Are All Training Data Useful? A Empirical Revisit of Subset Selection in Bayesian Optimization

Peili Mao
University of Electronic Science and Technology of China
Chengdu, 611731, China
peili.z.mao@gmail.com

Ke Li
Department of Computer Science
University of Exeter
Exeter, EX4 4QF, UK
k.li@exeter.ac.uk

Abstract

Bayesian optimization (BO) has been widely recognized as a powerful approach for black-box optimization problems with expensive objective function(s). Gaussian process (GP), which has been widely used for surrogate modeling in BO, is notorious for its cubic computational complexity grows with the increase of the amount of evaluated samples. This can lead to a significantly increased computational time for BO due to its sequential decision-making nature. This paper revisit the simple and effective subset selection methods to pick up a small group of representative data from the entire dataset to carry out the training and inference of GP in the context of BO. Empirical studies demonstrate that subset selection methods not only promote the performance of the vanilla BO but also significantly reduce the computational time for up to $\approx 98\%$.

1 Introduction

The black-box optimization problem considered in this paper is defined as:

$$\underset{\mathbf{x}\in\Omega}{\text{minimize}}\ f(\mathbf{x}), \tag{1}$$

where $\mathbf{x} = (x_1, \cdots, x_n)^\top$ is a decision vector (variable), and $\Omega = [x_i^L, x_i^U]_{i=1}^n \subset \mathbb{R}^n$ is the search space. $f : \Omega \to \mathbb{R}$ is the corresponding attainable set in the objective space. Bayesian optimization (BO) has been widely recognized as a powerful approach for such problem Shahriari et al. [2016]. Given its ability to estimate the uncertainty w.r.t. a prediction, Gaussian process (GP) Rasmussen and Williams [2006] is one of the most popular choices for surrogate modeling in BO. A main limitation of a GP model is its $\mathcal{O}(N^3)$ computational complexity for inverting the training data covariance matrix, where N is the number of training instances. Given the sequential decision-making nature of BO, the computational limit of a GP model can make BO become gradually computationally stuck or even intractable with the increase of evaluated samples. However, since the cost of surrogate modeling is assumed to be negligible compared against that of the objective function evaluation, the computationally demanding issue of the GP model is largely ignored in the context of BO. Whereas there have been a wealth of studies for mitigating the computational bottleneck of GP modeling itself alone. For example, Lawrence et al. [2002], Seeger [2003], Keerthi and Chu [2005] proposed to select a subset of data to approximate the exact GP by differential entropy, information gain, and matching pursuit, respectively. The theoretical error bounds of such strategy has recently analyzed in Hayashi et al. [2020]. An alternative strategy is sparse GP approximation to accelerate the training and inference of a GP model (e.g., Williams and Seeger [2000], Candela and Rasmussen [2005], Snelson and Ghahramani [2005, 2007], Titsias [2009]). Its basic idea is to use a small set of $M \ll N$ support points, a.k.a. inducing points, as a representation of the complete dataset to constitute a sparse GP. By doing so, the time complexity is reduced to $\mathcal{O}(NM^2 + M^3)$. Another type of methods are

2022 NeurIPS Workshop on Gaussian Processes, Spatiotemporal Modeling, and Decision-making Systems.

developed to exploit the structural characteristics of the covariance matrix to implement an accurate and scalable inference (e.g., Cunningham et al. [2008]). However, since such methods are built upon a lattice structure of the training data, they are not directly applicable to the context of BO.

In this paper, we plan to revisit the computational limitation of a GP model in the context of BO. This paper studies three simple subset selection strategies to pick up some representative data from all the previously evaluated samples thus to progressively constitute a subset of training data. Our empirical study shows that the BO can be accelerated for up to 96% by using subset selection strategies. In the meanwhile, it is interesting to see that the performance of BO have been promoted in many cases by using a subset of representative data for the training and inference of a GP model. In addition, comparing to other two selected sparse GP approaches, the performance of subset selection methods is significantly better with a reduced variance.

2 Subset Selection Methods

In this paper, we empirically investigate three subset selection strategies to pick up a subset of data (denoted as $\mathcal{T}$) from the set of all the evaluated samples (denoted as $\mathcal{D}$) during the BO to serve the purpose of GP model training in the BO. Note that $|\mathcal{T}| \ll |\mathcal{D}|$. We briefly outline the basic ideas of these three strategies and their algorithmic implementations are delineated in Section 2 of the supplementary document of this paper.

- Random selection (`RS`): It simply picks up $|\mathcal{T}|$ data instances from $\mathcal{D}$ to constitute $\mathcal{T}$.
- k-means clustering selection (`KCS`): Its basic idea is to first use the k-means clustering Jain et al. [1999] procedure to divide $\mathcal{D}$ into k clusters. Then, the most representative data is selected from each cluster to constitute the truncated training dataset $\mathcal{T} \subset \mathcal{D}$.
- Seed clustering selection (`SCS`): Different from the k-means clustering, this strategy uses an experimental design method, the Latin hypercube sampling in particular, to seed k initial pivots in Ω. Clusters are generated according to the distance of the data in $\mathcal{D}$ w.r.t. each of the pivots. Then, the most representative data from each cluster are picked up to constitute $\mathcal{T}$ as done in `KCS`.

