Improving GP Hyperparameter Fitting for Bayesian Optimization with a Goal-Oriented Criterion

Anonymous[1]

[1]Anonymous Institution

Abstract Hyperparameters of Gaussian process (GP) surrogate models are typically fit by maximizing the log marginal likelihood (LML). In the specific context of Bayesian optimization, we might desire an alternative criterion for hyperparameter optimization that preferentially yields a better surrogate model fit in regions with high objective value, at the expense of a poor fit in regions with low objective value. With this motivation in mind, we introduce the **Probability Density of Improvement (PDI) criterion function** for GP surrogate hyperparameters that naturally guides the surrogate model to attend to points with high objective function value. We show that the proposed criterion function achieves a statistically significant improvement over traditional Type-II MLE on Bayesian optimization tasks using a benchmark set of synthetic functions.

1 Introduction

The LML of a Gaussian process yields a natural and principled approach to hyperparameter selection via maximizing the LML — otherwise known as Type-II maximum likelihood estimation (MLE)[7]. However, conceptually, the LML treats all discrepancies between the data and the model with equal weight. This may not always be a desirable property.

For example, in the course of Bayesian optimization[3] we might accept a surrogate model that does a poor job explaining data with low objective values, provided that it more accurately models regions with high objective values. If the objective function has qualitatively different behavior (e.g., in terms of variability, smoothness, measurement noise, etc.), then emphasizing the ability of our surrogate to fit these "important" regions at the expense of "unimportant" ones may increase its utility for the narrow task of optimization. In the following sections, we develop intuition for how our proposed criterion function induces this behavior and demonstrate its effectiveness at Bayesian optimization on a range of synthetic benchmark tasks from [4].

1.1 Intuition

The PDI criterion function is inspired by the well-known Probability of Improvement acquisition function[5; 3]. Suppose we have an existing dataset $\mathcal{D} = \{(\vec{x}_i, y_i)\}_{i=1}^k$ of observations of the objective-function values y_i at locations $\vec{x}_i$, with a candidate threshold $y_{\text{thresh}} = y_{\text{thresh}}\left(\{y_i\}_{i=1}^k\right)$ that we take to be some function of the previously observed y-values, e.g., their maximum. The choice of hyperparameters $\vec{\theta}$ induces a belief over the amount by which a new observation $(\vec{x}_*, y_*)$ will *improve* on the given threshold. Defining this improvement as $v := \max(0, y_* - y_{\text{thresh}})$, we can write its density as a rectified normal distribution:

$$p_{\text{imp.}}\left(v \,\middle|\, \{(\vec{x}_i, y_i)\}_{i=1}^k\right) : [0, \infty) \to \mathbb{R}^+$$

$$p_{\text{imp.}}\left(v \,\middle|\, \{(\vec{x}_i, y_i)\}_{i=1}^k\right) = \begin{cases} \Phi\left(\frac{y_{\text{thresh}} - \mu_{x_*}}{\sigma_{x_*}}\right) & v = 0, \\ \mathcal{N}\left(v \,;\, \mu_{x_*} - y_{\text{thresh}},\, \sigma_{x_*}^2\right) & \text{otherwise,} \end{cases}$$

where μ_{x_*} and σ_{x_*} are the GP posterior predictive mean and standard deviation at x_*, conditioned on $\{(\vec{x}_i, y_i)\}_{i=1}^{k}$.

As a consequence of rectification, values below the threshold have the same density, leading to an implicit censoring: We do not care *how far* below the threshold value an observation lies, only *that* it lies below the threshold. In contrast, the density is non-constant above the threshold, and observations that agree with our belief about the specific *amount* of improvement will lead to a higher density.

Before articulating the details, we intuit that a surrogate model based on this density might preferentially fit points above the threshold at the cost of remaining less faithful to points below the threshold. This could lead to improved performance in the course of Bayesian optimization if the objective exhibits qualitatively different behavior close to its optima. One might expect such nonstationarity, for example, in a simulation of a physical phenomenon that is well behaved near the "correct" parameters but more pathological elsewhere.

1.2 Details

If our GP hyperparameters are well-chosen, then we expect this belief over improvement to be well-quantified. In other words, we seek to maximize the likelihood of some sequence of observed improvements $\{v_i\}_{i \in I}$ as a function of the hyperparameters. So far, we have not specified this choice of sequence. A natural choice, in the context of Bayesian optimization, is the sequence in which the points are actually acquired.

