Bayesian Optimization and Its Applications

Jungtaek Kim (jtkim@postech.ac.kr)

Department of Computer Science and Engineering, POSTECH,
77 Cheongam-ro, Nam-gu, Pohang 37673,
Gyeongsangbuk-do, Republic of Korea
https://jungtaek.github.io

April 22, 2021

Table of Contents

Bayesian Optimization

Mathematical Optimization

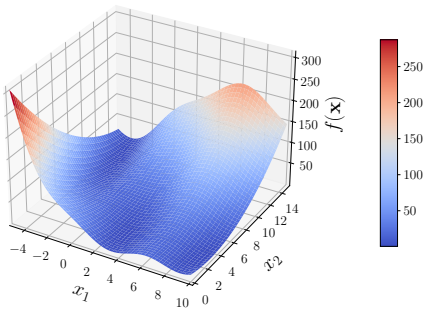


Figure 1: Branin function.

- Given an objective $f : \mathcal{A} \to \mathbb{R}$ where $\mathcal{A}$ is some set, it seeks minimum or maximum of the target function:

$$\mathbf{x}^* = \arg\min f(\mathbf{x}), \tag{1}$$

or

$$\mathbf{x}^* = \arg\max f(\mathbf{x}). \tag{2}$$

Mathematical Optimization

- To optimize an objective, we can select one of such strategies:
  - Gradient descent,
  - Convex programming,
  - Metaheuristic.
- Each strategy has the advantage in the corresponding conditions of optimization problem.
- However, under certain circumstances, Bayesian optimization is the most effective method to solve some class of mathematical optimization problem.

Target Functions in Bayesian Optimization

- Usually, an expensive black-box function f,
  which has unknown functional forms or local geometric features such as saddle points, global optima, and local optima, is optimized,
  where a d-dimensional search space $\mathcal{X} \subset \mathbb{R}^d$ is convex and compact.

- Moreover, we assume that the continuity of f can be unknown,
  and high-dimensional and mixed-variable domain space can be given.

Bayesian Optimization

- A powerful strategy for finding the extrema of objective function that is expensive to evaluate,
  - where one does not have a closed-form expression for the objective function,
  - but where one can obtain observations at samples to evaluate.
- Since we do not know a target function, it optimizes an acquisition function, instead of the target function.
- An acquisition function is defined with the outputs of Bayesian regression model.

Bayesian Optimization

Algorithm 1 Bayesian Optimization

Input: Initial data $\mathcal{D}_{1:k} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{k}$ and a time budget T.
Output: The best candidate of global optimum $\mathbf{x}^\dagger$.

1: **for** $t = 1, 2, \ldots, T$ **do**
2: Predict a function $\hat{f}(\mathbf{x} \mid \mathcal{D}_{1:k+t-1})$ considered as an objective function.
3: Find a query $\mathbf{x}_{k+t}$ that maximizes an acquisition function:

$$\mathbf{x}_{k+t} = \arg\max_{\mathbf{x}} a(\mathbf{x} \mid \hat{f}, \mathcal{D}_{1:k+t-1}). \tag{3}$$

4: Evaluate a true objective function, $y_{k+t} = f(\mathbf{x}_{k+t}) + \epsilon_{k+t}$.
5: Update historical data:

$$\mathcal{D}_{1:k+t} \leftarrow \mathcal{D}_{1:k+t-1} + \{(\mathbf{x}_t, y_t)\}. \tag{4}$$

6: **end for**
7: **return** the best query $\mathbf{x}^\dagger$: $(\mathbf{x}^\dagger, y^\dagger) = \arg\min_{(\mathbf{x},y)\in\mathcal{D}_{1:k+T}} y$.

Surrogate Models

- A surrogate model estimates a true objective function, where historical observations are given.
- To balance exploration and exploitation, it predicts a function estimate and its uncertainty estimate over any query $\mathbf{x} \in \mathcal{X}$.
- Gaussian process regression, random forests regression [Hutter et al., 2011], and Bayesian neural network [Springenberg et al., 2016] have been used.

