbf{x}_{1:d_p} - \mathbf{z}))$ where $\mathbf{x} \in [0, 1]^D$ and $\mathbf{z} \in \mathbb{R}^d$. For a given P_{d_p}, $\mathbf{z}$ is a vector in $[0, 1]^d$ that satisfies $\tilde{f}([\mathbf{x}^*; w]) = f_{\text{hart}}(\mathbf{x}^*), \forall w \in [0, 1]^{D-d}$ where $\mathbf{x}^*$ is the global optimum of the Hartmann function. The translation $\mathbf{z}$ guarantees that the global optimum value is attainable in the domain. We consider $d_p \in \{6, 18, 30\}$ and generate a random P_{d_p} and $\mathbf{z}$ for each embedded dimensionality; these are then used for all replications. EBO is excluded from this study, as it performed worse than Sobol in Fig. 3.

The results are shown in Fig. 10. We see that SAASBO outperforms the other methods even though the function has been rotated, thus straining the axis-aligned assumption. Despite the rotation, SAASBO quickly identifies the most

important parameters in the rotated space. We also notice that the worst-case performance of SAASBO is better than for the other methods across all projection dimensionalities considered.

E ADDITIONAL EXPERIMENTAL DETAILS

Apart from the experiment in Sec. 5.2 that is depicted in Fig. 2 we use $\alpha = 0.1$ in all experiments. Apart from Fig. 7 and Fig. 9, we use an RBF kernel in all experiments.

E.1 MODEL FIT EXPERIMENT

In the model fit experiment in Sec. 5.1 we take data collected from two different runs of SAASBO in $D = 100$. We use one run as training data and the second run as test data, each with $N = 100$ datapoints. To construct datasets in $D = 30$ dimensions we include the 6 relevant dimensions as well as 24 randomly chosen redundant dimensions and drop all remaining dimensions.

E.2 INFERENCE AND HYPERPARAMETER COMPARISON EXPERIMENT

For the experiment in Sec. 5.2 that is depicted in Fig. 2 we initialize SAASBO with $m = 10$ points from a Sobol sequence.

E.3 BASELINES

We compare SAASBO to ALEBO, CMA-ES, EBO, HeSBO, SMAC, Sobol, and TuRBO. For ALEBO and HeSBO we use the implementations in BoTorch [Balandat et al., 2020] with the same settings that were used by [Letham et al., 2020]. We consider embeddings of dimensionality $d_e = 5$ and $d_e = 10$ on the synthetic problems, which is similar to the $d_e = d$ and $d_e = 2d$ heuristics that were considered in [Nayebi et al., 2019] as well as [Letham et al., 2020]. As the true active dimensionality d of f_{obj} is unknown, we do not allow any method to explicitly use this additional information. For the three real-world experiments, $d_e = 5$ does not work well on any problem so we instead report results for $d_e = 10$ and $d_e = 20$.

For CMA-ES we use the pycma[5] implementation. CMA-ES is initialized using a random point in the domain and uses the default initial step-size of 0.25. Recall that the domain is normalized to $[0, 1]^D$ for all problems. We run EBO using the reference implementation by the authors[6]

with the default settings. EBO requires knowing the value of the function at the global optimum. Similarly to [Letham et al., 2020] we provide this value to EBO for all problems, but note that EBO still performs poorly on all problems apart from Branin and SVM.

Our comparison to SMAC uses SMAC4HPO, which is implemented in SMAC3[7]. On all problems we run SMAC in deterministic mode, as all problems considered in this paper are noise-free. For Sobol we use the SobolEngine implementation in PyTorch. Finally, we compare to TuRBO with a single trust region due to the limited evaluation budget; we use the implementation provided by the authors[8].

E.4 SYNTHETIC PROBLEMS

We consider three standard synthetic functions from the optimization literature. Branin is a 2-dimensional function that we embed in a 100-dimensional space. We consider the standard domain $[-5, 10] \times [0, 15]$ before normalizing the domain to $[0, 1]^{100}$. For Hartmann, we consider the $d = 6$ version on the domain $[0, 1]^6$ before embedding it in a 100-dimensional space. For Rosenbrock, we use $d = 3$ and the domain $[-2, 2]^3$, which we then embed and normalize so that the full domain is $[0, 1]^{100}$. Rosenbrock is a function that is challenging to model, as there are large function values at the boundary of the domain. For this reason all methods minimize $\log(1 + f_{obj}(x))$. All methods except for CMA-ES are initialized with $m = 10$ initial points for Branin and Rosenbrock and $m = 20$ initial points for Hartmann.

E.5 ROVER

We consider the rover trajectory optimization problem that was also considered in Wang et al. [2018]. The goal is to optimize the trajectory of a rover where this trajectory is determined by fitting a B-spline to 30 waypoints in the 2D plane. While the original problem had a pre-determined origin and destination, the resulting B-spline was not constrained to start and end at these positions. To make the problem easier, we force the B-spline to start and end at these pre-determined positions. Additionally, we use 50 waypoints points, which results in a 100-dimensional optimization problem. The reward function for the trajectory is computed in the same way as in Wang et al. [2018], namely we integrate over the trajectory penalizing collisions with potential objects. On this problem we initialize all methods except for CMA-ES with $m = 20$ initial points.