Remark 1 *It is anticipated that the amount of the representative data grows with the increase of $|\mathcal{D}|$. Instead of using a constant k, it makes more sense to dynamically increase k with the progression of the BO. In this paper, we set $k = \frac{N^{\mathrm{FE}}}{\alpha}$ where N^{FE} is the current number of function evaluations (FEs) and $\alpha > 0$ is a scaling factor.*

Remark 2 *Note that the subset selection is not always applied at each iteration of the main while loop. Instead, it is not incurred until the number of FEs approaches $30n$. Thereafter, it is re-called every $5n$ new FEs. In other words, $\mathcal{T}$ is refreshed every given number of FEs.*

3 Results and Analysis

In this section, we compare the performance of three `BO` variants by using the subset selection strategies proposed in Section B against the vanilla `BO` and other `BO` variants with GP approximation methods including the variational free energy (`VFE`) Titsias [2009] and the fully independent training conditional (`FITC`) Snelson and Ghahramani [2005]. Five synthetic benchmark test problems and hyperparameter optimization (HPO) problems for support vector machine (SVM) and feed-forward neural network (NN) are used to constitute our benchmark suite. The performance is evaluated on both the quality of the final solution and the computational efficiency achieved w.r.t. the vanilla `BO`. Each experiment is independently repeated 20 times with different random seeds. To have a statistical interpretation of the comparison results, we apply the Wilcoxon signed-rank test Wilcoxon [1945] and the Cliff's Delta effect size Hess and Kromrey [2004] as the statistical measures in our experiments. Our experimental setup is given in Section 3 of the supplementary document of this paper.

3.1 Results on Synthetic Problems

From the results shown in Table 1, it is clear to see that the three subset selection strategies are the best in most comparisons. In particular, it is surprising to note that even the `RS` achieves a

Table 1: Performance comparisons of the quality of solutions obtained by different algorithms.

	BO	RS	SCS	KCS	VFE	FITC
A4	2.120E+0(9.886-1)†	9.247E-1(3.569E-1)	**7.778E-1(3.708E-1)**	7.979E-1(3.849E-1)	2.161E+0(7.551E-1)†	2.170E+0(5.489E-1)†
A6	3.353E+0(1.379E+0)†	**1.204E+0(3.697E-1)**	2.208E+0(7.028E-1)†	2.073E+0(6.300E-1)†	2.505E+0(5.605E-1)†	2.754E+0(7.134E-1)†
A10	3.592E+0(5.894E-1)†	5.133E+0(8.983E-1) †	4.433E+0(1.070E+0)†	3.647E+0(9.356E-1)†	**2.690E+0(6.403E-1)**	4.036E+0(1.218E+0)†
S4	1.402E+2(8.681E+1)	1.220E+2(1.288E+2)	**7.370E+1(9.263E+1)**	1.483E+2(1.595E+2)	2.686E+2(1.277E+2)†	2.812E+2(1.290E+2)†
S6	3.126E+2(8.401E+1)	5.224E+2(1.831E+2)†	**2.946E+2(1.611E+2)**	2.963E+2(1.947E+2)	5.719E+2(2.127E+2)†	5.719E+2(2.127E+2)†
S10	1.415E+3(1.420E+2)†	1.378E+3(1.704E+2)	**1.270E+3(1.518E+2)**	1.402E+3(1.521E+2)†	1.631E+3(2.143E+2)†	1.628E+3(2.058E+2)†
L4	1.428E-1(6.130E-2)	1.488E-1(9.569E-2)	**1.307E-1(5.305E-2)**	1.479E-1(7.692E-2)	1.806E-1(1.206E-1)	1.595E-1(8.024E-1)†
L6	2.316E-1(8.305E-1)	3.014E-01(1.197E-1) †	2.162E-1(4.509E-1)	**1.904E-1(6.433E-2)**	3.749E-1(2.241E-1)†	2.091E-1(6.895E-2)
L10	**3.644E-1(6.999E-2)**	5.290E-1(1.546E-1)	4.259E-1(1.217E-1)	4.401E-1(1.435E-1)†	5.372E-1(2.150E-1)†	4.538E-1(1.667E-1)†
R4	**1.488E+0(1.079E+0)**	2.247E+0(9.262E-1)	1.857E+0(1.186E+0)	2.778E+0(2.332E+0)	3.723E+0(3.037E+0)†	3.277E+0(2.480E+0)†
R6	**3.603E+0(1.711E+0)**	3.761E+0(1.447E+0)	6.581E+0(4.760E+0)†	6.549E+0(5.327E+0)†	8.329E+0(6.088E+0)†	7.449E+0(4.856E+0)†
R10	3.191E+1(9.736E+0) †	**8.176E+0(2.276E+0)**	4.023E+1(9.302E+0)†	3.407E+1(8.197E+0) †	4.584E+1(1.437E+1)†	3.769E+1(9.216E+0) †
G4	3.670E-1(3.655E-1)†	3.072E-2(2.306E-2)	2.799E-1(2.530E-1)†	2.879E-1(2.702E-1)†	**2.952E-2(3.022E-2)**	3.687E-1(3.660E-1)†
G6	4.190E-1(4.079E-1)†	**4.455E-2(2.214E-2)**	3.533E-1(2.914E-1)†	3.493E-1(2.773E-1) †	3.420E-1(4.304E-1)†	4.190E-1(4.079E-1)†
G10	5.554E-1(2.186E+0)†	5.408E-1(9.343E-2) †	1.282E-1(4.104E-1) †	**9.698E-2(2.946E-1)**	5.554E-1(2.186E+0) †	5.399E-1(2.119E+0)†

The labels in the first column are the combination of the first letter of test problem and the number of variables, e.g., A4 is Ackley problem with $n = 4$.
† indicates that the best algorithm is significantly better than the other one according to the Wilcoxon signed-rank test at a 5% significance level.