Gaussian Process

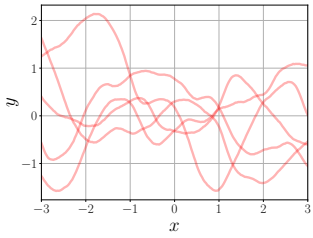
- A collection of random variables, any finite number of which have a joint Gaussian distribution [Rasmussen and Williams, 2006].
- Generally, Gaussian process (GP) is defined as

$$f \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')), \tag{5}$$

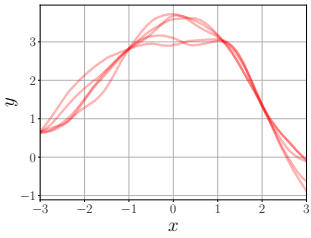
where

$$m(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})], \tag{6}$$
$$k(\mathbf{x}, \mathbf{x}') = \mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))]. \tag{7}$$

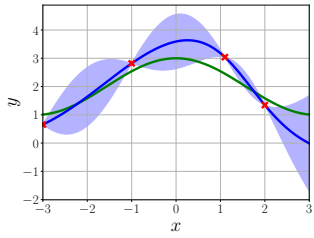
Gaussian Process Regression



(a) From prior function dist.



(b) From posterior function dist.



(c) Predictive dist.

Figure 2: Gaussian process regression for a function $\cos(x) + 2$ with observation noise.

Gaussian Process Regression

- One of popular covariance functions, the squared-exponential covariance function in one dimension is defined as

$$k\left(x, x'\right) = \sigma_f^2 \exp\left(-\frac{1}{2l^2}\left(x - x'\right)^2\right) + \sigma_n^2 \delta_{xx'}, \tag{8}$$

where σ_f is a signal level, l is a length scale and σ_n is a noise level [Rasmussen and Williams, 2006].

- Posterior mean function $\mu(\cdot)$ and covariance function $\Sigma(\cdot)$:

$$\mu(\mathbf{X}^*) = K(\mathbf{X}^*, \mathbf{X})(K(\mathbf{X}, \mathbf{X}) + \sigma_n^2 \boldsymbol{I})^{-1}\mathbf{y}, \tag{9}$$

$$\Sigma(\mathbf{X}^*) = K(\mathbf{X}^*, \mathbf{X}^*) - K(\mathbf{X}^*, \mathbf{X})(K(\mathbf{X}, \mathbf{X}) + \sigma_n^2 \boldsymbol{I})^{-1} K(\mathbf{X}, \mathbf{X}^*). \tag{10}$$

Gaussian Process Regression

- If non-zero mean prior is given, posterior mean and covariance functions:

$$\mu(\mathbf{X}^*) = K(\mathbf{X}^*, \mathbf{X})(K(\mathbf{X}, \mathbf{X}) + \sigma_n^2 \mathbf{I})^{-1}(\mathbf{y} - \mu_p(\mathbf{X})) + \mu_p(\mathbf{X}), \tag{11}$$

$$\Sigma(\mathbf{X}^*) = K(\mathbf{X}^*, \mathbf{X}^*) + K(\mathbf{X}^*, \mathbf{X})(K(\mathbf{X}, \mathbf{X}) + \sigma_n^2 \mathbf{I})^{-1} K(\mathbf{X}, \mathbf{X}^*), \tag{12}$$

  where $\mu_p(\cdot)$ is a prior mean function.

Acquisition Functions

- An acquisition function acquires the next point to evaluate for an expensive black-box function f.
- Traditionally, the probability of improvement (PI) [Kushner, 1964], the expected improvement (EI) [Močkus et al., 1978], and GP upper confidence bound (GP-UCB) [Srinivas et al., 2010] have been used.
- Several functions such as knowledge gradient [Frazier et al., 2009], entropy search [Hennig and Schuler, 2012] and a combination of existing functions [Kim and Choi, 2018] have been proposed.

Popular Acquisition Functions (Minimization Case)

- Suppose that $(\mathbf{x}^\dagger, y^\dagger) = \arg\min_{(\mathbf{x},y)\in\mathcal{D}} y$,

$$\mu(\mathbf{x}) \coloneqq \mu(\mathbf{x} \,|\, \mathcal{D}, \boldsymbol{\lambda}), \tag{13}$$

$$\sigma(\mathbf{x}) \coloneqq \sigma(\mathbf{x} \,|\, \mathcal{D}, \boldsymbol{\lambda}), \tag{14}$$

$$z = \begin{cases} \frac{f(\mathbf{x}^\dagger) - \mu(\mathbf{x})}{\sigma(\mathbf{x})}, & \text{if } \sigma(\mathbf{x}) > 0 \\ 0, & \text{if } \sigma(\mathbf{x}) = 0 \end{cases}. \tag{15}$$

- PI criterion [Kushner, 1964] is defined as

$$a_{\mathrm{PI}}(\mathbf{x} \,|\, \mathcal{D}, \boldsymbol{\lambda}) = \Phi(z), \tag{16}$$

where Φ is a cumulative distribution function of standard normal distribution.

