

## 7 Supplementary Material

### 7.1 Model inference

Assume an observation dataset  $\mathcal{D}_n = \{\mathbf{x}_i, y_i\}_{i=1}^n$ , we exploit an efficient element-wise Gibbs sampling algorithm for model inference.

#### 7.1.1 Sampling latent functions

Given the the Gaussian prior and Gaussian likelihood of the latent factors  $\mathbf{g}_d^r$ , their posterior distributions are still Gaussian. Let  $y_r^i = y_i - \sum_{h \neq r}^R \lambda_h \prod_{d=1}^D g_d^h(x_d^i)$  and  $\mathbf{y}_r = [y_r^1, \dots, y_r^n]^\top \in \mathbb{R}^n$  for  $r = 1, \dots, R$ . Every  $\mathbf{y}_r$  generates a  $D$ -dimensional tensor  $\mathbf{Y}_r \in \mathbb{R}^{|S_1| \times \dots \times |S_D|}$  in the Cartesian product space  $\prod_{d=1}^D S_d$ . We define a binary tensor  $\mathbf{O}$  with the same size of  $\mathbf{Y}$  indicating the locations of the observation data, where  $o(\mathbf{x}_i) = 1$  for  $i \in [1, n]$ , and other values are zero. The posterior of  $\mathbf{g}_d^r$  is given by:

$$\begin{aligned}
 p(\mathbf{g}_d^r | -) &= \mathcal{N}\left(\mathbf{g}_d^r \mid [\boldsymbol{\mu}_d^r]^*, ([\boldsymbol{\Lambda}_d^r]^*)^{-1}\right), \\
 [\boldsymbol{\mu}_d^r]^* &= ([\boldsymbol{\Lambda}_d^r]^*)^{-1} \underbrace{\left( \tau (\mathbf{Y}_{r(d)} \otimes \mathbf{O}_{(d)}) \begin{pmatrix} \lambda_r \bigotimes_{\substack{h=D \\ h \neq d}}^1 \mathbf{g}_h^r \end{pmatrix} \right)}_{\mathbf{a} \in \mathbb{R}^{|S_d|}}, \\
 [\boldsymbol{\Lambda}_d^r]^* &= \tau \underbrace{\text{diag} \left( \mathbf{O}_{(d)} \begin{pmatrix} \lambda_r \bigotimes_{\substack{h=D \\ h \neq d}}^1 \mathbf{g}_h^r \end{pmatrix}^2 \right)}_{\mathbf{b} \in \mathbb{R}^{|S_d|}} + (\mathbf{K}_d^r)^{-1},
 \end{aligned} \tag{11}$$

where  $\mathbf{Y}_{r(d)}$  and  $\mathbf{O}_{(d)}$  are mode- $d$  unfoldings of  $\mathbf{Y}_r$  and  $\mathbf{O}$ , respectively, with the size of  $|S_d| \times (\prod_{h=1}^D |S_h|)$ . Note that the vector term  $\mathbf{a}$  in  $[\boldsymbol{\mu}_d^r]^*$  and  $\mathbf{b}$  in  $[\boldsymbol{\Lambda}_d^r]^*$ , which are only relevant to the  $n$  observations and corresponding function values, can be computed element-wise instead of using matrix multiplication and Kronecker product. The point-wise computation can dramatically reduce the computational cost especially for a relatively large  $D$ , since the number of observations in BO is much smaller compared with the number of samples in the entire grid space, i.e.,  $n \ll \prod_{d=1}^D |S_d|$ .

