Analysis of the Histogram Intersection Kernel for Use in Bayesian Optimization

Dennis Becker

***Abstract*—In optimization problems, a typical assumption is that the objective function is cheap to evaluate. However, many problems are not conform to this assumption. When the objective function is expensive to evaluate, Bayesian optimization is a powerful strategy to estimate the optimum of the objective function. Typically, Bayesian optimization utilizes Gaussian process with an exponential kernel for the approximation of the objective function. However, the training time of Gaussian process scales cubically and prediction and memory requirement quadratically. This poses limitations on the number of utilized data points for the training of the Gaussian process. To potentially overcome this drawback, this paper analyzes the use of the histogram intersection kernel with approximation methods, which has been shown to scale linearly, as covariance function for the Gaussian process. The resulting algorithm is compared to the EGO optimization algorithm, which utilizes an exponential kernel, and random sampling on common optimization problems. The results show that the linear approximation of the histogram intersection kernel does not accurately approximate the error surface and introduces false minima which can prevent the algorithm to identify the global minimum of a function. However, the implemented algorithm performs better than random sampling and its potential for fast evaluation might make it attractive for time-limited optimization tasks.**

***Index Terms*—Bayesian optimization, histogram intersection kernel, gaussian process.**

I. Introduction

Nonconvex optimization problems arise in many research fields and are often solved using black box optimization algorithms. These algorithms typically do not rely on the computation of a gradient and are applicable to any optimization problem. The cost of optimizing such a problem is dominated by the number of objective function evaluations required to reach an acceptable solution. Difficulties in employing black box optimization algorithms arise when the optimization function is time intensive to evaluate.

A possibility to improve the search is to utilize the already evaluated solutions to build a model that approximates the function that is to be optimized. This approximation is then optimized instead of the true optimization function. Afterwards, the estimated solution of this model is estimated on the true cost function. This can often reduce the computation time in order to find an acceptable solution. Such models are also referred to as surrogates for the original objective function. However, a prerequisite for using such models is that the expense of model construction and prediction is lower than evaluating the true optimization function. An approach that aims at finding the global optimum with a small number of objective function evaluations is Bayesian optimization [1]-[3]. Typically, Bayesian optimization approximates the true objective function with a Gaussian process (GP) and estimates the next location to evaluate the true objective function using an acquisition function. An acquisition function balances between exploration and exploiting by considering the expected value of the surrogate function and the variance. Considering the uncertainty in the expected costs allows to search in so far unexplored areas. Research on Bayesian optimization dates back to Kushner (1964) and Mockus (1978) but subsided shortly after. New interest arose, when it was realized that Bayesian optimization provides a tool to estimate the hyperparameters of machine learning algorithms [6]-[8]. Hyperparameter estimation tends to be multi modal and expensive to evaluate functions, where Bayesian optimization has been used to successfully reduce the computational demand.

Although the Gaussian process can model a variety of functions and produce reasonable predictions, training time of the model scales cubically and memory usage quadratically with increasing data samples. This severely limits the sample size that can be used for function approximation. Sparse approximations can often provide a solution for this problem where only a subset of the data is used to alleviate the computational burden [9]-[11]. The utilized subset for the training of the model can consist of real training examples or pseudo inputs [12]. Another possibility is the distributed Gaussian process [13] where the computational and the memory load is distributed to many independent computational units. Each unit operates on a subset of the data and the results are recombined for an overall result. An alternative to these methods in order to reduce the memory and computational burden, that will be analyzed in this paper, is the use of Gaussian process with a histogram intersection kernel (HIK). This particular kernel has been utilized for computer vision due to its fast learning, classification and linear memory requirements [14], [15], [16]. Initially used as a kernel for support vector machines, it can also be used for the Gaussian process. Although the kernel only provides a piecewise linear approximation of the true function, its capability for large large-scale Gaussian process [17], [18] inference appears attractive for the use as surrogate function. The lower time and memory requirements could be beneficial for usage of mobile devices or when the optimization task poses limitations on the available time for optimization. Since the histogram intersection kernel has not been used in the context of optimization so far, we explore and analyze the use of this

Manuscript received September 23, 2017; revised December 12, 2017.

Dennis Becker is with Institute of Information Systems, Leuphana Universitiy, Lüneburg, Germany (e-mail: dbecker@leuphana.de).

DOI: 10.7763/IJMO.2017.V7.609

kernel in the context of surrogate function for Bayesian optimization. We compare the resulting Bayesian optimization algorithm with the EGO optimization algorithm [2] that utilizes an exponential kernel and has been shown to estimate the global optima quickly. Furthermore, we show the potential of parallelization of the estimation of the Gaussian process with HIK for additional speedup.

II. Method

A. Gaussian Process Regression

For the approximation of the true cost function, we consider the regression problem $y = f(\mathbf{x}) + \varepsilon \in \mathbb{R}$. In this function, it is assumed that the observed data y_i is generated by an unknown function $f(\mathbf{x}_i)$, and potentially corrupted by additional independent noise $\varepsilon \sim N(0, \sigma_y^2)$. A Gaussian process [19] defines a prior over functions that could have created the observed data with mean 0 and covariance given by $\Sigma_{ij} = \kappa(\mathrm{x_i}, \mathrm{x_j})$. The covariance matrix is built by the kernel function κ which describes the similarity among samples. For the prediction of new data points, the posterior predictive distribution of the new data sample $\mathbf{X}_*$ is calculated by marginalizing over all possible functions. The posterior predictive distribution of the corresponding function value $y_* = f(\mathbf{x}_*)$ is a Gaussian with mean and variance given by,

$$p(f_*|\mathbf{X}_*, \mathbf{X}, \mathbf{y}) = N(f_*|\boldsymbol{\mu}_*, \boldsymbol{\Sigma}_*), \tag{1}$$

$$\boldsymbol{\mu}_* = \mathbf{K}_*^T \mathbf{K}_\mathrm{y}^{-1} \mathbf{y}\,, \tag{2}$$

$$\boldsymbol{\Sigma}_* = \mathbf{K}_{**} - \mathbf{K}_*^T \mathbf{K}_\mathrm{y}^{-1} K_* \tag{3}$$

where $\mathbf{K}_\mathbf{y} = \kappa(\mathbf{X}, \mathbf{X}) + \sigma_y^2 \mathbf{I} \in \mathbb{R}^{N \times N}$, $\mathbf{K}_* = \kappa(\mathbf{X}, \mathbf{X}_*) \in \mathbb{R}^{N \times N_*}$, and $\mathbf{K}_{**} = \kappa(\mathbf{X}_*, \mathbf{X}_*) \in \mathbb{R}^{N_* \times N_*}$. For the Gaussian process regression, all these computations can be executed in closed form. However, the creation of the $N \times N$ kernel matrix requires $\mathcal{O}(N^2)$ space and inversion of this matrix $\mathcal{O}(N^3)$ time. These requirements practically limit the use of Gaussian process to data sets of size $\mathcal{O}(10^4)$.