Popular Acquisition Functions (Minimization Case)

- EI criterion [Močkus et al., 1978] is defined as

$$a_{\mathrm{EI}}(\mathbf{x} \mid \mathcal{D}, \boldsymbol{\lambda}) = \begin{cases} (f(\mathbf{x}^{\dagger}) - \mu(\mathbf{x}))\Phi(z) + \sigma(\mathbf{x})\phi(z), & \text{if} \quad \sigma(\mathbf{x}) > 0 \\ 0, & \text{if} \quad \sigma(\mathbf{x}) = 0 \end{cases}, \tag{17}$$

  where ϕ is a probability density function of standard normal distribution.

- GP-UCB criterion [Srinivas et al., 2010] is defined as

$$a_{\mathrm{UCB}}(\mathbf{x} \mid \mathcal{D}, \boldsymbol{\lambda}) = -\mu(\mathbf{x}) + \beta\sigma(\mathbf{x}), \tag{18}$$

  where β is a trade-off hyperparameter.

Acquisition Function Optimization

- We should find a global optimum of acquisition function.
- But, in practice, either global optimizer or local optimizer is used.
- Multi-started local optimizers can be a good option as a substitute of global optimum.
- Analyses on these selections are provided in [Kim and Choi, 2020].

On Local Optimizers of Acquisition Functions in Bayesian Optimization [Kim and Choi, 2020]

Theorem 1. *Given $\delta_l \in [0,1)$ and $\epsilon_l, \epsilon_1, \epsilon_2 > 0$, the regret difference for a local optimizer $\mathbf{x}_{t,l}$ at round t, $|r_{t,g} - r_{t,l}|$ is less than ϵ_l with a probability at least $1-\delta_l$:*

$$\mathbb{P}\big(|r_{t,g} - r_{t,l}| < \epsilon_l\big) \geq 1 - \delta_l, \tag{10}$$

where $\delta_l = \frac{\gamma}{\epsilon_1}(1-\beta_g) + \frac{M}{\epsilon_2}$, $\epsilon_l = \epsilon_1\epsilon_2$, $\gamma = \max_{\mathbf{x}_i,\mathbf{x}_j \in \mathcal{X}} \|\mathbf{x}_i - \mathbf{x}_j\|_2$ is the size of $\mathcal{X}$, β_g is the probability that a local optimizer of the acquisition function collapses with its global optimizer, and M is the Lipschitz constant explained in Lemma 8.

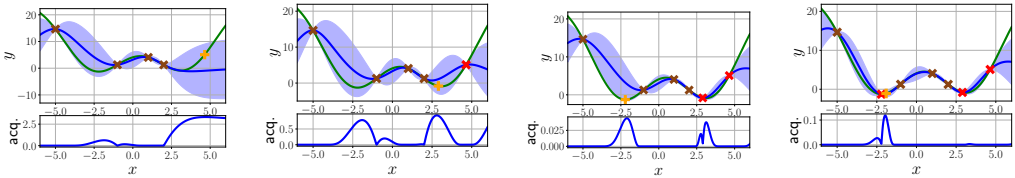
On Local Optimizers of Acquisition Functions in Bayesian Optimization [Kim and Choi, 2020]

Theorem 2. *Given $\delta_m \in [0, 1)$ and $\epsilon_m, \epsilon_2, \epsilon_3 > 0$, a regret difference for a multi-started local optimizer $\mathbf{x}_{t,m}$, determined by starting from N initial points at round t, is less than ϵ_m with a probability at least $1 - \delta_m$:*

$$\mathbb{P}\big(|r_{t,g} - r_{t,m}| < \epsilon_m\big) \geq 1 - \delta_m, \tag{11}$$

where $\delta_m = \frac{\gamma}{\epsilon_3}(1 - \beta_g)^N + \frac{M}{\epsilon_2}$, $\epsilon_m = \epsilon_2\epsilon_3$, $\gamma = \max_{\mathbf{x}_i, \mathbf{x}_j \in \mathcal{X}} \|\mathbf{x}_i - \mathbf{x}_j\|_2$ is the size of $\mathcal{X}$, β_g is the probability that a local optimizer of the acquisition function collapses with its global optimizer, and M is the Lipschitz constant explained in Lemma 8.

