LEARNING TO [LOCALLY] OPTIMIZE A BLACK BOX

Anonymous authors
Paper under double-blind review

ABSTRACT

We consider black-box optimization with a limited number of function evaluations. This is a typical scenario when optimizing variable settings that are expensive to evaluate.

The traditional designs of optimization methods are *theory-driven*, so they obtain performance guarantees over the classes of problems specified by the theory. In sharp contrast, the concept of Learning to Optimize (L2O) represents a novel strategy that employs machine learning techniques to create optimization algorithms. This approach automates the design of an optimization algorithm for a class of optimization problems. This *data-driven* procedure generates methods that can efficiently solve problems similar to those in the training set.

1 INTRODUCTION

Black box optimization (BBO) refers to optimization techniques used when the objective function is complex, expensive to evaluate, or lacks a known analytical form. Here, the term "black box" indicates that the internal workings of the function are unknown or not utilized. Instead, the function is treated as an oracle that provides output values (or performance measures) for given input values without revealing the underlying process.

Given an objective function $f : \mathbb{R}^n \to \mathbb{R}$, black box optimization aims to find the input $x^* \in \mathbb{R}^n$ that minimizes $f(x)$. Mathematically, the optimization problem can be stated as:

$$x^* = \arg\min_{x \in \mathbb{R}^n} f(x), \tag{1}$$

where f is the black box function whose internal structure is unknown and only its input-output behavior can be observed.

Classic optimization algorithms such as Covariance Matrix Adaptation Evolution Strategy (CMA-ES) (Hansen & Ostermeier, 2001), Particle Swarm Optimization (PSO) Kennedy & Eberhart (1995), Differential Evolution (DE) Storn & Price (1997), and Bayesian Optimization (Pelikan & Pelikan, 2005) have been extensively used for black box optimization. These methods rely on hand-engineered rules and heuristics to explore and exploit the search space. They are effective for a broad range of problems but often require significant manual tuning to achieve optimal performance for specific tasks.

In contrast, a promising *data-driven* approach, Learning to Optimize (L2O), seeks to replace traditional human-tuned optimization algorithms with neural network-parameterized optimizers trained to learn update rules from data. This method leverages machine learning to automate the optimization process, reducing the need for extensive manual adjustments. While classical algorithms are *theory-driven* and provide performance guarantees based on theoretical analysis, L2O algorithms are designed through empirical training on a set of problems, aiming to generalize and perform well on new, similar tasks.

In this work, we consider the optimization of unimodal noise-free real-parameter black box functions with a single objective under a limited evaluation budget. Unimodal functions are characterized by having a single local minimum.

Let $f : \mathbb{R}^n \to \mathbb{R}$ be a unimodal black box function that needs to be optimized. Our goal is to find $x^* \in \mathbb{R}^n$ that minimizes $f(x)$ within the constraints of a limited number of function evaluations.

By employing L2O, we aim to develop optimization methods that can adaptively learn from data and improve performance over time, providing a significant advantage over traditional optimization techniques in scenarios where the objective function evaluations are costly and the problem-specific tuning is impractical.

2 Related work

Li and Malik (Li & Malik, 2017) introduced an alternative approach using reinforcement learning to optimize, where the gradient history and objective values are used as observations and step vectors as actions.This method demonstrated the potential of reinforcement learning in optimization tasks, particularly in leveraging historical data to make informed decisions.

L2O represents a paradigm shift from these traditional methods by leveraging machine learning to automate the design of optimization algorithms. The first significant contribution to L2O was by Andrychowicz (Andrychowicz et al., 2016), who formulated the update rules of optimizers as the input and output features for a Recurrent Neural Network (RNN) optimizer. This foundational work paved the way for subsequent advancements in the field.

In the work Chen et al. (2017) proposed a gradient-free optimization method that employs Gaussian Processes (GP) for training. This approach is particularly useful in scenarios where gradient information is unavailable or expensive to compute. Similarly, in the work TV et al. (2019) utilized Gaussian Mixture Models (GMM) for training optimization algorithms, focusing on global optimization tasks. This work highlighted the effectiveness of probabilistic models in capturing the underlying distribution of optimization landscapes. In contrast, our focus is on local optimization where the training and testing functions belong to the same family, characterized by random. This approach requires the training function to have a gradient, allowing for efficient use of gradient-based training methods. This ensures that the optimization algorithm is well-suited for the specific family of functions encountered during testing, providing a consistent and relevant optimization process.