The most widely used kernel in machine learning might be the squared exponential kernel (SEK) that is given by

$$\kappa_{\mathrm{SE}}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left(-\gamma \parallel \boldsymbol{x}_i - \boldsymbol{x}_j \parallel^2\right). \tag{4}$$

where the free parameter γ is called a hyperparameter that influences the fit of the Gaussian process to the data. Usually, the hyperparameters are inferred from the data by optimizing the likelihood of the Gaussian process given the data.

B. Histogram Intersection Kernel

To reduce the computational effort of the Gaussian process, the histogram intersection kernel is utilized. The histogram intersection kernel is defined as,

$$\kappa_{\mathrm{HIK}}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sum_{d=1}^{D} \min\left(\boldsymbol{x}_i(d), \boldsymbol{x}_j(d)\right). \tag{5}$$

It has been shown that properties of this kernel allow to speed up model learning and prediction of new data samples [14], [15], [16]. The multiplication of the kernel matrix **K** by an arbitrary vector *v* can be done without explicitly creating the kernel matrix, which enables sub quadratical calculation of the matrix-vector product. This alleviates computational and memory storage cost during the training of the model. Furthermore, the prediction of new data samples scales sub-linear, which improves prediction time respectively. However, these considerations only apply when each dimension of the kernel matrix is sorted. For illustration of these properties, we first consider the prediction of a new data sample given the matrix $\mathbf{K}_*$ and $\boldsymbol{\alpha} = \mathbf{K}_y^{-1}\boldsymbol{y}$,

$$\mathbf{K}_*^T \boldsymbol{\alpha} = \sum_{i=1}^{N} \boldsymbol{\alpha}(i) \sum_{d=1}^{D} \min(\boldsymbol{x}_i(d), \boldsymbol{x}_*(d))\,, \tag{6}$$

$$= \sum_{d=1}^{D} \left(\sum_{i:\, x_i(d) < x_*(d)} \alpha(i) x_i(d) + \; x_*(d) \sum_{j:\, x_j(d) \geq x_*(d)} \alpha(j) \right).$$

For each dimension, the summation can be separated into two parts. The first part consists of the samples **x** whose values are smaller than the value of the new sample to predict $\mathbf{x}_*$. For the calculation of this term, the values of **α** have to be multiplied by the values of **x** and are summed up. The second term consists of all samples where $\mathbf{x}_*$ is smaller than or equal to **x**. For this part of the kernel matrix $\mathbf{K}_*$, all values will be equal to $\mathbf{x}_*$. This allows to sum the remaining values of alpha and multiply the result by the value of $\mathbf{x}_*$. Similar observations apply to the multiplication of the kernel matrix **K** with an arbitrary vector $\boldsymbol{\nu}$. The following equation illustrates the calculation of one value of the kernel vector product for one dimension,

$$(\mathbf{K}\boldsymbol{\nu})_i = \sum_{j=1}^{N} \boldsymbol{\nu}(j) \cdot \kappa(\boldsymbol{x}_j, \boldsymbol{x}_i) \tag{7}$$

For the calculation of each value of the matrix-vector product, the kernel value will be $\mathbf{x}_j$ for all samples smaller than $\mathbf{x}_i$ and $\mathbf{x}_j$ for all remaining values. This allows the efficient multiplication of the kernel matrix with an arbitrary vector when the samples are sorted for each dimension. Therefore the kernel matrix has not to be stored, which reduces the memory requirements from $\mathcal{O}(N^2)$ to $\mathcal{O}(2ND)$ assuming the matrix with the sort indices is stored for later use.

The definition of the histogram intersection kernel can be further generalized to increase its flexibility. Boughorbel et al. (2005) have shown that any positive valued function $g(\cdot)$ can be applied to the data and the kernel remains positive definite. This allows transformation of the data and is often referred to as the generalized histogram intersection kernel:

$$\kappa_{\mathrm{GHIK}}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sum_{d=1}^{D} \min\left(g\left(\boldsymbol{x}_i(d)\right), g(\boldsymbol{x}_j(d))\right) \tag{8}$$

As transformation function, we utilize the exponential transformation:

$$g_{\exp}\left(\boldsymbol{x}(d)\right) = \frac{\exp(\eta\, |\boldsymbol{x}(d)|) - 1}{\exp(\eta) - 1}. \tag{9}$$

This transformation retains the order of the samples and therefore the fast evaluation of the kernel vector product can

still be applied. For each dimension, a different parameter $\boldsymbol{\eta}(d)$ is used in order to individually weight each dimension. Furthermore, we apply a positive shift to the data for each individual dimension. This allows to approximate functions that are also defined for negative values and circumvents that the prediction of the HIK at the origin will be zero

C. Bayesian Optimization

For the optimization of a function $f\boldsymbol{\mathcal{X}}: X \longrightarrow \mathbb{R}$ that is to be minimized on some domain $\boldsymbol{\mathcal{X}} \subseteq X$, we wish to find

$$x^* = \arg\min_{x \in X} f(x). \tag{10}$$

Typically, we aim to find the global optimum of a potential multi-modal function. Commonly, for the optimization of some function, assumptions are made such as non-linearity or in contrast smoothness. Bayesian optimization, although not strictly required, assumes a Gaussian process prior on the function f: $p(f)=\mathcal{GP}(f;\mu;\mathbf{K})$. Given observations of the function, the Gaussian process is conditioned on the data. The Gaussian process then approximates the underlying function and can be used to estimate an x that might provide a better solution. A function that estimates the next best location to evaluate is called an acquisition function. The simplest acquisition function would be to use the minimum of the conditioned Gaussian process. But this procedure can easily lead to a local minimum because the acquisition function does not consider the uncertainty about the surface. Therefore, this acquisition function would focus on exploitation rather than balancing the trade-off between exploration and exploitation. Exploration represents a global search in order to identify further minima and exploitation emphasis on the local search for refining a solution.

