CLUSTERING-GUIDED GP-UCB FOR BAYESIAN OPTIMIZATION

Jungtaek Kim and Seungjin Choi

Department of Computer Science and Engineering, POSTECH, Korea
{jtkim,seungjin}@postech.ac.kr

ABSTRACT

Bayesian optimization is a powerful technique for finding extrema of an objective function, a closed-form expression of which is not given but expensive evaluations at query points are available. Gaussian Process (GP) regression is often used to estimate the objective function and uncertainty estimates that guide GP-Upper Confidence Bound (GP-UCB) to determine where next to sample from the objective function, balancing exploration and exploitation. In general, it requires an auxiliary optimization to tune the hyperparameter in GP-UCB, which is sometimes not easy to carry out in practice. In this paper we present a simple practical method which improves GP-UCB, especially in cases where the objective function is not smooth with sharp peaks and valleys. We first present a geometric interpretation of GP-UCB on which we base our development of the clustering-guided method to select the next observation. Clustering is applied to two-dimensional vectors whose entries correspond to the posterior mean and standard deviation computed by GP regression, which is followed by utility maximization with GP-UCB, in order to determine where next to sample from the objective function. Experiments on various functions demonstrate our method alleviates the chance of being trapped in local extrema, making small efforts for auxiliary optimization.

Index Terms— Bayesian optimization, black-box function optimization, GP-UCB.

1. INTRODUCTION

Bayesian optimization involves finding minima (or maxima) of a black-box objective function $f(\mathbf{x})$ that is expensive to evaluate:

$$\arg\min_{\mathbf{x}\in\mathcal{X}\subset\mathbb{R}^D} f(\mathbf{x}), \tag{1}$$

where $\mathcal{X}$ is a compact set. In general, the closed-form expression of $f(\cdot)$ is not given. A popular approach to solving this problem is to search a minimum, gradually accumulating observations $\mathcal{D}_{1:t} = \{(\mathbf{x}_1, f_1), (\mathbf{x}_2, f_2), \ldots, (\mathbf{x}_t, f_t)\}$, evaluated at t sample points $\mathbf{x}_1, \ldots, \mathbf{x}_t$, with t increasing, where $f_t = f(\mathbf{x}_t)$. We define

$$\begin{aligned} \mathbf{X}_t &= [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_t] \in \mathbb{R}^{D\times t}, \\ \boldsymbol{f}_t &= [f_1, f_2, \ldots, f_t]^\top \in \mathbb{R}^t. \end{aligned}$$

Bayesian optimization provides an efficient approach in terms of the number of function evaluations required. To this end, Bayesian optimization places a prior on the latent function $f(\cdot)$ and combines it with the likelihood function $p(\mathcal{D}_{1:t}|f)$ to calculate the posterior distribution

$$p(f|\mathcal{D}_{1:t}) \propto p(\mathcal{D}_{1:t}|f)p(f). \tag{2}$$

This posterior distribution yields an estimate of the objective function $f(\cdot)$, which is also known as *surrogate function*, allowing us to evaluate the value of f_{t+1} at a new point $\mathbf{x}_{t+1}$ by calculating the predictive distribution $p(f_{t+1}|\mathbf{x}_{t+1}, \mathcal{D}_{1:t})$. A Gaussian Process (GP) prior with zero mean function and covariance function $k(\cdot,\cdot)$ is widely used as a prior over the latent function $f(\cdot)$ in Bayesian optimization. We define

$$\begin{aligned} \mathbf{k}_t^\top(\mathbf{x}_{t+1}) &= [k(\mathbf{x}_{t+1}, \mathbf{x}_1), \ldots, k(\mathbf{x}_{t+1}, \mathbf{x}_t)], \\ \mathbf{K}_t &= \begin{bmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & \cdots & k(\mathbf{x}_1, \mathbf{x}_t) \\ \vdots & \ddots & \vdots \\ k(\mathbf{x}_t, \mathbf{x}_1) & \cdots & k(\mathbf{x}_t, \mathbf{x}_t) \end{bmatrix}. \end{aligned}$$

In this paper we use the Matérn class of covariance functions, which involves a smoothness parameter ν and a length-scale parameter l

$$k(\mathbf{x}, \mathbf{x}') = \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\frac{\sqrt{2\nu}\|\mathbf{x} - \mathbf{x}'\|}{l} \right)^\nu H_\nu \left(\frac{\sqrt{2\nu}\|\mathbf{x} - \mathbf{x}'\|}{l} \right),$$

where $\Gamma(\cdot)$ is the gamma function and H_ν is a modified Bessel function [1].

