Materials Discovery 4 (2016) 18–21

Contents lists available at ScienceDirect

Materials Discovery

journal homepage: www.elsevier.com/locate/md

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, MA 02138, USA
[c] Center for Materials Research by Information Integration, National Institute for Materials Science, 1-2-1 Sengen, Tsukuba, Ibaraki 305-0047, Japan
[d] Institute of Industrial Science, The University of Tokyo, 4-6-1, Komaba, Meguro, Tokyo 153-8505, Japan

ARTICLE INFO

Article history:
Received 15 February 2016
Received in revised form 29 March 2016
Accepted 2 April 2016
Available online 21 June 2016

Keywords:
Bayesian optimization
Python library
Global optimization
Materials design

ABSTRACT

In many subfields of chemistry and physics, numerous attempts have been made to accelerate scientific discovery using data-driven experimental design algorithms. Among them, Bayesian optimization has been proven to be 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 using COMBO to determine the atomic structure of a crystalline interface are presented. COMBO is available at https://github.com/tsudalab/combo.

© 2016 Elsevier Ltd. All rights reserved.

1. Introduction

Algorithmic design of experiments appears in many different contexts in chemistry and physics [1–5]. The scientific process of discovering new knowledge is often characterized as *search* through a space of candidate hypotheses. 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 out of all possible structures [2]. In the first steps of drug discovery, the potential compounds which could bind to a target protein are sought from numerous candidates [1,3]. Experimental design is an iterative process for selecting the next candidates for experiments, where the outcome of the experiments are exploited for making further choices. In many cases, experiments are substituted by calculations, e.g., first-principles calculations 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 [6–8]. 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. The next candidates for experiments are chosen based on the predicted values and variances (Fig. 1). Bayesian optimization gained momentum in machine learning research due to the success of 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]. However, in these successful applications the size of the training set was small (i.e., several hundreds at most). For upcoming large scale problems, the size of the training set may increase to the millions. Existing packages (e.g., scikit-learn [12]) are hopelessly slow in this case, as will be 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.

* Corresponding author at: 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.
E-mail address: tsuda@k.u-tokyo.ac.jp (K. Tsuda).

http://dx.doi.org/10.1016/j.md.2016.04.001
2352-9245/© 2016 Elsevier Ltd. All rights reserved.

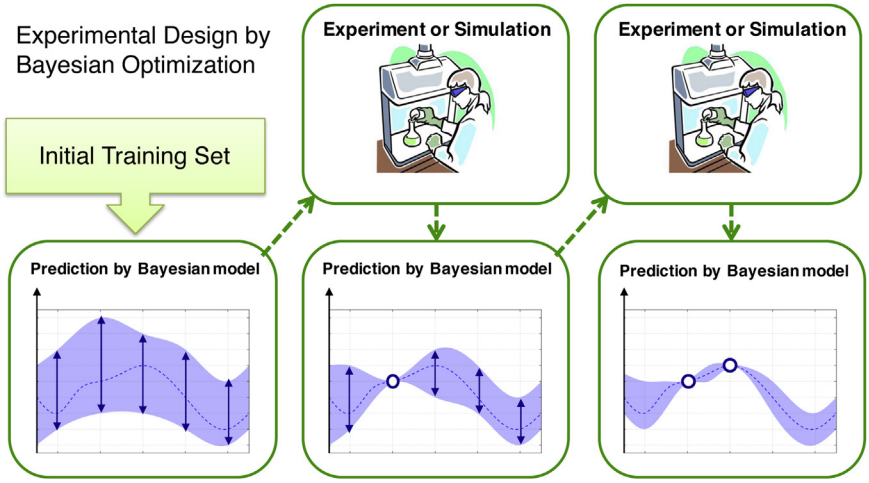

Fig. 1. Bayesian optimization. Initially, the black-box function is estimated from a training set. Based on the prediction by a Bayesian model, the next point on which to apply an experiment is chosen from a set of candidate points. An experiment or simulation is performed to obtain the value of the 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.