There are a variety of acquisition functions [1] that consider the trade-off between exploration and exploitation such as the probability of improvement [4], expected improvement, entropy search, and upper confidence bound [21]. In the following, the expected improvement and upper confidence bound acquisition function are introduced because these will be utilized for optimization.

For the calculation of the expected improvement, first the lowest observed function value so far $f_{min} = \min(y_1, \ldots, y_n)$ is needed. To estimate if the evaluation of the function at a new point y might be beneficial, the prediction of the mean and variance of the Gaussian process are required next. This prediction is considered as a random variable Y and the improvement is defined as $I(x) = \max(f_{min} - Y, 0)$. This again represents a random variable and the expected value of this variable is estimated to obtain the expected improvement:

$$\mathbf{E} \equiv \mathbf{E}[I(\boldsymbol{x}) = \max(f_{min} - Y, 0)]. \tag{11}$$

The expected improvement (EI) acquisition function can be evaluated analytically [2]:

$$\mathbf{E}[I(\boldsymbol{x})] = \big(f_{min} - \mu(\boldsymbol{x})\big)\Phi(\boldsymbol{Z}) + \sigma(\boldsymbol{x})\phi(\boldsymbol{Z}), \tag{12}$$

$$Z = \frac{f_{min} - \mu(\boldsymbol{x})}{\sigma(x)}, \tag{13}$$

where $\phi(\cdot)$ and $\Phi(\cdot)$ denote the PDF and CDF of the standard normal distribution respectively. Another acquisition function that considers the mean and the weighted variance is the upper confidence bound (UCB) [21]. New sample points are selected based on $\mu(x) - \beta\sigma(x)$. Where the choice of the weight parameter $\beta \geq 0$ is left to the user.

An example of Bayesian optimization for the squared exponential and generalized histogram intersection kernel is shown in Fig. 1. Different points from an unknown cost function are estimated and fitted using the Gaussian process. By optimizing the acquisition function, in this case, the expected improvement, the next point that is to be evaluated on the true cost function is identified.

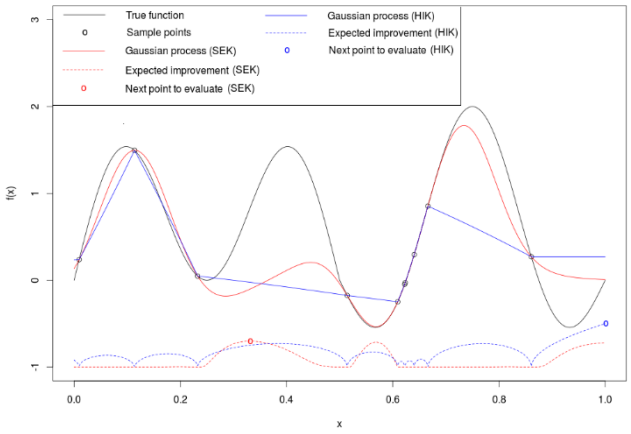

Fig. 1. Example of Bayesian optimization utilizing squared exponential and histogram intersection kernel.

D. Utilized Approximations

The implementation of the Gaussian process with HIK mostly follows the implementation described in Rodner et al. (2016). For the prediction of the mean value of new samples, the $\boldsymbol{\alpha}$ vector is required. The $\boldsymbol{\alpha}$ values are estimated using conjugate gradient descent, which allows to avoid estimation and inversion of kernel matrix $\mathbf{K}_y$ estimated from the data. In the absence of round-off errors, this would allow to obtain the exact solution after N iterations. In practice, however, the algorithm can be stopped significantly earlier when the norm of the residual drops below a predefined threshold.

For the estimation of the sample variance, which is required for the acquisition function of the Gaussian process, the inverse of the matrix $\mathbf{K}_y$ is needed. However, to reduce the computational demand, the variance can be approximated with a fewer number of eigenvectors. For the estimation of the eigenvalues, the Lanczos algorithm [22] with full reorthogonalization is used. To estimate the eigenvalues from the resulting tridiagonal matrix, the strum sequences are estimated and bounded by bisection. The corresponding eigenvectors are estimated using inverse iteration. Although this requires inversion of the kernel matrix, this is achieved by applying the conjugate gradient descent again. The determinant of the kernel matrix, likelihood, and prediction variance are approximated using the approach presented in Rodner et al. (2016). The implementation of the Gaussian processes with the

described approximations for the HIK is implemented as an *R* [23] package, to provide other researchers an easy access to the package for their own research.

The Bayesian optimization algorithm is implemented in *R* using the implemented package of the Gaussian process. For the optimization of the acquisition function, the *genoud* [24] package is used, which is a combination of a genetic algorithm for the global and quasi-Newton method for the local optimization. The hyperparameters that are required for the exponential transformation and shift of the data are estimated by maximizing the likelihood of the Gaussian process. The likelihood is optimized using the downhill simplex algorithm [25], which is a gradient-free optimization algorithm. This circumvents estimation of the gradient, which would also require an inversion of the kernel matrix. An example of the resulting approximation of the variance with a varying number of eigenvalues is displayed in Fig. 2. From the unknown cost function, 30 points were randomly sampled. The Gaussian process with HIK is fitted to the data and the standard deviation as well as their approximation with the largest eigenvector and largest five largest eigenvectors are plotted for comparison.

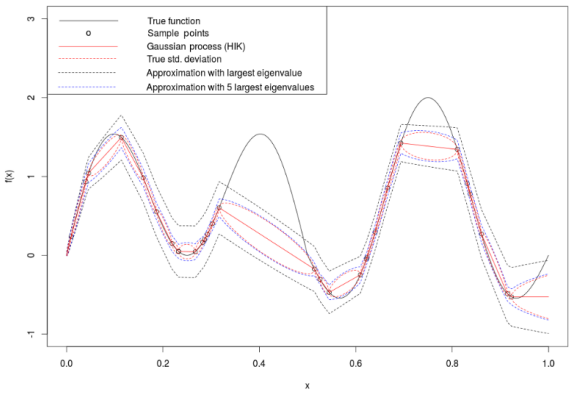

Fig. 2. HIK variance approximation.

