Better call Surrogates: A hybrid Evolutionary Algorithm for Hyperparameter optimization

Subhodip Biswas
Computer Science, Virginia Tech
Arlington, VA 22203
subhodip@cs.vt.edu

Adam D. Cobb
US Army Research Laboratory
Adelphi, MD 20783
cobb.derek.adam@gmail.com

Andreea Sistrunk
US Army, ERDC-GRL
Alexandria, VA 22315
sistrunk@vt.edu

Naren Ramakrishnan
Computer Science, Virginia Tech
Arlington, VA 22203
naren@cs.vt.edu

Brian Jalaian
US Army Research Laboratory
Adelphi, MD 20783
brian.a.jalaian.civ@mail.mil

Abstract

In this paper, we propose a surrogate-assisted evolutionary algorithm (EA) for hyperparameter optimization of machine learning (ML) models. The proposed STEADE model initially estimates the objective function landscape using Radial Basis Function interpolation, and then transfers the knowledge to an EA technique called Differential Evolution that is used to evolve new solutions guided by a Bayesian optimization framework. We empirically evaluate our model on the hyperparameter optimization problems as a part of the black box optimization challenge at NeurIPS 2020, and demonstrate the improvement brought about by STEADE over the vanilla EA.

1 Introduction

The empirical performance of ML models can be enhanced by configuring its hyperparameters. However, manually searching for a good configuration of hyperparameters is nearly impossible due to the exponentially large size of the parameter space. This has led to an increased interest from the research community of recent in devising automated ways of searching for optimal hyperparameter configuration, popularly known as *hyperparameter optimization* (HPO) [10]. HPO is a blackbox optimization problem, where the objective function f is usually *expensive* to evaluate, and no other information like gradient, presence of special structures like linearity/concavity is available. The objective in HPO is to seek the optimum of blackbox function f, which is usually some error metric quantifying the performance of a ML model, by expending as little computational budget as possible.

EAs belong to a class of stochastic derivative-free optimization algorithms [8]. In spite of being fast, EAs employ random search moves that result in low sample efficiency, i.e., the expected gain in the objective function per unit of function evaluation (FE). Hence EAs are preferred when the FEs are cheap to compute and a high evaluation budget is available. On the other hand, surrogate-based optimization algorithms have high data efficiency as they approximate the objective function [12], but incur high cost of suggesting new points and hence they are used in solving computationally expensive optimization problems [16, 3]. Radial Basis Functions (RBFs) [4] and Gaussian Processes (GPs) [19] are popular choices when it comes to surrogate models. Motivated by this, we devise a hybrid search algorithm for adapting EAs to low budget optimization problems, like HPO, by balancing the randomness of EA search moves with strategically generated search points via informative surrogates. As detailed next, the proposed STEADE uses a mix of surrogate models (RBF and GP) and mutation mechanism (based on EA) for solving HPO problems.

Preprint. Under review.

2 The STEADE algorithm

Our algorithm Surrogate-assisTed BayEsiAn Differential Evolution (STEADE) starts with a stochastic RBF method [14, 15], which is parallel surrogate optimization algorithm, to estimate the functional landscape and then switches to an EA-based scheme guided by Bayesian Optimization (BO) [3]. The design rationale is to perform initial exploration of the parameter space using the RBF model, and then transfer the knowledge to a GP-based Bayesian optimization framework, which is further augmented by a Differential Evolution (DE) algorithm [18]. DE is a population-based real-parameter global optimization technique that has gained widespread popularity as it is fast and reliable, easy to implement, highly flexible, and robust to simple transformations [7]. Recent work has shown the advantage of information sharing in DE framework specially for solving multimodal optimization [2].

STEADE begins by instantiating an experimental design like symmetric Latin hypercube design (SLHD)[1] to generate the design points (trial solutions) and evaluate them. The RBF-based surrogate model $\mathcal{R}$ is then used to approximate the objective function [14]. After building the model, we solve an auxiliary problem by optimizing the acquisition function to generate trial solution(s) for evaluation. In an iterative manner, we evaluate the trial solution(s) and update the surrogate model till λ iterations. Then we switch to DE-based search for generating trial solution(s) till a termination criterion is met.

