Machine Learning (2020) 109:1925–1943
https://doi.org/10.1007/s10994-020-05899-z

High-dimensional Bayesian optimization using low-dimensional feature spaces

Riccardo Moriconi[1] · Marc Peter Deisenroth[2] · K. S. Sesh Kumar[3]

Received: 3 November 2019 / Revised: 29 July 2020 / Accepted: 11 August 2020 /
Published online: 21 September 2020

Abstract
Bayesian optimization (BO) is a powerful approach for seeking the global optimum of expensive black-box functions and has proven successful for fine tuning hyper-parameters of machine learning models. However, BO is practically limited to optimizing 10–20 parameters. To scale BO to high dimensions, we usually make structural assumptions on the decomposition of the objective and/or exploit the intrinsic lower dimensionality of the problem, e.g. by using linear projections. We could achieve a higher compression rate with nonlinear projections, but learning these nonlinear embeddings typically requires much data. This contradicts the BO objective of a relatively small evaluation budget. To address this challenge, we propose to learn a low-dimensional feature space jointly with (a) the response surface and (b) a reconstruction mapping. Our approach allows for optimization of BO's acquisition function in the lower-dimensional subspace, which significantly simplifies the optimization problem. We reconstruct the original parameter space from the lower-dimensional subspace for evaluating the black-box function. For meaningful exploration, we solve a constrained optimization problem.

1 Introduction

Bayesian optimization (BO) is a useful model-based approach to global optimization of black-box functions, which are expensive to evaluate (Kushner 1964; Jones et al. 1998). This sample-efficient technique for optimization has been effective in experimental

Editors: Ira Assent, Carlotta Domeniconi, Aristides Gionis, Eyke Hüllermeier.

✉ Riccardo Moriconi
r.moriconi16@imperial.ac.uk

Marc Peter Deisenroth
m.deisenroth@ucl.ac.uk

K. S. Sesh Kumar
s.karri@imperial.ac.uk

[1] Department of Computing, Imperial College London, London, UK

[2] Department of Computer Science, University College London, London, UK

[3] Data Science Institute, Imperial College London, London, UK

design of machine learning algorithms (Bergstra et al. 2011), robotics applications (Cully et al. 2015; Calandra et al. 2016b) and medical therapies (Sui et al. 2015) for optimization of spinal-cord electro-stimulation. Despite its great success, BO is practically limited to optimizing 10–20 parameters. A large body of literature has been devoted to address scalability issues to elevate BO to high-dimensional optimization problems, such as discovery of chemical compounds (Gomez-Bombarelli et al. 2018) or automatic software configuration (Hutter et al. 2011).

The standard BO routine consists of two key steps: (1) estimating the black-box function from data through a probabilistic surrogate model, usually a Gaussian process (GP), referred to as the *response surface*; (2) maximizing an *acquisition function* that trades off exploration and exploitation according to uncertainty and optimality of the response surface. As the dimensionality of the input space increases, these two steps become challenging. The sample complexity to ensure good coverage of inputs for learning the response surface is exponential in the number of dimensions (Shahriari et al. 2016). With only a small evaluation budget, the learned response surface and the resulting acquisition function are characterized by vast flat regions interspersed with highly non-convex landscapes (Rana et al. 2017). This renders the maximization of the acquisition in high dimensions inherently hard (Garnett et al. 2014).

High-dimensional optimization is often translated into low-dimensional problems, which are defined on subsets of variables (Moriconi et al. 2020; Kandasamy et al. 2015; Rolland et al. 2018). These approaches apply a divide and conquer approach to decompose the problem into independent (Moriconi et al. 2020; Kandasamy et al. 2015) and potentially dependent components (Rolland et al. 2018). However, high-dimensional data often possesses a lower intrinsic dimensionality, which can be exploited for optimization. A feature mapping can then be used to map the original D-dimensional data onto a $d \ll D$-dimensional manifold. For example, in Wang et al. (2013), the authors used random linear mappings to reduce dimensionality of the optimization problem. Similar approaches, which use linear dimensionality reduction, drive exploration in BO to actively learn this linear embedding (Garnett et al. 2014). While these methods perform well in practice, they are restricted to linear subspaces of the original domain. With nonlinear embeddings, higher compression rates are possible. In our work, we focus on this nonlinear setting.

Using BO with nonlinear feature spaces was proposed in Gomez-Bombarelli et al. (2018), Gonzalez et al. (2015), Kusner et al. (2017) and Griffiths and Hernández-Lobato (2017). In Gomez-Bombarelli et al. (2018), a low-dimensional data representation is learned with variational autoencoders (VAEs) (Rezende et al. 2014; Kingma and Welling 2014). However, this approach requires both large amounts of data and learning the model offline without the possibility to update the learnt feature space during optimization. Nevertheless, in the specific application of automatic discovery of molecules, where libraries of existing compounds are readily available prior to optimization, this approach makes much sense. To accommodate fairly small evaluation budgets, in our work, we exploit a probabilistic model based on GPs, which features superior data efficiency with respect to VAE-based approaches (Gomez-Bombarelli et al. 2018; Gonzalez et al. 2015; Kusner et al. 2017; Griffiths and Hernández-Lobato 2017). VAE models (Lu et al. 2018) were used to propagate uncertainty of latent space representations through the response surface model with Gaussian process latent variable models (Lawrence 2005; Titsias and Lawrence 2010; Lawrence and Quiñonero-Candela 2006). However, in Lu et al. (2018), the latent space

representation is not learned specifically for the regression task (learning the response surface). Gradient-based methods (Abbati et al. 2018) have been used to learn a lower-dimensional Riemannian manifold for optimization and sampling.

Nonlinear embeddings also allow for modeling non-stationary objective functions. In this context, a hierarchical composition of GPs, referred to as *deep GPs* (Damianou and Lawrence 2013; Salimbeni and Deisenroth 2017; Dai et al. 2016; Damianou 2015; Hensman and Lawrence 2014), is especially useful when the response surface is characterized by abrupt changes or has constraints. An extensive investigation on the employment of deep GP models in BO is presented in Dai et al. (2016) and Hebbal et al. (2019). In our work, we also exploit the idea of learning highly nonlinear functions through the composition of simpler functions (LeCun et al. 2015), but we focus on deterministic dimensionality reduction and optimization in feature space.

In this paper, we propose a BO algorithm for high-dimensional optimization, which learns a nonlinear feature mapping $\mathbf{h} : \mathbb{R}^D \to \mathbb{R}^d$ to reduce the dimensionality of the inputs, and a *reconstruction mapping* $\mathbf{g} : \mathbb{R}^d \to \mathbb{R}^D$ based on GPs to evaluate the true objective function, jointly, see Fig. 1. This allows us to optimize the acquisition function in a lower-dimensional feature space, so that the overall BO routine scales to high-dimensional problems that possess an intrinsic lower dimensionality. Finally, we use constrained maximization of the acquisition function in feature space to prevent meaningless reconstructions.

2 Bayesian optimization

Bayesian optimization is a powerful tool for globally optimizing black-box functions that are expensive to evaluate (Jones et al. 1998; Kushner 1964; Močkus 1975). In our setting, we consider the global minimization problem

$$\mathbf{x}^* = \arg\min_{\mathbf{x}\in\mathcal{X}} f_X(\mathbf{x}) \tag{1}$$

with input space $\mathcal{X} = [0,1]^D$ and objective function $f_X : \mathcal{X} \to \mathbb{R}$. We consider functions f_X that are costly to evaluate and for which we are allowed a small budget of evaluation queries to express our best guess of the optimum's location $\mathbf{x}^*$ in at most T_{end} iterations. We further assume we have access only to noisy evaluations of the objective $y = f_X + \varepsilon$, where $\varepsilon \sim \mathcal{N}(0, \sigma_n^2)$ is i.i.d. Gaussian measurement noise with variance σ_n^2. We restrict ourselves

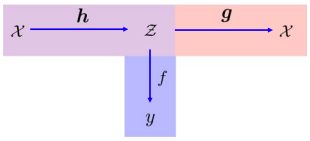


Fig. 1 Model for Bayesian optimization on data manifolds, jointly solving two distinct tasks: (1) a regression from feature space to observations (in blue) and (2) a reconstruction mapping from feature space to high-dimensional space (in red) (Color figure online)

to the typical setting, where neither gradients nor convexity properties of f_X are available.

Algorithm 1 Key steps of Bayesian optimization in feature space. The response surface learning and acquisition function maximization are performed in feature space with dimensionality $d \ll D$. The reconstruction step in line 9 allows us to run experiments with the original objective function, f_X.