---

[5] https://github.com/CMA-ES/pycma
[6] https://github.com/zi-w/Ensemble-Bayesian-Optimization

[7] https://github.com/automl/SMAC3
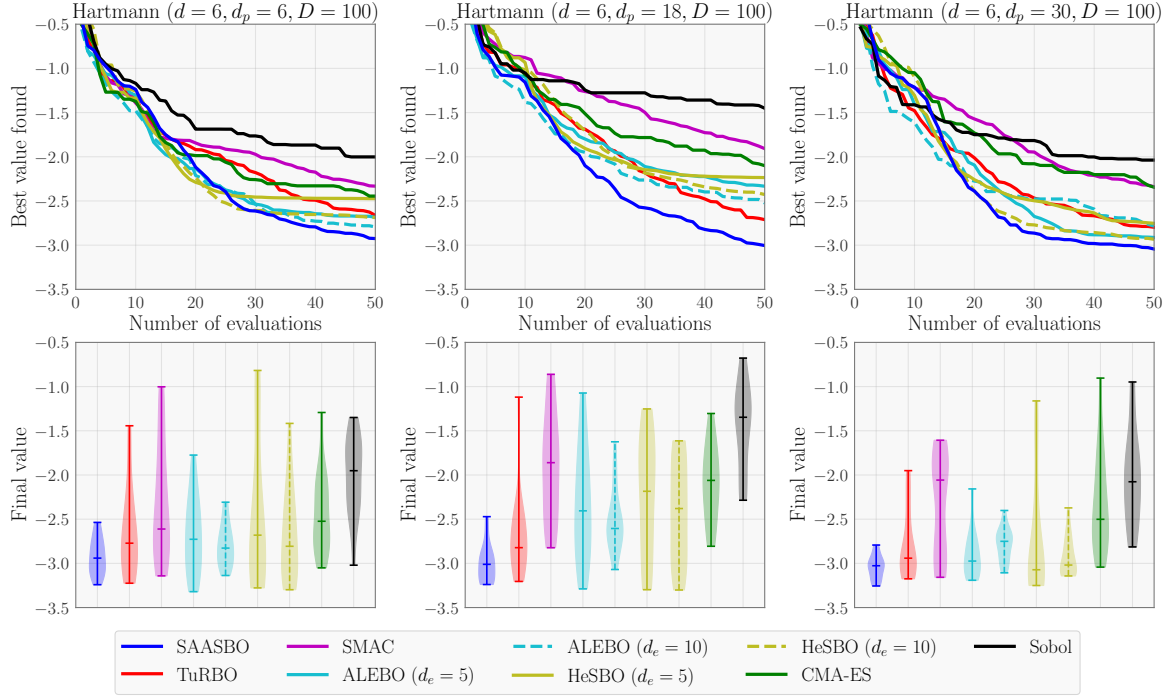[8] https://github.com/uber-research/TuRBO

Figure 10: We consider a rotated version of the Hartmann function with $d = 6$. We generate random linear projection matrices P_{d_p} for different projection dimensionalities $d_p \in \{6, 18, 30\}$ and optimize the resulting rotated function. SAASBO outperforms the other methods and is able to quickly identify the most important parameters in the rotated coordinate system.

E.6 SVM

We randomly choose 5000 training and 5000 test points from the 385-dimensional "CT slice"[9] UCI dataset [Dua and Graff, 2019]. We normalize the inputs and scalar output so that e.g. the test RMSE of a trivial zero prediction is given by 1.0. Our domain $\mathcal{D}$ then consists of 385 kernel (log) length scales and 3 regularization hyperparameters for a kernel support vector machine fit with Scikit-learn [Pedregosa et al., 2011]. The log length scales are restricted to the interval $[-2, 2]$. The 3 regularization hyperparameters, which are likewise represented in log space, are denoted epsilon, C, and gamma in the SVR class constructor. We restrict epsilon to $[0.01, 1.0]$, gamma to $[0.1, 3.0]$, and C to $[0.01, 5.0]$. Aftering fitting the SVM regressor to the training data we compute the test RMSE (root mean squared error). This test RMSE is the quantity we seek to minimize. We use the default settings of SVR, which among other things means the kernel used is a RBF kernel. On this problem we initialize all methods except for CMA-ES with $m = 20$ initial points.

E.7 MOPTA VEHICLE DESIGN

We consider the vehicle design problem MOPTA08 which is a challenging 124-dimensional real-world high-dimensional BO problem [Jones, 2008]. The goal in this problem is to minimize the mass of a vehicle subject to 68 performance constraints. The $D = 124$ design variables describe materials, gauges, and vehicle shape. While this problem is originally formulated as a constrained optimization problem, we make it unconstrained by converting the constraints into a soft constraint. In particular, we consider minimizing $f_{\text{obj}}(x) + 10 \sum_{i=1}^{68} \max(0, c_i(x))$ where the 68 constraints are of the form $c_i(x) \leq 0$. This penalty is chosen to be small enough to have most of the signal come from f_{obj} while at the same time discouraging large constraint violations. While it is worth emphasizing that there are constrained optimization methods that can explicitly handle the constraint, this problem shows that SAASBO can quickly exploit structure in f_{obj} even though there is no obvious low-dimensional structure.

For SAASBO we use the NUTS settings described in Sec. A.1 for $t \leq 150$. To lower the runtime after iteration $t > 150$ we collect $384 = 192 + 192$ NUTS samples and retain every 24th of the final 192 samples, resulting in a total of $L = 8$ retained samples. We note while this may hurt the accuracy of the inferred GP model, SAASBO

---

[9]https://archive.ics.uci.edu/ml/datasets/
Relative+location+of+CT+slices+on+axial+
axis

still performs very well on this problem and outperforms other methods by a large margin. As we consider a larger evaluation budget on this problem we initialize all methods except for CMA-ES with $m = 50$ initial points.