COMBO: An Efficient Bayesian Optimization Library for Materials Science

Tsuyoshi Ueno[a], Trevor David Rhone[b], Zhufeng Hou[c], Teruyasu Mizoguchi[d], Koji Tsuda[a,c,*]

[a]*Department of Computational Biology and Medical Sciences, Graduate School of Frontier Sciences, The University of Tokyo, 5-1-5-CB02 Kashiwanoha, Kashiwa 277-8561, Japan*
[b]*Department of Physics, Harvard University, 17 Oxford Street, Cambridge, Massachusetts 02138, USA*
[c]*National Institute for Materials Science, 1-2-1 Sengen, Tsukuba 305-0047, Japan*
[d]*Institute of Industrial Science, The University of Tokyo, 4-6-1, Komaba, Meguro, Tokyo 153-8505, Japan*

Abstract

In many subfields of chemistry and physics, numerous attempts have been made to accelerate scientific discovery by data-driven experimental design algorithms. Among them, Bayesian optimization has been proven as an effective tool. A standard implementation (e.g., scikit-learn), however, can accommodate only small training data. We designed an efficient protocol for Bayesian optimization that employs Thompson sampling, random feature maps, one-rank Cholesky update and automatic hyperparameter tuning, and implemented it as an open-source python library called COMBO (COMmon Bayesian Optimization library). Promising results are presented in determination of atomic structure of a crystalline interface. COMBO is available at `https://github.com/tsudalab/combo`.

Keywords: Bayesian Optimization, Python Library, Global Optimization, Materials Design.

1. Introduction

Algorithmic design of experiments appears in many different contexts in chemistry and physics [1, 2, 3, 4, 5]. The scientific process of discovering new knowledge is often characterized as *search* from a space of candidates. The candidates are either stored in a database or generated on the fly. For example, in molecular structure optimization, one is required to find the most stable structure in all possible structures [2]. In the first steps of drug discovery,

*Corresponding author
Email address: `tsuda@k.u-tokyo.ac.jp` (Koji Tsuda)

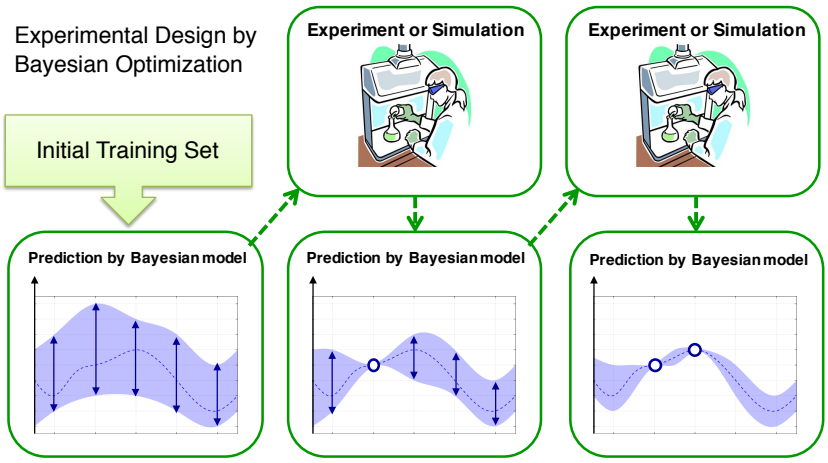

Figure 1: Bayesian optimization. Initially, the black-box function is estimated from a training set. Based on the prediction by a Bayesian model, the next point to apply an experiment on is chosen from a set of candidate points. An experiment or simulation is performed to obtain the value of the point. The obtained value is then added to the training set, the model is updated, and the next point is chosen. This procedure is repeated until the predetermined number of experiments are done.

the compounds binding to a target protein are seeked from numerous candidates [1, 3]. Experimental design is an iterative process to choose next ones to apply experiments, where the outcome of the experiments are exploited to make further choices. In many cases, experiments are substituted by simulators, e.g., first-principles calculation and docking simulators. Mathematically, it is often formulated as an optimization problem of a black-box function [6].