III. Results

To analyze and visualize the fitting capabilities of the HIK, we consider a 1-D case of the Rastrigin function and 2-D case of the Branin function for optimization. Afterwards, we will compare the results with the EGO algorithm because it uses a non linear approximation of the error function and has been shown to find global optima surprisingly quick. Finally, we will conclude with the speedup that is provided by a parallel implementation of the fast kernel matrix-vector multiplication.

A. Visual Analysis

For the illustration of the Bayesian optimization algorithm with HIK we visually analyze the optimization of the 1-D Rastrigin and 2-D Branin function. The EGO algorithm uses the expected improvement acquisition function to identify the next point to evaluate on the true objective function. Therefore, we will analyze the use the expected improvement, but will also consider the use of the UCB acquisition function for reasons that will become apparent later. For the Rastrigin function to be optimized, we assume the function range [-10, 10]. Since the HIK can only be used for positive numbers, the data is shifted into the positive range. As stated in the previous section, any transformation can be applied to the data while the kernel still remains positive definite. Additionally, the exponential transformation is used to increase the fitting capabilities and an additional positive shift of the data because otherwise, the HIK will always predict a value of 0 for an x-position of 0. The magnitude of the additional shift to the data is estimated from the data as additional hyperparameter.

For the optimization of the Rastrigin function, initially, six equally spaced points are chosen to start the Bayesian optimization. The number of utilized eigenvectors influences the quality of the predicted variance. Since the estimation of the eigenvectors is considered time intensive, we divide the number of samples by four and round up this value and estimate this amount of eigenvectors for the variance approximation. This appears to provide a good appropriation to the true variance especially in areas with only a few samples. However, in areas with many samples, the approximation loses accuracy, which can be hindering during the exploitation phase.

The initial approximation with the six estimated function values is shown in Fig. 3. The HIK can approximate the overall shape of the function and also the variance appears to capture the overall uncertainty of the Rastrigin function.

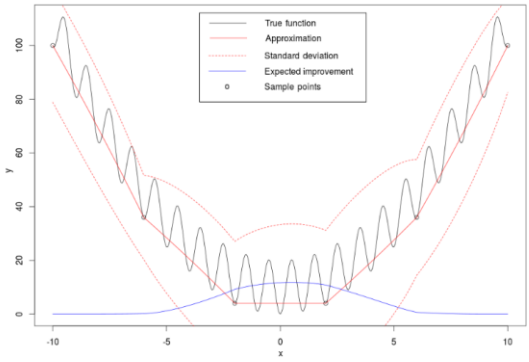

Fig. 3. Bayesian optimization of the Rastrigin function with an initial estimation of six points.

Furthermore, the expected improvement signals where to estimate the next true function value and also indicates the correct location of the global minimum. The acquisition function is optimized to estimate the point with the highest expected improvement and estimated as next point on the true cost function.

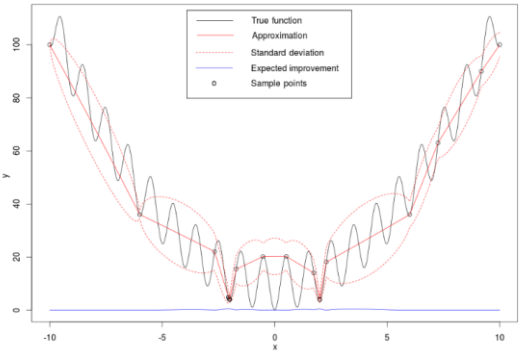

Fig. 4. Optimization results of the Rastrigin function after 20 function estimations utilizing the EI acquisition function.

Fig. 4 shows the status of the Bayesian optimization algorithm with 20 estimated true function values (including the six initial estimations). The algorithm was not able to locate the global minimum and got stuck in two local optima while underestimating the variance. This too small variance effectively prevents the algorithm to search near the area of the global optimum. This results in a flattened expected improvement function, which suggests that the optimal solution might already be found and prevents further search. The EI apparently underestimates the variance and stagnates in a local minimum. To counter this shortcoming, we apply the UCB acquisition function, which potentially emphasizes more on exploration. The UCB function requires a parameter β which is a factor that weights the standard deviation. It effectively balances the trade-off between exploration and exploitation. The choice of this parameter might require domain knowledge, and no clear recommendations can be given. However, for the following analysis of the Rastrigin function, a parameter of $\beta = 2.5$ is chosen to upscale the influence of the variance.

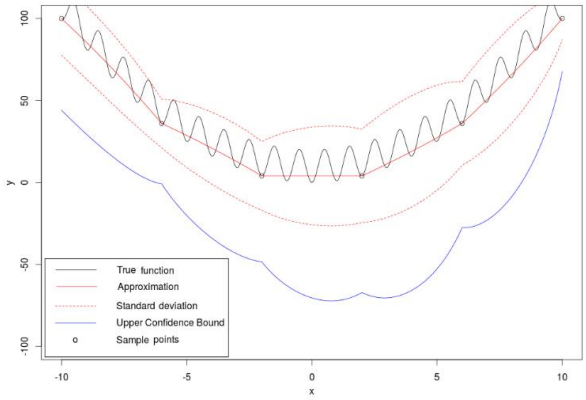

Fig. 5. Initial estimate of Rastrigin function utilizing the UCB acquisition function.

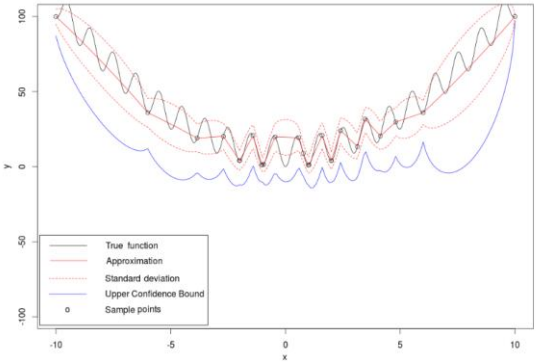

Fig. 6. Optimization results of the Rastrigin function after 20 function estimations utilizing the UCB acquisition function.