Recent advancements in genetic programming have further expanded the L2O framework. Stanovov et al. (2022) demonstrated the use of genetic programming to develop symbolic expressions for parameter adaptation in differential evolution algorithms. This method automates the search for optimal parameter control strategies, leading to enhanced performance and the extraction of novel algorithmic insights. Such techniques complement our local optimization approach by offering adaptive mechanisms that can be integrated into gradient-based methods, enhancing their efficiency and adaptability.

Furthermore, Exploratory Landscape Analysis (ELA) (Mersmann et al., 2011) subsumes a number of techniques employed to obtain knowledge about the properties of an unknown optimization problem, particularly those that impact the performance of optimization algorithms. These features are not only relatively cheap to compute but also highly effective in distinguishing problem groups and relating them to high-level features. This capability paves the way for automatic algorithm selection, enhancing the L2O approach by providing a deeper understanding of optimization landscapes and guiding the selection of the most suitable algorithms for specific problem classes. However, a significant challenge with ELA is that among all features, the Y-values have been found to be the most significant. These Y-values correspond to specific types of functions, which can limit the generalization and adaptability of these features across diverse optimization problems. Despite this limitation, the low-level features generated through ELA remain valuable for L2O.

3 Learning to optimize black box

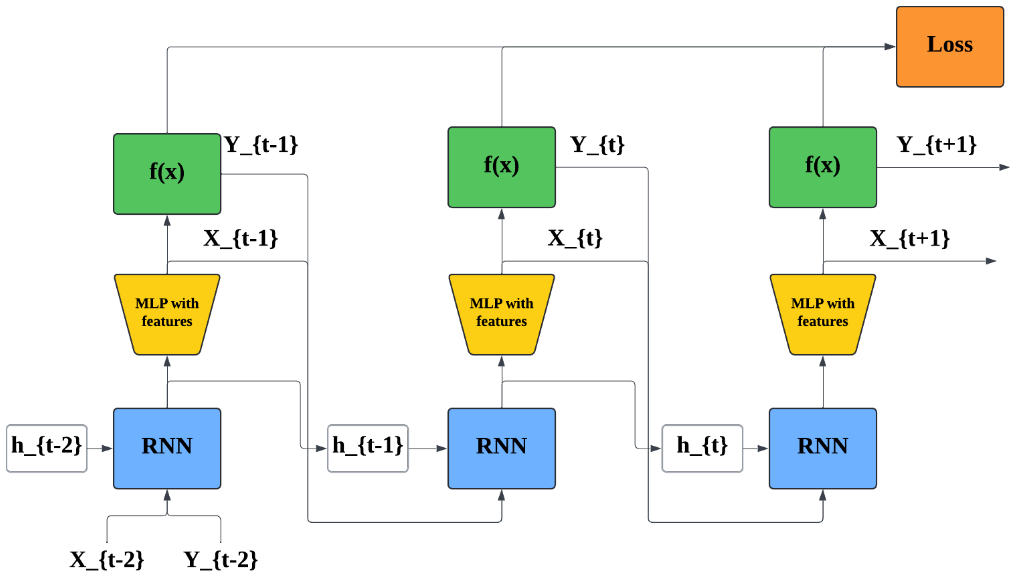
A black box optimization algorithm can be described through an iterative loop consisting of the following steps:

1. Based on the current state of knowledge h_t, propose a next query point x_t.
2. Observe the resultant response $y_t = f\left(x_t\right)$.
3. Update the internal state to obtain h_{t+1}.

Here, a recurrent neural network (RNN), parameterized by θ, updates its hidden state using data from the previous time step, and a simple linear layer derives the new query point x_t from the hidden state. The first query x_1 is generated by initializing the parameters to ones.