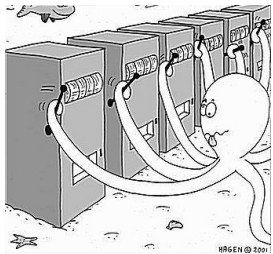
Synthetic Example



(a) Iteration 1 (b) Iteration 2 (c) Iteration 3 (d) Iteration 4

Figure 3: $y = 4.0\cos(x) + 0.1x + 2.0\sin(x) + 0.4(x - 0.5)^2$. EI is used to optimize.

Relationship to Multi-Armed Bandit Problem



- Each machine returns a reward $\hat{r}_a \sim p_{\boldsymbol{\theta}_a}(r_a)$ where $a \in \{1, \ldots, K\}$.
- It minimizes a cumulative regret $T\mu^* - \sum_{t=1}^{T} \hat{r}_{a_t}$ where $\mu^* = \max_{a \in \{1,\ldots,K\}} \mu_a$.
- Bayesian optimization can be considered as infinite bandits with dependent arms.

Relationship to Thompson Sampling

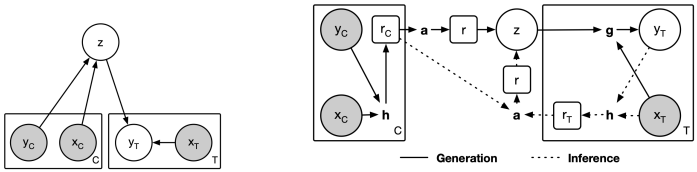
- Thompson sampling is usually applied in multi-armed bandit problems.
- For the case of a beta-Bernoulli bandit, Thompson sampling is defined as

Algorithm 2 Thompson Sampling for a Beta-Bernoulli Bandit

for $t = 1, 2, \ldots, T$ **do**
  for $k = 1, \ldots, K$ **do**
    Sample $\hat{\theta}_k \sim \text{beta}(\alpha_k, \beta_k)$.
  end for
  $x_t \leftarrow \arg\max_k \hat{\theta}_k$.
  Apply x_t and observe r_t.
  $(\alpha_{x_t}, \beta_{x_t}) \leftarrow (\alpha_{x_t} + r_t, \beta_{x_t} + 1 - r_t)$.
end for

- After sampling the possibilities, it chooses a maximizer of those sampled values.

Relationship to Neural Processes



(a) Graphical model

(b) Computational diagram

Figure 1. **Neural process model. (a)** Graphical model of a neural process. x and y correspond to the data where $y = f(x)$. C and T are the number of context points and target points respectively and z is the global latent variable. A grey background indicates that the variable is observed. **(b)** Diagram of our neural process implementation. Variables in circles correspond to the variables of the graphical model in (a), variables in square boxes to the intermediate representations of NPs and unbound, bold letters to the following computation modules: h - encoder, a - aggregator and g - decoder. In our implementation h and g correspond to neural networks and a to the mean function. The continuous lines depict the generative process, the dotted lines the inference.

- Neural processes model distributions over functions, similar to Gaussian processes [Garnelo et al., 2018a,b, Lee et al., 2020].
- It might be used as a surrogate model in meta-learning manner.

BayesO [Kim and Choi, 2017]

- Current version: 0.5.0
- Supported Python version: 3.6, 3.7, 3.8, 3.9 (tested by Travis CI)
- Web page: `https://bayeso.org`
- GitHub repo: `https://github.com/jungtaekkim/bayeso`
- Documentation: `https://bayeso.readthedocs.io`
- License: MIT license

Comparison to Related Software in Python

	Input		Surrogate			Native	GPU
	Vector	Set	GP	TP	RF	Implement	Support
SMAC [Hutter et al., 2011]	✓				✓		
BayesOpt [Martinez-Cantin, 2014]	✓		✓				
GPyOpt [The GPyOpt authors, 2016]	✓		✓			✓	
GPFlowOpt [Knudde et al., 2017]	✓		✓				✓
BoTorch [Balandat et al., 2020]	✓		✓				✓
BayesO [Kim and Choi, 2017]	✓	✓	✓	✓		✓	

Table 1: Comparison to related software in Python. GP, TP, and RF stand for Gaussian process regression, Student-t process regression, and random forests. Native implement indicates an implementation with Python and the minimum extra packages (including NumPy and SciPy).