1: **Inputs:** $\mathbf{X}_0 \in \mathbb{R}^{N_0 \times D}$, $\mathbf{y}_0 \in \mathbb{R}^{N_0}$
2: **for** $t = 0, 1, 2, ..., T_{\text{end}}$ **do**
3: **Response surface learning**
4: $f_X = f_Z \circ \mathbf{h}$ ▷ Composition of feature map and low-dimensional response surface
5: $\mathbf{Z}_t = \mathbf{h}(\mathbf{X}_t)$ ▷ Dimensionality reduction
6: $p(f_Z|\mathbf{Z}_t, \mathbf{y}_t)$ ▷ Learning of the response surface in low-dimensional feature space
7: **Optimal input selection** $\mathbf{x}_{t+1}$
8: $\mathbf{z}_* = \underset{\mathbf{z}\in\mathcal{Z}}{\text{argmax}}\ \alpha(\mathbf{z})$ ▷ Acquisition function maximization in feature space
9: $\mathbf{x}_{t+1} := \mathbf{g}(\mathbf{z}_*)$ ▷ Reconstruction of high-dimensional input
10: **Evaluation**
11: $y_{t+1} = f_X(\mathbf{x}_{t+1}) + \varepsilon$ ▷ Evaluation of noisy high-dimensional objective function
12: $\mathbf{X}_t \cup \{\mathbf{x}_{t+1}\}, \quad \mathbf{y}_t \cup \{y_{t+1}\}$
13: **end for**
14: **Return** $\mathbf{x}^* = \arg\min \mathbf{y}_t$ ▷ Minimizer of the objective function f_X

The main steps of a BO routine at iteration t involve (1) *response surface learning*, (2) *optimal input selection* $\mathbf{x}_{t+1}$ and (3) *evaluation* of the objective function f_X at $\mathbf{x}_{t+1}$. The first step trains a probabilistic surrogate model $p(f_X)$, the response surface, which describes the black-box relationship between inputs $\mathbf{x}$ and observations y. In the $(t+1)$st iteration of BO, the optimal input selection step finds an input $\mathbf{x}_{t+1}$ that maximizes an *acquisition function* $\alpha(\cdot)$, which describes the added value of input $\mathbf{x}_{t+1}$. The evaluation step returns a noisy observation of the true objective function $f_X(\mathbf{x}_{t+1}) + \varepsilon$ at the selected location. These steps are summarized in lines 4, 7 and 10 of Algorithm 1, respectively. Having defined a probabilistic surrogate model for our objective function, which is usually modeled by a GP (Rasmussen and Williams 2006), we can compute posterior predictions of objective function values at test locations. These posterior predictions are then fed to the acquisition function, which drives exploration during optimization. Posterior predictions of the GP are Gaussian distributed with mean μ and variance σ^2. Defining $Z(\mathbf{x}) := (f_{\min} - \mu(\mathbf{x}))/\sigma(\mathbf{x})$ and $f_{\min} := \min_{\mathbf{x}\in\mathbf{X}_t} f(\mathbf{x})$, this allows us to define three different acquisition functions to maximize:

$$\alpha(\mathbf{x}) = \Phi(Z(\mathbf{x})) \text{ Probability of improvement (PI) (Kushner 1964)} \tag{2}$$

$$\alpha(\mathbf{x}) = \sigma(\mathbf{x})Z(\mathbf{x})\Phi(Z(\mathbf{x})) + \sigma(\mathbf{x})\phi(Z(\mathbf{x})) \text{ Expected improvement (EI) (Mockus 1975)} \tag{3}$$

$$\alpha(\mathbf{x}) = -\mu(\mathbf{x}) + \beta_t\sigma(\mathbf{x}) \text{ Upper confidence bound (UCB) (Srinivas et al. 2010).} \tag{4}$$

Here, ϕ and Φ denote the probability density function and the cumulative density function of the standard normal $\mathcal{N}(0, 1)$, respectively. The parameter β_t controls the exploration exploitation trade-off. For a complete review on acquisition function the reader is referred to Shahriari et al. (2016). In high-dimensional settings ($D > 20$), both the response surface learning and optimal input selection via optimization of the acquisition function are computationally challenging.

3 Bayesian optimization in low-dimensional feature spaces

In this section, we consider a setting, where the input space $\mathcal{X}$ is high-dimensional and the objective function possesses an intrinsic lower dimensionality. In our work, we exploit the effective low dimensionality of the objective function for BO in a lower-dimensional *feature space* $\mathcal{Z} \subset \mathbb{R}^d$, where $d \ll D$. In particular, we express the true objective $f_X : \mathbb{R}^D \to \mathbb{R}$ as a composition of a feature mapping $\mathbf{h} : \mathbb{R}^D \to \mathbb{R}^d$ and a function $f_Z : \mathcal{Z} \to \mathbb{R}$ so that $f_X = f_Z \circ \mathbf{h}$. The lower-dimensional feature space allows for both learning the response surface f_X and maximizing an acquisition function α with domain $\mathcal{Z}$, which yields optimizer $\mathbf{z}_*$. Since we cannot evaluate the true objective f_X directly at the low-dimensional features $\mathbf{z}_*$, we project $\mathbf{z}_*$ back into the D-dimensional data space $\mathcal{X}$ by means of a *reconstruction* mapping $\mathbf{g} : \mathcal{Z} \to \mathcal{X}$. We can think of this mapping as a decoder within an auto-encoder framework. We model both the composition $f_X := f_Z \circ \mathbf{h}$ and the reconstruction with GPs (Rasmussen and Williams 2006). Algorithm 1 summarizes the main steps of this feature-space BO.

In the following, we detail the model (see Fig. 1) for jointly learning the feature map $\mathbf{h}(\cdot)$, the low-dimensional response surface in feature space f_Z, and the reconstruction mapping $\mathbf{g}(\cdot)$.

3.1 Manifold Gaussian processes for response surface learning in feature space

We expect the response surface to predict the value of the black-box objective function f_X with calibrated uncertainty associated with each prediction. GPs are probabilistic models that allow for an analytic computation of posterior predictive function values within a Bayesian framework, and they are the standard model in BO for modeling the response surface.

A GP is a distribution over functions $f_Z \sim \mathcal{GP}(m(\cdot), k(\cdot,\cdot))$ and is fully specified by a *mean function* $m : \mathcal{Z} \to \mathbb{R}$, and a *covariance function/kernel* $k : \mathcal{Z} \times \mathcal{Z} \to \mathbb{R}$. The kernel computes the covariance between pairs of function values as a function of the corresponding inputs, i.e. $\mathrm{Cov}(f_Z(\mathbf{z}), f_Z(\mathbf{z}')) = k(\mathbf{z}, \mathbf{z}')$, and thereby encodes regularity assumptions about f_Z, such as smoothness or periodicity. Common kernel choices in the BO literature include the *squared exponential* and *Matérn* kernels (Frazier 2018).

In our feature space optimization, we phrase lines 5–6 of Algorithm 1 as a single learning problem. Therefore, we need a GP that learns useful representations $\mathbf{z}$ of inputs $\mathbf{x}$ for the regression task together with f_Z. A manifold GP (MGP) (Calandra et al. 2016a; Wilson et al. 2016) addresses this issue by composing two mappings: The deterministic feature map $\mathbf{h}$ with parameters $\boldsymbol{\theta}_h$ and a GP $f_Z \sim \mathcal{GP}(m, k)$ with kernel hyper-parameters $\boldsymbol{\theta}_k$. The GP models the relationship between features $\mathbf{z}$ and function values $y \in \mathbb{R}$ in observation space. The resulting composite model $f_X := f_Z \circ \mathbf{h}$ is a GP so that $f_X \sim \mathcal{GP}(m_M, k_M)$ with mean function given by $m_M(\mathbf{x}) = m(\mathbf{h}(\mathbf{x}))$ and the covariance function given by $k_M(\mathbf{x}, \mathbf{x}') = k(\mathbf{h}(\mathbf{x}), \mathbf{h}(\mathbf{x}'))$ respectively. Given high-dimensional training inputs $\mathbf{X}$ and corresponding observations $\mathbf{y}$ of the objective function, we find model parameters $\{\boldsymbol{\theta}_h, \boldsymbol{\theta}_k\}$ that maximize the marginal likelihood (evidence) $\{\boldsymbol{\theta}_h^*, \boldsymbol{\theta}_k^*\} \in \arg\max_{\boldsymbol{\theta}_h, \boldsymbol{\theta}_k} p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}_h, \boldsymbol{\theta}_k)$. This objective allows us to learn a low-dimensional embedding as a by-product of the supervised GP regression.

Unsupervised dimensionality reduction usually solves an orthogonal task to that of learning a response surface. Algorithms, such as PCA or variational auto-encoders (Rezende et al. 2014), achieve compact data representations by optimizing objectives that are not necessarily useful in a supervised setting (Wahlström et al. 2015). The MGP, instead, leads to low-dimensional representations that are optimal (locally) for the regression task at hand.