#### 7.1.2 Sampling kernel hyperparameters

We update the kernel hyperparameters  $\{l_d^r : d = 1, \dots, D, r = 1, \dots, R\}$  from their marginal posteriors by integrating out the latent factors. Let  $[\mathbf{y}_r]_d = \mathbf{O}_d \text{vec}(\mathbf{Y}_{r(d)}) \in \mathbb{R}^n$ , where  $\mathbf{O}_d \in \mathbb{R}^{n \times (\prod_{d=1}^D |S_d|)}$  is a binary matrix obtained by removing the rows corresponding to zeros in  $\text{vec}(\mathbf{O}_{(d)})$  from  $\mathbf{I}_{\prod_{d=1}^D |S_d|}$ . When sampling the posteriors for kernel hyperparameters under a given  $d$  and  $r$ , their marginal likelihoods only relate to  $[\mathbf{y}_r]_d$ . The log marginal likelihood of  $l_d^r$ , for example, is:

$$\begin{aligned}
 \log p([\mathbf{y}_r]_d | l_d^r) &\propto -\frac{1}{2} ([\mathbf{y}_r]_d)^\top \boldsymbol{\Sigma}_{[\mathbf{y}_r]_d | l_d^r}^{-1} [\mathbf{y}_r]_d - \frac{1}{2} \log \det(\boldsymbol{\Sigma}_{[\mathbf{y}_r]_d | l_d^r}) \\
 &\propto \frac{\tau^2}{2} \underbrace{([\mathbf{y}_r]_d)^\top \mathbf{H} ([\boldsymbol{\Lambda}_d^r]^*)^{-1} \mathbf{H}^\top [\mathbf{y}_r]_d}_c - \frac{1}{2} \log \det([\boldsymbol{\Lambda}_d^r]^*) - \frac{1}{2} \log \det(\mathbf{K}_d^r),
 \end{aligned} \tag{12}$$

where  $\mathbf{H} = \mathbf{O}_d \left( \lambda_r \bigotimes_{\substack{h=D \\ h \neq d}}^1 \mathbf{g}_h^r \otimes \mathbf{I}_{(|S_d|)} \right) \in \mathbb{R}^{n \times |S_d|}$  and  $\boldsymbol{\Sigma}_{[\mathbf{y}_r]_d | l_d^r} = \mathbf{H} \mathbf{K}_d^r \mathbf{H}^\top + \tau^{-1} \mathbf{I}_n$ . The term  $c$  in Eq. (12) is a scalar that can be computed with  $\mathbf{u}^\top \mathbf{u}$ , where  $\mathbf{u} = (\mathbf{L}_d^r)^{-1} \mathbf{a}$  is a vector

of length  $|S_d|$ ;  $\mathbf{L}_d^r = \text{chol}([\mathbf{\Lambda}_d^r]^*)$  is the Cholesky factor matrix of  $[\mathbf{\Lambda}_d^r]^*$ . This means that the complicated term  $c$  can also be calculated element-wise, and it leads to a fast learning process. With the marginal likelihoods and pre-defined log-normal hyperpriors, we can get the marginal posteriors of the kernel hyperparameters straightforwardly; and we update them by using the slice sampling algorithm presented in [13].

### 7.1.3 Sampling weight vector

Every observed data point has the following distribution:

$$y_i \sim \mathcal{N}\left(\sum_{r=1}^R \lambda_r \prod_{d=1}^D g_d^r(x_d^i) = \left[\prod_{d=1}^D g_d^1(x_d^i), \dots, \prod_{d=1}^D g_d^R(x_d^i)\right] \boldsymbol{\lambda}, \tau^{-1}\right), i = 1, \dots, n.$$

Let  $\mathbf{g}(x_i) = \left(\prod_{d=1}^D g_d^1(x_d^i), \dots, \prod_{d=1}^D g_d^R(x_d^i)\right)^\top \in \mathbb{R}^R$  and  $\mathbf{y} = (y_1, \dots, y_n)^\top \in \mathbb{R}^n$ ; we then have  $\mathbf{y} \sim \mathcal{N}(\tilde{\mathbf{G}}^\top \boldsymbol{\lambda}, \tau^{-1} \mathbf{I}_n)$ , where  $\tilde{\mathbf{G}} = [\mathbf{g}(x_1), \dots, \mathbf{g}(x_n)] \in \mathbb{R}^{R \times n}$ . The posterior of  $\boldsymbol{\lambda}$  follows a Gaussian distribution  $p(\boldsymbol{\lambda} | -) \sim \mathcal{N}(\boldsymbol{\mu}_\lambda^*, (\boldsymbol{\Lambda}_\lambda^*)^{-1})$ , where  $\boldsymbol{\mu}_\lambda^* = \tau (\boldsymbol{\Lambda}_\lambda^*)^{-1} \tilde{\mathbf{G}}^\top \mathbf{y}$  and  $\boldsymbol{\Lambda}_\lambda^* = \tau \tilde{\mathbf{G}} \tilde{\mathbf{G}}^\top + \mathbf{I}_R$ .