Fig. 5 shows the initial estimates of the UCB acquisition function. In this Figure, the acquisition function is represented by the mean value plus times 2.5 the predicted standard deviation. This emphasizes the uncertainty because it could be noticed from the previous analyzes, that the HIK kernel appears to underestimate the variance. In contrast to the expected improvement plot, the representation of the acquisition function is inverted to provide better visibility. Therefore, we wish to estimate the solution with the lowest UCB value as next point on the true function.

Fig. 6 shows the status of the Bayesian optimization after 20 function evaluations using the UCB acquisition function. Although the UCB acquisition function was not able to estimate the global optimum either, it can be noticed that in contrast to the expected improvement function, the local optimum can still be estimated due to the remaining uncertainty.

After this visual analysis of the one dimensional case, we move on to the two dimensional case. For the two dimensional case, the 2-D Branin function in an interval [0, 1] is considered. For the optimization of the Branin function, an initial grid of nine sample points is utilized and the resulting approximation of the HIK is displayed in Fig. 7. It can be noticed that the approximation is quite poor and does not reflect the true function and that the sampled points are not represented accurately.

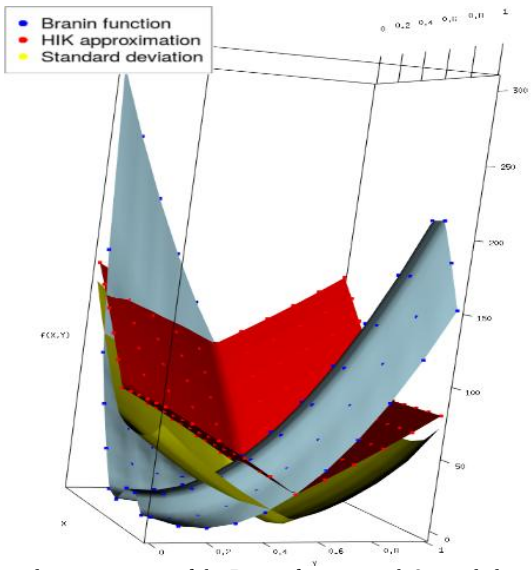

Fig. 7. Initial approximation of the Branin function with 9 sampled points.

Fig. 8 illustrates the optimization results after a total of 28 function evaluations. It can be seen that the sampled points are all sampled in the area of the HIK approximation falsely introduced minima. Additionally, the variance of the HIK approximation is considerably low, which prevents the algorithm from searching in areas with a lower error.

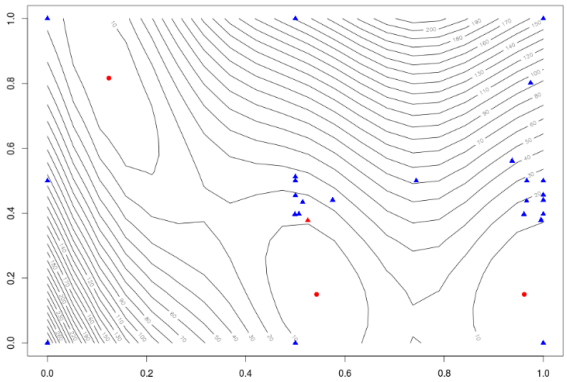

Fig. 8. Bayesian optimization after 28 evaluations. Estimated points are shown in blue triangles, next point to evaluate is shown with red triangle. The location of the global optima are shown as red dots.

The considerably low variance could be explained with the assumptions of the linear kernel, which reflects the assumption made on the true cost function. It assumes that unknown values are within a linear range of two known values. This leads to underestimation of the variance which effectively prevents the algorithm from exploring the optimal regions. Based on these observations, we assume that using the EI acquisition function does not allow to move away from the falsely introduced minima. Therefore the UCB acquisition function is applied in the following analysis.

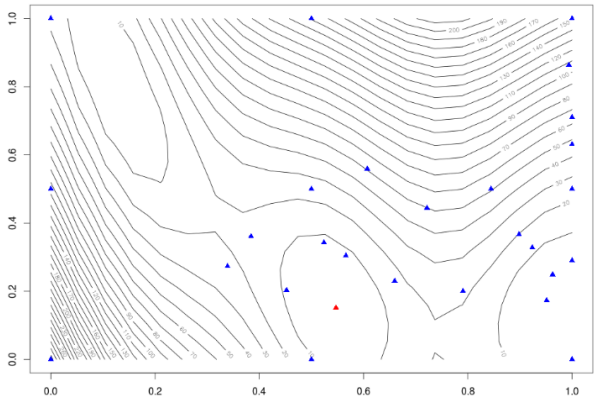

Fig. 9. Results of the optimization of the Branin function using the UCB acquisition function. Estimated points are shown in blue, next point to evaluate is shown in red.

Fig. 9 shows that the UCB acquisition function explores the error surface better than the EI acquisition function. Apparently, the search focuses around two of the tree available minima. A further observation that can be made from the 2-D case is that the HIK kernel is not able to fit abrupt increase in the cost function. It accordingly fails to approximate the true cost function and even the true measurement values. To alleviate the problem of rapidly increasing values, we apply the log-transformation to the problem. Results of this transformation for the UCB function are shown in Fig. 10.

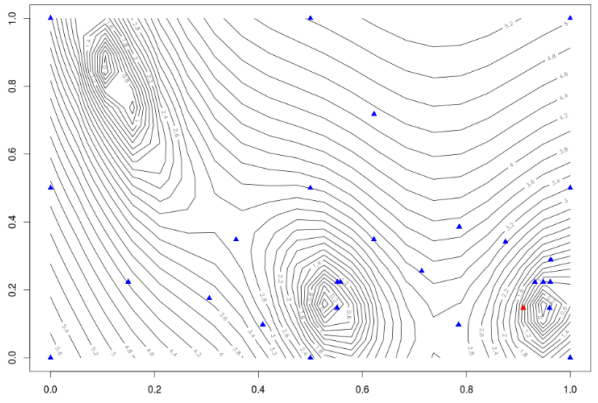

Fig. 10. Contour plot of the log Branin function after 28 function evaluations. Estimated points are shown in blue, next point to evaluate is shown in red.