Bayesian optimization (aka kriging) is a well-established technique for black-box optimization [7, 8, 6]. Bayesian prediction models, most commonly Gaussian processes [9], are employed to predict the black-box function, where the uncertainty of the predicted function is also evaluated as predictive variance. Next candidates for experiments are chosen based on the predicted values and variances (Figure 1). It particularly gained momentum in machine learning research due to success in hyperparameter tuning in deep learning algorithms [7].

In materials science, we are aware of at least four successful applications of Bayesian optimization: elastic properties [10], melting temperature [4], lattice thermal conductivity [5], and the design of a grain boundary interface [11]. Despite their successes, the size of training set is still small (i.e., several hundreds at most). For upcoming large scale problems, the size of training set may grow up to millions. Existing packages (e.g., scikit-learn [12]) are hopelessly slow in this case, as shown later by numerical experiments.

We designed a new efficient protocol involving state-of-the-art machine learning techniques for optimal efficiency, and created an open source library termed COMmon Bayesian Optimization Library (COMBO). COMBO is amenable to

large scale problems, because the computational time grows only linearly as the number of candidates increases. This library frees materials scientists from the need to implement complex machine learning algorithms and has a potential to become a central tool in the emerging field of materials informatics.

2. Technical Features and Details

A black-box optimization problem is defined as follows. Let $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m \in \Re^d$ denote a set of candidate points represented as d-dimensional vectors. By an experiment, the value of the black-box function at a candidate point can be obtained. We would like to identify the points with maximum values using as few experiments as possible. In some cases, the user may aim to find the best from the entire set of candidate points. In most cases, however, it is sufficient to discover at least one point from top-k points.

In sequential experimental design, it is assumed that the values $y_1, \ldots, y_n$ are already measured for n candidate points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$. Based on a prediction model learned from the training set $D = \{\boldsymbol{x}_i, y_i\}_{i=1}^n$, an algorithm determines which one to apply an experiment on from the remaining $m - n$ points. This process is repeated until the predetermined number of experiments are done.

2.1. Thompson Sampling

Thompson sampling is a popular algorithm for sequential experimental design [8]. The basic idea of Thompson sampling is *probability matching*, namely choosing the candidate point according to the probability of being optimal. A prediction model is required to facilitate Thompson sampling. Here we employ the Bayesian linear regression model,

$$y = \boldsymbol{w}^\top \phi(\boldsymbol{x}) + \epsilon,$$

where $\boldsymbol{x} \in \Re^d$ is an input vector corresponding to a candidate point, $\phi : \Re^d \to \Re^\ell$ is a feature map, $\boldsymbol{w} \in \Re^\ell$ is a weight vector and ϵ is the noise subject to $\mathcal{N}(0, \sigma^2)$. Let Φ denote the $\ell \times n$ matrix whose i-th column corresponds to $\phi(\boldsymbol{x}_i)$. The posterior distribution of $\boldsymbol{w}$ given data D is described as

$$\boldsymbol{w} \mid D \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$$

where

$$\boldsymbol{\mu} = (\Phi\Phi^\top + \sigma^2 I)^{-1}\Phi\boldsymbol{y}, \;\; \Sigma = \sigma^2(\Phi\Phi^\top + \sigma^2 I)^{-1}. \tag{1}$$

The predicted value for candidate point $\boldsymbol{x}_i$ is described as $\boldsymbol{w}^\top\phi(\boldsymbol{x}_i)$. The region of $\boldsymbol{w}$ where choosing $\boldsymbol{x}_i$ is optimal is defined as

$$W_i = \{\boldsymbol{w} \in \Re^\ell \mid \boldsymbol{w}^\top\phi(\boldsymbol{x}_i) = \max_j \boldsymbol{w}^\top\phi(\boldsymbol{x}_j)\}.$$

Thompson sampling chooses the candidate point according to the probability of being optimal, $p_i = P(\boldsymbol{w} \in W_i \mid D)$. It can be done without computing p_i as follows [8].

- Sample a vector $\boldsymbol{s}$ from $P(\boldsymbol{w} \mid D)$.
- Determine the candidate point with maximum score with respect to $\boldsymbol{s}$,

$$i^* = \operatorname*{argmin}_i \boldsymbol{s}^\top \boldsymbol{x}_i. \tag{2}$$

Thompson sampling for the Bayesian linear model is particularly efficient, because the computational time for evaluating a candidate point (2) is $O(\ell)$. In comparison, other methods such as maximum probability of improvement [4] and maximum expected improvement [6] require us to compute the predictive variance, which takes $O(\ell^2)$ time.