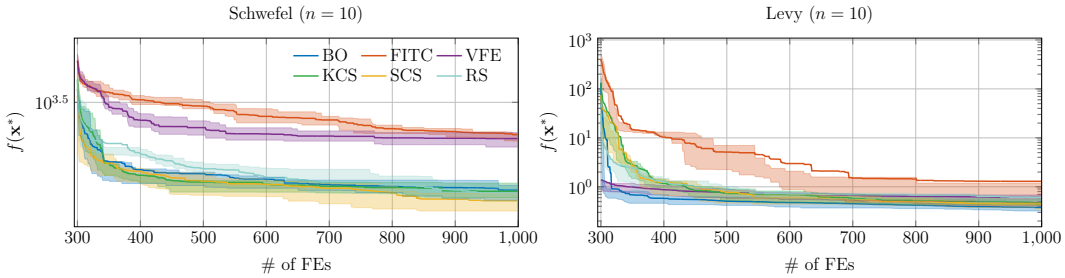


Figure 1: Plots of the convergence trajectories with confidence bounds across the optimization process obtained by different algorithms.

comparable performance with the KCS and SCS. Moreover, these BO variants with subset selection strategies are relatively more stable with a smaller variance in most cases. This observation is very inspiring as it demonstrates that the BO can achieve similar results without using all the data collected during the optimization process. This can be explained as the data collected at different stages of the optimization process might only be useful for a specific period of time. In other words, it is more recommended to strategically leverage the most relevant data to guide the GP model training and the optimization of the acquisition function.

Figure 1 plots the convergence trajectories with confidence bounds obtained by different algorithms on some selected problem instances while the full results are given in Section 4 of the supplementary document. Almost all algorithms are able to converge to the global optimum on the Levy problem while the convergence trajectory of FITC is clearly slower than the other peer algorithms. As for the Schwefel and Rastrigin problems, the convergence trajectories of both VFE and FITC are clearly worse than that of the BO by using our proposed subset selection strategies. For the Griewank problem, the performance of our proposed methods are relatively stable but they are outperformed by VFE when $n = 4$. From these results, we can see that the convergence trajectories BO are accelerated by using a representative subset of the data collected during the optimization process.

In addition to the quality of solution, we also investigate the potential reduction on the computational time. To this end, we keep a record of the average CPU wall clock time (in seconds) cost by the vanilla BO and the other BO variants. From the results shown in Table 2, it is clear to see that the computational time of BO can be significantly reduced by all peer algorithms. In particular, the percentage of CPU wall clock time reduction achieved by KCS and SCS are over 96% on all benchmark test problems while VFE and FITC are not as good and robust as our proposed subset selection strategies. It is also interesting, even surprising, to see that RS is not as efficient as the other two clustering strategies.

Last but not the least, we investigate three other settings of hyperparameter $\alpha = \{5, 10, 15\}$. We use the Cliff's Delta effect size to interpret the difference of $\alpha = 20$ against the others. From the results shown in Table 3, we can see that there is no comparison classified to be a large difference for the BO variants with the KCS and SCS. A smaller α means more representative data will be selected to constitute the truncated training dataset $\mathcal{T}$. That is to say augmenting more data for GP model training does not lead to a visible improvement to our proposed subset selection strategies. Another

Table 2: The performance comparison of the average CPU wall clock time (in seconds) with variance cost by different peer algorithms on each benchmark test problem.

	BO	RS		SCS		KCS		VFE		FITC	
	CPU time	CPU time	ROI	CPU time	ROI	CPU time	ROI	CPU time	ROI	CPU time	ROI
A10	40.72(6.31)	5.19(0.38)	87.25%	1.23(0.15)	**96.97%**	1.32(0.12)	96.76%	5.11(1.28)	87.45%	4.39(1.06)	89.22%
S10	48.77(12.12)	4.88(0.60)	90.00%	1.48(0.22)	96.96%	1.45(0.15)	**97.03%**	2.52(1.89)	94.84%	10.99(3.13)	77.47%
L10	123.92(25.94)	4.79(0.24)	96.14%	3.46(0.15)	97.20%	3.39(0.18)	**97.27%**	21.12(6.49)	82.96%	14.88(4.91)	87.99%
R10	59.30(33.97)	4.01(0.32)	93.24%	1.87(0.07)	96.85%	1.75(0.10)	**97.04%**	13.50(4.07)	77.23%	12.30(1.53)	79.25%
G10	169.33(26.60)	4.53(0.29)	97.32%	6.11(0.17)	96.39%	6.34(0.17)	96.26%	5.21(1.58)	96.92%	4.39(1.94)	**97.41%**

ROI measures the percentage of reduction of CPU wall clock time w.r.t. the vanilla BO. Its calculation method is given in Section 3.4 of the supplementary document.

Table 3: Comparison of the Cliff's Delta effect size w.r.t. different α settings in KCS and SCS.

	KCS			SCS			RS		
	$\alpha = 5$	$\alpha = 10$	$\alpha = 15$	$\alpha = 5$	$\alpha = 10$	$\alpha = 15$	$\alpha = 5$	$\alpha = 10$	$\alpha = 15$
A10	0.03	0.05	0.04	**0.175**	0.14	0.085	**0.2775**	**0.365**	**0.215**
S10	0.085	**0.2975**	**0.215**	0.05	**0.19**	0.065	**0.18**	**0.15**	**0.195**
L10	**0.24**	0.0025	0.065	**0.17**	**0.1775**	0.0575	**0.24**	**0.3625**	0.1375
R10	**0.175**	0.09	0.045	**0.225**	0.005	**0.195**	0.11	0.025	0.115
G10	0.005	0.0775	0.0025	0.075	**0.1575**	0.0825	**0.36**	**0.16**	0.0525

Table 4: The performance comparison of mean squared error (MSE) and the percentage of reduction of CPU wall clock time achieved by the BO variants w.r.t. the vanilla BO on HPO problems for SVM and NN.