The combined update and query operations can be formulated as follows:

$$h_t = \text{RNN}_\theta(h_{t-1}, x_{t-1}, y_{t-1}), \tag{2}$$
$$x_t = \text{MLP}_\omega(h_t, \text{features}) \tag{3}$$



Here, h_t is the hidden state at time step t, and MLP_ω is a multi-layer perceptron parameterized by ω.

In our approach, the linear layer is structured with hidden dimensions $256 \to 128 \to$ output size, using LeakyReLU activation functions between each linear transformation.

The feature vectors x_{feat} and y_{feat} are constructed by concatenating the last four observed values and the best-known values. To ensure the values are within a comparable range and to improve numerical stability, we apply L2 normalization.

$$x_{\text{feat}} = [x_{\text{best}}, x_i, x_{i-1}, x_{i-2}, x_{i-3}], \tag{4}$$
$$y_{\text{feat}} = [y_{\text{best}}, y_i, y_{i-1}, y_{i-2}, y_{i-3}], \tag{5}$$
$$y_{\text{feat}} = \frac{y_{\text{feat}}}{\|y_{\text{feat}}\|_2} \tag{6}$$

This ensures that the feature vector y_{feat} is scaled appropriately, making it more suitable for further processing in algorithms sensitive to varying scales of input data.

The combined features to the linear layer is then given by:

$$\text{features} = [x_{\text{feat}}, y_{\text{feat}}] \tag{7}$$

3.1 Optimizing weighs

Lion (Evolved Sign Momentum) is a unique optimizer that uses the sign of the gradient to determine the update direction of the momentum (Chen et al., 2024). This makes Lion more memory-efficient and faster than AdamW which tracks and store the first and second-order moments.

3.2 Incorporating LSTM for Enhanced Memory

To further enhance the optimization process, we incorporate Long Short-Term Memory (LSTM) networks (Hochreiter & Schmidhuber, 1997). LSTMs are a type of RNN specifically designed to capture long-term dependencies and mitigate the vanishing gradient problem.

The LSTM equations are as follows:

$$f_t = \sigma(\mathbf{W}_f[h_{t-1}, x_{t-1}] + \mathbf{b}_f), \tag{8}$$
$$i_t = \sigma(\mathbf{W}_i[h_{t-1}, x_{t-1}] + \mathbf{b}_i), \tag{9}$$
$$o_t = \sigma(\mathbf{W}_o[h_{t-1}, x_{t-1}] + \mathbf{b}_o), \tag{10}$$
$$\tilde{c}_t = \tanh(\mathbf{W}_c[h_{t-1}, x_{t-1}] + \mathbf{b}_c), \tag{11}$$
$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t, \tag{12}$$
$$h_t = o_t \odot \tanh(c_t). \tag{13}$$

Here, f_t, i_t, o_t, and $\tilde{c}_t$ represent forget, input, output gates, and candidate cell state, respectively. The LSTM cell state c_t allows the network to retain long-term information.

3.3 Orthogonal Initialization of Hidden States

For stable and efficient training of RNNs, it is crucial to properly initialize the hidden states. Orthogonal initialization is a widely adopted technique that ensures the initial states are mutually orthogonal, which helps in preserving the gradient flow across many time steps (Vorontsov et al., 2017).

3.4 Validation

The validation process is crucial for assessing the generalization capability of the model. Validation data, which is a subset of the dataset not used in training, is employed to evaluate the model's performance after each epoch. This helps in ensuring that the model generalizes well to unseen data.

3.5 Iteration-Weighted Loss

To address the exploration-exploitation trade-off in black box optimization, we introduce a novel loss function termed *Iteration-Weighted Loss*. This function dynamically adjusts the weight of each iteration, thereby balancing the focus between exploring new areas and exploiting known promising regions.

The Iteration-Weighted Loss can be mathematically described as follows:

Given:

- $\mathbf{y}_{\text{best}}$: the best observed value at each iteration.
- $\mathbf{y}_{\text{found}}$: the value found at the current iteration.
- $\mathbf{w}$: the weight vector for iterations, with only the last n iterations having non-zero weights.