In practice, due to the possibility of the presence of measurement noise, the noisy observation of the objective function at $\mathbf{x}_t$ is considered, i.e., $y_t = f(\mathbf{x}_t) + \epsilon_t$ where $\epsilon_t \sim \mathcal{N}(0, \rho^2)$ is assumed to be Gaussian. Naturally, we can define $\mathbf{y}_t = [y_1, y_2, \ldots, y_t]$ for the noisy observations. With these definitions, the predictive distribution over y_{t+1} given $\mathbf{x}_{t+1}$ is easily computed:

$$p(y_{t+1}|\mathbf{x}_{t+1}, \mathcal{D}_{1:t}) = \mathcal{N}\left(\mu_t(\mathbf{x}_{t+1}), \sigma_t^2(\mathbf{x}_{t+1}) + \rho^2\right), \tag{3}$$

where posterior mean $\mu(\cdot)$ and variance $\sigma^2(\cdot)$ are calculated as

$$\mu_t(\mathbf{x}) = \mathbf{k}_t^\top(\mathbf{x}) \left[\mathbf{K}_t + \rho^2\mathbf{I}\right]^{-1} \mathbf{y}_t, \tag{4}$$

$$\sigma_t^2(\mathbf{x}) = k(\mathbf{x}, \mathbf{x}) - \mathbf{k}_t^\top(\mathbf{x}) \left[\mathbf{K}_t + \rho^2\mathbf{I}\right]^{-1} \mathbf{k}_t^\top(\mathbf{x}). \tag{5}$$

We have briefly explained how to estimate the black-box objective function, placing a prior over the latent function and calculating the posterior by combining it with the likelihood, in the framework of GP regression. The next step in Bayesian optimization is to determine where next to sample from the objective function, balancing exploration and exploitation. Given the posterior mean $\mu(\mathbf{x})$ and standard deviation $\sigma(\mathbf{x})$ computed by GP regression using $\mathcal{D}_{1:t}$, the maximization of an acquisition function $a(\mathbf{x}|\mathcal{D})$ yields the selection of the next point at which to evaluate the objective function:

$$\mathbf{x}_{t+1} = \arg\max_{\mathbf{x}} a(\mathbf{x}|\mathcal{D}_{1:t}). \tag{6}$$

The Upper Confidence Bound (UCB) in the case of GP regression, which is referred to as GP-UCB, is defined as

$$a(\mathbf{x}|\mathcal{D}_{1:t}) = -\mu(\mathbf{x}) + \kappa\sigma(\mathbf{x}), \tag{7}$$

where κ is a hyperparameter that controls the tightness of the confidence bounds [2]. Bayesian optimization is summarized in Algorithm 1.

Algorithm 1 Bayesian Optimization with GP regression

Require: Initial data $\mathcal{D}_{1:I} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_I, y_I)\}$, and $T \in \mathbb{N} > 0$

1: **for** $t = 1, 2, \ldots, T$ **do**
2: Find $\mathbf{x}_{I+t}$ that maximizes the acquisition function over the current GP: $\mathbf{x}_{I+t} = \arg\max_{\mathbf{x}} a(\mathbf{x}|\mathcal{D}_{1:I+t-1})$.
3: Sample the objective function: $y_{I+t} = f(\mathbf{x}_{I+t}) + \epsilon_{I+t}$.
4: Augment the data: $\mathcal{D}_{1:I+t} = \{\mathcal{D}_{1:I+t-1}, (\mathbf{x}_{I+t}, y_{I+t})\}$.
5: Update the GP, computing $\mu_{I+t}(\mathbf{x}), \sigma^2_{I+t}(\mathbf{x})$:
6: **end for**
7: **return** $\mathbf{x}^* = \arg\max_{\mathbf{x}\in\{\mathbf{x}_1,\ldots,\mathbf{x}_{I+T}\}} \mu_{I+T}(\mathbf{x})$