Applying the log-transformation allows the HIK to provide a more accurate fit of the surface and enables the Bayesian optimization algorithm to further reduce the error. Although the HIK appears to approximate the function to some degree, it still assumes very large values around the third minima as shown in Fig. 11, which prevents discovery of this minima.

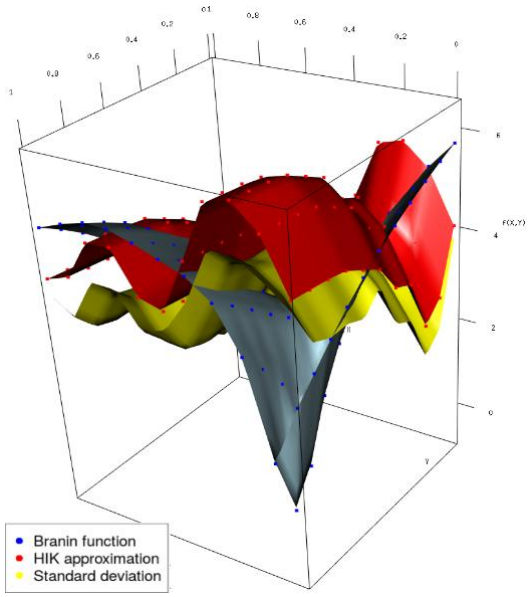

Fig. 11. Approximation of the log Branin function after 28 function evaluations.

A. Comparison of Optimization Results

After the visual analysis of the fitting capabilities of the HIK for Bayesian optimization, we compare the algorithm to the EGO algorithm [2]. Therefore, we analyze the same test functions as presented in the paper of Jones et al. (1998) as well as their archived optimization results.

For further comparison, the results of a Bayesian optimization with a squared experimental kernel applying the same settings as the HIK implementation are provided. To emphasize the exploration and avoid underestimation of the variance, a $\beta = 5.576$ is chosen for the UCB acquisition function. For all problems, we use one-fourth of the eigenvectors for variance approximation.

TABLE I: TEST FUNCTION RESULTS FOR THE EGO ALGORITHM, THE BAYESIAN OPTIMIZATION WITH HIK AND SEK, AND RANDOMLY TAKEN SAMPLES

	Samples evaluated	Error EGO	SEK UCB mean (std)	HIK UCB mean (std)	Random sample mean (std)
Branin	28	0.2%	8% (10%)	179% (270%)	407% (400%)
log Branin				94% (128%)	
Goldstein-Price	32	0.1%	25% (13%)	35% (10%)	34% (16%)
log Goldstein-Price				30% (6%)	
Hartman 3	34		4% (5%)	13% (7%)	16% (12%)
log Hartman 3		1.7%		5% (3%)	
Hartman 6	84		7% (6%)	49% (18%)	42% (14%)
log Hartman 6		1.9%		36% (9%)	

Table I shows the results on the test problems for the EGO, the Bayesian optimization with HIK and SEK utilizing the UCB acquisition function, and a reference measure where the same amount of samples is taken

randomly from the search space. Since the implementation of the Bayesian optimizations does not provide deterministic results, due to the non-deterministic optimization of the acquisition function, the function was optimized 50 times and the mean and standard deviation are provided. The same applies for the reference measure of randomly taken samples.

B. Time Requirements and Parallelization

Finally, we analyze the improvement in each iteration and runtime of an implementation of the EGO algorithm and the implemented Bayesian optimization using the HIK. For the estimation of the runtime and error of the EGO algorithm, the *DiceOptim* [26] package for R is used. This package provides an implementation of the EGO optimization algorithm. However, in contrast to the original EGO algorithm presented in the paper by Jones at al. (1998), that utilizes a bound on the expected improvement function for their optimization, the *DiceOptim* package relies on the *genoud* package for the optimization of the acquisition function. In order to make both algorithms comparable, the same settings for the *genoud* package have been used in both algorithms. The population size was set to 20 and the number of generations was set to 12. The generation size of 12 is the default setting utilized in the *DiceOptim* package and a population size of 20 has been utilized in the example, provided in the package documentation, for the optimization of the Branin function. However, it might be argued that these settings are quite low for higher dimensional problems. Since both algorithms will not provide deterministic outcomes, the optimization was repeated 50 times. The results for the log-transformed Hartman3 optimization function are shown Fig. 12.

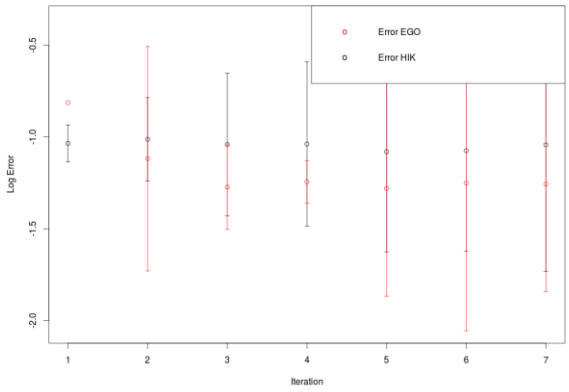

Fig. 12. Comparison of the error of the log-Hartman3 function in each iterations of the Bayesian optimization using EGO and HIK.

For the comparison on the log-transformed Hartman3 function, it appears that the Bayesian optimization with the HIK does not reduce the error as much as the EGO algorithm within the same amount of true objective function evaluations. The execution time of the EGO algorithm was on average 2.40 seconds with a standard deviation of 0.18 seconds. The average execution time of the Bayesian optimization implementation with HIK was 1.66 seconds with a standard deviation of 0.15 seconds.

The comparison of both algorithms for the log-transformed Hartman6 problem is shown in Fig. 13. In the early iterations it appears that the EGO algorithm emphasis more on exploring than the HIK. With increasing iterations, the EGO algorithm can improve more than the HIK implementation and also yields an overall better optimization result, which is also indicated by the larger variance of the error. Surprisingly, the average of execution time for the HIK was longer than the execution time of the EGO algorithm. The EGO implementation required on average 8.61 seconds with a standard deviation of 0.45 seconds. In comparison, the HIK implementation required on average 17.43 seconds with a standard deviation of 1.58 seconds. This might be due to the increasing number of eigenvectors that are to be estimated for the six-dimensional Hartman6 function. The influence of the number of estimated eigenvectors on the runtime is discussed next as well as a possibility to reduce this runtime.