	BO (SVM)	RS (SVM)		SC (SVM)		KC (SVM)		VFE (SVM)		FITC (SVM)	
	MSE	MSE	ROI	MSE	ROI	MSE	ROI	MSE	ROI	MSE	ROI
1	6.641E-3 (0)	6.641E-3 (0)	35.58%	6.641E-3 (0)	**54.87%**	6.641E-3 (0)	48.69%	6.831E-3 (3.795E-4)	33.21%	6.641E-3 (0)	-74.06%
2	1.456E-2 (0)	1.456E-2 (0)	48.38%	1.456E-2 (0)	56.17%	1.456E-2 (0)	48.38%	1.456E-2 (0)	**58.31%**	1.456E-2 (0)	-86.65%
3	3.478E-2 (0)	3.478E-2 (0)	11.90%	3.565E-2 (1.739E-3)	**32.83%**	3.478E-2 (0)	28.00%	3.478E-2 (0)	24.44%	3.478E-2 (0)	-99.43%
	BO (NN)	RS (NN)		SC (NN)		KC (NN)		VFE (NN)		FITC (NN)	
1	**7.590E-3 (6.001E-4)**	8.918E-3 (9.676E-4)	80.16%	9.677E-3 (1.840E-3)	**86.23%**	9.677E-3 (1.106E-3)	83.46%	1.480E-2 (3.823E-3)	84.64%	1.195E-2 (4.648E-4)	78.35%
2	2.816E-2 (3.633E-3)	3.010E-2 (3.633E-3)	83.98%	**2.427E-2 (5.318E-3)**	83.60%	2.718E-2 (2.378E-3)	86.63%	3.398E-2 (7.520E-3)	83.44%	2.913E-2 (3.070E-3)	**90.51%**
3	3.043E-2 (0)	3.043E-2 (0)	94.97%	2.957E-2 (1.739E-3)	91.59%	3.043E-2 (0)	88.22%	2.957E-2 (1.739E-3)	**95.35%**	**2.870E-2 (2.130E-3)**	95.08%

The dataset indices in this paper are abbreviation of the dataset IDs in the OpenML project, i.e., $1 \rightarrow \#167149, 2 \rightarrow \#167151, 3 \rightarrow \#167153$.

interesting observation is that the RS is relatively more susceptible to different α settings, except for the Rastrigin problem.

3.2 Results on Hyperparameter Optimization Problems

For HPO problems on SVM and NN, we choose three datasets selected from the OpenML project Vanschoren et al. [2013] in our experiments. The experimental setup of our HPO problems is detailed in Section 3.2 of the supplementary document. From the comparison results shown in Table 4, we find that all algorithms achieve the same performance for tuning SVM on all three datasets, but the BO variants by using our proposed subset selection strategies lead to a significant speedup where they reduce up to $\approx 57\%$ CPU wall clock time. In contrast, it is surprising to note that the BO variant using the FITC even cost much more (up to $\approx 100\%$ more) CPU wall clock time comparing to the vanilla BO when tuning SVM. As for tuning NN, we can still observe a significant speedup (more than 80%) when comparing the BO variants with the vanilla BO. In addition, the performance of BO is not compromised or even promoted by using a subset of data for GP modeling and inference as shown in Table 4 and Figure 3 in the supplementary document.

4 Conclusions

This paper empirically investigate three simple but effective subset selection strategies to pick up a subset of data from all the evaluated samples during the BO to serve the training and inference of a GP in BO. Experimental results demonstrate that the proposed subset selection strategies not only promote the performance of the baseline BO, but also significantly reduce the computational time. In addition, the performance of our proposed subset selection strategies are robust to the number of representative data picked up for constituting the training subset.

5 Acknowledgment

This work was supported by UKRI Future Leaders Fellowship (MR/S017062/1), NSFC (62076056), EPSRC (2404317), Royal Society (IES/R2/212077) and Amazon Research Award.

References

Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P. Adams, and Nando de Freitas. Taking the human out of the loop: A review of bayesian optimization. *Proc. IEEE*, 104(1):148–175, 2016.

Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian processes for machine learning*. Adaptive computation and machine learning. MIT Press, 2006. ISBN 026218253X.

Neil D. Lawrence, Matthias W. Seeger, and Ralf Herbrich. Fast sparse gaussian process methods: The informative vector machine. In Suzanna Becker, Sebastian Thrun, and Klaus Obermayer, editors, *Advances in Neural Information Processing Systems 15 [Neural Information Processing Systems, NIPS 2002, December 9-14, 2002, Vancouver, British Columbia, Canada]*, pages 609–616. MIT Press, 2002.

Matthias W. Seeger. *Bayesian Gaussian process models : PAC-Bayesian generalisation error bounds and sparse approximations*. PhD thesis, University of Edinburgh, UK, 2003.

S. Sathiya Keerthi and Wei Chu. A matching pursuit approach to sparse gaussian process regression. In *Advances in Neural Information Processing Systems 18 [Neural Information Processing Systems, NIPS 2005, December 5-8, 2005, Vancouver, British Columbia, Canada]*, pages 643–650, 2005.

Kohei Hayashi, Masaaki Imaizumi, and Yuichi Yoshida. On random subsampling of gaussian process regression: A graphon-based analysis. In Silvia Chiappa and Roberto Calandra, editors, *The 23rd International Conference on Artificial Intelligence and Statistics, AISTATS 2020, 26-28 August 2020, Online [Palermo, Sicily, Italy]*, volume 108 of *Proceedings of Machine Learning Research*, pages 2055–2065. PMLR, 2020.

Christopher K. I. Williams and Matthias W. Seeger. Using the nyström method to speed up kernel machines. In Todd K. Leen, Thomas G. Dietterich, and Volker Tresp, editors, *NIPS'00: Proc. of the 13th Advances in Neural Information Processing Systems*, pages 682–688. MIT Press, 2000.

Joaquin Quiñonero Candela and Carl Edward Rasmussen. A unifying view of sparse approximate gaussian process regression. *J. Mach. Learn. Res.*, 6:1939–1959, 2005.

Edward Lloyd Snelson and Zoubin Ghahramani. Sparse Gaussian processes using pseudo-inputs. In *NIPS'05: Proc. of the 2005 Advances in Neural Information Systems*, pages 1257–1264. MIT Press, 2005.