Bayesian optimization has been used in various problems, including hyperparameter optimization [3, 4], networks architecture optimization [5], molecule-surface interaction optimization [6], and biological structure recombination optimization [7]. As shown in Algorithm 1, Bayesian optimization is constituted by two ingredients: (i) estimating a surrogate function via GP regression; (ii) determining where next to sample from the objective function via the maximization of an acquisition function (e.g., GP-UCB in this paper). Hyperparameters appearing in GP regression as well as in the acquisition function should be carefully tuned, which is done by an auxiliary optimization [8]. However, auxiliary optimization requires expensive computation, which is often problematic while its convergence is proved [2, 9, 10]. Methods that bypass the auxiliary optimization are also available [11–13].

In this paper we present a simple practical method which improves GP-UCB, especially in cases where the objective function is not smooth with sharp peaks and valleys. Our method allows for the model to better explore regions far from the current location in determining where next to sample from the objective function. Detailed description of the method is given in the next section.

2. CLUSTERING-GUIDED GP-UCB

In this section we present our main contribution, the clustering-guided GP-UCB method, in detail. We begin with a novel geometric view of GP-UCB on which we base our clustering-guided GP-UCB.

2.1. Geometric View of GP-UCB

GP-UCB [2] uses the following acquisition function constructed by the current GP inferred on $\mathcal{D}_{1:t}$

$$a(\mathbf{x}|\mathcal{D}_{1:t}) = -\mu(\mathbf{x}) + \kappa\sigma(\mathbf{x}), \tag{8}$$

where μ is the posterior mean, σ is the posterior standard deviation, and κ is the trade-off hyperparameter with its value being increased gradually as iterations proceed. It determines the next query point by

$$\mathbf{x}_{t+1} = \arg\max_{\mathbf{x}} a(\mathbf{x}|\mathcal{D}_{1:t}),$$

which is expected to have small mean and large variance. We write the acquisition function as a mapping from $\mu(\mathbf{x})$ to $\sigma(\mathbf{x})$:

$$\sigma(\mathbf{x}) = \frac{1}{\kappa}\Big(\mu(\mathbf{x}) + a(\mathbf{x}|\mathcal{D}_{1:t})\Big),$$

where $a(\mathbf{x}|\mathcal{D}_{1:t})/\kappa$ is interpreted as a σ-axis intercept in the μ-σ space. Fig. 1 illustrates that the maximum of the acquisition function corresponds to a σ-axis intercept whose value is the largest.

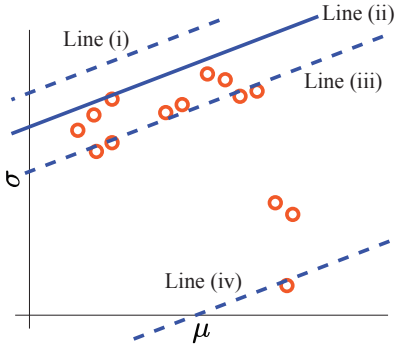


Fig. 1. The horizontal axis is the posterior mean μ and the vertical axis is the posterior standard deviation σ. Small circles (with orange color) represent points associated with two-dimensional vectors whose entries are sampled from $\mu(\mathbf{x})$ and $\sigma(\mathbf{x})$. Lines (i)-(iv), described by (9), are cases with different y-axis intercepts. The point that meets the line (ii) gives the maximal acquisition value.

2.2. Query Point Selection

A query point is determined by searching where the acquisition function is maximized. As shown in Fig. 1, nearby points in the μ-σ space exhibit similar characteristics in the perspective of posterior mean and variance. This suggests to group these points into a few coherent clusters, in order to reduce the search space over which the GP-UCB acquisition function is maximized. In GP-UCB, the search space consists of whole possible candidates to be considered. In contrast, our clustering-guided GP-UCB (CG-GPUCB) reduces the search space to a set of candidates in a single cluster. Only centers of clusters are considered to determine which cluster is the best in terms of the values of σ-axis intercepts.