However, nothing in the motivation or definition of the PDI criterion demands that we consider only *one* choice of ordering. We could imagine averaging the sequential PDI over all possible permutations of an existing set of observations, demanding that our belief over improvement is well-quantified no matter the order of observations. Of course, this would be computationally infeasible, but the idea inspires an alternative "leave-one-out" (LOO) version of the PDI criterion:

$$\mathrm{PDI}_{\mathrm{LOO}}\left(\{(\vec{x}_i, y_i)\}_{i=1}^{n}\right) := \prod_{k=1}^{n} p_{\mathrm{imp.}}\left(\tilde{v}_k \,\middle|\, \{(\vec{x}_i, y_i)\}_{i=1}^{n} \setminus \{(\vec{x}_k, y_k)\}\right), \tag{1}$$

where $v_k := \max\left(0, y_k - y_{\mathrm{thresh}}\left(\{y_i\}_{i=1}^{n} \setminus \{(\vec{x}_k, y_k)\}\right)\right)$ is the observed improvement of the kth acquisition, as conditioned on the complementary observations — all those *except* the kth. Experimentally, we find that the sequential version of PDI is outperformed by $\mathrm{PDI}_{\mathrm{LOO}}$, and so we focus on the latter for the rest of this work.

There is one final detail to consider, which is the choice of threshold function $y_{\mathrm{thresh}}(\cdot)$. While taking $y_{\mathrm{thresh}}(\cdot) \equiv \max(\cdot)$ is appealing in its simplicity, we observe that in the case of the $\mathrm{PDI}_{\mathrm{LOO}}$ this choice would generally result in only *one* term in the product impacting the likelihood – the term k where y_k attains the maximum $\max\left(\{y_i\}_{i=1}^{n}\right)$. Empirically, this choice of threshold function leads to an ill-behaved hyperparameter optimization landscape. Choosing y_{thresh} to be some quantile q (with $q < 1$) of the existing observations alleviates this issue, but reduces the desirable censoring behavior of the criterion. We choose instead to compute $\mathrm{PDI}_{\mathrm{LOO}}$ by considering y_{thresh} over a range of quantiles, from $q = 0.75$ to $q = 1.0$, and average the resulting values. This produces a smoother and more well-behaved optimization landscape.

2 Methods

We compare our proposed method ($\mathrm{PDI}_{\mathrm{LOO}}$) against standard Type-II MLE using the log marginal likelihood (LML) across a range of synthetic benchmark tasks found in [4], with the only control variable being the criterion for hyperparameter optimization. We provide further implementation details in the appendix.

We conduct 30 independent runs of each Bayesian optimization task for each method in order to obtain a statistical picture of performance. The random seed, and hence the initial points, are

identical for paired runs of PDI_{LOO} and LML. We then compute the relative optimization gap — the fraction of progress from the best initial evaluation to the true global maximum — as a function of number of objective evaluations, allowing us to straightforwardly compare results across random seeds and across tasks.

3 Results

We present our results in table 1. We consider the final gap at exhaustion of the objective function evaluation budget. For all tasks in aggregate, and for each task individually, we report the mean final gap (± the associated standard error), and p-values of a relative t-test, paired by task and by random seed.

In some cases, we observe that one method closes the gap significantly more quickly than the other, but both converge eventually, leading to a negligibly small final gap. To account for this, we define the "hit time" of a run as the number of iterations required to reach a gap value of 99% of the mean final gap of the method that performs best on that problem. If a run reaches the function evaluation budget before this point, we right-censor the number of required iterations to the function evaluation budget.

We report the mean hit time for both methods, and the p-value of a relative t-test on their difference. For a uniform comparison, we report the mean hit time for all tasks in percentage of objective function evaluation budget, averaged across tasks. In all other rows, the mean hit time is in units of objective function evaluations. For convenience, we also report the relative speedup between the mean hit time of LML and the mean hit time of PDI_{LOO} as the fraction $\Delta = \frac{\text{mean hit time(LML)}}{\text{mean hit time}(\text{PDI}_{\text{LOO}})}$.

4 Conclusions and Discussion

We observe that our proposed method (PDI_{LOO}) statistically outperforms standard Type-II MLE (LML) on our set of benchmark tasks. LML surpasses PDI_{LOO} on 9 of 43 tasks in terms of final gap achieved, and on only 5 of 43 tasks in terms of hit time. On the other hand, PDI_{LOO} surpasses LML on 17 of 43 tasks — as well as on all tasks in aggregate — by either metric. The difference in hit times is not only statistically significant: in many cases, it represents a substantial reduction in the number of objective function evaluations required for optimization.