Bayesian Optimization on Structured Search Space

- Graph structure [Cui and Yang, 2018, Oh et al., 2019]
- Set structure [Buathong et al., 2020, Kim et al., 2021]
- Combinatorial structure [Baptista and Poloczek, 2018, Oh et al., 2019]
- Topological structure [Shiraishi et al., 2020]

Combinatorial Bayesian Optimization using the Graph Cartesian Product [Oh et al., 2019]

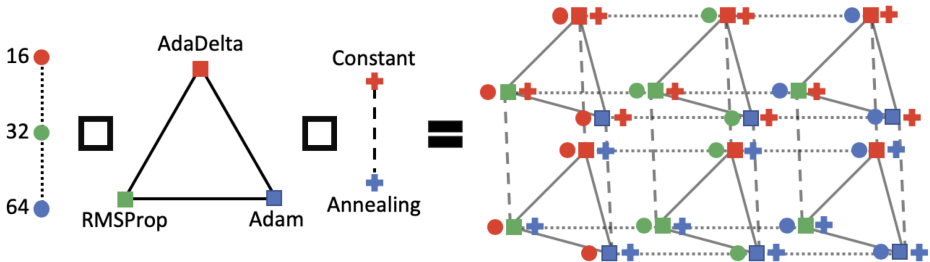


Figure 1: Combinatorial Graph: graph Cartesian product of sub-graphs $\mathcal{G}(C_1)\square\mathcal{G}(C_2)\square\mathcal{G}(C_3)$

Topological Bayesian Optimization with Persistence Diagrams [Shiraishi et al., 2020]

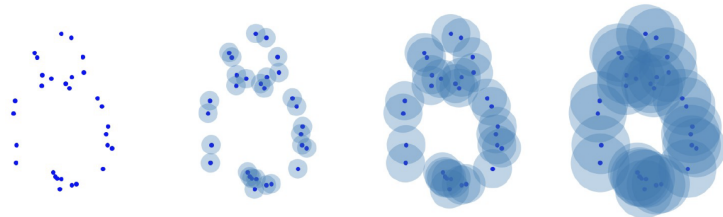


Figure 1. Examples of S_r. We can observe that topological structures like connected components and rings appear and disappear while increasing r.

Bayesian Optimization with Approximate Set Kernels [Kim et al., 2021]

- Denote that a set of m vectors is $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$.
- If we define a set kernel

$$k_{\text{set}}(\mathbf{X}^{(1)}, \mathbf{X}^{(2)}) = \frac{1}{|\mathbf{X}^{(1)}||\mathbf{X}^{(2)}|} \sum_{i=1}^{|\mathbf{X}^{(1)}|} \sum_{j=1}^{|\mathbf{X}^{(2)}|} k(\mathbf{x}_i^{(1)}, \mathbf{x}_j^{(2)}), \tag{19}$$

  a stochastic process can be directly employed.

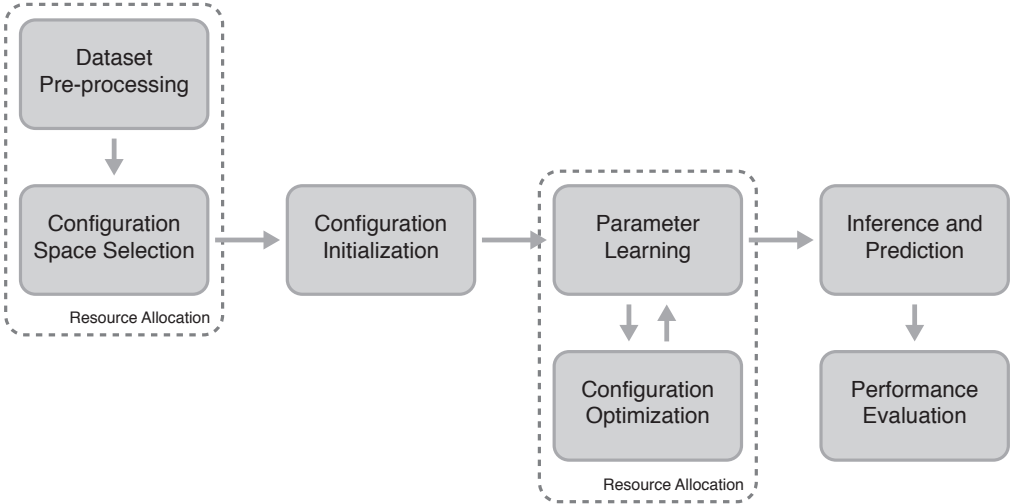
- To reduce the computational complexity $\mathcal{O}(n^2 m^2 d)$, we propose an approximation of the set kernel:

$$\tilde{k}_{\text{set}}(\mathbf{X}^{(1)}, \mathbf{X}^{(2)}; \pi, \mathbf{w}, L) = k_{\text{set}}(\tilde{\mathbf{X}}^{(1)}, \tilde{\mathbf{X}}^{(2)}), \tag{20}$$

  where π is a permutation function, $\mathbf{w} \in \mathbb{R}^d$ is a vector for random scalar projection, and L is the number of subsamples.