Denote by $\mathbf{c}_i \in \mathbb{R}^2$ the center of cluster i in the μ-σ space, for $i = 1, \ldots, K$, where K is the number of clusters pre-specified. The best cluster is determined by

$$i^* = \arg\max_{i=1,\ldots,K} \left[-c_{i,1} + \kappa c_{i,2}\right], \tag{9}$$

where $c_{i,j}$ represents entry j in $\mathbf{c}_i$, for $j = 1, 2$. Denote by $\mathcal{C}_i$ cluster i, then the best cluster corresponds to $\mathcal{C}_{i^*}$. Depending on how the final query point is selected in the best cluster, we present two methods:

- CG-GPUCB-NN where the next query point is chosen as the nearest point to the center of the best cluster;
- CG-GPUCB^2 where we consider acquisition values at only points in the best cluster to finally determine the next query point.

That is, in CG-GPUCB-NN, the next query point is calculated as

$$\mathbf{x}_{t+1} = \arg\min_{\mathbf{x}\in\mathcal{C}_{i^*}} \left\| [\mu(\mathbf{x}), \sigma(\mathbf{x})]^\top - \mathbf{c}_{i^*} \right\|_2^2, \tag{10}$$

where $\|\cdot\|_2$ is the Euclidean norm. In CG-GPUCB^2, the next query point is calculated as

$$\mathbf{x}_{t+1} = \arg\max_{\mathbf{x}\in\mathcal{C}_{i^*}} a(\mathbf{x}|\mathcal{D}_{1:t}), \tag{11}$$

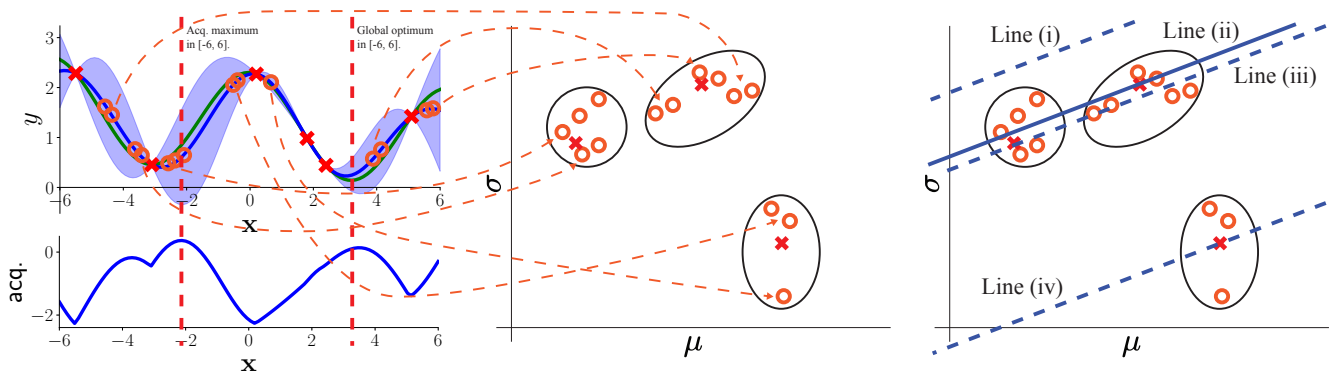


Fig. 2. Left panel: The upper plot shows the mean function (blue solid line) with the confidence interval (shaded region) measured by the variance in the current GP. The entire search space is [-6, 6], and the true function (green solid line) has the global optimum at 3.14. Exemplary candidate points are marked with small circles (orange circles). These candidate points are mapped to corresponding locations in the μ-σ space (the middle panel). The lower plot shows the value of GP-UCB acquisition function over [-6, 6]. Middle panel: Three clusters are shown in the μ-σ space, where each center is x-marked. Right panel: Lines are moving from the line (i) to the line (iv) to see which center (among three centers) first meets the line, giving the largest value of intercept. The cluster whose center first meets the line is chosen as the best cluster. The next query point is searched over candidates in the best cluster.

where the search space is restricted to $\mathcal{C}_{i^*}$. The CG-GPUCB algorithm is summarized in Algorithm 2. An example is illustrated in Fig. 2.