2.2. Quick Sampling

Sampling from a multidimensional Gaussian distribution requires a triangular decomposition of the covariance matrix [13]. In our case, we need to sample a vector $\boldsymbol{s}$ from the posterior distribution $p(\boldsymbol{w} \mid D)$. Defining

$$A = \frac{1}{\sigma^2}\Phi\Phi^\top + I,$$

the posterior distribution is described as $\boldsymbol{w} \mid D \sim \mathcal{N}(\frac{1}{\sigma^2}A^{-1}\Phi\boldsymbol{y}, A^{-1})$. Thus we need a triangular decomposition of A^{-1} in each step of Thompson sampling. After an experiment is done, a new training example $(\boldsymbol{x}', y')$ is obtained, and the matrix A is updated as

$$A' = A + \frac{1}{\sigma^2}\phi(\boldsymbol{x}')\phi(\boldsymbol{x}')^\top.$$

It is well known that Cholesky decomposition of a one-rank-modified matrix is computed from that of the original matrix in $O(\ell^2)$ time (i.e., fast-update) [14], while computing it from scratch takes $O(\ell^3)$ time. So we maintain Cholesky decomposition L of A throughout the process (i.e., $A = L^\top L$) and apply fast-update to L whenever a new example is added.

The sample $\boldsymbol{s}$ is obtained as $\boldsymbol{\mu} + \boldsymbol{s}_0$ where $\boldsymbol{s}_0$ is a sample from $\mathcal{N}(0, A^{-1})$. Substituting $A = L^\top L$ to (1), the mean vector $\boldsymbol{\mu}$ is described as the solution of the following equation,

$$L^\top L\boldsymbol{\mu} = \frac{1}{\sigma^2}\Phi\boldsymbol{y}.$$

The solution is obtained by applying back substitution twice, first for $L^\top$ and second for L [13]. Also, $\boldsymbol{s}_0$ is obtained by sampling $\boldsymbol{z} \sim N(0, I)$ and then solving the linear equation $\boldsymbol{z} = L\boldsymbol{s}_0$ by back substitution. The sampling process is done in $O(\ell^2)$ time.

2.3. Random Feature Map

The choice of feature map ϕ crucially affects the performance of Bayesian optimization. In COMBO, we employ a random feature map that approximates

the mapping induced by Gaussian kernel [15]. Let us describe the Gaussian kernel as $k(\Delta) = \exp(-\|\Delta\|^2/2)$. Due to Bochner's theorem, it is rewritten as

$$k(\boldsymbol{x} - \boldsymbol{x}') = \int \exp(j\boldsymbol{\omega}^\top(\boldsymbol{x} - \boldsymbol{x}'))p(\boldsymbol{\omega})d\boldsymbol{\omega},$$

where j is the imaginary unit and

$$p(\boldsymbol{\omega}) = (2\pi)^{-d/2}\exp(-\|\boldsymbol{\omega}\|^2/2).$$

Let us define $z_{\boldsymbol{\omega},b}(\boldsymbol{x}) = \sqrt{2}\cos(\boldsymbol{\omega}^\top\boldsymbol{x}+b)$. If $\boldsymbol{\omega}$ is drawn from $p(\boldsymbol{\omega})$ and b is drawn uniformly from $[0, 2\pi]$, we have $E[z_{\boldsymbol{\omega},b}(\boldsymbol{x})z_{\boldsymbol{\omega},b}(\boldsymbol{x}')] = k(\boldsymbol{x} - \boldsymbol{x}')$. In COMBO, the feature map is defined using ℓ random samples $\{\boldsymbol{\omega}_i, b_i\}_{i=1}^{\ell}$ as

$$\phi(\boldsymbol{x}) = (z_{\boldsymbol{\omega}_1,b_1}(\boldsymbol{x}), \ldots, z_{\boldsymbol{\omega}_\ell,b_\ell}(\boldsymbol{x}))^\top.$$

Then, the inner product in the feature space is approximately equal to the Gaussian kernel. The width of Gaussian kernel can be changed by rescaling vectors $\boldsymbol{x}$. In the limit $\ell \to \infty$, the Bayesian linear model with the random feature map converges to a Gaussian process.