Applications of Bayesian Optimization

Automated Machine Learning



Automated Machine Learning

- Data pre-processing
- Configuration space selection
  - Specify a searching space $\mathcal{X}$ (e.g., a Cartesian space of kernel type, penalty parameter, kernel coefficient, and degree)
- Configuration initialization
  - Random initialization, Low discrepancy initialization
  - Learn to initialize [Kim et al., 2017]
- Configuration optimization
  - Random search [Bergstra and Bengio, 2012], Grid search
  - Bayesian optimization
  - Neural architecture search [Zoph and Le, 2017]
- Resource allocation

Automated Machine Learning

- It automatically finds the optimal machine learning model without human intervention.
- Feature transformation, algorithm selection, and hyperparameter optimization are usually included.
- Given a training dataset $\mathcal{D}_{\text{train}}$ and a validation dataset $\mathcal{D}_{\text{val}}$, the optimal hyperparameter vector $\boldsymbol{\lambda}^*$ for an automated machine learning system is found:

$$\boldsymbol{\lambda}^* = \text{AutoML}(\mathcal{D}_{\text{train}}, \mathcal{D}_{\text{val}}, \Lambda), \tag{21}$$

where AutoML is an automated machine learning system.

AutoML-Zero: Evolving Machine Learning Algorithms From Scratch [Real et al., 2020]

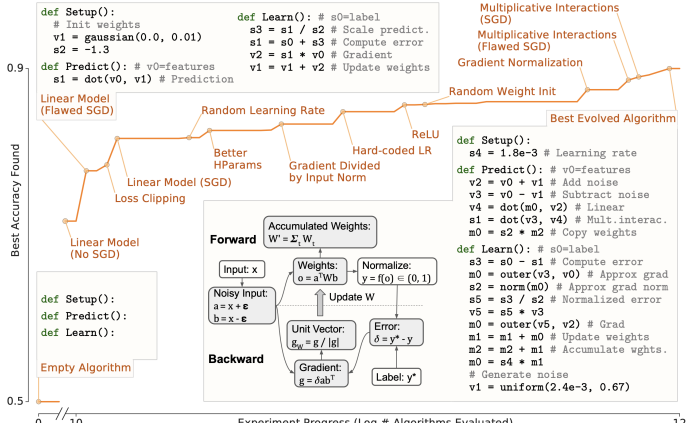


Figure 6: Progress of one evolution experiment on projected binary CIFAR-10. Callouts indicate some beneficial discoveries. We also print the code for the initial, an intermediate, and the final algorithm. The last is explained in the flow diagram. It outperforms a simple fully connected neural network on held-out test data and transfers to features 10x its size. Code notation is the same as in Figure 5. The x-axis gap is due to infrequent recording due to disk throughput limitations.

Combinatorial 3D Shape Generation via Sequential Assembly [Kim et al., 2020]

- 3D shape generation via sequential assembly mimics a human assembling process, by allocating a budget of primitives given.
- We solve a sequential problem with Bayesian optimization-based framework of combinatorial 3D shape generation
- It creates a 3D shape with a set of geometric primitives.
- We also introduce a new combinatorial 3D shape dataset that consists of 14 classes and 406 instances.

Sequential Assembly with Unit Primitives

- Instead of employing other 3D representations such as point clouds, triangular meshes, and voxels, we create a sequence of unit primitives such as 2×4 LEGO bricks.

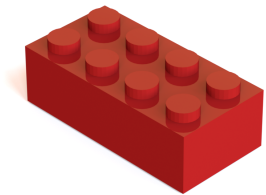


Figure 4: 2×4 LEGO brick.

- This 2×4 LEGO bricks make our problem more combinatorial and more complex, compared to other primitives.

Combinatorial 3D Shape Generation

- To determine the position of the next primitive, we define two evaluation functions regarding occupiability and stability.
- Occupiability encourages us to follow a target shape and stability helps to create a physically stable combination.
- We determine the position of the next primitive via Bayesian optimization.
- To avoid a suboptimal sequence, our framework includes a rollback step.