Technical features include Thompson sampling [8], one-rank Cholesky update [13], random feature maps [14] and automatic hyperparameter tuning [9]. Thompson sampling does not require costly computation of predictive variance. Cholesky decomposition required in Thompson sampling is made efficient by one-rank Cholesky update. A random feature map allows us to approximate a Gaussian process with a Bayesian linear model that is more efficient for learning. By sacrificing a little loss of efficiency, users can enable automatic hyperparameter tuning using type-II likelihood maximization. This library frees materials scientists from the need to implement complex machine learning algorithms and has the potential to become a key 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. By means of Bayesian optimization, we would like to identify the points with minimum values using as few experiments as possible.

In Ref. [11], for instance, Bayesian optimization was applied to determine the most stable translation parameters of a grain boundary. In this case, the dimensionality of candidate points d is three. The candidate points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m \in \Re^3$ are created by splitting the ranges of x, y, z-axes into equal intervals. If each axis is divided into u intervals, the number of candidate points turns out to be $m = (u+1)^3$. The value of the black box function y_i is defined as the grain boundary energy at translation $\boldsymbol{x}_i$ obtained by a static lattice calculation [15]. So, a calculation plays the role of an experiment here.

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 candidate point on which to apply an experiment 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 ith 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

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

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

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

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 minimum 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 [16].

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 $\boldsymbol{w} \mid D \sim \mathcal{N}((1/\sigma^2) A^{-1}\Phi\boldsymbol{y}, A^{-1})$. 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) [13], 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 [16]. 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 the Gaussian kernel [14]. Let us describe the Gaussian kernel of unit width 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

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

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 such that the inner product $\phi(\boldsymbol{x})^\top\phi(\boldsymbol{x}')$ approximates the Gaussian kernel of width η, i.e., $\exp(-((\|\boldsymbol{x} - \boldsymbol{x}'\|^2)(2\eta^2)))$. Using ℓ random samples $\{\boldsymbol{\omega}_i, b_i\}_{i=1}^{\ell}$, the feature map is defined as

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

In the limit $\ell \to \infty$, the Bayesian linear model with the random feature map converges to a Gaussian process.

2.4. Hyperparameter tuning

The manual determination of hyperparameters σ and η is often tricky for non-experts. COMBO has the capability to set the hyperparameters automatically by maximizing the type-II likelihood [9]. The type-II likelihood is described as $p(D|\sigma, \eta)$, i.e., the parameters except σ and η are marginalized out by means of the prior distribution. Before optimization, the hyperparameters are initialized according to a heuristic procedure described in Ref. [17]. An online optimization algorithm ADAM [18] is applied to maximize the type-II likelihood with respect to the hyperparameters, where the gradient is approximated by subsampling.

Hyperparameter tuning is repeatedly applied in the sequential experimental design process. In the beginning, n_I candidate points are selected randomly to create the initial set of training examples. At this point in time, the hyperparameters are tuned for the first time. They are retuned every time n_E candidate points are added. Notice that hyperparameter tuning can take more computational time than the Bayesian optimization itself. Users can configure n_I and n_E to balance time and accuracy.

3. Benchmarking

COMBO is applied to optimize rigid body translation of the $\Sigma 5[001](210)$ simplified coincidence lattice grain boundary of face-centered cubic copper [11]. As already mentioned in Section 2, a candidate point $\boldsymbol{x}_i$ is a three dimensional vector describing the translation along x-, y- and z-axes. The corresponding value y_i is the grain boundary energy obtained by a static lattice calculation using an empirical potential method [15]. See Ref. [11] for details about the calculation.

In this study, the number of total experiments is fixed to 300. Among them, the first 20 points are chosen randomly. The number of features ℓ in the random feature map is set to 2000 and 5000, where a larger ℓ leads to a more accurate approximation of the Gaussian process but consumes more time. We have 16,983 candidate points in total. For benchmarking purposes, the grain boundary energies are pre-computed for all points. Top-k points are defined as the ones of lowest energy.

COMBO was applied 30 times with different random seeds. The success probability for top-k points is defined as the fraction of COMBO applications that succeeded in finding at least one of the top-k points. Fig. 2 shows the success probability for different values of k. 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% ($\ell = 5000$).