With respect to wall-clock time, we note that our implementation of PDI_{LOO} introduces additional computational overhead compared to LML (see figure 3). The implementation could likely be further optimized to reduce this overhead. However, in real-world Bayesian optimization tasks, the cost of evaluating the objective function typically outweighs the cost of fitting hyperparameters[3]. Therefore, we focus here on performance as a function of objective evaluations.

We note that PDI_{LOO} tends to outperform LML more reliably and to a greater degree in higher-dimensional problems (see figure 2). This may be due to the relative ease of lower-dimensional tasks, which give "less room" for either method to outperform the other. Alternatively, it may indicate that PDI_{LOO} is especially well suited to higher-dimensional problems, perhaps because they are more likely to exhibit variation in structure near the optimum versus away from it.

In any case, these preliminary results suggest that further exploration may be fruitful. In particular, the following lines of research interest us:

- Testing these alternative PDI criteria on real-world, non-synthetic objective functions.
- Identifying conditions under which PDI is likely to outperform LML, e.g., by quantifying the heterogeneity of the objective function near versus far from the optimum.
- Considering generalizations of the idea to other active learning goals besides optimization, e.g., Bayesian quadrature: Can we develop an alternative hyperparameter criterion function that outperforms LML in a quadrature setting?

	Mean Final Gap			Mean Hit Time				
Task, Dimension	PDI_{LOO}	LML	P-value	PDI_{LOO}	LML	P-value	Δ	# Trials
All Tasks, —	**0.85**	0.79	**0.000**	**52.3%**	62.5%	**0.000**	119%	1259
Gramacy and Lee (2012), 1	0.81	**0.95**	**0.979**	**22.9**	29.3	**0.018**	128%	30
Beale, 2	0.75	**0.90**	**0.998**	46.4	39.3	0.834	85%	30
Bohachevsky, 2	1.00	1.00	0.058	13.6	16.3	0.197	120%	30
Branin, 2	0.94	**1.00**	**0.982**	40.2	**24.3**	**0.998**	61%	30
Bukin 6, 2	0.90	0.85	0.071	**37.2**	56.7	**0.002**	152%	30
De Jong 5, 2	0.82	0.85	0.738	42.9	45.3	0.378	106%	30
Drop-Wave, 2	**0.70**	0.57	**0.050**	54.1	63.8	0.098	118%	30
Eggholder, 2	0.66	**0.83**	**1.000**	56.5	**47.3**	**0.957**	84%	30
Goldstein-Price, 2	0.90	0.91	0.568	33.8	28.8	0.760	85%	30
Holder Table, 2	0.89	**0.96**	**0.998**	52.9	44.5	0.912	84%	30
Kim1, 2	0.87	0.88	0.567	49.1	52.3	0.327	107%	30
Kim2, 2	0.82	0.84	0.816	52.9	48.1	0.794	91%	30
Kim3, 2	0.85	0.87	0.731	40.6	45.3	0.254	112%	30
Michalewicz, 2	**0.96**	0.82	**0.007**	**27.7**	53.9	**0.000**	195%	30
Rosenbrock, 2	0.91	**0.98**	**0.977**	44.6	**31.7**	**0.996**	71%	30
Shubert, 2	**0.40**	0.26	**0.016**	**56.3**	70.3	**0.028**	125%	30
Six-Hump Camel, 2	0.85	**0.98**	**0.999**	65.7	**44.4**	**1.000**	68%	30
Three-Hump Camel, 2	0.85	0.92	0.855	39.1	34.7	0.738	89%	30
Hartmann 3D, 3	0.98	0.99	0.742	33.5	25.9	0.907	77%	30
Ackley, 4	**0.80**	0.50	**0.000**	**104.6**	141.5	**0.000**	135%	30
Colville, 4	0.96	**0.98**	**0.966**	75.2	66.5	0.751	88%	30
Cosines, 4	0.90	0.89	0.298	85.2	89.1	0.387	105%	30
Griewank, 4	0.98	0.98	0.609	62.7	71.5	0.166	114%	30
Levy, 4	0.94	0.96	0.831	96.1	82.0	0.893	85%	30
Rastrigin, 4	**0.81**	0.54	**0.000**	**105.4**	148.8	**0.001**	141%	30
Rosenbrock, 4	0.93	**0.98**	**0.999**	115.3	**82.5**	**0.986**	72%	30
Zakharov, 4	**0.79**	0.71	**0.032**	68.7	85.6	0.120	125%	30
Hartmann 6D, 6	**0.97**	0.76	**0.000**	**131.7**	215.6	**0.000**	164%	30
Ackley, 8	**0.62**	0.16	**0.000**	**246.3**	321.0	**0.000**	130%	30
Cosines, 8	0.88	0.88	0.945	145.6	130.8	0.742	90%	29
Griewank, 8	**0.98**	0.88	**0.000**	**122.2**	288.8	**0.000**	236%	30
Levy, 8	**0.94**	0.83	**0.001**	**180.5**	263.0	**0.002**	146%	30
Rastrigin, 8	**0.66**	0.30	**0.000**	**221.5**	312.6	**0.000**	141%	30
Rosenbrock, 8	**0.95**	0.78	**0.000**	**114.0**	278.7	**0.000**	244%	30
Zakharov, 8	0.89	0.87	0.117	63.6	62.2	0.737	98%	30
Ackley, 16	**0.48**	0.11	**0.000**	**460.2**	641.0	**0.000**	139%	30
Cosines, 16	0.88	0.88	0.703	179.0	205.8	0.230	115%	30
Griewank, 16	**0.93**	0.82	**0.001**	**284.1**	528.1	**0.000**	186%	30
Levy, 16	**0.91**	0.80	**0.000**	**347.1**	578.1	**0.000**	167%	30
Rastrigin, 16	**0.57**	0.27	**0.000**	**459.7**	641.0	**0.000**	139%	30
Rosenbrock, 16	**0.94**	0.75	**0.000**	**341.0**	609.4	**0.000**	179%	30
Zakharov, 16	1.00	1.00	0.755	34.9	56.1	0.281	161%	30