### 7.1.4 Sampling model noise precision

For precision  $\tau$ , we have a Gamma posterior  $p(\tau | -) = \text{Gamma}(\tau | a^*, b^*)$ , where  $a^* = a_0 + \frac{1}{2}n$  and  $b^* = b_0 + \frac{1}{2} \sum_{i=1}^n \left(y_i - \sum_{r=1}^R \lambda_r \prod_{d=1}^D g_d^r(x_d^i)\right)^2$ .

## 7.2 Algorithm of BKTF for BO

---

### Algorithm 2: BKTF for BO

---

**Input:** Initial dataset  $\mathcal{D}_0$ .

**for**  $n = 1 : N$  **do**

**for**  $k = 1 : K$  **do**

**for**  $r = 1 : R$  **do**

**for**  $d = 1 : D$  **do**

                Draw kernel length-scale hyperparameters  $l_d^r$ ;

                Draw latent factors  $(\mathbf{g}_d^r)^{(k)}$ ;

        Draw model noise precision  $\tau^{(k)}$ ;

        Draw weight vector  $\boldsymbol{\lambda}^{(k)}$ ;

**if**  $k > K_0$  **then**

            Compute and collect  $\tilde{\mathcal{F}}^{(k)}$ .

    Compute mean  $\boldsymbol{\mu}$  and variance  $\boldsymbol{\Sigma}$  of  $\{\tilde{\mathcal{F}}^{(k)}\}$ ;

    Compute  $\alpha_{\text{B-UCB}}(\mathbf{x} | \mathcal{D}_{n-1}, \beta)$  as a tensor;

    Find next  $\mathbf{x}_n = \arg \max_{\mathbf{x}} \alpha_{\text{B-UCB}}(\mathbf{x} | \mathcal{D}_{n-1}, \beta)$ ;

    Augment the data  $\mathcal{D}_n = \mathcal{D}_{n-1} \cup \{\mathbf{x}_n, y_n\}$ .

---

## 7.3 Optimization for nonstationary and nonseparable function

### 7.3.1 Data generation

The function in Figure 1 (a) is a modification of the case study used in [14]. It is  $40 \times 40$  2D process generated in a  $[1, 2] \times [-1, 0]$  square, with

$$\mathbf{Y}(x_1, x_2) = (\cos(4[f_1(x_1) + f_2(x_2)]) + \sin(4[f_1(x_2) - f_2(x_1)]) - 1) \times \exp\left(-(x_1 - 0.5)^2 + \frac{(x_2 - 1)^2}{5}\right), \quad (13)$$

where  $f(x_1) = x_1 (\sin 2x_1 + 2)$ ,  $f(x_2) = 0.2x_2 \sqrt{99(x_2 + 1) + 4}$ ,  $x_1 \in [1, 2]$ ,  $x_2 \in [-1, 0]$ . This is a nonstationary, nonseparable, and multimodal function, with the global maximum  $f(\mathbf{x}^*) = 0.6028$  at  $\mathbf{x}^* = (1.75, -0.55)$ .