Algorithm 2 Bayesian Optimization with CG-GPUCB

Require: Initial data $\mathcal{D}_{1:I} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_I, y_I)\}$, $T \in \mathbb{N} > 0$, and $K \in \mathbb{N} > 0$ (number of clusters)

1: **for** $t = 1, 2, \ldots, T$ **do**
2: Calculate centers $\mathbf{c}_i$ of K clusters determined in the μ-σ space, given the current GP.
3: Find the best cluster $\mathcal{C}_{i^*}$ via (9).
4: Find $\mathbf{x}_{I+t}$ by (10) or (11).
5: Sample the objective function: $y_{I+t} = f(\mathbf{x}_{I+t}) + \epsilon_{I+t}$.
6: Augment the data: $\mathcal{D}_{1:I+t} = \{\mathcal{D}_{1:I+t-1}, (\mathbf{x}_{I+t}, y_{I+t})\}$.
7: Update the GP via (4) & (5).
8: **end for**

2.3. Hyperparameter Setting

CG-GPUCB-NN and CG-GPUCB^2 show the convergence performance with various hyperparameter settings, as shown in Fig. 3(a)-3(d). Fig. 3(a)-3(b) represent both CG-GPUCB-NN and CG-GPUCB^2 have the best result where scaling-down hyperparameter is 10.0 for the synthetic function, (12). Our methods have another hyperparameter, the number of clusters for a mixture of Gaussians. We present how the cluster number affects the convergence performance. As shown in Fig. 3(c)-3(d), the convergence of CG-GPUCB-NN and CG-GPUCB^2 is varied with respect to the number of clusters. Intuitively, if the cluster number is increased, the minimum function value results of CG-GPUCBs will be converged to the result of GP-UCB, which can be understood that the number of cluster is data size. Thus, the best configuration for a cluster number can be found. As in Fig. 3(c)-3(d), CG-GPUCB-NN has the best result at which 5 clusters are grouped, but CG-GPUCB^2 has the best result at 10 clusters case for the synthetic function.

3. EXPERIMENTS

We applied GP-UCB, CG-GPUCB-NN, and CG-GPUCB^2 in a synthetic function, and Probability of Improvement (PI) [14], PI MCMC, Expected Improvement (EI) [15], EI MCMC, GP-UCB, CG-GPUCB-NN, and CG-GPUCB^2 in global optimization benchmarks and real-world problems. The versions of MCMC, PI MCMC and EI MCMC marginalize out their hyperparameters for GP regression [5]. The details of PI and EI are described in [16]. The scaling-down hyperparameters of GP-UCB and CG-GPUCBs for every experiment were searched with the best effort. The mixture of Gaussians that has three clusters was used. We implemented our methods based on GPyOpt library [17].

3.1. Synthetic Function

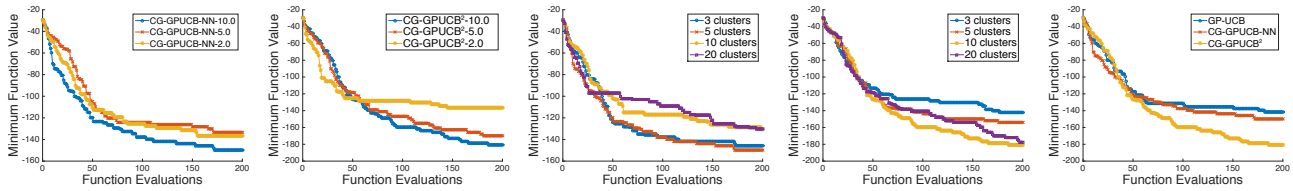
The synthetic function is

$$y = \begin{cases} -100 & \text{if } 35.0 < x < 35.5 \\ -200 & \text{if } 45.0 < x < 45.5 \\ 50\sin(\frac{8\pi x}{50})\sin(\frac{3x}{100}) & \text{otherwise.} \end{cases} \quad (12)$$

This function whose range is $[0, 100]$ has a global optimum, -200 at the second condition. If only the third condition of (12) existed, then the variations of CG-GPUCB performed slightly better than GP-UCB. However, if two conditions that make a synthetic function discrete were added, CG-GPUCBs outperformed rather than GP-UCB, as shown in Fig. 3(e).