Table 1: Benchmark statistics for problems in [4]. Highlighted cells indicate that one method outperforms the alternative to a statistically significant degree: $p <= 0.05$ where PDI_{LOO} outperforms LML, highlighted in **bold blue**, or $p >= 0.95$ where LML outperforms PDI_{LOO}, highlighted in **bold red**.

References

[1] Balandat, M., Karrer, B., Jiang, D., Daulton, S., Letham, B., Wilson, A. G., and Bakshy, E. BoTorch: A framework for efficient monte-carlo bayesian optimization. In *Advances in Neural Information Processing Systems* (2020), H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., vol. 33, Curran Associates, Inc., pp. 21524–21538.

[2] Gardner, J., Pleiss, G., Weinberger, K. Q., Bindel, D., and Wilson, A. G. GPyTorch: Blackbox matrix-matrix gaussian process inference with GPU acceleration. In *Advances in Neural Information Processing Systems* (2018), S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., vol. 31, Curran Associates, Inc.

[3] Garnett, R. *Bayesian Optimization*, 1 ed. Cambridge University Press, Jan. 2023.

[4] Kim, J. BayesO Benchmarks: Benchmark Functions for Bayesian Optimization. Zenodo, Mar. 2024.

[5] Kushner, H. J. A New Method of Locating the Maximum Point of an Arbitrary Multipeak Curve in the Presence of Noise. *Journal of Basic Engineering 86*, 1 (Mar. 1964), 97–106.

[6] Močkus, J. On bayesian methods for seeking the extremum. In *Optimization Techniques IFIP Technical Conference Novosibirsk, July 1–7, 1974*, G. Goos, J. Hartmanis, P. Brinch Hansen, D. Gries, C. Moler, G. Seegmüller, N. Wirth, and G. I. Marchuk, Eds., vol. 27. Springer Berlin Heidelberg, Berlin, Heidelberg, 1975, pp. 400–404.

[7] Rasmussen, C. E., and Williams, C. K. I. *Gaussian Processes for Machine Learning*, 3. print ed. Adaptive Computation and Machine Learning. MIT Press, Cambridge, Mass., 2008.

[8] Sobol', I. On the distribution of points in a cube and the approximate evaluation of integrals. *USSR Computational Mathematics and Mathematical Physics 7*, 4 (Jan. 1967), 86–112.

A Appendix

A.1 Implementation Details

We conduct our experiments using the BoTorch framework for Bayesian optimization[1], with expected improvement[6; 3] as the choice of acquisition function. We also employ GPyTorch[2] to implement our alternative hyperparameter criterion.