### 7.3.2 Results

We randomly select  $n_0 = 30$  data points as the initial data and run 50 iterations (i.e., budget) of evaluation for optimization. To compare different surrogates, we run the optimization for 20 replications with different initial datasets. For the proposed BKTF surrogate, we place Matérn 3/2 kernel functions on the latent factors, set the rank  $R = 4$ , and run 1000 MCMC samples for model inference where the first 600 samples are burn-in. For comparison BO methods, we consider typical GP surrogate with both EI and UCB ( $\beta = 2$ ) as the AF, denoted by GP  $\alpha_{EI}$  and GP  $\alpha_{UCB}$  respectively, and use the same Matérn 3/2 kernel for GP surrogate. Figure 1(b) shows the medians along with 25% and 75% quantiles of the optimization results from 20 runs. We see that GP  $\alpha_{EI}$  and GP  $\alpha_{UCB}$  cannot find the global optimum in most cases, and they easily get stuck in the lower left flat area which contains easily find local optima. Figure 1(c) illustrates the estimation surface for the function and the estimated AF surface from one run. It is clear that BKTF can capture global correlations with limited data. The search points contain areas of almost every peak in the true function, and the peaks of its AF surface also reflect the peak area in  $f$ . Figure 4(a)-(d) shows the posterior distributions of model parameters, the mean of latent factors after burn-in, and the last 20 MCMC samples for latent factors in two dimensions, respectively. The samples of latent factors in panel (c) and (d) explain the uncertainty learned by BKTF for this optimization problem.

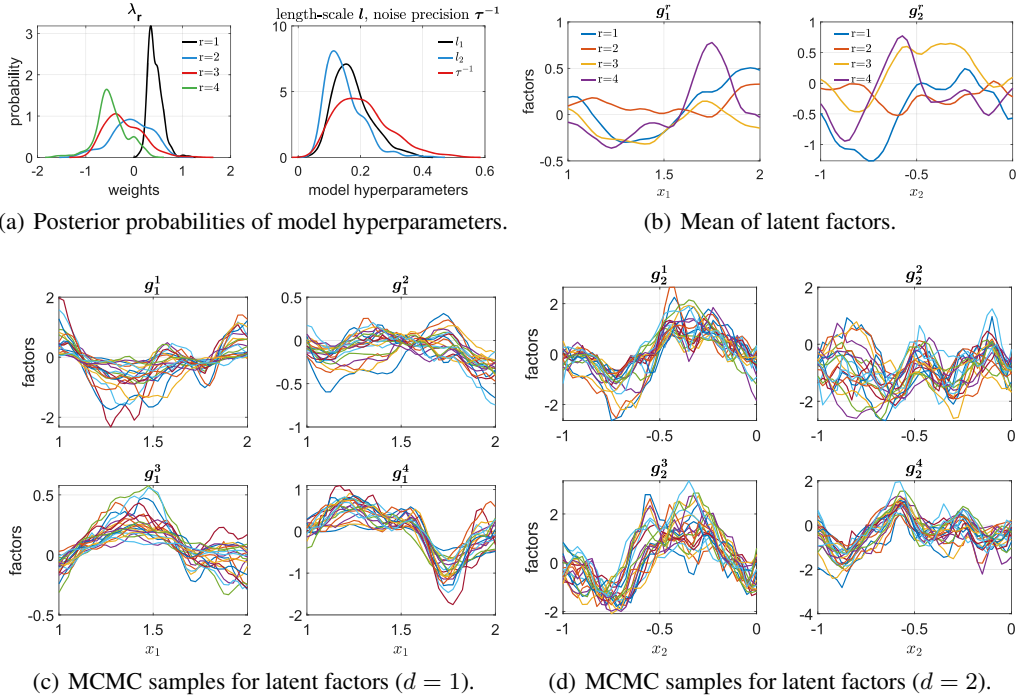


Figure 4: Results of BKTF for optimization on the nonstationary and nonseparable multimodal 2D function in Figure 1(a).

### 7.4 Benchmark test functions

The functional expressions and characteristics of the used test functions in Section 5.1 are summarized as below.