Edward Lloyd Snelson and Zoubin Ghahramani. Local and global sparse gaussian process approximations. In *AISTATS'07: Proc. of the 11th International Conference on Artificial Intelligence and Statistics*, volume 2 of *JMLR Proceedings*, pages 524–531. JMLR.org, 2007.

Michalis K. Titsias. Variational learning of inducing variables in sparse gaussian processes. In *AISTATS'03: Proc. of the 2009 12th International Workshop on Artificial Intelligence and Statistics*, pages 567–574. JMLR.org, 2009.

John P. Cunningham, Krishna V. Shenoy, and Maneesh Sahani. Fast gaussian process methods for point process intensity estimation. In *ICML'08: Proc. of the 25th International Conference on Machine Learning*, volume 307 of *ACM International Conference Proceeding Series*, pages 192–199. ACM, 2008.

Anil K. Jain, M. Narasimha Murty, and Patrick J. Flynn. Data clustering: A review. *ACM Comput. Surv.*, 31(3):264–323, 1999.

Frank Wilcoxon. Individual comparisons by ranking methods. 1945.

Melinda R. Hess and Jeffrey D. Kromrey. Robust confidence intervals for effect sizes: A comparative study of Cohen's d and Cliff's delta under non-normality and heterogeneous variances. In *Annual Meeting of the American Educational Research Association*, 2004.

Joaquin Vanschoren, Jan N. van Rijn, Bernd Bischl, and Luís Torgo. OpenML: networked science in machine learning. *SIGKDD Explor.*, 15(2):49–60, 2013.

A A Gentle Tutorial of Bayesian Optimization

To be self-contained, this section provides a gentle tutorial of Bayesian optimization.

Algorithm 1: Pseudo code of a vanilla BO

Input: Related hyperparameter settings.
Output: The best solution found so far.

1 Sample a set of initial solutions $\mathcal{X} \leftarrow \{\mathbf{x}^i\}_{i=1}^{N_{\mathrm{I}}}$ from Ω and evaluate their objective function values $\mathcal{Y} \leftarrow \{f(\mathbf{x}^i)\}_{i=1}^{N_{\mathrm{I}}}$. Set the training dataset $\mathcal{D} \leftarrow \{(\mathbf{x}^i, f(\mathbf{x}^i))\}_{i=1}^{N_{\mathrm{I}}}$;
2 **while** *stopping criteria is not met* **do**
3 Build a GP model based on $\mathcal{D}$;
4 Optimize an acquisition function to obtain a candidate solution $\mathbf{x}^*$;
5 Evaluate the objective function value $f(\mathbf{x}^*)$ and update $\mathcal{D} \leftarrow \mathcal{D} \bigcup \{(\mathbf{x}^*, f(\mathbf{x}^*))\}$;
6 **return** $\underset{\mathbf{x} \in \mathcal{D}}{\operatorname{argmin}} f(\mathbf{x})$

The pseudo-code of a vanilla BO is given in Algorithm 1. It starts from a space-filling experimental design (e.g., Latin hypercube sampling) **?** to obtain a set of initialized solutions. During the main while loop from line 2 to line 5, it strategically search the next point of merit in a sequential manner until the prescribed computational budget is exhausted. In a nutshell, there are two key components in a BO.

- Surrogate model: This paper considers using a GP model to serve the surrogate modeling purpose. Given a set of training data $\mathcal{D} = \left\{\left(\mathbf{x}^i, f\left(\mathbf{x}^i\right)\right)\right\}_{i=1}^{N}$, a GP model aims to learn a latent function $g(\mathbf{x})$ by assuming $f\left(\mathbf{x}^i\right) = g\left(\mathbf{x}^i\right) + \epsilon$ where $\epsilon \sim \mathcal{N}\left(0, \sigma_n^2\right)$ is an independently and identically distributed Gaussian noise. For each testing input vector $\mathbf{z}^* \in \Omega$, the mean and variance of the target $f\left(\mathbf{z}^*\right)$ are predicted as:

$$\begin{aligned} \overline{g}\left(\mathbf{z}^*\right) &= m\left(\mathbf{z}^*\right) + \mathbf{k}^{*T}\left(K + \sigma_n^2 I\right)^{-1}\left(\mathbf{f} - \mathbf{m}(X)\right) \\ \mathbb{V}\left[g\left(\mathbf{z}^*\right)\right] &= k\left(\mathbf{z}^*, \mathbf{z}^*\right) - \mathbf{k}^{*T}\left(K + \sigma_n^2 I\right)^{-1}\mathbf{k}^* \end{aligned}, \tag{2}$$

where $X = (\mathbf{x}^1, \cdots, \mathbf{x}^N)^T$ and $\mathbf{f} = (f(\mathbf{x}^1), \cdots, f(\mathbf{x}^N))^T$. $\mathbf{m}(X)$ is the mean vector of X, $\mathbf{k}^*$ is the covariance vector between X and $\mathbf{z}^*$, and K is the covariance matrix of X. In this paper, we use the radial basis function as the covariance function to measure the similarity between a pair of two solutions $\mathbf{x}$ and $\mathbf{x}' \in \Omega$:

$$k(\mathbf{x}, \mathbf{x}') = \gamma \exp(-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{\ell}), \tag{3}$$

where $\|\cdot\|$ is the Euclidean norm and γ and length scale ℓ are two hyperparameters. The predicted mean $\overline{g}(\mathbf{z}^*)$ is directly used as the prediction of $f(\mathbf{z}^*)$, and the predicted variance $\mathbb{V}[g(\mathbf{x}^*)]$ quantifies the uncertainty. In practice, the hyperparameters associated with the mean and covariance functions are learned by maximizing the log marginal likelihood function as recommended in Rasmussen and Williams [2006]:

$$\begin{aligned} \log p(\mathbf{f} \mid X) = &-\frac{1}{2}(\mathbf{f} - \mathbf{m}(X))^\top \left(K + \sigma_n^2 I\right)^{-1}\left(\mathbf{f} - \mathbf{m}(X)\right) \\ &-\frac{1}{2}\log\left|K + \sigma_n^2 I\right| - \frac{N}{2}\log 2\pi, \end{aligned} \tag{4}$$

For the sake of simplicity, here we assume that the mean function is a constant 0 and the inputs are noiseless. The computational bottleneck of training a GP model is the Cholesky decomposition of $\left(K + \sigma_n^2 I\right)^{-1}$, of which the time complexity is $\mathcal{O}\left(N^3\right)$ and the space complexity is $\mathcal{O}\left(N^2\right)$.