We use the BayesO Benchmarks[4] package of synthetic optimization to provide our objective functions. We report statistics for all functions in the benchmark package except `Easom`, which was excluded due to incorrect recording of objective function values during optimization, and `Sphere`, which was excluded because the objective function ($f(x) := x^2$) is too simple to be of use for benchmarking purposes. For functions that have a variable input dimension (`Cosines`, `Griewank`, `Levy`, `Rastrigin`, `Rosenbrock`, and `Zakharov`), we create task instances with 4, 8, and 16 input dimensions. Additionally, we create a 2-dimensional task for the classic `Rosenbrock` function.

We budget $40 \cdot d$ objective function evaluations for each task, where d is the dimension of the task. The initial design for each task consists of five points sampled from a Sobol sequence[8]. After these initial five acquisitions, and after every following acquisition, we fit the hyperparameters by maximizing either PDI_{LOO} or LML. In order to efficiently evaluate PDI_{LOO}, we make use of the fact that the complete set of leave-one-out predictive distributions can be computed at once (see equation 5.12 in [7]).

For both criterion functions, we conduct hyperparameter optimization using L-BFGS with multiple starts (the current values plus four random samples from the hyperparameter priors) in order to avoid bad local minima. We further limit L-BFGS to 50 steps per optimization and a relative objective function tolerance of $1e-3$, as optimizing to high tolerance is unnecessary and becomes expensive with large numbers of observations.

Rarely, the resulting hyperparameters yield a numerically non-positive definite covariance matrix, even after adding "jitter"[7, p. 47] — in such cases, we continue with hyperparameter optimization for up to 200 more steps. In case this fails, we restart optimization from a new set of hyperparameters drawn from their corresponding priors.

A.2 Additional Figures

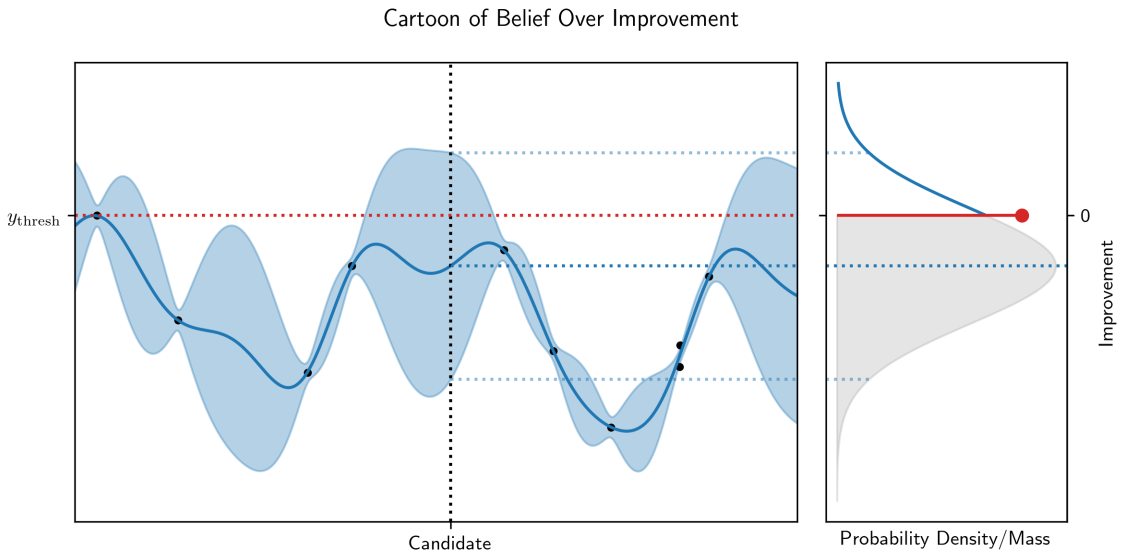

Figure 1: A one-dimensional cartoon showing a GP and its induced belief over improvement at a candidate location. Observations made at this location that result in no improvement are censored by the discrete component (in red) of the mixed discrete/continuous density.