DE maintains a population $\mathbf{X}$ of Np randomly generated D-dimensional real-parameter vectors representing trial solutions to a problem. We can represent the i^{th} solution (also called target vector), $i \in [1, 2, \ldots, Np]$, of the population at the current generation G, $G = 0, 1, \ldots, G_{\max}$, as

$$\vec{X}_{i,G} = \left[x_{1,i,G}, x_{2,i,G}, x_{3,i,G}, \ldots, x_{D,i,G}\right].$$

Following the initialization, the solutions are improved iteratively through a series of steps−*mutation*, *crossover* or recombination, and *selection*−till a termination criterion is met. For an in-depth understanding of the DE algorithm, kindly refer to [7]. In our approach, we propose a *Bayesian mutation* mechanism to evolve new solutions (also called donor vectors) as follows.

$$\vec{V}_{i,G} = \vec{B}_{r_1^i,G} + F \cdot \left(\vec{R}_{r_2^i,G} - \vec{R}_{r_3^i,G}\right) + \frac{1}{G} \cdot rand\,(1, D) \cdot \left(\vec{X}_{r_2^i,G} - \vec{X}_{r_3^i,G}\right), \tag{1}$$

where the indices r_1^i, r_2^i, and r_3^i are mutually exclusive integers randomly chosen from the range $[1, Np]$ such that $r_1^i \neq r_2^i \neq r_3^i \neq i$, $rand\,(1, D)$ is a D-dimensional vector of uniformly distributed random numbers lying between 0 and 1, F is a scaling factor that typically lies in the interval $[0.4, 1]$, $\vec{B}$ and $\vec{R}$ are the populations of trial solution(s) generated by a BO-based model and the RBF-based method, respectively. The Bayesian mutation leverages the sample-efficiency of BO to search for solutions at promising regions of the parameter-space. The second component in Eq. (1) acts as a global search (exploratory) component while the third component gradually transitions from a global to local search as G increases (and $1/G$ shrinks accordingly).

Next, the trial vectors are generated via the crossover operation in which every donor vector exchanges its components/features with its corresponding target vector. DE can use two types of crossover− *binomial* and *exponential*. Due to its flexibility, we adopt the binomial crossover as

$$u_{j,i,G} = \begin{cases} v_{j,i,G} & \text{if } (rand_{i,j}[0,1] \leq Cr \text{ or } j == j_{rand}) \\ x_{j,i,G} & \text{otherwise} \end{cases}, \tag{2}$$

where the crossover rate Cr approximates the probability of recombination, $rand_{i,j}[0, 1]$ is an uniform random number generated for every (i, j) pair, $j_{rand} \in [1, \ldots, D]$ is a randomly chosen index to ensure that $\vec{U}_{i,G}$ has at least one component from $\vec{V}_{i,G}$. Binomial crossover ensures that the number of components inherited by the trial vector from the donor vector roughly follows a binomial distribution. Finally, a fitness-based selection mechanism takes place between the trial vector and the mutant vector to select the fitter one. The pseudocode of the STEADE is outlined in Algorithm 1.

We use the DE/rand/2/bin model where two scaled difference vectors are used to mutate the target vector (generated by BO). We set the scaling factor F and crossover rate Cr to 0.7, λ to 10 for activating the Bayesian mutation, and use population size Np of 16 as per the competition guidelines.

[1] We use SLHD as it allows an arbitrary number of design points/trial solutions and works well in practice [6]

Algorithm 1: STEADE algorithm

Input : Objective function f, Parameter space $\mathbf{\Lambda}$, Population size Np, Evaluation budget $G_{\max}$
Result: Best possible solution to f
begin
  Select an experimental design to instantiate trial solutions $\mathbf{X}_0 \in \mathbf{\Lambda}$ and evaluate them $f(\mathbf{X}_0)$
  Use $f(\mathbf{X}_0)$ to build a surrogate model $\mathcal{A}(\mathbf{X}_0)$ based on RBF
  `// Iterative improvement of the solutions till a termination criterion is met`
  for $G = 1, 2, \ldots, G_{\max}$ **do**
    if $G < \lambda$ **then**
      Generate trial solutions $\vec{U}_{i,G}$, $i \in [1, \ldots, Np]$ using the RBF-based method
    else
      Generate solutions $\vec{\mathcal{B}}$ and $\vec{\mathcal{R}}$ from the RBF-based and BO-based model respectively
      Perform Bayesian mutation (1) to generate donor vectors/solutions
      Simulate binomial crossover (2) to produce trial solutions $\vec{U}_{i,G}$
    Perform fitness-based selection to update the trial solutions as

$$\vec{X}_{i,G+1} = \begin{cases} \vec{U}_{i,G} & \text{if } f(\vec{U}_{i,G}) \leq f(\vec{X}_{i,G}) \\ \vec{X}_{i,G} & \text{otherwise} \end{cases},$$

    Use the trial solutions $\mathbf{X}_{G+1}$ to update the surrogate models.

Output the best solution $\vec{X}_* \in \mathbf{X}_{G_{\max}}$

2.1 Implementation details

Our STEADE is built on top of the Bayesmark[2] package, and uses its in-built space functionality to translate the parameter-space into a continuous search space using the idea of warping [17].