- Acquisition function: Instead of optimizing the surrogate objective function, the search process of BO is driven by an acquisition function which naturally strikes a balance between exploitation of the predicted minimum and exploration of the model uncertainty. In this paper, we consider the widely used expected improvement (EI) **?** for a proof-of-concept purpose:

$$\begin{aligned} \mathrm{EI}(\mathbf{x}) &= \mathbb{E}\left[\max(f(\mathbf{x}^*) - f(\mathbf{x}), 0]\right] \\ &= (f(\mathbf{x}^*) - \overline{g}(\mathbf{x}))\Phi\left(\frac{f(\mathbf{x}^*) - \overline{g}(\mathbf{x})}{\mathbb{V}[g(\mathbf{x})]}\right), \\ &+ \mathbb{V}[g(\mathbf{x})]\phi\left(\frac{f(\mathbf{x}^*) - \overline{g}(\mathbf{x})}{\mathbb{V}[g(\mathbf{x})]}\right) \end{aligned} \tag{5}$$

where $\mathbf{x}^*$ is the current best sample point and $\phi(\cdot)$ is the probability density function of the standard normal distribution.

B Proposed Methods

Algorithm 2: Pseudo code of the BO with subset selection

Input: Related hyperparameter settings.
Output: The best solution found so far.

1 $\mathcal{D} \leftarrow \emptyset$, $\mathcal{T} \leftarrow \emptyset$;
2 Sample a set of initial solutions $\mathcal{X} \leftarrow \{\mathbf{x}^i\}_{i=1}^{N_\mathrm{I}}$ from Ω and evaluate their objective function values $\mathcal{Y} \leftarrow \{f(\mathbf{x}^i)\}_{i=1}^{N_\mathrm{I}}$. Set the training dataset $\mathcal{D} \leftarrow \{(\mathbf{x}^i, f(\mathbf{x}^i))\}_{i=1}^{N_\mathrm{I}}$ and the truncated training dataset $\mathcal{T} \leftarrow \mathcal{D}$;
3 **while** *stopping criteria is not met* **do**
4 **if** *subset selection criterion is met* **then**
5 Use a subset selection strategy to pick a subset $\mathcal{T}$ from $\mathcal{D}$;
6 Build a GP model based on $\mathcal{T}$;
7 Optimize an acquisition function to obtain a candidate solution $\mathbf{x}^*$;
8 Evaluate the objective function value $f(\mathbf{x}^*)$ and set $\mathcal{D} \leftarrow \mathcal{D} \bigcup \{(\mathbf{x}^*, f(\mathbf{x}^*))$, $\mathcal{T} \leftarrow \mathcal{T} \bigcup \{(\mathbf{x}^*, f(\mathbf{x}^*))\}$;
9 **return** $\underset{\mathbf{x} \in \mathcal{D}}{\operatorname{argmin}} f(\mathbf{x})$

As shown in line 5 of Algorithm 1, the training dataset $\mathcal{D}$ grows in a sequential manner. Since the training and inference of a GP model are notoriously time consuming and they grow cubically with the size of $\mathcal{D}$, we can anticipate that the BO gradually becomes stacked with the increase of $\mathcal{D}$. To mitigate this issue, this paper proposes three subset selection strategy to pick up a subset of $\mathcal{D}$ to train the GP model. Since the random selection is straightforward, this supplementary document mainly focus on delineating the other two subset strategies.

B.1 Subset Selection Strategy based on Clustering

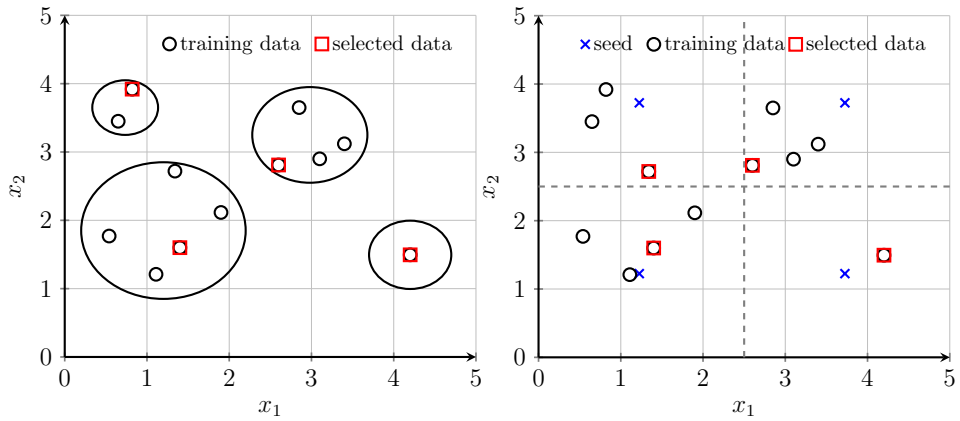
The basic idea of the first strategy is to use a clustering procedure to pick up the most representative data from each cluster to constitute the truncated training dataset $\mathcal{T} \subset \mathcal{D}$. The working mechanism is given as follows.

Step 1: Use the k-means algorithm to divide $\mathcal{D}$ into $k > 1$ clusters $\mathcal{C} = \{\mathcal{C}^i\}_{i=1}^k$ where $|\mathcal{D}| = \sum_{i=1}^k \left|\mathcal{C}^i\right|$ and $|*|$ indicates the cardinality of a set.