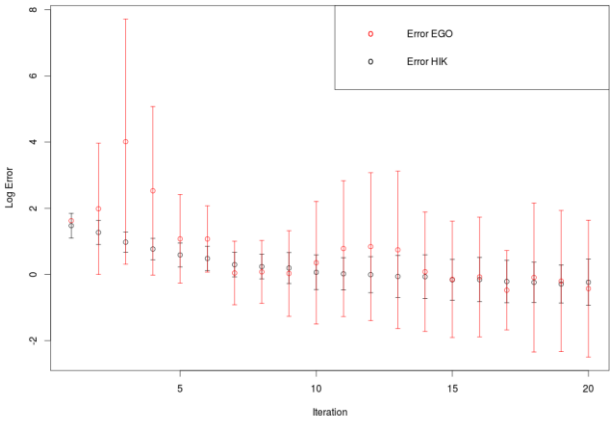

Fig. 13. Comparison of the error of the log-transformed Hartman6 function in each iterations of the Bayesian optimization using EGO and HIK.

The required execution time to estimate the Gaussian process utilizing the HIK can be reduced, by recognizing that the fast vector kernel multiplication is independent in each dimension. Therefore, the multiplication of the histogram intersection kernel matrix with an arbitrary vector can be parallelized. To analyze this potential speed improvement 1000 data points with 100 dimensions were randomly sampled from the Rastrigin function. Then the Gaussian process with HIK with an increasing number of eigenvectors was estimated. This has been repeated 25 times and the results for sequential matrix-vector and parallel matrix-vector multiplication are shown Fig. 14. For the parallel estimation of the Gaussian process, 8 threads were used.

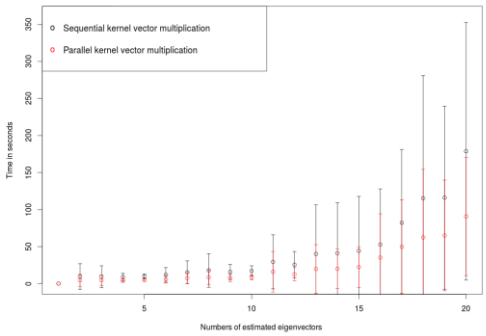

Fig. 14. Time to estimate Gaussian process with HIK for an increasing number of eigenvectors.

It can be noticed that the runtime quadratically increases with the number of estimated eigenvectors. This complexity increase might mostly be due to the utilized Lanczos algorithm for the estimation of the eigenvalues. Furthermore, a large variance in the estimation time can be noticed. This is due to the varying convergence speed of the individual parts

of the implementation. The convergence speed of the utilized conjugate gradient descent depends on the starting vector. Therefore, the algorithm requires more or fewer iterations till convergence. The same applies for the inverse iteration method for estimation of the eigenvectors, which also utilizes the conjugate gradient descent function. Depending on the starting vector it requires more or fewer iterations until the vector has the desired accuracy. However, parallelization of the fast vector-matrix multiplication can speed up the estimation of the Gaussian process for an increasing number of estimated eigenvectors.

IV. Discussion

The implemented Bayesian optimization algorithm with HIK for the approximation of the true cost function does not provide solutions that are comparable to the EGO algorithm, which utilizes a squared exponential kernel for function approximation. Furthermore, the analyses presented in this paper, show that the piecewise linear approximation of the HIK cannot accurately approximate the true cost function. The underestimation of the variance and inability to represent observed data points, as shown in the visual analysis, leads to the introduction of false minima. This suggests that poor approximation models can lead to large approximation errors and introduce false minima, which has already been shown in the case of genetic algorithms with surrogate function [27]. The same problem appears to apply here for the optimization with Bayesian optimization and the HIK. In conclusion, this emphasizes the role of the covariance function as assumptions that are made on the cost function to be optimized. Although the use of a log-transformation to the problem, to smooth out steep increases in the cost function, improves the performance of the algorithm, the problems remain.

This analysis further suggests that the fitting capability of the HIK appears to decrease with an increase in dimensions. Where the illustration of the one-dimensional problem indicates a sufficient fit of the surface, in the two-dimensional case, the fitting capabilities greatly suffers. This is also suggested by the results of the comparison of different optimization problems. Where the difference between the remaining error of the HIK and the random sample decreases with an increase in dimension. However, it appears that the HIK can greatly benefit from a log-transformation of the objective function, and is then able to perform better than random sampling. The effect of introducing false minima and underestimation of the variance might be specifically apparent in the cause of the Goldstein-price optimization function. The Goldstein-price optimization function has 3 minima where one of them is the global minimum. Apparently, the optimization using the HIK focuses around one local minimum and the lack of approximation capabilities prevents the algorithm to explore the global optimum similar. This effect might be identical the one illustrated for the log-transformed Branin function.

Although the implemented Bayesian optimization with the HIK has linear memory requirements, the estimation of the eigenvectors scales quadratically in terms of runtime for the utilized approximations. However, a higher number of eigenvalues is only required for a more accurate approximation of the sample variance. A lower number of eigenvalues might be sufficient for the envisioned optimization problem and the runtime for an increasing number of eigenvectors can be reduced utilizing parallelization.

V. Conclusion

In this paper, the use of Bayesian optimization with HIK was analyzed and compared to the results of the EGO optimization algorithm. The HIK can provide linear runtime and memory requirements, which makes it attractive for large problems. However, estimation of the eigenvectors that are required for prediction of the sample variance is computationally expensive. To improve the speed and memory requirements, approximations for the prediction variance and log-likelihood for hyperparameter estimation are utilized. Despite the benefits, the HIK posses limitations and only provides a piecewise linear approximation of the true function. Although the definition of the kernel can be generalized to improve its fitting capabilities, it appears unsuited for the use as the surrogate function in optimization.

The visual inspection and comparison with the EGO algorithm show that a poor fitting surrogate function can be hindering for optimization purposes. Although the HIK kernel can provide better results in image recognition tasks than other traditional kernels, potentially due to generalizing well among multiple dimensions, this generalizability is not suited for optimization tasks, where an accurate fit of the surrogate function is preferable.