**(1) Branin Function** ( $D = 2$ )

$$f(x_1, x_2) = \left( x_2 - \frac{5.1}{4\pi} x_1^2 + \frac{5}{\pi} x_1 - 6 \right)^2 + 10 \left( 1 - \frac{1}{8\pi} \right) \cos(x_1) + 10, \quad (14)$$

511 where  $x_1 \in [-5, 10]$  and  $x_2 \in [0, 15]$ . It is a smooth but multimodal function with the global minima  
 512  $f(\mathbf{x}^*) = 0.3978873$  at three input points  $\mathbf{x}^* = (-\pi, 12.275), (\pi, 2.275), (3\pi, 2.425)$ .

**(2) Damavandi Function** ( $D = 2$ )

$$f(x_1, x_2) = \left[ 1 - \left| \frac{\sin[\pi(x_1 - 2)] \sin[\pi(x_2 - 2)]}{\pi^2(x_1 - 2)(x_2 - 2)} \right|^5 \right] [2 + (x_1 - 7)^2 + 2(x_2 - 7)^2], \quad (15)$$

513 where  $x_d \in [0, 14]$ . This is a multimodal function with the global minimum  $f(\mathbf{x}^*) = 0$  at  $\mathbf{x}^* = (2, 2)$ .

**(3) Schaffer Function** ( $D = 2$ )

$$f(x_1, x_2) = 0.5 + \frac{\sin^2 \sqrt{x_1^2 + x_2^2} - 0.5}{[1 + 0.001(x_1^2 + x_2^2)]^2}, \quad (16)$$

514 where  $x_d \in [-10, 10]$ . The global minimum value is  $f(\mathbf{x}^*) = 0$  at  $\mathbf{x}^* = (0, 0)$ . One characteristic  
 515 of this function is that the global minimum is located very close to the local minima.

**(4) Griewank Function** ( $D = 3, 4$ )

$$f(\mathbf{x}) = 1 + \sum_{d=1}^D \frac{x_d^2}{4000} - \prod_{d=1}^D \cos\left(\frac{x_d}{\sqrt{d}}\right), \quad (17)$$

516 where  $x_d \in [-10, 10]$ . This is a multimodal function with the global minimum  $f(\mathbf{x}^*) = 0$  at  
 517  $\mathbf{x}^* = (0, 0)$ .

518 **(5) Hartmann Function** ( $D = 6$ ) A nonseparable function with multidimensional inputs and  
 519 multiple local minima.

$$f(\mathbf{x}) = - \sum_{j=1}^4 c_j \exp\left(- \sum_{d=1}^6 a_{jd}(x_d - b_{jd})^2\right), \quad (18)$$

520 where

$$\begin{aligned} \mathbf{A} = [a_{jd}] &= \begin{bmatrix} 10 & 3 & 17 & 3.5 & 1.7 & 8 \\ 0.05 & 10 & 17 & 0.1 & 8 & 14 \\ 3 & 3.5 & 1.7 & 10 & 17 & 8 \\ 17 & 8 & 0.05 & 10 & 0.1 & 14 \end{bmatrix}, \quad \mathbf{c} = [c_j] = \begin{bmatrix} 1 \\ 1.2 \\ 3 \\ 3.2 \end{bmatrix}, \\ \mathbf{B} = [b_{jd}] &= \begin{bmatrix} 0.1312 & 0.1696 & 0.5569 & 0.0124 & 0.8283 & 0.5586 \\ 0.2329 & 0.4135 & 0.8307 & 0.3736 & 0.1004 & 0.9991 \\ 0.2348 & 0.1451 & 0.3522 & 0.2833 & 0.3047 & 0.6650 \\ 0.4047 & 0.8828 & 0.8732 & 0.5743 & 0.1091 & 0.0381 \end{bmatrix}, \end{aligned} \quad (19)$$

521  $x_d \in [0, 1]$ . The six dimensional case has 6 local minima, and the global minimum is  $f(\mathbf{x}^*) =$   
 522  $-3.32237$  at  $\mathbf{x}^* = (0.20169, 0.150011, 0.476874, 0.275332, 0.311652, 0.657301)$ .