Step 2: For each cluster $\mathcal{C}^i$, pick up the best data $\mathbf{x}^{i*} = \underset{\mathbf{x} \in \mathcal{C}^i}{\operatorname{argmin}} f(\mathbf{x})$ where $i \in \{1, \cdots, k\}$.

Step 3: Set $\mathcal{T} \leftarrow \{\mathbf{x}^{i*}\}_{i=1}^k$.

Remark 3 *It is anticipated that the amount of the representative data grows with the increase of $|\mathcal{D}|$. In this case, instead of using a constant k, it makes more sense to dynamically increase k with the*



(a) clustering (b) space partitioning

Figure 2: Illustrative examples of the subset selection strategies based on (a) clustering and (b) space partitioning.

progression of the BO. In this paper, we set $k = \frac{N^{\mathrm{FE}}}{\alpha}$ *where* N^{FE} *is the current number of function evaluations (FEs) and* $\alpha > 0$ *is a scaling factor.*

Remark 4 *As the illustrative example shown in Figure 2(a), we can see that the four red squares are the representative data picked up by the subset selection strategy based on clustering.*

B.2 Subset Selection Strategy based on Space Partitioning

The basic idea of this strategy is to divide the space into different segments. Then, the most representative data are picked up from each of these segments to constitute the truncated training dataset $\mathcal{T} \subset \mathcal{D}$. The working mechanism is given as follows.

Step 1: Use a Latin hypercube sampling to set k seeds $\mathcal{C} = \{\mathbf{c}^i\}_{i=1}^k$. For each $\mathbf{c}^i$, we initialize a corresponding set $\mathcal{S}^i \leftarrow \emptyset$.

Step 2: For each $\mathbf{x} \in \mathcal{D}$, identify $i = \underset{\mathbf{c}^i \in \mathcal{C}}{\operatorname{argmin}}\, d(\mathbf{x}, \mathbf{c}^i)$ where $d(*,*)$ indicates the Euclidean distance, and set $\mathcal{S}^i \leftarrow \mathcal{S}^i \bigcup \{\mathbf{x}\}$.

Step 3: For each $\mathcal{S}^i$ where $i \in \{1, \cdots, k\}$, identify the best data as $\mathbf{x}^{i*} = \underset{\mathbf{x} \in \mathcal{S}^i}{\operatorname{argmin}} f(\mathbf{x})$.

Step 4: Set $\mathcal{T} \leftarrow \{\mathbf{x}^{i*}\}_{i=1}^k$.

Remark 5 *We can envisage that the space is divided into* k *segments as illustrated in Figure 2(b). By this means, the best data to each set, as the red square shown in Figure 2(b), can be regarded as a pivot of each segment.*

Remark 6 *The BO by using a subset selection strategy is given in Algorithm 2. It follows the same routine as the vanilla BO shown in Algorithm 1 except the use of a subset selection in lines 4 and 5.*

Remark 7 *During the GP model training, we use this truncated training dataset* $\mathcal{T}$ *as an alternative of* $\mathcal{D}$ *to train a GP model. Note that* $|\mathcal{T}| \ll |\mathcal{D}|$ *and both of them will be augmented with the newly evaluated point of merit.*

Remark 8 *Note that the subset selection is not always applied at each iteration of the main while loop. Instead, it is not incurred until the number of FEs approaches* $30n$*. Thereafter, it is re-called every* $5n$ *new FEs. In other words,* $\mathcal{T}$ *is refreshed every given number of FEs.*

C Experimental Settings

This section introduces the experimental setup used to validate the effectiveness of our strategies proposed in Section B. The source code of different peer algorithms are available in our Github repository[1]. They are developed based on the two prevalent GP packages, i.e., GPy **?** and GPyOpt **?**.

C.1 Benchmark Test Problems

In our experiments, we choose five test problems including Ackley, Levy, Schwefel, Rastrigin and Griewank constitute our benchmark suite. The number of variables is set as $n = \{4, 6, 10\}$. All these test problems are with many local optima and their detailed mathematical properties can be found in **?**.

C.2 Hyperparameter Optimization Problems

In addition to the synthetic test problems, we also consider the hyperparameter optimization (HPO) problems for support vector machine (SVM) and feed-forward neural network (NN). In our experiments, we choose three supervised classification datasets, the ID of which are $\{167149, 167151, 167153\}$, from the OpenML project[2] Vanschoren et al. [2013]. In practice, we follow the protocol of the HPOBench **?**, a widely recognized HPO prackage, to implement our HPO environment. In particular, according to the recommendation in the HPOBench, the hyperparameters considered in our experiments are listed in Table 5. Each experiment is independently repeated 5 times with different random seeds.

Table 5: A lookup table of the hyperparameters consider in our HPO experiments

SVM			
Hyperparameter	Type	Lower Bound	Higher Bound
c	float	-10.0	10.0
$gamma$	float	-10.0	10.0
NN			
depth	integer	1	3
width	integer	16	1024
batch size	integer	4	256
alpha	float	1.00E-08	1.0
initial learning rate	float	1.00E-05	1.0

C.3 Peer Algorithms and Parameter Settings

The effectiveness of our proposed strategies is compared with three peer algorithms including the vanilla `BO`, variational free energy (`VFE`) Titsias [2009] and fully independent training conditional (`FITC`) Snelson and Ghahramani [2005]. Note that both of `VFE` and `FITC` apply an inducing point method Candela and Rasmussen [2005] to constitute a sparse GP as an alternative of the exact GP, to mitigate the exponentially increased training time with the increase of the training data. Interested readers can refer to their original papers for their detailed working mechanisms. Some algorithmic parameters of the selected peer algorithms are listed as follows.