The loss for iteration t is defined as:

$$L_t = w_t \cdot \left(\frac{1}{N} \sum_{i=1}^{N} (y_{\text{found},i} - y_{\text{best},i}) \right) \tag{14}$$

Where w_t is the weight at iteration t, given by:

$$w_t = \begin{cases} \frac{\alpha^t}{\sum_{j=T-n+1}^{T} \alpha^j} & \text{for } T - n < t, \\ 0 & \text{otherwise.} \end{cases} \tag{15}$$

Here, T is the total number of iterations, n is the number of last iterations to consider, and α is a scaling coefficient where $\alpha \geq 1$.

The weight distribution w_t follows an exponentially increasing pattern for the last n iterations, with more recent iterations receiving higher weights. This ensures that the most recent observations have a larger influence on the optimization process, promoting better exploitation of recent findings while still allowing for exploration.

3.6 Advantages

- **Balancing Exploration and Exploitation**: By assigning higher weights to recent iterations, the loss function effectively balances the exploration of new regions and the exploitation of known good regions.
- **Dynamic Adjustment**: The weights are dynamically adjusted based on the progress of the optimization, allowing more flexible and responsive search behavior.
- **Enhanced Memory with LSTM**: Incorporating LSTM networks enables the optimization process to better handle long-term dependencies and improves overall performance.
- **Stable Training with Orthogonal Initialization**: Orthogonal initialization of hidden states ensures stable gradient flow, facilitating more effective training of RNNs.

4 Numerical experiments

For the experiments conducted in this study, we select unimodal functions from the Black-Box Optimization Benchmarking (BBOB) suite (Hansen et al., 2021). These functions are designed to evaluate the performance of optimization algorithms on single-objective, continuous optimization problems. The choice of unimodal functions provides a controlled environment to test the algorithms' efficiency in converging to the local optimum.

We compare our approach with various algorithms available in the Nevergrad optimization platform (Bennet et al., 2021). Nevergrad offers a wide range of algorithms, allowing a comprehensive evaluation and comparison of algorithms' performance.

4.1 Selection of Benchmark Functions

The unimodal functions from BBOB include F2, F6, F8, F11, F12, F13, F14. Each function is subjected to a random shift x. See details in Appendix B.

4.2 Black Box Optimizers

The algorithms selected from Nevergrad include Random Search, BOBYQA, Bayesian Optimization (implemented as "BayesOptimBO" and "BO" classes), CMA-ES, PSO, and DE, see details in Appendix A.

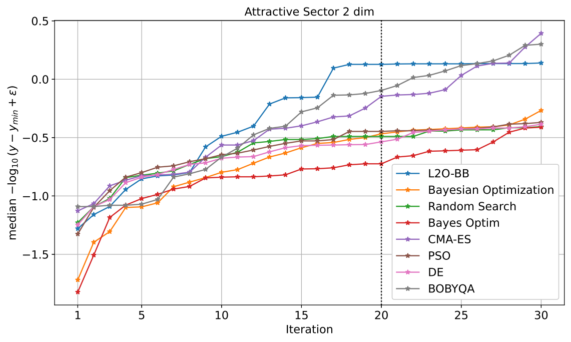
4.3 Iteration Scheme

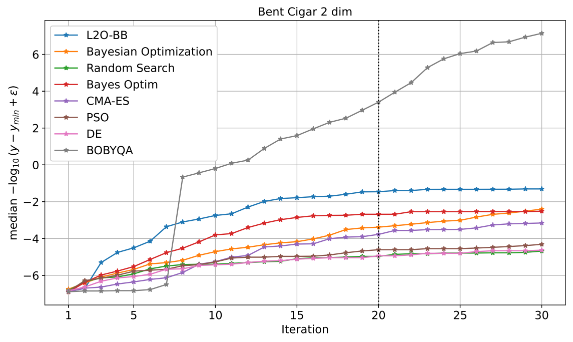
The first $10 \times D$ (where D is the dimensionality of the function) iterations are conducted under the algorithm's architecture. The subsequent $5 \times D$ iterations are not. This is done to test the algorithm's ability to predict independently of the iteration size.