3. Benchmarking

COMBO is applied to optimize rigid body translation of the $\Sigma5[001](210)$ simplified coincidence lattice grain boundary of face-centered cubic copper [11]. From 17,983 grid points of translation, we need to find the best translation to yield lowest grain boundary energy. A candidate point $\boldsymbol{x}_i$ is a three dimensional vector describing the translation along x,y and z-axis. The corresponding value y_i is the grain boundary energy obtained by a static lattice calculation using an empirical potential method [16]. See [11] for details about the calculation.

To measure the performance, a 'success' is defined as finding at least one point in top-k points within 300 experiments, where an experiment in this case corresponds to a static lattice calculation. Initial 20 experiments are chosen randomly and the remaining ones are designed by COMBO. The number of features ℓ in the random feature map is set to 2,000 and 5,000. In each case, COMBO is applied 30 times with different initial experiments. The success probability of COMBO at different values of k is shown in Figure 2. The dotted line shows the success probability of random design, analytically computed using hypergeometric distributions. Compared to random design, COMBO was far more effective in finding top points. At $k = 30$, for instance, the success probability of random design was 40%. It is enhanced by COMBO to 83% ($\ell = 2000$) and 90% ($\ell = 5000$).

Figure 3 compares computational time of scikit-learn and COMBO. The computational time of scikit-learn showed quadratic growth, whereas that of COMBO grew linearly. For very small problems, scikit-learn was faster, but COMBO was far more efficient for large-scale problems as expected.

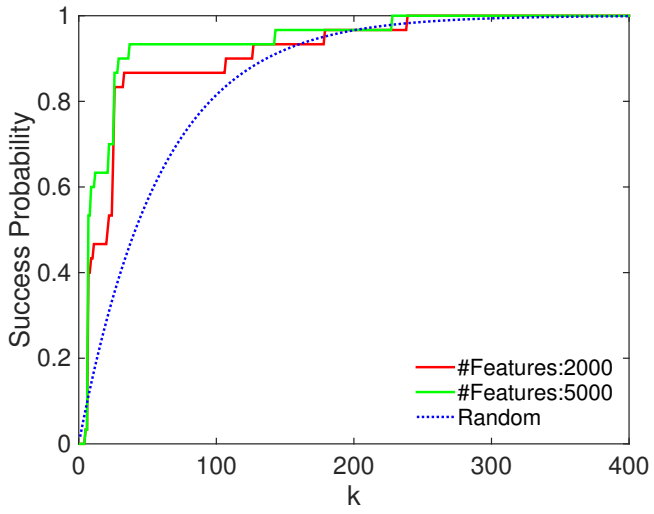

Figure 2: Success probability in 300 experiments. Red and green solid lines indicate the results of COMBO with 2,000 and 5,000 features, respectively. Blue dotted line corresponds to that of random design.

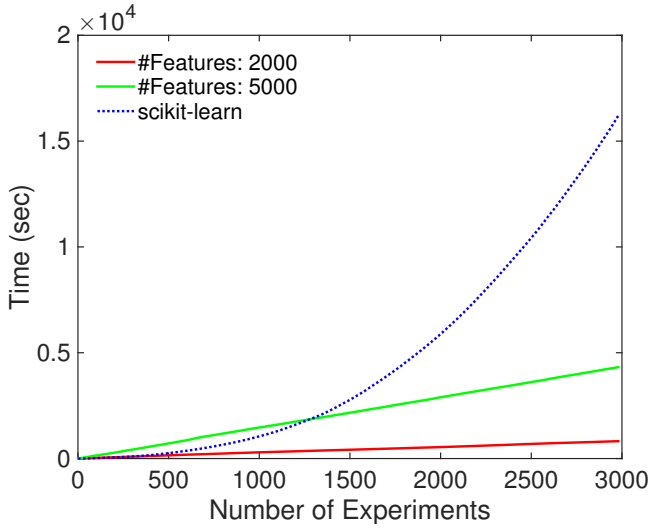

Figure 3: Computational time. Red and green solid lines indicate the results of COMBO with 2,000 and 5,000 features, respectively. Blue dotted line corresponds to that of scikit-learn.