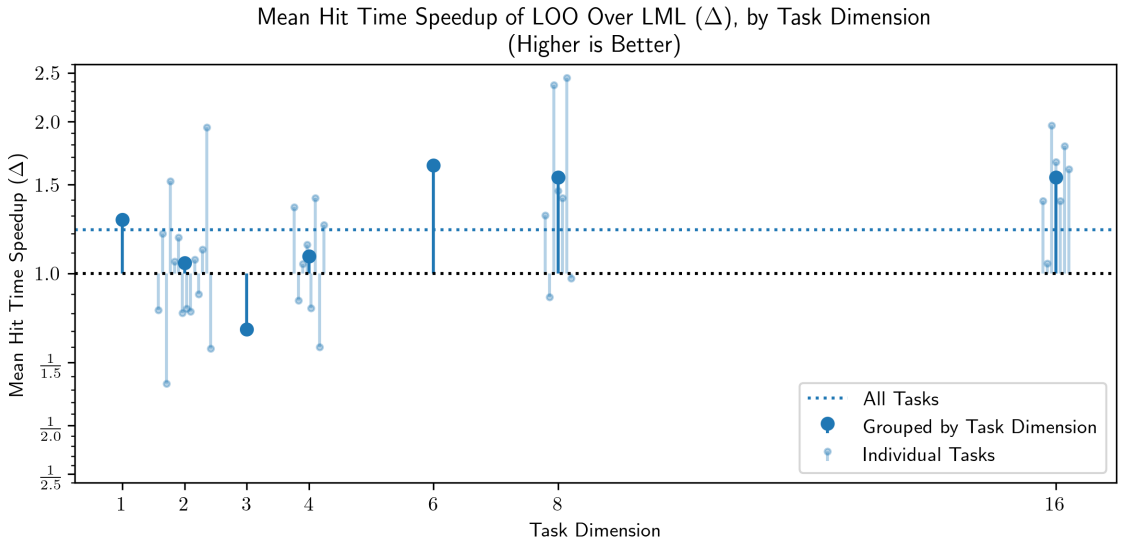

Figure 2: The speedup, Δ, in mean hit time between LML and PDI_{LOO}, plotted for individual tasks (small blue stems) and averaged across tasks with the same dimensionality (large blue stems). PDI_{LOO} tends to outperform LML more reliably on tasks with dimensionality ≥ 6.

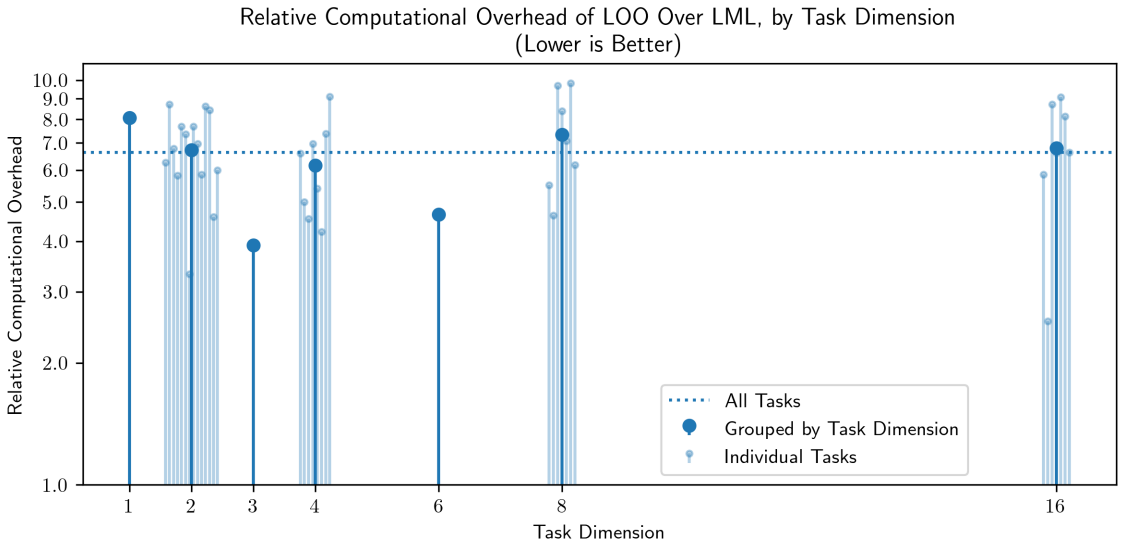

Figure 3: The relative computational overhead of PDI LOO compared to LML, plotted for individual tasks (small blue stems) and averaged across tasks with the same dimensionality (large blue stems). PDI_{LOO} is on average about 6.6 times more costly than LML, in terms of computational overhead, i.e., ignoring the cost of objective function evaluations.