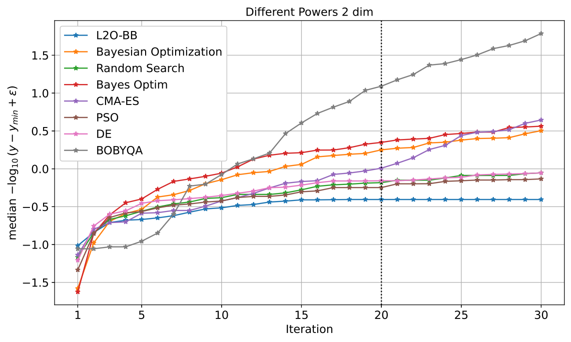
4.4 Performance Metric

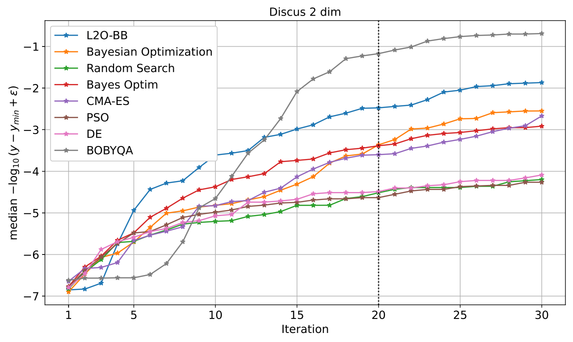
We use a logarithmic transformation to assess the performance more precisely. Specifically, the metric $-\log_{10}(y - y_{min})$ is used, where y is the current solution value and y_{min} is the global optimum. This transformation helps in distinguishing small differences in performance, making it easier to compare algorithms that converge very close to the optimal solution. The x-axis represents the iteration number, while the y-axis shows the median $-\log_{10}(y - y_{min})$, where y is the solution value at a given iteration and y_{min} is the known optimal value.

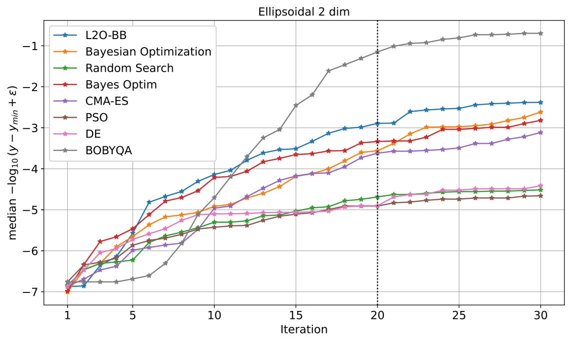
In the loss function, all iterations were considered significant (since the tasks are unimodal) and alpha = 1.5.