We use a multi-layer feed-forward neural network with sigmoid activation functions as a feature map (encoder) $\mathbf{h}$, resulting in a feature space $\mathcal{Z} = [0, 1]^d$. Neural networks as an explicit feature map within an MGP have already been applied successfully for modeling non-smooth responses in robot locomotion (Calandra et al. 2016a; Cully et al. 2015). Deep networks have also proven useful for learning the orientation of images from high-dimensional images (Wilson et al. 2016). With a Gaussian likelihood, the MGP posterior predictive distribution at a test point $\mathbf{x}_\star \in \mathcal{X}$ is Gaussian distributed with mean and variance given by

$$\begin{aligned}\mathbb{E}[f_X(\mathbf{x}_\star)] &= m_M(\mathbf{x}_\star) + k_M(\mathbf{x}_\star, \mathbf{X})\mathbf{K}_{My}^{-1}(\mathbf{y} - m_M(\mathbf{X})) \\ &= m(\mathbf{z}_\star) + k(\mathbf{z}_\star, \mathbf{Z})\mathbf{K}_y^{-1}(\mathbf{y} - m(\mathbf{Z}))\end{aligned} \tag{5}$$

$$\begin{aligned}\mathbb{V}[f_X(\mathbf{x}_\star)] &= k_M(\mathbf{x}_\star, \mathbf{x}_\star) - k_M(\mathbf{x}_\star, \mathbf{X})\mathbf{K}_{My}^{-1}k_M(\mathbf{X}, \mathbf{x}_\star) \\ &= k(\mathbf{z}_\star, \mathbf{z}_\star) - k(\mathbf{z}_\star, \mathbf{Z})\mathbf{K}_y^{-1}k(\mathbf{Z}, \mathbf{z}_\star),\end{aligned} \tag{6}$$

respectively, with $\mathbf{z}_\star := \mathbf{h}(\mathbf{x}_\star)$ and $\mathbf{Z} := \mathbf{h}(\mathbf{X})$. Moreover, $k_M(\mathbf{x}_\star, \mathbf{X}) = k(\mathbf{z}_\star, \mathbf{Z}) = [k(\mathbf{z}_\star, \mathbf{z}_i)]_{i=1}^N$, where N is the size of the training dataset, $\mathbf{K}_{My} := k_M(\mathbf{X}, \mathbf{X}) + \sigma_n^2\mathbf{I}$, $\mathbf{K}_y := k(\mathbf{Z}, \mathbf{Z}) + \sigma_n^2\mathbf{I}$, $k_M(\mathbf{X}, \mathbf{X}) := k(\mathbf{Z}, \mathbf{Z})$, and $m_M(\mathbf{X}) := m(\mathbf{Z}) = [m(\mathbf{z}_i)]_{i=1}^N$ computes the prior mean function evaluated at the embedded training inputs $\mathbf{Z}$. Posterior predictions can be computed using both the feature and data space. Equations (5)–(6) appear in the definition of the acquisition functions in (2)–(4) as mean $\mu(\mathbf{x}) := \mathbb{E}[f_X(\mathbf{x})]$ and standard deviation $\sigma(\mathbf{x}) := \sqrt{\mathbb{V}[f_X(\mathbf{x})]}$ of the posterior predictions of the surrogate model.

The MGP defines a GP on $\mathcal{X}$, but allows us to learn a response surface in the lower-dimensional feature space $\mathcal{Z}$. This is key for optimizing the acquisition function in a low-dimensional space $\mathcal{Z}$ instead of the original data/parameter space $\mathcal{X}$. Thus far, we have detailed the feature-space BO procedure up to line 8 in Algorithm 1. Once we found an optimizer $\mathbf{z}_*$ of the acquisition function, we need to project it back into the original data space $\mathcal{X}$ in order to evaluate the true objective f_X, whose domain is $\mathcal{X}$. This can be done by means of a reconstruction mapping (decoder), which we detail in the following.

3.2 Input reconstruction with manifold multi-output Gaussian processes

In the following, we present the reconstruction part (decoder) of our feature space optimization model described in Fig. 1. We are interested in modeling the functional relationship between the feature space $\mathcal{Z}$ and the data space $\mathcal{X}$ for step 9 in Algorithm 1, which requires us to evaluate f_X. We therefore consider a vector-valued function $\mathbf{g} = \{g_i\}_{i=1}^D$, where each component $g_i : \mathcal{Z} \to \mathcal{X}_i$ maps vectors in feature space to the i-th coordinate of high-dimensional data, i.e. $g_i(\mathbf{z}) = \tilde{x}^{(i)} \in \mathcal{X}_i$. Multi-output GPs (MOGPs) (Alvarez et al. 2011; Alvarez and Lawrence 2011; Byron et al. 2009; Wilson et al. 2012; Alvarez and Lawrence 2009; Osborne et al. 2008; Seeger et al. 2005; Boyle and Frean 2005) define a

prior over vector-valued functions and explicitly allow for output correlations. An MOGP $\mathcal{GP}(\mathbf{m}, \mathbf{K})$ is fully specified by a mean vector function $\mathbf{m} : \mathcal{Z} \to \mathbb{R}^D$ and a positive, semi-definite matrix-valued covariance function $\mathbf{K} : \mathcal{Z} \to \mathbb{R}^{D \times D}$, which computes the correlation between observations in the same output coordinate and cross-correlations between the D different outputs.

Here we consider the *intrinsic coregionalization model* (ICM) (Goovaerts 1997; Wackernagel 2013), which structures the covariance matrix as a Kronecker product. This model is particularly suitable for trading off number of model parameters and expressiveness of the vector valued function. In particular, the ICM facilitates information sharing across different tasks by adopting the same covariance function. It has been successfully adopted in robotics for learning inverse dynamics (Williams et al. 2009). Hence, this model requires fewer parameters than the linear model of coregionalization (Alvarez et al. 2011) and allows for exploiting properties of the Kronecker product for efficient training and prediction.

In our reconstruction model, we need to ensure that the output space is exactly $\mathcal{X}$. If we start from a data space $\mathcal{X} = [0, 1]^D$ the reconstructions need to belong to this hypercube. This property is not guaranteed by the MOGP. In order to satisfy this constraint, we consider a strictly monotonic output squashing function Ψ, as introduced in the context of warped GPs (Snelson et al. 2004). This allows us to define a corresponding inverse transformation Ψ^{-1} that is applied to the data in input to the model. The resulting output of the MOGP at test time is then squashed through the transformation Ψ. Since the reconstruction of the MOGP is a distribution $p(\tilde{\mathbf{x}}_\star|\mathbf{X}, \tilde{\mathbf{X}}, \mathbf{z}_\star)$, we evaluate the expectation with respect to this distribution of the transformed outputs, i.e. $\mathbf{x}_{t+1} = \mathbb{E}_p[\Psi(\tilde{\mathbf{x}}_\star)|\mathbf{X}, \tilde{\mathbf{X}}, \mathbf{z}_\star]$. In this paper, we choose the Gaussian cumulative density function as a monotonic squashing function $\Psi := \Phi$ for warping the outputs of our reconstruction model (Snelson et al. 2004). The motivation for this choice is twofold: the inverse mapping Ψ^{-1} is defined as the Probit function, which is a well known function, and the expectation $\mathbb{E}_p[\Psi(\tilde{\mathbf{x}}_\star)|\mathbf{X}, \tilde{\mathbf{X}}, \mathbf{z}_\star]$ with respect to the distribution $p(\tilde{\mathbf{x}}_\star|\mathbf{X}, \tilde{\mathbf{X}}, \mathbf{z}_\star)$ at test reconstructions can be derived analytically (Rasmussen and Williams 2006).

Intrinsic coregionalization model. The ICM (Goovaerts 1997; Wackernagel 2013) applies a linear mapping to a set of latent functions. In particular, we consider a set of P latent functions $u_i : \mathcal{Z} \to \mathbb{R}$, that are assumed to be *sample paths*, i.e. sample functions independently drawn from the same GP prior $\mathcal{GP}(m_c, k_c)$. The ICM model expresses the vector-valued function as a linear combination of these sample functions $\mathbf{g}(\mathbf{z}) = \mathbf{A}\mathbf{u}(\mathbf{z})$, where $\mathbf{u}(\mathbf{z}) \in \mathbb{R}^P$ is the collection of the P sample paths' evaluations at $\mathbf{z}$, and $\mathbf{A} \in \mathbb{R}^{D \times P}$ is the linear mapping that couples the independent vector and parameterizes the ICM model. As a result, $\mathbf{g}$ is an MOGP $\mathcal{GP}(\mathbf{m}, \mathbf{K})$ with mean function $\mathbf{m} = \mathbf{A}\mathbf{m}_c$, where $\mathbf{m}_c = [m_c]_{i=1}^P$ is obtained by repeating the single-valued mean function m_c in a P-vector. The covariance function is $\mathbf{K}(\mathbf{z}, \mathbf{z}') = \mathbf{A}\mathbf{A}^T \otimes k_c(\mathbf{z}, \mathbf{z}')$, where k_c is the covariance function for the GP prior, $\otimes$ is the Kronecker product and the matrix $\mathbf{A}\mathbf{A}^T$ is denoted as the coregionalization matrix. Note that k_c may differ from the covariance function k used for the response surface f_Z.

Reconstruction model. For the reconstruction task in line 9 of Algorithm 1, we introduce the manifold MOGP with intrinsic coregionalization model (mMOGP), which shares the feature map $\mathbf{h}$ with the MGP used for learning the response surface; see Sect. 3.1. Without loss of generality, we assume a prior zero-mean vector function for the mMOGP $\mathcal{GP}(\mathbf{0}, \mathbf{B} \otimes k_{MO})$, where $k_{MO}(\mathbf{x}, \mathbf{x}') = k_c(\mathbf{h}(\mathbf{x}), \mathbf{h}(\mathbf{x}'))$ and the matrix $\mathbf{B} = \mathbf{A}\mathbf{A}^T$. We can interpret this model as an auto-encoder, where the MGP $\mathbf{g} \circ \mathbf{h} : \mathcal{X} \to \mathcal{Z}$ plays the role of the encoder, and the MOGP the role of the decoder, mapping low-dimensional features back into data space.