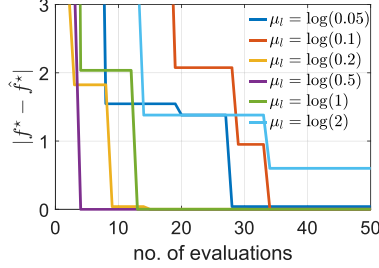
523 Note that all these minimization problems can be easily transformed as a maximization optimization  
 524 problem, i.e.,  $\max -f(\mathbf{x})$ .

**7.5 Supplementary results on benchmark test functions**

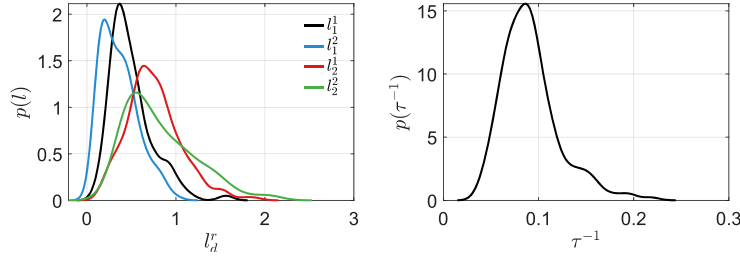
**7.5.1 Effects of hyperpriors on kernel hyperparameters**

527 We show the optimization results on Branin function with different hyperprior settings in Figure 5 as  
 528 an example to illustrate the effects of hyperpriors. In panel (a), the optimization results under several  
 529 hyperprior assumptions are compared, where  $\tau_l^{-1}$  is set as 0.5. As can be seen, BKTF is not able to

reach the global minimum with too small or too large mean assumptions (comparable to  $[0, 1]$ ) on the kernel length-scales  $l$ , for example in the cases where  $\mu_l = \{\log(0.05), \log(2)\}$ . In contrast, it finds the global optimum after 4 iterations of function evaluations when  $\mu_l = \log(0.5)$ , see the purple line in panel (a). These imply the importance of hyperprior selection. The reason is that in the first several evaluations, since the observations are rare, the prior basically determines the exploration-exploitation balance and guides the search process. Panel (b) shows the approximated posterior distributions for kernel hyperparameters and model noise variance when  $\tau_l^{-1} = 0.5$ ,  $\mu_l = \log(0.5)$ . We see that for the re-scaled input space and normalized function output, the sampled length-scales are around half of the input domain. Such settings are reasonable to capture the correlations between the observations and are also interpretable.



(a) Results with different hyperpriors.



(b) Posterior probability distributions of length-scales and model noise variance.

Figure 5: Effects of hyperpriors for kernel hyperparameters on Branin function.

## 7.5.2 Performance profiles

When computing the performance profiles (PPs), i.e., Dolan-Moré curves [30], we consider the number of function evaluations to find the global optimum as the performance measure. Specifically, let  $t_{p,a}$  denote the number of evaluations used by method/solver  $a$  to reach the global solution in experiment  $p$  (a lower value is better). The value equals to  $N_p + 100$  if the method cannot find the global optimum with  $N_p$  being the given observation budget for experiment  $p$ . The performance ratio  $\gamma_{p,a} = \frac{t_{p,a}}{\min\{t_{p,a}: a \in \mathcal{A}\}}$ , where  $\mathcal{A}$  represents the set including all comparing models, and the performance profile for each method is the distribution of  $p(\gamma_{p,a} \leq \rho)$  in terms of a factor  $\rho$ . We set  $\rho = 1 : N_{\max} + 1$ , where  $N_{\max} = \max N_p$  is the largest observation budget assumed for the compared experiments. We define the problem set  $\{\mathcal{P} \mid \forall p \in \mathcal{P}\}$  as the 10 runs for every function and draw the performance profiles of each model, also set  $\mathcal{P}$  as all 60 experiments across the 6 tested functions and estimate the overall performance profiles. We show the obtained PPs on the three 2D functions and across test functions (i.e., overall PPs) in Figure 2(c) for illustration; the full results on all the tested functions along with the overall plot are shown below in Figure 6. Table 2 gives the AUC (area under the curve) values of these curves, where the AUC of the overall performance profiles are taken as the metric to compare the overall performances. As can be seen, BKTF obtains the best performance across all the considered functions. In addition, for most of the test functions, the AUC of grid-based baseline models is comparable with those of continuous GP-based models, suggesting that the discretization of the continuous space is feasible to simplify the optimization problem.