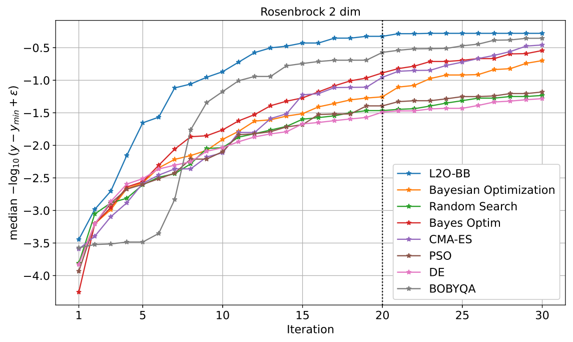


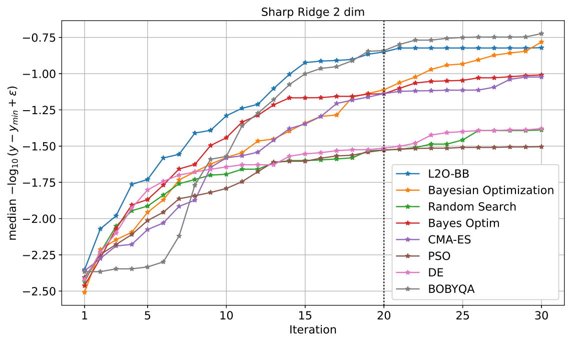


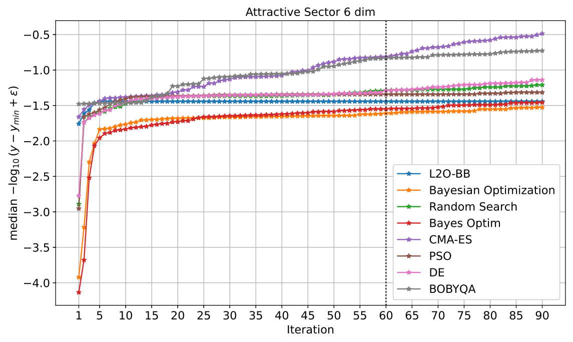


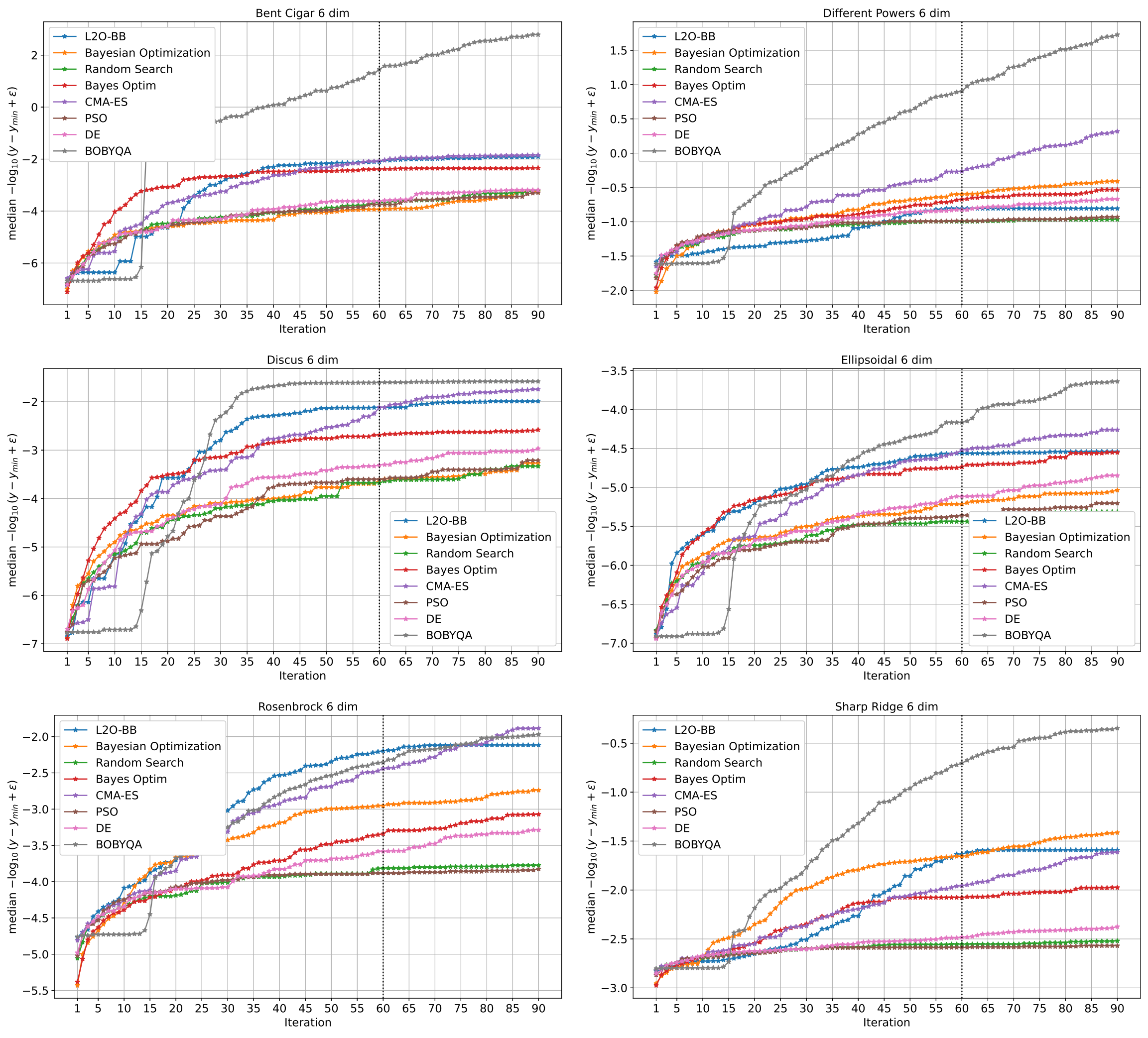












5 Discussion and Limitations

Our experiments revealed several important insights:

1. Random search, as expected, does not perform well due to its lack of guided exploration.
2. The performance variation in Bayesian optimization models is due to differences in implementation and hyperparameters, demonstrating a limitation of theory-driven approaches.
3. BOBYQA excels on unimodal functions, likely due to its utilization of quadratic approximations to efficiently model the local search space.