3.3 Joint training

The joint training of the MGP, which models the response surface, and the mMOGP, which is used for the reconstruction (see also Fig. 1), is performed by maximizing a rescaled version of the log-marginal likelihood

$$\mathcal{L} \propto -\mathbf{y}^T\mathbf{K}_y^{-1}\mathbf{y} - \log|\mathbf{K}_y| - \frac{1}{D}\left(\mathbf{x}_V^T\mathbf{K}_V^{-1}\mathbf{x}_V + \log|\mathbf{K}_V|\right) + \text{const.} \tag{7}$$

Here, $\mathcal{L}$ comprises terms from both the MGP and mMOGP models, where $\mathbf{K}_y$ is defined in (5), and the covariance matrix of the mMOGP $\mathbf{K}_V = \bar{\mathbf{K}} + \sigma_n^2\mathbf{I}$ is obtained by evaluating the Kronecker product $\bar{\mathbf{K}} = \mathbf{B} \otimes k_c(\mathbf{Z}, \mathbf{Z})$ with the mMOGP kernel k_c. The vector $\mathbf{x}_V$ is a concatenation of the columns of the data $\mathbf{X}$. The maximizers $[\boldsymbol{\theta}_h^*, \boldsymbol{\theta}_k^*, \boldsymbol{\theta}_c^*]$ of the log-marginal likelihood are the parameters $\boldsymbol{\theta}_h^*$ of the feature map $\mathbf{h}$ (which is shared between the MGP and the mMOGP), the hyper-parameters $\boldsymbol{\theta}_k^*$ of the kernel k and the hyper-parameters $\boldsymbol{\theta}_c^*$ of k_c including the coregionalization matrix $\mathbf{B}$ for the mMOGP, respectively. The rescaling factor $1/D$ balances the contributions of the two log-marginal likelihood terms involved in training. The dimensions of the matrix $\mathbf{K}_V$ are $ND \times ND$ which correspond to repeating the $\mathbf{K}_y$ matrix D times in a block-diagonal fashion. This block diagonal would then have quadratic form equal to $D\mathbf{y}^T\mathbf{K}_y^{-1}\mathbf{y}$ and log determinant equal to $D\log|\mathbf{K}_y|$. Thus, an equivalent rescaling is to divide the reconstruction terms $\mathbf{x}_V^T\mathbf{K}_V^{-1}\mathbf{x}_V$ and $\log|\mathbf{K}_V|$ by D. Optimization of (7) is performed via gradient-based methods (Byrd et al. 1995; Zhu et al. 1997).

Modeling the black-box objective function f_X is orthogonal to the reconstruction problem. However, when training these tasks jointly, they have a regularization effect on the optimization of the parameters $\boldsymbol{\theta}_h$ of the feature embedding in the sense that the mapping $\mathbf{h}$ will not overfit to a single regression task: the parameters $\boldsymbol{\theta}_h$ will give rise to a feature space embedding that is useful for both the modeling of the objective and the reconstruction of the original inputs.

The major computational bottleneck for evaluating the marginal likelihood comes from the term $\mathbf{x}_V^T\mathbf{K}_V^{-1}\mathbf{x}_V$, which requires inverting an $ND \times ND$ covariance. We reduce the computational complexity of this operation to $\mathcal{O}(N^3) + \mathcal{O}(D^3)$ by exploiting the properties of the Kronecker product, tensor algebra (Riley et al. 1999) and structured GPs (Gilboa et al. 2015; Saatçi 2012) as shown in the following section.

3.4 Computationally efficient mMOGP

For the reconstruction mapping $\mathbf{g}$, we use the posterior mean of the mMOGP with intrinsic coregionalization model and apply exact inference and training via rescaled marginal likelihood maximization. While the ICM enables modeling correlation between arbitrary pairs of dimensions, it also requires computing a Kronecker product to evaluate the full covariance matrix of all outputs

$$\bar{\mathbf{K}} = \mathbf{B} \otimes k_c(\mathbf{Z}, \mathbf{Z}), \tag{8}$$

where $k_c(\mathbf{Z}, \mathbf{Z})$ is the covariance matrix obtained from the training inputs $\mathbf{Z}$ in feature space and the $\mathbf{B}$ matrix is the coregionalization matrix of the ICM. Inverting the full covariance matrix $\bar{\mathbf{K}}$ requires $\mathcal{O}(N^3D^3)$ and easily becomes intractable in high-dimensional spaces even for small N. Storing this full covariance matrix required $\mathcal{O}(N^2D^2)$ space and also becomes challenging in high dimensions. For an efficient implementation of the ICM, we exploit properties of the Kronecker product and apply results from structured GPs (Gilboa

et al. 2015; Saatçi 2012) that allow for efficient training and predictions in $\mathcal{O}(N^3) + \mathcal{O}(D^3)$ time and $\mathcal{O}(ND)$ space. In particular, we are interested in the full covariance matrix under the assumption of a Gaussian likelihood for the multi-output observations, i.e. $\bar{\mathbf{K}} + \sigma_n^2\mathbf{I}$. We first express the full covariance matrix in terms of its eigendecomposition, i.e. $\bar{\mathbf{K}} = \mathbf{Q}\boldsymbol{\Lambda}\mathbf{Q}^T$. This allows expressing the inverse of the covariance from noisy targets as

$$\left(\bar{\mathbf{K}} + \sigma_n^2\mathbf{I}\right)^{-1} = \mathbf{Q}\left(\boldsymbol{\Lambda} + \sigma_n^2\mathbf{I}\right)^{-1}\mathbf{Q}^T, \tag{9}$$

where both $\boldsymbol{\Lambda}$ and $\sigma_n^2\mathbf{I}$ are diagonal and can be trivially inverted. However, the eigendecomposition of an $ND \times ND$ matrix would still be cubic in the product between the number of dimensions and the number of data. By the properties of the Kronecker product, we can express the eigendecomposition itself with a Kronecker structure, i.e.

$$\bigotimes_{l=1}^{W} \mathbf{K}_l = \bigotimes_{l=1}^{W} \mathbf{Q}_l \bigotimes_{l=1}^{W} \boldsymbol{\Lambda}_l \left(\bigotimes_{l=1}^{W} \mathbf{Q}_l \right)^T, \tag{10}$$

where each term of the Kronecker product on the left-hand side $\mathbf{K}_l \in \mathbb{R}^{G_l \times G_l}$ has eigendecomposition $\mathbf{K}_l = \mathbf{Q}_l\boldsymbol{\Lambda}_l\mathbf{Q}_l^T$ for $l = 1, \ldots, W$, where W is number of factors in the Kronecker product. In our ICM model $W = 2$, because the coregionalization matrix $\mathbf{B}$ Kronecker multiplies $k_c(\mathbf{Z}, \mathbf{Z})$, the covariance matrix of the observations; see (8). Thus, from (9)–(10), we are allowed to invert the covariance from noisy targets by separately decomposing the covariance matrix $k_c(\mathbf{Z}, \mathbf{Z}) = \mathbf{Q}_k\boldsymbol{\Lambda}_k\mathbf{Q}_k^T$ and the coregionalization matrix $\mathbf{B} = \mathbf{Q}_b\boldsymbol{\Lambda}_b\mathbf{Q}_b^T$, which require $\mathcal{O}(N^3)$ and $\mathcal{O}(D^3)$ time, respectively; see line 5 of Algorithm 2.

Algorithm 2 Efficient computation of the inverse for matrices that have a Kronecker structure and spherical additive noise. Subroutine `matvecmul`: fast matrix-vector multiplication for matrices that can be expressed as a Kronecker product. Here the function `eigh` returns the eigen-decomposition of a matrix.

1: **Input matrices:** $\{\mathbf{K}_l \in \mathbb{R}^{G_l \times G_l}\}_{l=1}^{W}$
2: **Input vector:** $\mathbf{x}_V \in \mathbb{R}^{N_V}$, $N_V = \prod_{l=1}^{W} G_l$
3: **Input variable:** σ_n^2
4: **for** $l = 1, 2, ..., W$ **do**
5: $\quad \boldsymbol{\Lambda}_l, \mathbf{Q}_l, = \texttt{eigh}(\mathbf{K}_l)$ ▷ Eigen-decomposition of each input matrix
6: **end for**
7: $\mathbf{s} = \texttt{matvecmul}(\bigotimes_{l=1}^{W} \mathbf{Q}_l^T, \mathbf{x}_V)$ ▷ Fast matrix-vector multiplication
8: $\mathbf{D} = \bigotimes_{l=1}^{W} \boldsymbol{\Lambda}_l + \sigma_n^2\mathbf{I}$ ▷ Diagonal term with eigenvalues and noise
9: $\mathbf{w} = \mathbf{D}^{-1}\mathbf{s} = [s_i / D_{i,i}]_{i=1}^{N_V}$ ▷ Standard matrix-vector multiplication
10: **Return** $\mathbf{r} = \texttt{matvecmul}\left(\bigotimes_{l=1}^{W} \mathbf{Q}_l, \mathbf{w}\right) = \left(\bigotimes_{l=1}^{W} \mathbf{K}_l + \sigma_n^2\mathbf{I}\right)^{-1} \mathbf{x}_V$ ▷ Fast matrix vector multiplication of an inverse with Kronecker structure and a vector

1: **Procedure** $\texttt{matvecmul}(\bigotimes_{l=1}^{W} \mathbf{K}_l, \mathbf{x})$
2: **Input matrices:** $\{\mathbf{K}_l \in \mathbb{R}^{G_l \times G_l}\}_{l=1}^{W}$
3: **Input vector:** $\mathbf{x} \in \mathbb{R}^{N_V}$, $N_V = \prod_{l=1}^{W} G_l$
4: $\mathbf{r} = \mathbf{x}$ ▷ Initialize result
5: **for** $l = W, W-1, ..., 1$ **do**
6: $\quad \mathbf{R} = \texttt{reshape}(\mathbf{r}, [G_l, N_V / G_l])$ ▷ Reshape results
7: $\quad \mathbf{Z} = \mathbf{K}_l\mathbf{R}$ ▷ Matrix-tensor product
8: $\quad \mathbf{r} = \texttt{vec}(\mathbf{Z}^\top)$ ▷ Reshape results
9: **end for**
10: **Return** $\mathbf{r} = \left(\bigotimes_{l=1}^{W} \mathbf{K}_l\right)\mathbf{x}$ ▷ Fast matrix vector multiplication for matrix with Kronecker structure