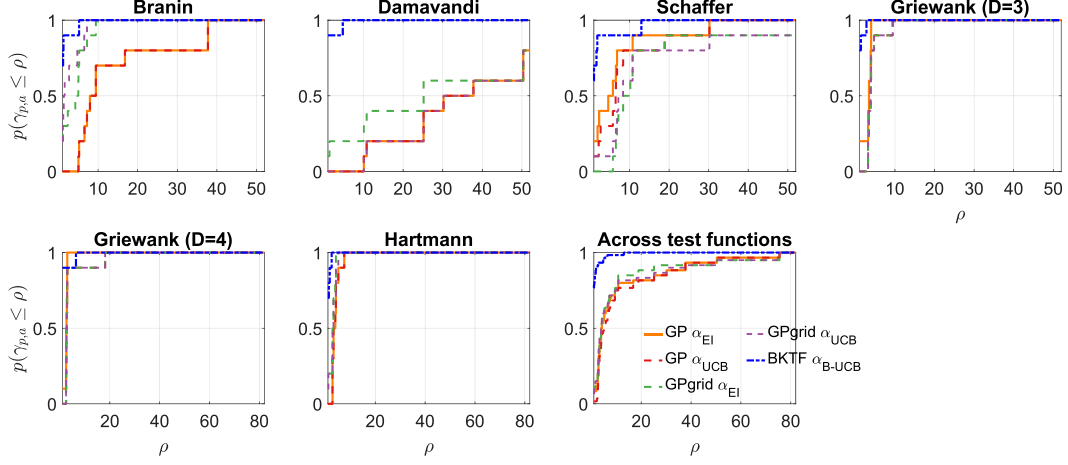


Figure 6: Performance profiles on the standard test functions.

### 7.5.3 Interpretation of results

The basis functions learned by BKTf are interpretable. For example, Figure 7 shows the latent factors ( $r = 1$ ) obtained at the last iteration of function evaluation in one run on 3D Griewank function. We see that BKTf can learn the periodicity (global structure) of the function benefited from the low-rank modeling. On the other hand, a stationary and separable GP cannot, other than using a specific kernel function such as the periodic kernel, which however requires strong prior knowledge to set the periodicity kernel hyperparameter.

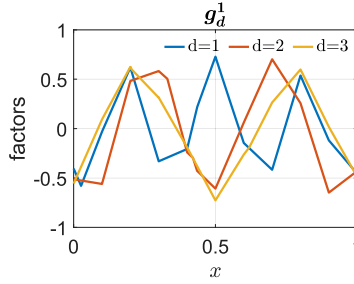


Figure 7: Examples of latent factors learned by BKTf on 3D Griewank function.

## 7.6 Hyperparameter tuning for machine learning

Table 4 lists all the hyperparameters in the tuning tasks.

Table 4: Hyperparameters of the tested ML algorithms.

Dataset	Algorithm	Hyperparameters	Type	Search space
MNIST	RF classifier	no. of estimators	discrete	[10, 100]
		max depth	discrete	[5, 50]
		max features	discrete	[1, 64]
		min samples split	discrete	[2, 11]
	NN classifier	neurons	discrete	[10, 100]
		batch size	discrete	[16, 64]
		epochs	discrete	[20, 50]
Boston housing	RF regressor	no. of estimators	discrete	[10, 100]
		max depth	discrete	[5, 50]
		max features	discrete	[1, 13]
		min samples split	discrete	[2, 11]
	NN regressor	neurons	discrete	[10, 100]
		batch size	discrete	[16, 64]
		epochs	discrete	[20, 50]