Experimental Results

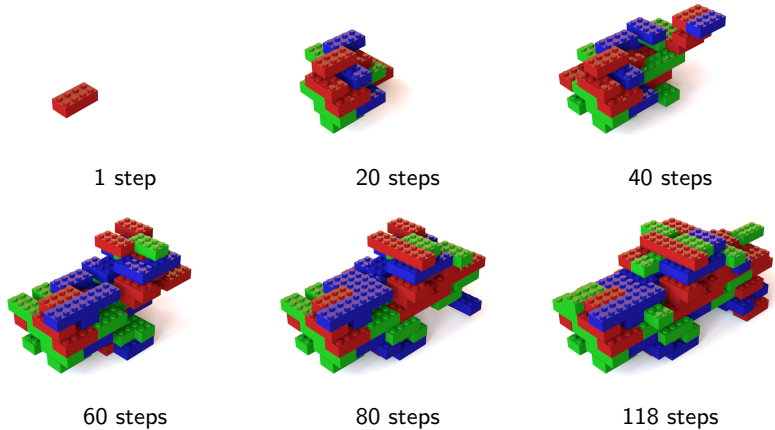
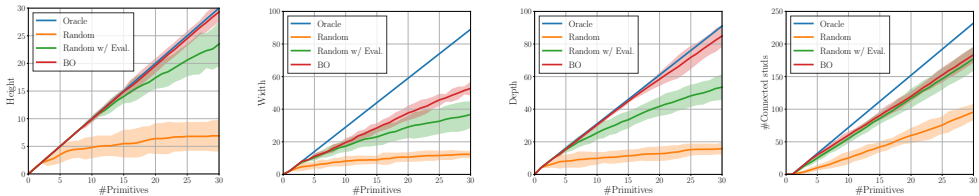


Figure 5: Generated assembling sequence that creates a *car* shape with 118 unit primitives.

Experimental Results

- We apply our framework in optimizing specific explicit functions.



(a) Height (b) Width (c) Depth (d) #Conn. studs

Figure 6: Quantitative results on maximizing explicit evaluation functions.

Combinatorial 3D Shape Dataset

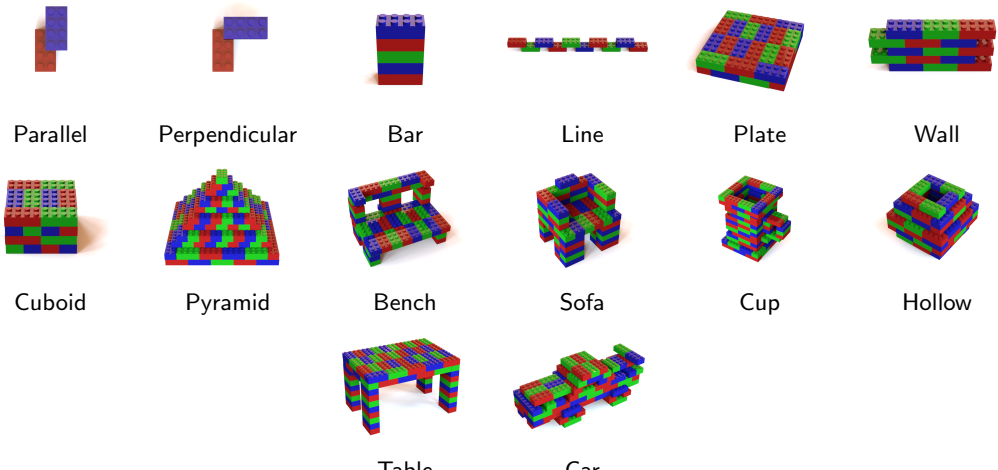


Figure 7: Selected examples from our dataset.

Takeaway

- Bayesian optimization is a powerful method to optimize a black-box function.
- Instead of methods based on heuristic or prior knowledge, it provides a structured approach to find an optimal solution.
- Bayesian optimization is expanding into various real-world applications.

Thank you.

References I

M. Balandat, B. Karrer, D. R. Jiang, S. Daulton, B. Letham, A. G. Wilson, and E. Bakshy. BoTorch: A framework for efficient Monte-Carlo Bayesian optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, Virtual, 2020.

R. Baptista and M. Poloczek. Bayesian optimization of combinatorial structures. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 462–471, Stockholm, Sweden, 2018.