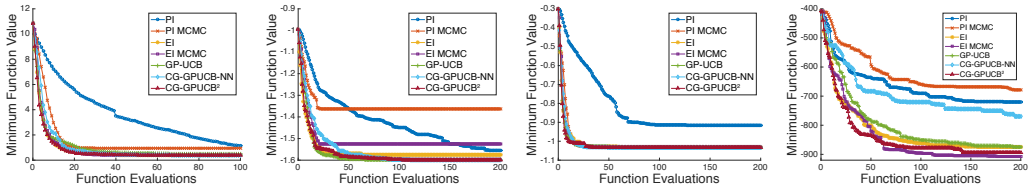
3.2. Global Optimization Benchmarks

The well-known global optimization benchmark functions: Branin-Hoo, Cosines, Sixhumpcamel, and Eggholder functions were tested as shown in Fig. 4(a)-4(d). For Branin-Hoo function, all acquisition functions except PI and EI showed that minimum function values were converged to almost the global optimum for 200 evaluations. For Cosines function, EI, GP-UCB, and CG-GPUCB^2 found the global optimum for less than 100 iterations. For Sixhumpcamel



(a) Effect of scaling-down hyp. for CG-GPUCB-NN (b) Effect of scaling-down hyp. for CG-GPUCB^2 (c) Effect of the number of clusters for CG-GPUCB-NN (d) Effect of the number of clusters for CG-GPUCB^2 (e) Comparison for synthetic function

Fig. 3. Function evaluation results with respect to the hyperparameters, scaling-down hyperparameter for κ and the number of clusters (Fig. 3(a)-3(d)). Scaling-down hyperparameter for κ was set to 2.0, 5.0, and 10.0 where the number of clusters was fixed to 5 for CG-GPUCB-NN and 10 for CG-GPUCB^2. The number of clusters was also set to 3, 5, 10, and 20 where the scaling-down hyperparameter was fixed to 10.0. Function evaluation results of GP-UCB and CG-GPUCBs (Fig. 3(e)). All hyperparameters were found with the best effort for Fig. 3(e).

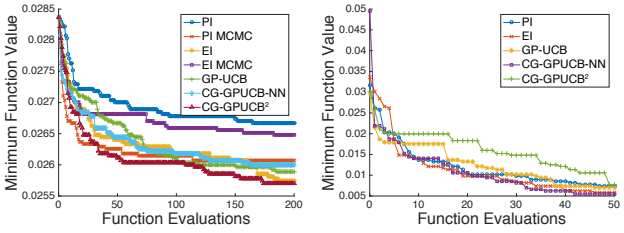


(a) Branin-Hoo function (b) Cosines function (c) Sixhumpcamel function (d) Eggholder function

Fig. 4. Seven acquisition functions: PI, PI MCMC, EI, EI MCMC, GP-UCB, CG-GPUCB-NN, and CG-GPUCB^2 found global optima of four benchmark functions: Branin-Hoo, Cosines, Sixhumpcamel, and Eggholder functions for 50 rounds of the experiments. Standard deviation is not depicted for the clarity of the figures.

function, most acquisition functions, except PI found the global optimum with similar performance. For Eggholder function, EI MCMC and CG-GPUCB^2 converged to almost -900 for 200 function evaluations.

3.3. Real-world Problems



(a) Logistic regression (b) Deep convolutional networks

Fig. 5. Two real-world problems: logistic regression and deep convolutional networks were optimized with our methods and the existing methods.

3.3.1. Logistic Regression

Logistic regression, implemented in the Scikit-learn library [18] was optimized by Bayesian optimization with various acquisition functions for 50 times. The logistic regression model was trained and tested on 8×8 digit images from UCI Repository. Three-dimensional hyperparameter space composed of regularization, bias, and stopping tolerance is optimized. EI and CG-GPUCB^2 showed almost similar convergence performance, but CG-GPUCB^2 was slightly better.

3.3.2. Deep Convolutional Networks

Deep convolutional networks, based on TensorFlow library [19] return an error rate as an actual response. The networks were learned and inferred by MNIST dataset. The domain space is three-dimensional space composed of learning rate, batch size, and dropout rate. Deep convolutional networks were iterated 10 times. PI MCMC and EI MCMC were excluded, because they took too much time to be employed in this problem. A y-axis stands for error rate of those models. In addition, since the different deep networks are trained even if the same hyperparameter setting is given, the error rate of deep convolutional networks at 0 evaluation can differ as shown in Fig. 5(b). EI and CG-GPUCB-NN converged to near error rate 0.005% during 50 evaluations.