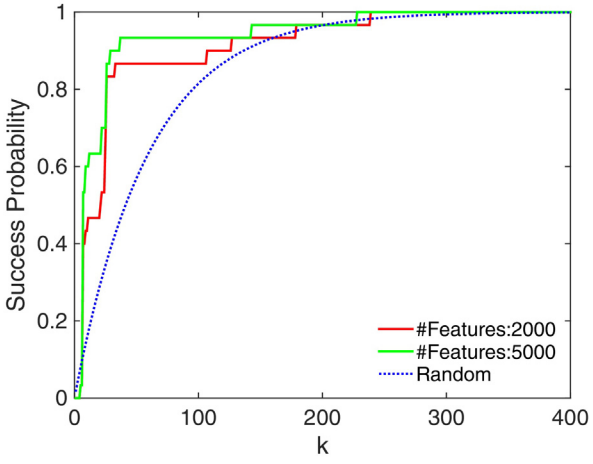

Fig. 2. Success probability in 300 experiments. Red and green solid lines indicate the results of COMBO with 2000 and 5000 features, respectively. Blue dotted line corresponds to that of random design. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of the article.)

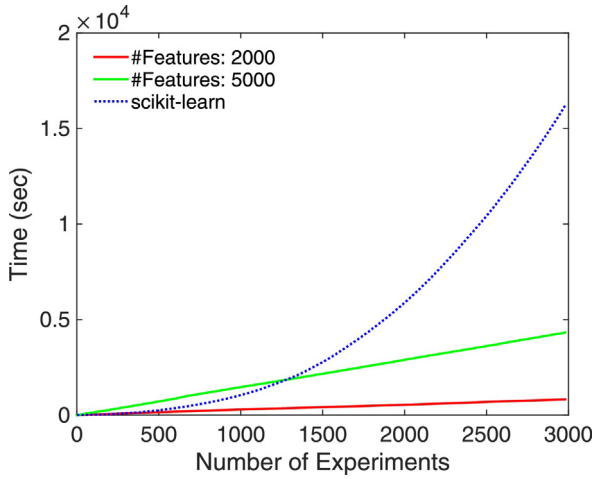

Fig. 3. Computational time. Red and green solid lines indicate the results of COMBO with 2000 and 5000 features, respectively. Blue dotted line corresponds to that of scikit-learn. Hyperparameter tuning was not employed in this experiment. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of the article.)

Fig. 3 compares computational time of scikit-learn and COMBO. Hyperparameter tuning was turned off in this experiment. 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.

4. Conclusion

Bayesian optimization can be applied to various kinds of problems. A prerequisite, however, is that each candidate point is represented as a numerical vector of identical dimensionality (i.e., descriptors). In grain boundary optimization, the x, y, z-coordinates are natural descriptors. For other applications such as crystal structures, the choice of descriptors may not be obvious [5].

COMBO will be further developed to include more models and functions. Everyone is invited to join the development in github. As a python package, it will be easily combined with other materials informatics packages such as pymatgen [19] and ASE [20].

Acknowledgements

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

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, Adv Neural Inf. Process. Syst. (2012) 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. 55 (2016) 045502.
[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] P.E. Gill, G.H. Golub, W. Murray, M.A. Saunders, Methods for modifying matrix factorizations, Math. Comput. 28 (1974) 505–535.
[14] A. Rahimi, B. Recht, Random features for large-scale kernel machines, in: Advances in Neural Information Processing Systems, 2007, pp. 1177–1184.
[15] J.D. Gale, Gulp A computer program for the symmetry-adapted simulation of solids, J. Chem. Soc. Faraday Trans. 93 (1997) 629–637.
[16] G.H. Golub, C.F. Van Loan, Matrix Computations, vol. 3, JHU Press, 2012.
[17] Z. Yang, A. Wilson, A. Smola, L. Song, A la Carte-Learning Fast Kernels, in: Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics, 2015, pp. 1098–1106.
[18] D. Kingma, J. Ba, Adam A Method for Stochastic Optimization, 2014 arXiv:1412.6980.
[19] 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, Comput. Mater. Sci. 68 (2013) 314–319.
[20] S.R. Bahn, K.W. Jacobsen, An object-oriented scripting interface to a legacy electronic structure code, Comput. Sci. Eng. 4 (2002) 56–66.