Storing this inverse matrix and multiplying it by a vector still requires $\mathcal{O}(N^2D^2)$ space and run time, respectively, so that this step becomes the main bottleneck for efficient mMOGP training and predictions computation. To address this issue we represent the expensive matrix-vector multiplication as a sequence of small matrix-tensor multiplications without computing the full Kronecker product (Riley et al. 1999). In particular, we are interested in efficiently evaluating

$$\mathbf{r} = \left(\bigotimes_{i=1}^{W} \mathbf{K}_i\right)\mathbf{x}. \tag{11}$$

We first represent the multiplication of a matrix with Kronecker structure by a vector as a tensor product. A *tensor* $\mathbf{T}_{i_1,\ldots,i_V}$ can be interpreted as an extension of matrices to objects where elements are indexed using a set of V indices: $i_1, \ldots, i_V$, where the number V is referred to as the *order* of the tensor. With the definition of the Kronecker product we express the left-hand side of (10) as a tensor

$$\left[\bigotimes_{l=1}^{W} \mathbf{K}_l\right]_{i,j} = \left[\mathbf{K}_1\right]_{i_1,j_1} \cdot \ldots \cdot \left[\mathbf{K}_W\right]_{i_W,j_W}, \quad 1 \le i_l \quad ,j_l \le G_l, \quad 1 \le i,j \le \prod_{l=1}^{W} G_l. \tag{12}$$

The right-hand side of (12) coincides with a tensor $\mathbf{T}^K_{i_1,j_1,\ldots,i_W,j_W}$, and a similar tensor-representation can be obtained for the $\prod_{l=1}^{W} G_l$-long vector $\mathbf{x}$, i.e. $\mathbf{T}^X_{j_W,\ldots,j_1}$. A tensor product between the tensors $\mathbf{T}^K_{i_1,j_1,\ldots,i_W,j_W}$ and $\mathbf{T}^X_{j_W,\ldots,j_1}$ applies a contraction along the indices of the second tensor, i.e.

$$\sum_{j_1} \cdots \sum_{j_W} \mathbf{T}^K_{i_1,j_1,\ldots,i_W,j_W} \mathbf{T}^X_{j_W,\ldots,j_1}. \tag{13}$$

This tensor contraction can be expressed in terms of a sequence of tensor-transposed matrix-tensor products

$$\left(\bigotimes_{l=1}^{W} \mathbf{K}_l\right)\mathbf{x} = \mathrm{vec}\left(\left(\mathbf{K}_1 \cdots \left(\mathbf{K}_W \mathbf{T}^X\right)^\top\right)^\top\right) \tag{14}$$

with $\mathbf{K}_l\mathbf{T}^X = \sum_{k=1}^{G_l} \left[\mathbf{K}_l\right]_{i_1,k} \mathbf{T}^X_{k,j_2,\ldots,j_W}$. The function $\mathrm{vec}(\cdot)$ returns the vectorized form of a matrix by stacking its columns vertically. The tensor transposition $\top$ applies a cyclic permutation to the order of the indices in a tensor. As a result, the right-hand side in (14) allows us to evaluate the expensive matrix-vector product without computing and storing the Kronecker product. Algorithm 2 shows the main steps of the efficient matrix inversion and matrix-vector multiplication for matrices that feature a Kronecker structure. The matrix vector multiplication subroutine is expressed as a sequence of tensor-transpose matrix-tensor products.

4 Constrained acquisition

We defined a joint probabilistic model for the response surface learning and the input reconstruction tasks; see lines 4–6 and 9 of Algorithm 1, respectively. We are now concerned with the maximization of the acquisition function in feature space; see line 8 of Algorithm 1. We aim at maximizing the acquisition function in a low-dimensional feature space of the original data/parameter space $\mathcal{X}$. However, one problem that arises with the mMOGP decoding is that locations in feature space, which are too far away from data,

will be mapped back to the mMOGP prior. Since the acquisition function is a key driver of exploration in BO, this is a problem. We address this limitation by introducing a constraint based on the Lipschitz continuity of the mMOGP posterior. This will ensure that candidates $\mathbf{z}_* \in \mathcal{Z}$ selected in feature space will not collapse to the origin $\mathbf{0} \in \mathbb{R}^D$ if the reconstruction is defined as $\tilde{\mathbf{x}}_* = \boldsymbol{\mu}(\mathbf{z}_*)$, where $\boldsymbol{\mu}$ is the posterior mean of the mMOGP.

We want to leverage information from observed data for the multi-output mapping and exploit it when optimizing the acquisition function in feature space. This can be achieved by upper-bounding the Euclidean distance

$$\text{dist}(\mathbf{z}, \mathbf{Z}_t) = \min_{1 \leq i \leq N_t} \|\mathbf{z}_i - \mathbf{z}\|_2 \tag{15}$$

in feature space between the optimization variable $\mathbf{z}$ and the embedded training data $\mathbf{Z}_t = [\mathbf{z}_1, \ldots, \mathbf{z}_{N_t}]$. Here, N_t is the number of data points available at BO iteration t. The desired upper bound is obtained by exploiting the Lipschitz continuity property of the multi-output posterior mean for which

$$|[\boldsymbol{\mu}(\mathbf{z})]_i - [\boldsymbol{\mu}(\mathbf{z}')]_i| \leq L\|\mathbf{z} - \mathbf{z}'\|. \tag{16}$$

Here, L denotes the Lipschitz constant of the posterior mean $\boldsymbol{\mu}$ of the mMOGP. For common kernels, such as Matérn$_{52}$ and squared exponential, the posterior mean is Lipschitz continuous. The upper bound

$$\text{dist}(\mathbf{z}, \mathbf{Z}_t) \leq \mu_{\max}(\mathbf{z}^*)/L \tag{17}$$

allows us to specify how far from the data we can move in feature space without falling back to the prior on all coordinates of the reconstruction. Here $\mathbf{z}^*$ minimizes the distance in (15), while the numerator on the right-hand side is the component-wise maximum of $\boldsymbol{\mu}(\mathbf{z}^*)$. We estimate the Lipschitz constant as the maximum norm of the Jacobian of the posterior mean of the mMOGP (González et al. 2016)

$$L = \max_{\mathbf{z} \in \mathcal{Z}} \|\nabla_{\mathbf{z}} \boldsymbol{\mu}(\mathbf{z})\|. \tag{18}$$

This maximization returns a valid Lipschitz constant (González et al. 2016) for the multi-output mapping for any choice of norm in (18). The Jacobian of the posterior mean is represented by a $D \times d$ matrix and we adopt the max norm $\|\nabla_{\mathbf{z}} \boldsymbol{\mu}(\mathbf{z})\|_\infty = \max |\mu'_{i,j}|$ for $i = 1, \ldots, D$ and $j = 1, \ldots, d$. Lower values of valid Lipschitz constants L allow for exploration in larger regions of the feature space that still satisfy the nonlinear constraint in (17).

5 Experiments

We report results on a set of high-dimensional benchmark functions that possess an intrinsic low dimensionality. In particular, we (1) assess the benefits of adopting a model structure as presented in Fig. 1; (2) analyze the benefits of the constrained optimization of the acquisition function. Our purpose is to compare empirical performances across (a) different characterizations of the feature spaces, e.g. linear/nonlinear subspaces; (b) different properties of the objective function, e.g. additivity/non additivity; (c) a real problem set.

Approaches We compare our approach (MGPC-BO) with the random embeddings optimization (REMBO) (Wang et al. 2013), which performs BO on a random linear subspace of the inputs. Additional baselines include additive models (ADD-BO) (Kandasamy et al.

2015), which assumes an additive structure (across dimensions) of the objective f_X, and one recently proposed VAE-based model (VAE-BO) (Gomez-Bombarelli et al. 2018) that learns a feature space with deep networks offline. We also include a version of our model presented in Fig. 1 (HMGPC-BO) that uses a hierarchical ICM for the input reconstruction mapping $\mathbf{g}$. The hierarchical ICM partitions the data space into low-dimensional disjoint subsets, i.e. $\{\mathcal{X}_i\}_{i=1}^{Q}$, $\mathcal{X}_i \subset \mathbb{R}^3$, and assumes independence between reconstructions of different subsets, i.e. $\tilde{\mathbf{x}}^{(i)} \perp \tilde{\mathbf{x}}^{(j)}$, where $\tilde{\mathbf{x}}^{(i)} \in \mathcal{X}_i$, $\tilde{\mathbf{x}}^{(j)} \in \mathcal{X}_j$ for $i \neq j$. Moreover, the baselines MGP-BO and HMGP-BO correspond to same modeling as in MGPC-BO and HMGPC-BO, respectively, but without applying the nonlinear constraint in (17). We also compare with a different parametrization of the covariance function of the decoder $\mathbf{g}$. The baseline DMGP-BO and DMGPC-BO define a single kernel k_c for the reconstruction task while HMGP-BO and HMGPC-BO define different kernels $\{k_c^i\}_{i=1}^{Q}$, one for each subset of the partitioning. Here, DMGPC-BO and DMGP-BO denote the baseline with and without Lipschitz regularization, respectively. For all the approaches we specify the dimensionality d_{fs} of a feature space where the optimization is performed. Note that this value may differ from the intrinsic dimensionality d of the objective functions i.e. $d_{fs} \neq d$.