The python Surrogate Optimization Toolbox (pySOT) is a library of surrogate optimization techniques leveraging the Plumbing for Optimization with Asynchronous Parallelism (POAP) framework. For more details about these frameworks, kindly refer to [9]. Our code is based on pySOT v0.3.3 and POAP v0.1.26. We use a SLHD experimental design, a stochastic RBF surrogate with a cubic kernel and linear tail [14], and the DYCORS strategy [15] for generating trial solutions, i.e., $\vec{\mathcal{R}}$.

The other surrogate model is GP-based and is implemented using BOTORCH, a recent programming framework that includes advanced Bayesian optimization techniques [1]. Since GPs involve expensive computation, we use batch BO to reduce the computation time. In batch BO, the design points are selected in parallel by doing joint optimization over the multiple design points. To do that we use the parallel counterpart of the sequential acquisition functions (EI, PI, UCB) [20]. In our implementation, we use qEI acquisition function [11] with randomized quasi-Monte Carlo (RQMC) sampling based on scrambled Sobol sequences for sample average approximation [5, 13]. The code of STEADE is made available at `https://github.com/subhodipbiswas/BayesianEvolution`.

3 Experimentation

3.1 Methodology

The STEADE model was developed for the blackbox optimization challenge[3] at NeurIPS 2020 under the team name `Better call Bayes`. The experimental setup consists of ML hyperparameter tuning problems. The competition guidelines allowed each optimizer to run for 16 iterations using a batch size of 8 for making suggestions on each benchmark problem. There was a strict time limit of 640 s (or 40 s/iteration) which was not to be exceeded. Optimizers exceeding the time limits were cut off from making further suggestions and the best optima found till then was used. We simulated 15 runs[4] of STEADE on the benchmark resulting in 1620 independent studies. The Bayesmark library was used to compute the performance of the optimizer on different benchmark problems.

[2] Available in GitHub at https://github.com/uber/bayesmark.

[3] The starter kit and implementation details are available at github.com/rdturnermtl/bbo_challenge_starter_kit.

[4] Each run consists of testing 9 ML models on 6 datasets using 2 metric, thereby resulting in 108 studies.

Since STEADE is an composed of three components (RBF, BO and DE), we do ablation testing by selectively simulating each component under identical experimental conditions. The `visible_to_opt`, `generalization` and leaderboard[5] scores are reported in Table 1. The `visible_to_opt` is the score seen by an optimizer (e.g, in a ML hyper-parameter tuning problem this can be the cross-validation error), whereas `generalization` is a related metric the optimizer does not get to see (e.g, error on held-out test set in a ML hyper-parameter tuning context). For more details on how the mean, median and normalized mean are calculated, kindly refer to the documentation[6]. We also plot the convergence profile of the different optimizers in Figure 1 based on the `visible_to_opt` score.

Table 1: Comparative analysis of STEADE with its component algorithms.

Models	`visible_to_opt`			`generalization`			`Leaderboard`
	Median	Mean	Normalized Mean	Median	Mean	Normalized Mean	
BO	0.10906	0.03151	0.37376	2.36782	0.29934	2.58823	96.8487
DE	0.21536	0.02817	0.33408	2.5	0.29478	**2.54886**	97.1833
RBF	0.02002	0.01998	0.23693	2.62130	0.30189	2.61027	98.0024
STEADE	0.10301	0.01304	**0.15463**	2.51204	0.297115	2.56901	**98.6963**

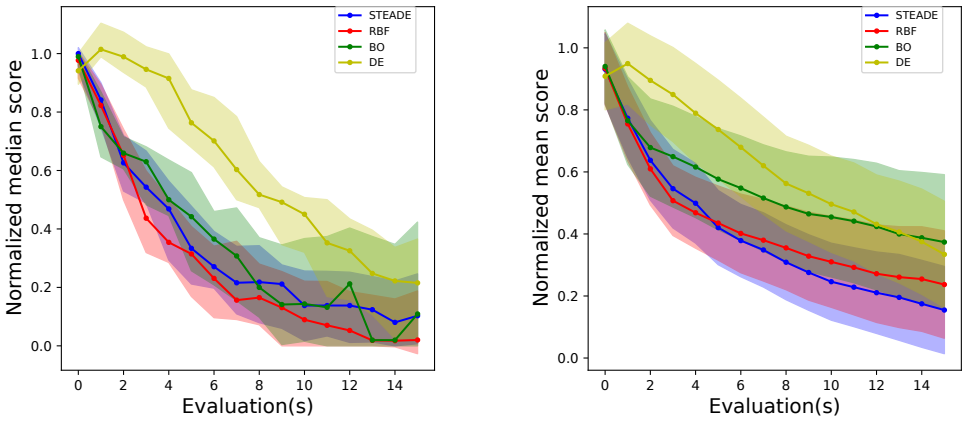

Figure 1: The median and mean `visible_to_opt` score obtained by the different algorithms.