When training Learning to Optimize (L2O) models, having gradients of the black box functions is essential. These gradients are crucial for the optimization process within the L2O framework. However, obtaining gradients directly from black box functions is challenging. The solutions to this problem lie in function approximation or gradient approximation.

6 Future Works

In this section, we discuss potential future research directions to further enhance Learning to Optimize (L2O) methods, particularly focusing on global optimization and the generalization problem.

- **Gradient Approximation in Black Box Optimization:** Developing effective methods for approximating gradients in black box functions remains a critical challenge. Future research could explore advanced techniques for gradient approximation, such as using surrogate models or leveraging ensemble methods to estimate gradients more accurately.
- **Global Optimization Variants:** Enhancing L2O models for global optimization could involve designing different strategies for the initial and final stages of the optimization process. For instance, an adaptive approach that employs more exploratory methods at the beginning and more exploitative methods towards the end could be beneficial.
- **Generalization and Transfer Learning:** Improving the generalization capabilities of L2O algorithms is essential for their application to a wider range of problems. Research could focus on transfer learning techniques that allow models trained on certain types of optimization problems to be effectively adapted to new, unseen problems (Wolpert & Macready, 1997).
- **Exploiting Exploratory Landscape Analysis (ELA) Features:** Incorporating ELA features into the L2O framework can provide valuable insights into the structure of optimization problems. This could guide the learning process and improve the efficiency and robustness of the optimization algorithms.
- **Learning from Optimization Trajectories:** Analyzing and learning from optimization trajectories can offer insights into the optimization process and help in refining L2O models. Future work could involve creating methods that leverage historical optimization data to inform and improve current optimization strategies.

These research directions highlight the potential for further advancements in the field of Learning to Optimize, aiming to make optimization algorithms more efficient, robust, and generalizable across diverse optimization problems.

REFERENCES

Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando De Freitas. Learning to learn by gradient descent by gradient descent. *Advances in neural information processing systems*, 29, 2016.

Pauline Bennet, Carola Doerr, Antoine Moreau, Jeremy Rapin, Fabien Teytaud, and Olivier Teytaud. Nevergrad: black-box optimization platform. *ACM SIGEVOlution*, 14(1):8–15, 2021.

Xiangning Chen, Chen Liang, Da Huang, Esteban Real, Kaiyuan Wang, Hieu Pham, Xuanyi Dong, Thang Luong, Cho-Jui Hsieh, Yifeng Lu, et al. Symbolic discovery of optimization algorithms. *Advances in neural information processing systems*, 36, 2024.

Yutian Chen, Matthew W Hoffman, Sergio Gómez Colmenarejo, Misha Denil, Timothy P Lillicrap, Matt Botvinick, and Nando Freitas. Learning to learn without gradient descent by gradient descent. In *International Conference on Machine Learning*, pp. 748–756. PMLR, 2017.

Nikolaus Hansen and Andreas Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary computation*, 9(2):159–195, 2001.

Nikolaus Hansen, Anne Auger, Raymond Ros, Olaf Mersmann, Tea Tušar, and Dimo Brockhoff. Coco: A platform for comparing continuous optimizers in a black-box setting. *Optimization Methods and Software*, 36(1):114–144, 2021.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

James Kennedy and Russell Eberhart. Particle swarm optimization. In *Proceedings of ICNN'95-international conference on neural networks*, volume 4, pp. 1942–1948. ieee, 1995.

Ke Li and Jitendra Malik. Learning to optimize neural nets. *arXiv preprint arXiv:1703.00441*, 2017.

Olaf Mersmann, Bernd Bischl, Heike Trautmann, Mike Preuss, Claus Weihs, and Günter Rudolph. Exploratory landscape analysis. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, pp. 829–836, 2011.