When applying approximations for the predictive variance, the amount of the eigenvalues has an influence on the performance in the context of Bayesian optimization. Although the estimation of fewer eigenvalues is beneficial in terms of runtime and memory requirement, the approximated variance can significantly differ from the variance estimated with all eigenvalues in areas with many samples. This mainly influences the exploitation phase, where the Bayesian optimization refines the solutions around a currently estimated minimum. Therefore, the choice of the numbers of eigenvalues influences the performance of the algorithm. Also, the choice of the β parameter when applying the UCB acquisition function influences the behavior and performance of the Bayesian optimization.

It appears that a well-fitting model that adequately represents the data and the variance is necessary in order to steer search into promising areas and to allow an efficient exploitation to enable good results.

References

[1] E. Brochu, V. M. Cora, and N. De Freitas, “A tutorial on Bayesianoptimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning,” ArXiv, p. 49, 2010.

[2] D. R. Jones, M. Schonlau, and W. J. Welch, "Efficient global optimization of expensive black-box functions," *Journal of Global Optimization*, vol. 13, pp. 455–492, 1998.
[3] J. P. C. Kleijnen, "Simulation-optimization via Kriging and bootstrapping: a survey," *Journal of Simulation*, vol. 8, no. 4, 2014.
[4] H. J. Kushner, "A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise," *Journal of Basic Engineering*, vol. 86, no. 1, pp. 97+, 1964.
[5] J. Mockus, V. Tiesis, and A. Zilinskas, "The application of Bayesian methods for seeking the extremum," *Towards Global Optimisation. II*, no. December 2016, pp. 117–129, 1978.
[6] J. Snoek, H. Larochelle, and R. Adams, "Practical bayesian optimization of machine learning algorithms." *Nips*, pp. 1–9, 2012.
[7] K. Swersky, J. Snoek, and R. P. Adams, "Multi- task bayesian optimization," *Advances in Neural Information Processing Systems*, pp. 2004–2012, 2013
[8] M. A. Gelbart, J. Snoek, and R. P. Adams, "Bayesian optimization with unknown constraints," in *Proc. the Thirtieth Conference on Uncertainty in Artificial Intelligence*, Arlington, Virginia, United States: AUAI Press, 2014, pp. 250–259
[9] C. Williams and M. W. Seeger, "Using the nystrom method to speed up kernel machines," *NIPS Proceedings*, vol. 13, pp. 682–688, 2001.
[10] J. Qui˜nonero-candela, C. E. Rasmussen, and R. Herbrich, "A unifying view of sparse approximate Gaussian process regression," *Journal of Machine Learning Research*, vol. 6, pp. 1935–1959, 2005.
[11] J. Hensman, N. Fusi, and N. D. Lawrence, "Gaussian processes for big data," in UAI, 2013, pp. 282–290.
[12] E. Snelson and Z. Ghahramani, "Sparse gaussian processes using pseudo-inputs," *Advances in Neural Information Processing Systems*, MIT Press, 2006, pp. 1257–1264.
[13] M. P. Deisenroth and J. W. Ng, "Distributed gaussian processes," in *Proc. the International Conference on Machine Learning*, vol. 37, p. 10, 2015.
[14] J. Wu, W. Tan, and J. Rehg, "Efficient and effective visual codebook generation using additive kernels," *The Journal of Machine Learning Research*, vol. 12, pp. 3097–3118, 2011.
[15] J. Wu, "A fast dual method for HIK SVM learning," *Lecture Notes in Computer Science*, vol. 6312 LNCS, no. PART 2, pp. 552–565, 2010.
[16] S. Maji, A. C. Berg, and J. Malik, "Classification using intersection kernel support vector machines is efficient classification using intersection kernel support vector machines is efficient," *in Slides*, 2008, pp. 1–8.
[17] E. Rodner, A. Freytag, P. Bodesheim, and J. Denzler, "Large-scale gaussian process classification with flexible adaptive histogram kernels," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 7575 LNCS, no. PART 4, 2012, pp. 85–98.
[18] E. Rodner, A. Freytag, P. Bodesheim, B. Fr ̈ohlich, and J. Denzler, "Large-scale gaussian process inference with generalized histogram intersection kernels for visual recognition tasks," pp. 1–28, 2016
[19] C. Rasmussen and C. Williams, *Gaussian Processes for Machine Learning, ser. Adaptive Computation and Machine Learning. Cambridge*, MA, USA: MIT Press, Jan. 2006.
[20] S. Boughorbel, J.-P. Tarel, and N. Boujemaa, "Generalized histogram intersection kernel for image recognition," in *Proc. the fIEEEg International Conference on Image Processing*, vol. 3, no. October, pp. III—-161, 2005.
[21] D. D. Cox and S. John, "SDO: A statistical method for global optimization," *Multidisciplinary Design Optimization*, pp. 315–329, 1997.
[22] C. Lanczos, "An iteration method for the solution of the eigenvalue problem of linear differential and integral operators," *Journal of Research of the National Bureau of Standards*, vol. 45, no. 4, p. 255, 1950.
[23] R. Core, "Development team, R: A language and environment for statistical computing," *R Foundation for Statistical Computing*, Vienna, Austria, 2014.
[24] S. C. Goslee, D. L. Urban, J. H. Jones, and A. D. Models, *Journal Of Statistical Software*, vol. 22, no. 11, 2007.
[25] J. A. Nelder and R. Mead, "A simplex method for function minimization," *The Computer Journal*, vol. 7, no. 4, 1965.
[26] O. Roustant, D. Ginsbourger, and Y. Deville, "Dicekriging, diceoptim: Two r packages for the analysis of computer experiments by kriging-based metamodeling and optimization," *Journal of Statistical Software, Articles*, vol. 51, no. 1, pp. 1–55, 2012.
[27] Y. Jin, M. Olhofer, and B. Sendhoff, "On evolutionary optimisation with approximate fitness functions," in *Proc. the 2000 Genetic and Evolutionary Conference (fGECCOg 2000)*, pp. 786–793, 2000.

Dennis Becker is with Institute of Information Systems, Leuphana Universitiy, L ̈uneburg, Germany