- The computational budget is set as $100n$ function evaluations (FEs) in total while the number of initial samples is set as $20n$. Each experiment is independently repeated 20 times with different random seeds.
- `VFE` and `FITC`: The number of inducing points is set as $5n$ and the corresponding inducing variables are initialized by a uniform sampling. The inducing point method is not used until the number of FEs approaches $30n$.

[1]The source code will be released after the acceptance of this paper.

[2]`https://www.openml.org/`

C.4 Performance Metric and Statistical Test

In addition to compare the quality of the 'optimal' solution obtained by different algorithms. We are mainly interested in the computational efficiency achieved by our proposed strategies against the `VFE` and `FITC`. To this end, we use the rate of improvement (ROI) in terms of the CPU wall clock time cost by the vanilla `BO` against the other peer algorithms as the performance metric.

$$ROI = \frac{t_{\mathrm{BO}} - t}{t_{\mathrm{BO}}} \times 100\%, \tag{6}$$

where t_{BO} is the running time of the vanilla `BO` and t is the running time of a peer method introduced in Section C.3.

To have a statistical interpretation of the difference of the comparison results, we use the following statistical measures in our experiments.

- Wilcoxon signed-rank test Wilcoxon [1945]: This is a non-parametric statistical test that makes little assumption about the underlying distribution of the data. In particular, the significance level is set to $p = 0.05$ in our experiments.
- Cliff's Delta effect size Hess and Kromrey [2004]: This is a non-parametric effect size measure that quantifies the amount of difference between two groups of stochastic samples. In practice, the difference is classified to be *negligible* when the effect size is less than 0.147; it is classified to be *small* in case the effect size is with the range of $[0.147, 0.33]$; and it is classified to be *obvious* if effect size is larger than 0.33.

D Full Experimental Results

This section gives the full comparison results of our empirical study.

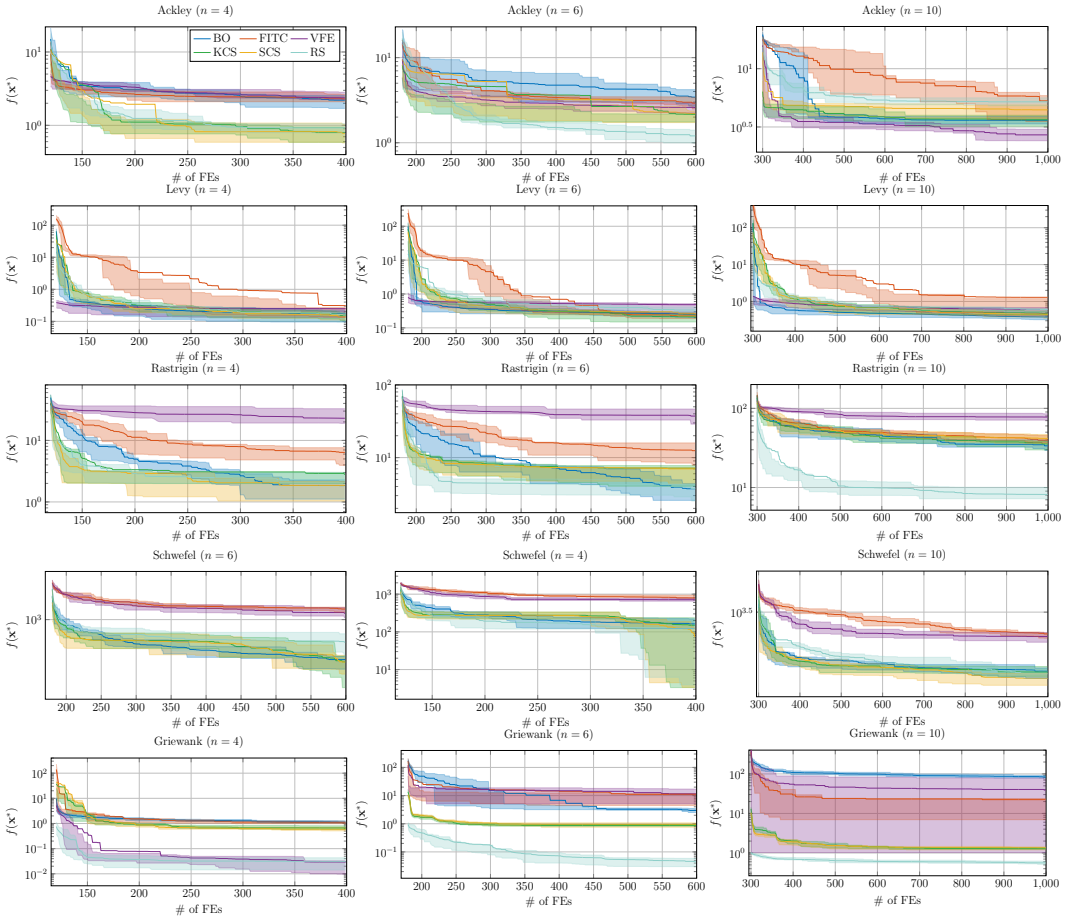


Figure 3: Plots of the convergence trajectories with confidence bounds across the optimization process obtained by different algorithms.

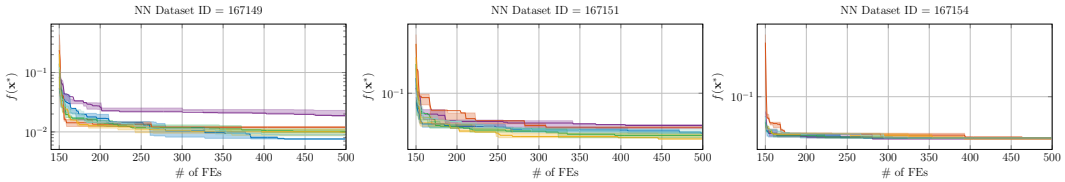


Figure 4: Plots of the convergence trajectories with confidence bounds across the optimization process obtained by different algorithms.