Acquisition functions We evaluate the performances of all baselines across common acquisition functions: EI (Močkus 1975), UCB (Srinivas et al. 2010) and PI (Kushner 1964), which are also given in (2)–(4). The motivation in selecting the above acquisition functions is that we wish to explore performances of our BO approach on a range of different decision strategies: aggressive exploitation (PI), aggressive exploration (UCB) and one-time-step optimal selection (EI). In our experiments we set the β_t parameter of UCB in (4) to $\sqrt{3}$. Moreover, we do not have access to the true $f_{\min}$ required in (2) and (3). Therefore, we compute the improvement based acquisitions (EI and PI) using $y_{\min} := \min \mathbf{y}_t$, which is the best noisy observation obtained up to iteration t. The maximization of the acquisition function is identical for all baselines: we first perform a random search step with 5000 samples drawn uniformly at random and select the best 100 locations to apply gradient-based optimization from these starting locations. For box-constrained acquisition optimization we use L-BFGS-B (Byrd et al. 1995; Zhu et al. 1997). For constrained acquisition optimization with nonlinear constraints we use a trust-region interior point method (Byrd et al. 1999).

Model parameters In our experiments, we select the $Matérn_{5/2}$ kernel as the covariance function for the GPs in each baseline. For the neural network employed in the encoder, the architecture was a single hidden layer with 20 units, and as the activation function we use the sigmoid activation $1/(1+\exp(-x))$.

Experiment setup Each BO progression curve shows the mean and standard error of the immediate logarithmic regret $\log_{10}|f(\mathbf{x}_{\text{best}}(t)) - f_{\min}|$, where $f_{\min}$ is the true minimum of f_X and $\mathbf{x}_{\text{best}}(t) \in \arg\min_{i=1:t} f_X(\mathbf{x}_i)$. Mean and standard error are computed over 20 experiments with different random initializations. All optimization experiments start with a budget of 10 data points and perform a total of 300 iterations. The noise variance is $\sigma_n^2 = 10^{-4}$.

5.1 Linear feature space

We consider benchmark functions that are defined in a $d = 10$-dimensional space. We map their input space to a $D = 60$-dimensional space using an orthogonal matrix $\mathbf{R}^{d\times D}$ so that the overall objective is $f_X(\mathbf{x}) = f(\mathbf{z}) = f(\mathbf{R}\mathbf{x})$.

5.1.1 Additive objective

We minimize the *Rosenbrock* benchmark function

$$f(\mathbf{z}) = \sum_{i=1}^{d-1} [100(z_{i+1} - z_i^2)^2 + (z_i - 1)^2] \tag{19}$$

in a $d_{fs} = 10$-dimensional feature space. Figure 2 shows that HMGPC-BO baseline descends quickly to relatively low regret in the early stages of optimization and recovers better regret at termination than the unconstrained baseline HMGP-BO. The VAE-BO baseline improves quickly but lacks exploration due to an insufficiently expressive reconstruction mapping from feature space to data space. We highlight that the VAE-BO model was trained on a budget of 500 inputs-observations pairs prior to starting the BO experiments. This additional budget, however, still does not allow the VAE-BO to compare well with baselines that learn a feature mapping during optimization. REMBO shows a competitive descent for two main reasons: the fact that the baseline conforms with the linear embedding assumption that characterizes the objective function and the employment of an orthonormal linear mapping which is supposed to improve performances and conforms to structural assumption about the linear embedding $\mathbf{R}$. The ADD-BO baseline suffers from the coupling effects of the linear dimensionality reduction $\mathbf{R}$. Overall, Fig. 2 highlights the fast learning of feature space representations that are effective for optimization with MGPC-BO, HMGPC-BO and DMGPC-BO baselines.

5.1.2 Non-additive objective

Here, we optimize the *product of sines* with intrinsic dimensionality $d = 10$

$$f(\mathbf{z}) = 10 \sin(z_1) \prod_{i=1}^{d} \sin(z_i) \tag{20}$$

and compare results when the additivity assumption is not satisfied. Figure 3 shows the regret curves obtained optimizing the objective on a $d_{fs} = 10$-dimensional feature space. Solid lines describe the Lipschitz-regularized baselines MGPC-BO, HMGPC-BO and

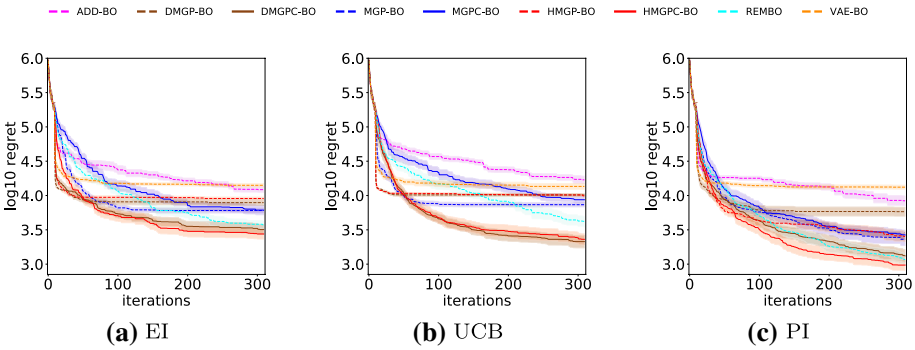


Fig. 2 Results with Rosenbrock objective function of BO in feature space. The objective function is characterized by a linear embedding to reach $D = 60$ dimensions. Baselines MGPC-BO, HMGPC-BO and DMGPC-BO (solid) apply nonlinearly constrained acquisition maximization and recover no worse regret at termination than the unconstrained versions MGP-BO, HMGP-BO and DMGP-BO

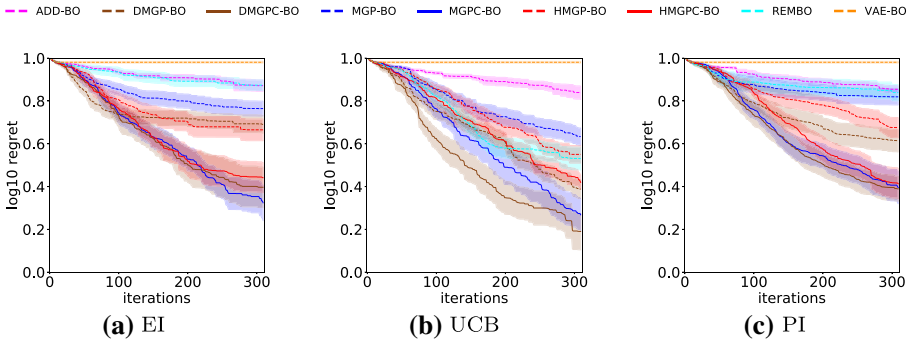


Fig. 3 Optimization progression on product of sines characterized by linear embedding with EI (**a**), UCB (**b**) and PI (**c**). Baselines MGPC-BO, HMGPC-BO and DMGPC-BO learn low-dimensional representations of the objective that are useful for optimization

DMGPC-BO (with nonlinear constraint), while dashed lines are baselines that apply box-constrained maximization of the acquisition in feature space. The HMGP-BO, MGP-BO and DMGP-BO regrets flatten early in both improvement-based acquisition functions (EI and PI) since these acquisition functions highlight locations in feature space that are too far away from the training data. In this setting, the decoder $\mathbf{g}$ returns the same high-dimensional reconstruction, which prevents BO from exploring. The constrained maximization of the acquisition is beneficial for all our models. We also note that the REMBO baseline conforms to the intrinsic linear low-dimensionality assumption described in Sect. 5.1 and is the most competitive baseline especially for UCB acquisition. However, the linear reconstruction mapping applied by REMBO also suffers from non-injectivity, and this slows down exploration in the high-dimensional space. The linear projection deteriorates performances of the additive model. ADD-BO assumes independence between axis-aligned projections of the high-dimensional space, while the linear mapping $\mathbf{R}$ couples all subsets of dimensions. This linear mapping, therefore, penalizes optimization with independent additive components. The VAE-BO approach requires much larger amounts of data to learn a meaningful reconstruction mapping than available in our experiment. Thus, most locations in feature space are mapped to similar reconstructions. This explains the flat curve observed on all VAE-BO progressions with different acquisitions.