Martin Pelikan and Martin Pelikan. Bayesian optimization algorithm. *Hierarchical Bayesian optimization algorithm: toward a new generation of evolutionary algorithms*, pp. 31–48, 2005.

Vladimir Stanovov, Shakhnaz Akhmedova, and Eugene Semenkin. The automatic design of parameter adaptation techniques for differential evolution with genetic programming. *Knowledge-Based Systems*, 239:108070, 2022.

Rainer Storn and Kenneth Price. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, 11:341–359, 1997.

Vishnu TV, Pankaj Malhotra, Jyoti Narwariya, Lovekesh Vig, and Gautam Shroff. Meta-learning for black-box optimization. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 366–381. Springer, 2019.

Eugene Vorontsov, Chiheb Trabelsi, Samuel Kadoury, and Chris Pal. On orthogonality and learning recurrent networks with long term dependencies. In *International Conference on Machine Learning*, pp. 3570–3578. PMLR, 2017.

David H Wolpert and William G Macready. No free lunch theorems for optimization. *IEEE transactions on evolutionary computation*, 1(1):67–82, 1997.

A NEVERGRAD OPTIMIZERS

- **Random Search**: ng.optimizers.RandomSearch
- **BOBYQA**: ng.optimizers.BOBYQA
- **Covariance matrix adaptation evolution strategy (CMA)**: ng.optimizers.CMAbounded
- **Bayesian Optimization (BayesOptimBO)**: ng.optimizers.BayesOptimBO
- **Bayesian Optimization (BO)**: ng.optimizers.BO
- **Differential Evolution (DE)**: ng.optimizers.DE
- **Particle Swarm Optimization (PSO)**: ng.optimizers.PSO

B FUNCTIONS

B.1 DEFINITIONS

$$\mathbf{x}_{\text{opt}} \sim \mathcal{U}(-5, 5)$$
$$\mathbf{c}_1 \sim 10 \times \mathcal{U}(0, 1)$$

$$T_{\text{osz}}(x) = \text{sign}(x) \exp\left(\log(|x|) + 0.049\left(\sin(c_1 \log(|x|)) + \sin(c_2 \log(|x|))\right)\right)$$

where:

- $c_1 = 10$ if $x > 0$, else 5.5
- $c_2 = 7.9$ if $x > 0$, else 3.1

$$T_{\text{asy}}(x) = x + \beta \left(\frac{i}{D-1}\right)^{\frac{1}{D-1}} \sqrt{\max(x, 0)}$$

where β is a hyperparameter and i is the index of the element in $\mathbf{x}$.

B.2 ELLIPSOIDAL

$$f(\mathbf{x}) = \sum_{i=1}^{D} 10^{6\left(\frac{i}{D-1}\right)} \left(T_{\text{osz}}(x_i - x_{\text{opt},i})\right)^2$$

B.3 Discus

$$f(\mathbf{x}) = 10^6 (T_{\text{osz}}(x_1 - x_{\text{opt},1}))^2 + \sum_{i=2}^{D} (T_{\text{osz}}(x_i - x_{\text{opt},i}))^2$$

B.4 Bent Sigar

$$f(\mathbf{x}) = 10^6 (T_{\text{asy}}(x_1 - x_{\text{opt},1}))^2 + \sum_{i=2}^{D} (T_{\text{asy}}(x_i - x_{\text{opt},i}))^2$$

B.5 Attractive

$$f(\mathbf{x}) = \left(\sum_{i=1}^{D} s_i (x_i - x_{\text{opt},i})^2 \right)^{0.9}$$

where:

- $s_i = 100$ if $x_i x_{\text{opt},i} > 0$, else $s_i = 1$

B.6 Rosenbrock

$$f(\mathbf{x}) = \sum_{i=1}^{D-1} \left[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2 \right]$$

B.7 Different Powers

$$f(\mathbf{x}) = \sqrt{\sum_{i=1}^{D} |x_i|^{2+4\frac{i-1}{D-1}}}$$

B.8 Sharp Ridge

$$f(\mathbf{x}) = x_1^2 + 100\sqrt{\sum_{i=2}^{D} x_i^2}$$