3.2 Results and Discussions

In Figure 1, we notice that the random nature of DE search move results in low search efficiency during the initial stages of run while the surrogate-based models perform informed search due to their functional estimation capability. The search efficiency of the surrogate-based models gradually fall and after a certain point, there seems to be a saturation in a model's knowledge about the functional landscape. In such scenarios, one plausible way is to switch to a different model for carrying out further search. In STEADE, we transfer the knowledge from an RBF interpolation to a GP-based model. The solutions generated by the GP are further perturbed using a DE-based mutation. The efficacy of our approach is evident from Table 1, wherein STEADE is able to achieve better normalized mean score and leaderboard score in comparison to the baselines.

Even though DE has been extensively applied to solve cheap optimization problems, interestingly, very few works have explored its applicability to *low-budget* (computationally expensive) optimization problems. In this work, we take a step towards this direction and demonstrate how EAs can benefit from surrogate models in such scenarios. However, the increased cost of surrogates somewhat limits the applicability of such hybrid models to cheap/moderate budget optimization problems. In order

[5]This score is based on the simulations performed in the local machine, and is different from our actual leaderboard score of 89.846.

[6]See bayesmark.readthedocs.io/en/latest/scoring.html#analyze-and-summarize-results for scoring details.

to make that possible, we need to design careful learning mechanisms that will transition from the expensive surrogate-based model to a cheap EA model depending on the perceived gain in functional value with respect to computational time. This is a possible research direction that we wish to undertake in near future.

References

[1] Maximilian Balandat, Brian Karrer, Daniel Jiang, Samuel Daulton, Ben Letham, Andrew G Wilson, and Eytan Bakshy. Botorch: A framework for efficient monte-carlo bayesian optimization. *Advances in Neural Information Processing Systems*, 33, 2020.

[2] Subhodip Biswas, Souvik Kundu, and Swagatam Das. Inducing niching behavior in differential evolution through local information sharing. *IEEE Transactions on Evolutionary Computation*, 19(2):246–263, 2014.

[3] Eric Brochu, Vlad M Cora, and Nando De Freitas. A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *arXiv preprint arXiv:1012.2599*, 2010.

[4] Martin D Buhmann. *Radial basis functions: Theory and implementations*, volume 12. Cambridge university press, 2003.

[5] Russel E Caflisch et al. Monte carlo and quasi-monte carlo methods. *Acta numerica*, 1998:1–49, 1998.

[6] Alberto Costa and Giacomo Nannicini. RBFOpt: an open-source library for black-box optimization with costly function evaluations. *Mathematical Programming Computation*, 10(4):597–629, 2018.

[7] S. Das and P. N. Suganthan. Differential Evolution: A Survey of the State-of-the-Art. *IEEE Transactions on Evolutionary Computation*, 15(1):4–31, 2011.

[8] Agoston E Eiben and James E Smith. *Introduction to Evolutionary Computing*. Springer, 2015.

[9] David Eriksson, David Bindel, and Christine A Shoemaker. pysot and poap: An event-driven asynchronous framework for surrogate optimization. *arXiv preprint arXiv:1908.00420*, 2019.

[10] Matthias Feurer and Frank Hutter. Hyperparameter optimization. In *Automated Machine Learning*, pages 3–33. Springer, Cham, 2019.

[11] Donald R Jones, Matthias Schonlau, and William J Welch. Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, 13(4):455–492, 1998.

[12] Slawomir Koziel, David Echeverría Ciaurri, and Leifur Leifsson. *Surrogate-Based Methods*, pages 33–59. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[13] Art B Owen. Quasi-monte carlo sampling. *Monte Carlo Ray Tracing: SIGGRAPH*, 1:69–88, 2003.

[14] Rommel G Regis and Christine A Shoemaker. A stochastic radial basis function method for the global optimization of expensive functions. *INFORMS Journal on Computing*, 19(4):497–509, 2007.

[15] Rommel G Regis and Christine A Shoemaker. Combining radial basis function surrogates and dynamic coordinate search in high-dimensional expensive black-box optimization. *Engineering Optimization*, 45(5):529–555, 2013.

[16] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25, pages 2951–2959. Curran Associates, Inc., 2012.

[17] Jasper Snoek, Kevin Swersky, Rich Zemel, and Ryan Adams. Input warping for bayesian optimization of non-stationary functions. In Eric P. Xing and Tony Jebara, editors, *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 1674–1682, Bejing, China, 22–24 Jun 2014. PMLR.

[18] Rainer M. Storn and Kenneth V. Price. Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces. *Journal of global optimization*, 11(4):341–359, 1997.

[19] Christopher KI Williams and Carl Edward Rasmussen. *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA, 2006.

[20] James Wilson, Frank Hutter, and Marc Deisenroth. Maximizing acquisition functions for bayesian optimization. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31, pages 9884–9895. Curran Associates, Inc., 2018.