4. CONCLUSION

We have presented a clustering-guided extension of GP-UCB, where the maximum of GP-UCB acquisition function is searched over candidates in a single cluster chosen by our method while the ordinary GP-UCB requires the search over candidates in the entire space considered in the problem. We have based our development on a novel geometric view of GP-UCB which could be treated as another contribution, in addition to the CG-GPUCB method itself. Depending on how the final query point is selected in the best cluster, we have presented two methods: (1) CG-GPUCB-NN where the next query point is chosen as the nearest point to the center of the best cluster; (2) CG-GPUCB^2 where we consider acquisition values at only points in the best cluster to finally determine the next query point.

5. REFERENCES

[1] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*, MIT Press, 2006.

[2] N. Srinivas, A. Krause, S. Kakade, and M. Seeger, "Gaussian process optimization in the bandit setting: No regret and experimental design," in *Proceedings of the International Conference on Machine Learning (ICML)*, Haifa, Israel, 2010, pp. 1015–1022.

[3] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," in *Advances in Neural Information Processing Systems (NIPS)*, Granada, Spain, 2011, vol. 24, pp. 2546–2554.

[4] F. Hutter, H. H. Holger, and K. Leyton-Brown, "Sequential model-based optimization for general algorithm configuration," in *Proceedigns of the International Conference on Learning and Intelligent Optimization*, Rome, Italy, 2011, pp. 507–523.

[5] J. Snoek, H. Larochelle, and R. P. Adams, "Practical Bayesian optimization of machine learning algorithms," in *Advances in Neural Information Processing Systems (NIPS)*, Lake Tahoe, NV, USA, 2012, vol. 25, pp. 2951–2959.

[6] S. Carr, R. Garnett, and C. Lo, "BASC: Applying Bayesian optimization to the search for global minima on potential energy surfaces," in *Proceedings of the International Conference on Machine Learning (ICML)*, New York, NY, USA, 2016, pp. 898–907.

[7] M. Luna and E. Martínez, "A Bayesian approach to run-to-run optimization of animal cell bioreactors using probabilistic tendency models," *Industrial & Engineering Chemistry Research*, vol. 53, no. 44, pp. 17252–17266, 2014.

[8] D. R. Jones, M. Schonlau, and W. J. Welch, "Efficient global optimization of expensive black-box functions," *Journal of Global Optimization*, vol. 13, no. 4, pp. 455–492, 1998.

[9] E. Vazquez and J. Bect, "Convergence properties of the expected improvement algorithm with fixed mean and covariance functions," *Journal of Statistical Planning and Inference*, vol. 140, no. 11, pp. 3088–3095, 2010.

[10] A. D. Bull, "Convergence rates of efficient global optimization algorithms," *Journal of Machine Learning Research*, vol. 12, pp. 2879–2904, 2011.

[11] N. de Freitas, M. Zoghi, and A. J. Smola, "Exponential regret bounds for gaussian process bandits with deterministic observations," in *Proceedings of the International Conference on Machine Learning (ICML)*, Edinburgh, Scotland, UK, 2012, pp. 1743–1750.

[12] Z. Wang, B. Shakibi, L. Jin, and N. de Freitas, "Bayesian multi-scale optimistic optimization," in *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, Reykjavik, Iceland, 2014, pp. 1005–1014.

[13] K. Kawaguchi, L. P. Kaelbling, and T. Lozano-Pérez, "Bayesian optimization with exponential convergence," in *Advances in Neural Information Processing Systems (NIPS)*, Montreal, Canada, 2015, vol. 28, pp. 2791–2799.

[14] 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. 97–106, 1964.

[15] J. Mockus, V. Tiesis, and A. Zilinskas, "The application of Bayesian methods for seeking the extremum," *Towards Global Optimization*, vol. 2, pp. 117–129, 1978.

[16] E. Brochu, V. M. Cora, and N. de Freitas, "A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierachical reinforcement learning," 2010, arXiv preprint arXiv:1012.2599.

[17] The GPyOpt authors, "GPyOpt: A Bayesian optimization framework in python," `https://github.com/SheffieldML/GPyOpt`, 2016.

[18] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al., "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[19] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, et al., "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," 2015, arXiv preprint arXiv:1603.04467.