5.2 Nonlinear feature space with non-additive objective

We consider the product of sines functions and apply a nonlinear dimensionality reduction. We define a single-layer neural network mapping to elevate the dimensionality of the objective to $D = 60$, i.e. $f_X(\mathbf{x}) = f(\gamma(\mathbf{R}\mathbf{x}))$. Here γ is the sigmoid activation function. We select a dimensionality of the feature space as in previous sections $d_{fs} = 10$ which is equal to the intrinsic dimensionality of the objective function $d = 10$. Figure 4 shows the progression of the regret over 300 BO iterations. We can observe consistent improvements of MGPC-BO, HMGPC-BO and DMGPC-BO with respect to VAE-BO which also assumes a nonlinear embedding for the objective. The performance of MGPC-BO, HMGPC-BO and DMGPC-BO also retain better regret at termination than with box-constrained acquisition maximization (MGP-BO, HMGP-BO, DMGP-BO). Here we apply a significance testing with the Wilcoxon signed-rank test (Wilcoxon 1992) at termination of the optimization

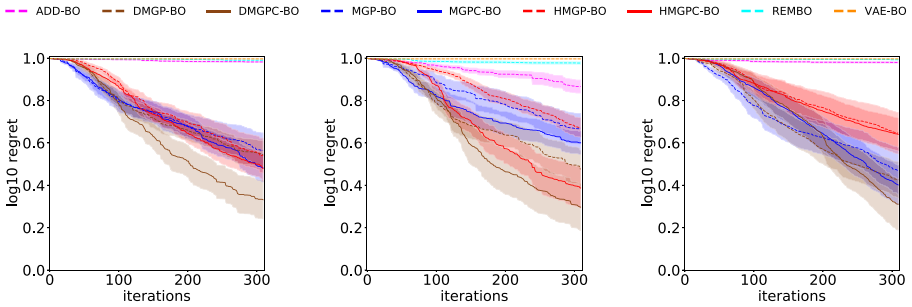


Fig. 4 BO performances expressed as log regret of the product of sines function in a nonlinear embedding. Results are shown for EI (**a**), UCB (**b**) and PI (**c**). All our baselines with nonlinear constraint, namely MGPC-BO, DMGPC-BO and HMGPC-BO learn useful representations in feature space for optimization. There is highly significant difference of 0.014% between DMGPC-BO and ADD-BO

between the best performing of our baselines, namely DMGPC-BO and the best competitive baseline that is ADD-BO. We observe a significance of at least 0.014% for all acquisition functions (largest p-value $p = 0.00014$ for UCB acquisition) meaning that our best baseline DMGPC-BO is highly significantly different than the ADD-BO baseline and attains better regret than ADD-BO at termination of the optimization.

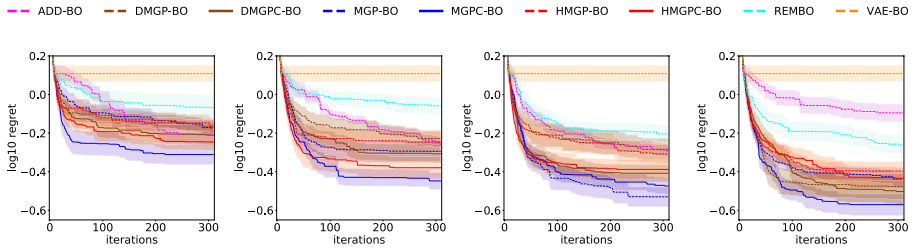
Overall, we observe that the constrained maximization of the acquisition function is beneficial for the proposed model. The advantages with respect to ADD-BO, REMBO and VAE-BO baselines are more evident with the product of sines objective with nonlinear embedding while with the Rosenbrock we retain no worse regret.

5.3 Sensitivity analysis on real data

Here we apply a sensitivity analysis with respect to the dimensionality of the feature space d_{fs} on a $D = 12$-dimensional real problem. We consider the Thomson problem of finding the lowest potential configuration of a set of electrons on a sphere (Dolan et al. 2004). This is a central problem in physics and chemistry for identifying a structure with respect to atomic locations (Dolan et al. 2004). The potential of a set of n_p electrons on a unit sphere is given by the objective

$$\sum_{i=1}^{n_p-1} \sum_{j=i+1}^{n_p} ((x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2)^{-1/2}. \tag{21}$$

This is a constrained minimization problem with constraints $x_i^2 + y_i^2 + z_i^2 = 1$ for $i = 1, \ldots, n_p$, which means that all electrons must lie on a unit sphere. We represent the variables of the problem as spherical coordinates with unit radius. This allows us defining two variables per point with a total number of $2n_p$ (azimuthal and polar angles) parameters to optimize within box constraints. For optimization, we select $n_p = 6$, which results in a $D = 12$-dimensional problem and we optimize it on low-dimensional feature spaces of dimensionalities $d_{fs} = 6, 4, 3, 2$ to observe the effect of this hyper-parameter in the optimization. Figure 5 shows a comparison of our approaches with ADD-BO, REMBO and VAE-BO baselines on a single acquisition function PI. Overall, we observe a deterioration of performances with diminishing dimensionality of the feature space. The regret clearly



(a) $d_{fs} = 2$, PI **(b)** $d_{fs} = 3$, PI **(c)** $d_{fs} = 4$, PI **(d)** $d_{fs} = 6$, PI

Fig. 5 Sensitivity analysis with respect to the dimensionality of the feature space d_{fs} on a real problem set. We test all approaches on a set of feature space dimensionalities $d_{fs} = 2, 3, 4, 6$. The performances of our baselines clearly deteriorate for $d_{fs} = 2$. Our baseline MGPC-BO show better performances than the best competing baseline ADD-BO and REMBO and reach the minimum in notably less iterations

increases for our baselines when we select $d_{fs} = 2$ meaning that, with a high compression rate, the probabilistic model for MGPC-BO, DGPC-BO and HMGPC-BO learns less useful features for optimization. We observe the most competitive baseline to be ADD-BO, which decomposes the 12-dimensional problems into D/d_{fs} sub-problems with dimensionality d_{fs}. Another competitive baseline is REMBO. We apply a significance test and compare our nonlinearly constrained baseline MGPC-BO with the most competitive baseline (ADD-BO or REMBO) for each plot of Fig. 5 at termination of the optimization. We select the Wilcoxon signed-rank test (Wilcoxon 1992), which does not assume that the difference between the sample populations is Gaussian. For feature space dimensionality $d_{fs} = 2$ we do not observe values significantly different since the p-value is $p = 0.135$. This is due to the deterioration of performances at $d_{fs} = 2$. For $d_{fs} = 3$ we observe a more significant difference between MGPC-BO and ADD-BO with p-value $p \leq 0.002$. With hyper-parameter values $d_{fs} \geq 4$ we observe significantly different baselines with significance at 0.6% (difference between MGPC-BO and ADD-BO for $d_{fs} = 4$ with p-value $p \leq 0.003$ and between MGPC-BO and REMBO for $d_{fs} = 6$ with p-value $p \leq 0.006$). Overall, we observe our constrained baselines to perform better than ADD-BO and REMBO and to reach the lowest value in notably less BO iterations.

5.4 Run-time complexity

The computational complexity of MGPC-BO is $\mathcal{O}(D^3 + N^3)$ due to the eigen-decomposition of both the coregionalization (D^3) and kernel matrix (N^3). The baseline HMGPC-BO scales with $\mathcal{O}(d_{out}^3 Q + N^3 Q)$ with Q being the number of independent subsets of dimensions, i.e. $Q = D/d_{out}$, with d_{out} being a small constant value ($d_{out} = 3$). This baseline achieves faster computations when having small number of data points N, for large number of data points and large number of dimensions (both tending to infinity) the MGPC-BO results more efficient. The baseline DMGPC-BO instead has complexity $\mathcal{O}(d_{out}^3 Q + N^3)$, which is faster than the MGPC-BO. MGP-BO, DMGP-BO and HMGP-BO have the same complexity of MGPC-BO, DMGPC-BO and HMGPC-BO, respectively. The remaining baselines have all computational complexity $\mathcal{O}(N^3)$ due to the matrix inversion of the covariance matrix for GP training which is used in ADD-BO, REMBO and VAE-BO. Our baseline has an additional overhead of at least a linear term $d_{out}^3 Q$, which implies slower

training times for our probabilistic model. This is a reasonable trade off for improved optimization performances and better data efficiency in our reconstruction model.

6 Conclusion

We proposed a framework for efficient Bayesian optimization of intrinsically low-dimensional black-box functions based on nonlinear embeddings. In our model, a manifold GP learns useful low-dimensional feature representations of high-dimensional data by jointly learning the response surface and a reconstruction mapping. Our approach allows for optimizing acquisition functions in a low-dimensional feature space. Since exploration in feature space (driven by the acquisition function) does not necessarily mean exploration in the high-dimensional parameter space, we introduce a nonlinear constraint based on Lipschitz continuity of predictions of the reconstruction mapping, which encourages exploration in the vicinity of the training data and mitigates un-identifiability issues in data space, which hinder optimization.

Acknowledgements We thank James T. Wilson for valuable feedback on early drafts of the manuscript. This work has been supported by the EPSRC Centre for Doctoral Training in High Performance Embedded and Distributed Systems (HiPEDS, Grant EP/L016796/1) and the Data Science Institute, Imperial College London.

References

Abbati, G., Tosi, A., Osborne, M. A., & Flaxman, S. (2018). Adageo: Adaptive geometric learning for optimization and sampling. In *International conference on artificial intelligence and statistics* (pp. 226–234).

Alvarez, M., & Lawrence, N. D. (2009). Sparse convolved Gaussian processes for multi-output regression. In *Advances in neural information processing systems* (pp. 57–64).

Alvarez, M. A., & Lawrence, N. D. (2011). Computationally efficient convolved multiple output Gaussian processes. *Journal of Machine Learning Research*, *12*, 1459–1500.

Alvarez, M. A., Rosasco, L., & Lawrence, N. D. (2011). Kernels for vector-valued functions: A review. *Foundations and Trends in Machine Learning*, *4*(3), 195–266.

Bergstra, J. S., Bardenet, R., Bengio, Y., & Kegl, B. (2011). Algorithms for hyper-parameter optimization. In *Advances in neural information processing systems* (pp. 2546–2554).

Boyle, P., & Frean, M. (2005). Dependent Gaussian processes. In *Advances in neural information processing systems* (pp. 217–224).