J. Bergstra and Y. Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13:281–305, 2012.

P. Buathong, D. Ginsbourger, and T. Krityakierne. Kernels over sets of finite sets using RKHS embeddings, with application to Bayesian (combinatorial) optimization. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 2731–2741, Virtual, 2020.

J. Cui and B. Yang. Graph Bayesian optimization: Algorithms, evaluations and applications. *arXiv preprint arXiv:1805.01157*, 2018.

P. I. Frazier, W. B. Powell, and S. Dayanik. The knowledge-gradient policy for correlated normal beliefs. *INFORMS Journal on Computing*, 21(4): 599–613, 2009.

M. Garnelo, D. Rosenbaum, C. J. Maddison, T. Ramalho, D. Saxton, M. Shanahan, Y. W. Teh, D. J. Rezende, and S. M. A. Eslami. Conditional neural processes. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 1690–1699, Stockholm, Sweden, 2018a.

M. Garnelo, J. Schwarz, D. Rosenbaum, F. Viola, D. J. Rezende, S. M. A. Eslami, and Y. W. Teh. Neural processes. *arXiv preprint arXiv:1807.01622*, 2018b.

P. Hennig and C. J. Schuler. Entropy search for information-efficient global optimization. *Journal of Machine Learning Research*, 13:1809–1837, 2012.

F. Hutter, H. H. Hoos, and K. Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *Proceedings of the International Conference on Learning and Intelligent Optimization (LION)*, pages 507–523, Rome, Italy, 2011.

J. Kim and S. Choi. BayesO: A Bayesian optimization framework in Python. `https://bayeso.org`, 2017.

J. Kim and S. Choi. Clustering-guided GP-UCB for Bayesian optimization. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 2461–2465, Calgary, Alberta, Canada, 2018.

J. Kim and S. Choi. On local optimizers of acquisition functions in Bayesian optimization. In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD)*, pages 675–690, Virtual, 2020.

J. Kim, S. Kim, and S. Choi. Learning to warm-start Bayesian hyperparameter optimization. *arXiv preprint arXiv:1710.06219*, 2017.

References II

J. Kim, H. Chung, J. Lee, M. Cho, and J. Park. Combinatorial 3D shape generation via sequential assembly. In *Neural Information Processing Systems Workshop on Machine Learning for Engineering Modeling, Simulation, and Design (ML4Eng)*, Virtual, 2020.

J. Kim, M. McCourt, T. You, S. Kim, and S. Choi. Bayesian optimization with approximate set kernels. *Machine Learning*, 2021.

N. Knudde, J. van der Herten, T. Dhaene, and I. Couckuyt. GPflowOpt: A Bayesian optimization library using TensorFlow. *arXiv preprint arXiv:1711.03845*, 2017.

H. J. Kushner. A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise. *Journal of Basic Engineering*, 86(1):97–106, 1964.

J. Lee, Y. Lee, J. Kim, E. Yang, S. J. Hwang, and Y. W. Teh. Bootstrapping neural processes. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 6606–6615, Virtual, 2020.

R. Martinez-Cantin. BayesOpt: A Bayesian optimization library for nonlinear optimization, experimental design and bandits. *Journal of Machine Learning Research*, 15:3735–3739, 2014.

J. Močkus, V. Tiesis, and A. Žilinskas. The application of Bayesian methods for seeking the extremum. *Towards Global Optimization*, 2:117–129, 1978.

C. Oh, J. Tomczak, E. Gavves, and M. Welling. Combinatorial Bayesian optimization using the graph cartesian product. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, pages 2914–2924, Vancouver, British Columbia, Canada, 2019.

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

E. Real, C. Liang, D. R. So, and Q. V. Le. AutoML-Zero: evolving machine learning algorithms from scratch. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 8007–8019, Virtual, 2020.

T. Shiraishi, T. Le, H. Kashima, and M. Yamada. Topological Bayesian optimization with persistence diagrams. In *Proceedings of the European Conference on Artificial Intelligence (ECAI)*, pages 1483–1490, Virtual, 2020.

J. T. Springenberg, A. Klein, S. Falkner, and F. Hutter. Bayesian optimization with robust Bayesian neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 29, pages 4134–4142, Barcelona, Spain, 2016.

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

The GPyOpt authors. GPyOpt: A Bayesian optimization framework in Python. `https://github.com/SheffieldML/GPyOpt`, 2016.

B. Zoph and Q. V. Le. Neural architecture search with reinforcement learning. In *Proceedings of the International Conference on Learning Representations (ICLR)*, Toulon, France, 2017.