4. Discussion and Conclusion

In addition to efficiency, COMBO has several functions for users' convenience. Firstly, COMBO has a interface function that users can easily modify for using their own simulators. Secondly, hyperparameters of a prediction model such as kernel width and noise variance can be automatically learned from data by maximizing type-II likelihood [9]. As the number of training samples increases, hyperparameters can be more accurately controlled. COMBO is designed to repeatedly update the hyperparameters with a user-specified interval.

COMBO will be further developed to include more models and functions, and everyone is invited to join the development in github. As a python package, it is easy to combine with other materials informatics packages such as pymatgen[17].

Acknowledgement

This study is supported by RIKEN PostK, NIMS MI2I, Kakenhi Nanostructure, Kakenhi 15H05711, JST CREST and JST ERATO.

References

[1] D. Reker, G. Schneider, Active-learning strategies in computer-assisted drug discovery, Drug Discov. Today 20 (2015) 458–465.

[2] A. R. Oganov, C. W. Glass, Crystal structure prediction using ab initio evolutionary techniques: Principles and applications, J. Chem. Phys. 124 (2006) 244704.

[3] M. Ahmadi, M. Vogt, P. Iyer, J. Bajorath, H. Fröhlich, Predicting potent compounds via model-based global optimization, J. Chem. Inf. Model. 53 (2013) 553–559.

[4] A. Seko, T. Maekawa, K. Tsuda, I. Tanaka, Machine learning with systematic density-functional theory calculations: Application to melting temperatures of single-and binary-component solids, Phys. Rev. B 89 (2014) 054303.

[5] A. Seko, A. Togo, H. Hayashi, K. Tsuda, L. Chaput, I. Tanaka, Prediction of low-thermal-conductivity compounds with first-principles anharmonic lattice-dynamics calculations and bayesian optimization, Phys. Rev. Lett. 115 (2015) 205901.

[6] D. R. Jones, M. Schonlau, W. J. Welch, Efficient global optimization of expensive black-box functions, J. Glob. Optim. 13 (1998) 455–492.

[7] J. Snoek, H. Larochelle, R. P. Adams, Practical bayesian optimization of machine learning algorithms, in: Advances in neural information processing systems, 2012, pp. 2951–2959.

[8] O. Chapelle, L. Li, An empirical evaluation of thompson sampling, in: Advances in neural information processing systems, 2011, pp. 2249–2257.

[9] C. E. Rasmussen, C. K. I. Williams, Gaussian processes for machine learning, MIT Press, Cambridge, Mass., 2006.

[10] P. V. Balachandran, D. Xue, J. Theiler, J. Hogden, T. Lookman, Adaptive strategies for materials design using uncertainties, Sci. Rep. 6 (2016) 19660.

[11] S. Kiyohara, H. Oda, K. Tsuda, T. Mizoguchi, Acceleration of stable interface structure searching using a kriging approach, Jpn. J. Appl. Phys., in press, (2016).

[12] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al., Scikit-learn: Machine learning in python, J. Mach. Learn. Res. 12 (2011) 2825–2830.

[13] G. H. Golub, C. F. Van Loan, Matrix computations, volume 3, JHU Press, 2012.

[14] P. E. Gill, G. H. Golub, W. Murray, M. A. Saunders, Methods for modifying matrix factorizations, Math. Comput. 28 (1974) 505–535.

[15] A. Rahimi, B. Recht, Random features for large-scale kernel machines, in: Advances in neural information processing systems, 2007, pp. 1177–1184.

[16] J. D. Gale, Gulp: A computer program for the symmetry-adapted simulation of solids, J. Chem. Soc. Faraday Trans. 93 (1997) 629–637.

[17] S. P. Ong, W. D. Richards, A. Jain, G. Hautier, M. Kocher, S. Cholia, D. Gunter, V. L. Chevrier, K. A. Persson, G. Ceder, Python materials genomics (pymatgen): A robust, open-source python library for materials analysis, Comp. Mater. Sci. 68 (2013) 314–319.