Byrd, R. H., Lu, P., Nocedal, J., & Zhu, C. (1995). A limited-memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing*, *16*(5), 1190–1208.

Byrd, R. H., Hribar, M. E., & Nocedal, J. (1999). An interior point algorithm for large-scale nonlinear programming. *SIAM Journal on Optimization*, *9*(4), 877–900.

Byron, M. Y., Cunningham, J. P., Santhanam, G., Ryu, S. I., Shenoy, K. V., & Sahani, M. (2009). Gaussian process factor analysis for low-dimensional single-trial analysis of neural population activity. In *Advances in neural information processing systems* (pp. 1881–1888).

Calandra, R., Peters, J., Rasmussen, C. E., & Deisenroth, M. P. (2016a). Manifold Gaussian processes for regression. In *International joint conference on neural networks* (pp. 3338–3345).
Calandra, R., Seyfarth, A., Peters, J., & Deisenroth, M. P. (2016b). Bayesian optimization for learning gaits under uncertainty. *Annals of Mathematics and Artificial Intelligence*, *76*, 5–23.
Cully, A., Clune, J., Tarapore, D., & Mouret, J. B. (2015). Robots that can adapt like animals. *Nature*, *521*, 503–507.
Dai, Z., Damianou, A., González, J., & Lawrence, N. D. (2016). Variational auto-encoded deep Gaussian processes. In *International conference on learning representations*.
Damianou, A. (2015). Deep Gaussian processes and variational propagation of uncertainty. PhD dissertation.
Damianou, A., & Lawrence, N. D. (2013). Deep Gaussian processes. In *International conference on artificial intelligence and statistics* (pp. 207–215).
Dolan, E. D., Moré, J. J., & Munson, T. S. (2004). Benchmarking optimization software with COPS 3.0. Technical Report.
Frazier, P. I. (2018). A tutorial on Bayesian optimization. arXiv preprint arXiv:1807.02811.
Garnett, R., Osborne, M. A., Hennig, P. (2014). Active learning of linear embeddings for Gaussian processes. In *Conference on uncertainty in artificial intelligence* (pp. 230–239).
Gilboa, E., Saatçi, Y., & Cunningham, J. P. (2015). Scaling multidimensional inference for structured Gaussian processes. *Institute of Electrical and Electronics Engineers*, *37*(2), 424–436.
Gomez-Bombarelli, R., Jennifer, N. W., Duvenaud, D., Hernández-Lobato, J. M., Sánchez-Lengeling, B., Sheberla, D., et al. (2018). Automatic chemical design using a data-driven continuous representation of molecules. *ACS Central Science*, *4*, 268–276.
González, J., Dai, Z., Hennig, P., & Lawrence, N. (2016). Batch Bayesian optimization via local penalization. In *International conference on artificial intelligence and statistics* (pp. 648–657).
Gonzalez, J., Longworth, J., James, D. C., & Lawrence, N. D. (2014). Bayesian optimization for synthetic gene design. In *Neural information processing systems workshop in bayesian optimization*.
Goovaerts, P. (1997). *Geostatistics for natural resources evaluation*. Oxford: Oxford University Press.
Griffiths, R. R., & Hernández-Lobato, J. M. (2017). Constrained Bayesian optimization for automatic chemical design. arXiv preprint arXiv:1709.05501.
Hebbal, A., Brevault, L., Balesdent, M., Talbi, E. G., & Melab, N. (2019). Bayesian optimization using deep Gaussian processes. arXiv preprint arXiv:1905.03350.
Hensman, J., & Lawrence, N. D. (2014). Nested variational compression in deep Gaussian processes. arXiv preprint arXiv:1412.1370.
Hutter, F., Hoos, H. H., & Leyton-Brown, K. (2011). Sequential model-based optimization for general algorithm configuration. In *International conference on learning and intelligent optimization* (pp. 507–523).
Jones, D. R., Schonlau, M., & Welch, W. J. (1998). Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, *13*(4), 455–492.
Kandasamy, K., Schneider, J., & Poczos, B. (2015). High dimensional Bayesian optimisation and bandits via additive models. In *International conference on machine learning* (pp. 295–304).
Kingma, D. P., & Welling, M. (2014). Auto-encoding variational Bayes. In *International conference on learning representations*.
Kushner, H. J. (1964). A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise. *Journal of Basic Engineering*, *86*(1), 97–106.
Kusner, M. J., Paige, B., & Hernández-Lobato, J. M. (2017). Grammar variational autoencoder. *International Conference on Machine Learning*, *70*, 1945–1954.
Lawrence, N. D. (2005). Probabilistic non-linear principal component analysis with Gaussian process latent variable models. *Journal of Machine Learning Research*, *6*, 1783–1816.
Lawrence, N. D., & Quiñonero-Candela, J. (2006). Local distance preservation in the gp-lvm through back constraints. In *International conference on machine learning* (pp. 513–520).
LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*, 436–444.
Lu, X., Gonzalez, J., Dai, Z., & Lawrence, N. (2018). Structured variationally auto-encoded optimization. In *International conference on machine learning* (pp. 3267–3275).
Močkus, J. (1975). On Bayesian methods for seeking the extremum. In *Optimization techniques IFIP technical conference* (pp. 400–404).
Moriconi, R., Kumar, K. S. S., & Deisenroth, M. P. (2020). High-dimensional Bayesian optimization with projections using quantile Gaussian processes. *Optimization Letters*, *14*, 1–14.
Osborne, M. A., Roberts, S. J., Rogers, A., Ramchurn, S. D., & Jennings, N. R. (2008). Towards real-time information processing of sensor network data using computationally efficient multi-output

Gaussian processes. In *International conference on information processing in sensor networks* (pp. 109–120). Institute of Electrical and Electronics Engineers.
Rana, S., Li, C., Gupta, S., Nguyen, V., & Venkatesh, S. (2017). High dimensional Bayesian optimization with elastic Gaussian Process. In *International conference on machine learning* (pp. 2883–2891).
Rasmussen, C. E., & Williams, C. K. I. (2006). *Gaussian processes for machine learning*. Cambridge: The MIT Press.
Rezende, D. J., Mohamed, S., & Wierstra, D. (2014). Stochastic backpropagation and variational inference in deep latent Gaussian models. In *International conference on machine learning* (Vol. 2).
Riley, K. F., Hobson, M. P., & Bence, S. J. (1999). *Mathematical methods for physics and engineering*. Cambridge: Cambridge University Press.
Rolland, P., Scarlett, J., Bogunovic, I., & Cevher, V. (2018). High-dimensional Bayesian optimization via additive models with overlapping groups. In *International conference on artificial intelligence and statistics* (pp. 298–307).
Saatçi, Y. (2012). Scalable inference for structured Gaussian process models. PhD dissertation.
Salimbeni, H., & Deisenroth, M. P. (2017). Doubly stochastic variational inference for deep Gaussian processes. In *Advances in neural information processing systems* (pp. 4588–4599).
Seeger, M., Teh, Y. W., & Jordan, M. (2005). Semiparametric latent factor models. Technical Report.
Shahriari, B., Swersky, K., Wang, Z., Adams, R. P., & De Freitas, N. (2016). Taking the human out of the loop: A review of Bayesian optimization. *Institute of Electrical and Electronics Engineers*, *104*(1), 148–175.
Snelson, E., Ghahramani, Z., & Rasmussen, C. E. (2004). Warped Gaussian processes. In *Advances in neural information processing systems* (pp. 337–344).
Srinivas, N., Krause, A., Kakade, S. M., & Seeger, M. W. (2010). Gaussian process optimization in the bandit setting: No regret and experimental design. In *International conference on machine learning* (pp. 1015–1022).
Sui, Y., Gotovos, A., Burdick, J., & Krause, A. (2015). Safe exploration for optimization with Gaussian processes. In *International conference on machine learning* (pp. 997–1005).
Titsias, M., & Lawrence, N. D. (2010). Bayesian Gaussian process latent variable model. In *International conference on artificial intelligence and statistics* (pp. 844–851).
Wackernagel, H. (2013). *Multivariate geostatistics: An introduction with applications*. Berlin: Springer.
Wahlström, N., Schön, T. B., & Deisenroth, M. P. (2015). From pixels to torques: policy learning with deep dynamical models. In *International conference of machine learning workshop on deep learning*.
Wang, Z., Zoghi, M., Hutter, F., Matheson, D., & De Freitas, N. (2013). Bayesian optimization in high dimensions via random embeddings. In *International joint conference on artificial intelligence* (pp. 1778–1784).
Wilcoxon, F. (1992). Individual comparisons by ranking methods. In S. Kotz & N. L. Johnson (Eds.), *Breakthroughs in statistics* (pp. 196–202). New York: Springer.
Williams, C., Klanke, S., Vijayakumar, S., & Chai, K. M. (2009). Multi-task Gaussian process learning of robot inverse dynamics. In *Advances in neural information processing systems* (pp. 265–272).
Wilson, A. G., Hu, Z., Salakhutdinov, R., & Xing, E. P. (2016). Deep kernel learning. In *International conference on artificial intelligence and statistics* (pp. 370–378).
Wilson, A. G., Knowles, D. A., & Ghahramani, Z. (2012). Gaussian process regression networks. In *International conference on machine learning* (pp. 1139–1146).
Zhu, C., Byrd, R. H., Lu, P., & Nocedal, J. (1997). Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization. *ACM Transactions on Mathematical Software*, *23*(4), 550–560.