ECAI 2020
G.D. Giacomo et al. (Eds.)

doi:10.3233/FAIA200255

Topological Bayesian Optimization with Persistence Diagrams

Tatsuya Shiraishi[1] and **Tam Le**[2] and **Hisashi Kashima**[3] and **Makoto Yamada**[4]

Abstract. Finding an optimal parameter of a black-box function is important for searching stable material structures and optimal neural network structures, and Bayesian optimization algorithms are widely used for the purpose. However, most of existing Bayesian optimization algorithms can only handle vector data and cannot handle complex structured data. In this paper, we propose the topological Bayesian optimization, which can efficiently find an optimal solution from structured data using *topological information*. More specifically, in order to apply Bayesian optimization to structured data, we extract useful topological information from a structure and measure the proper similarity between structures. To this end, we utilize persistent homology, which is a topological data analysis method that was recently applied in machine learning. Moreover, we propose the Bayesian optimization algorithm that can handle multiple types of topological information by using a linear combination of kernels for persistence diagrams. Through experiments, we show that topological information extracted by persistent homology contributes to a more efficient search for optimal structures compared to the random search baseline and the graph Bayesian optimization algorithm.

1 INTRODUCTION

In recent years, many studies have been actively conducted on the analysis of data with complex structures like graph structures. Graph structure optimization involves searching for graph structures with optimal properties, and it is one of the fundamental tasks in graph structured data analysis. Examples of graph structure optimization include searching for stable lowest-energy crystal structures [23] and searching for road networks with optimal traffic volume [7]. Another example of graph structure optimization would be neural network architecture search [19, 11], which is an important task in deep learning architecture research. Thus, learning from complex structure is very important in various research fields.

The objective function of graph structure optimization (e.g., energy of a crystal structure and traffic volume of a road network) is an expensive-to-evaluate function, which needs to be measured by performing a long time experiment or a large scale investigation, and is a black-box function, which cannot be written explicitly. Therefore, an optimization method that can optimize even an unknown objective function with fewer evaluations is desirable. Bayesian optimization is one of methods that satisfies this condition. However, studies on Bayesian optimization often assume vector data as the input, and few studies focus on structured data. In standard Bayesian optimization methods, we tend to use the Gaussian kernel, which expresses the similarity between input vectors. Thus, to handle structured data by Bayesian optimization, we need to design a similarity that properly captures the structure. For example, a method using graph kernels was proposed for handling arbitrary graph structures by Bayesian optimization [5], and this method outperforms vector based Bayesian optimization in tasks such as identifying the most active node in a social network and searching for optimal transportation networks.

Recently, the topological data analysis (TDA) has received considerable attention in machine learning as a technique for extracting topological features from complex structured data. Persistent homology is a TDA method that is actively studied for application to statistical machine learning. The result of persistent homology is represented by a point cloud on $\mathbb{R}^2$ like Figure 2 called a persistence diagram (PD). As one of the applications of persistent homology to machine learning, several kernels for PD have been proposed, and the effectiveness has been demonstrated by classification tasks using the support vector machines and change point detection [12, 14]. However, to the best of our knowledge, there is no Bayesian optimization method that utilizes topological data analysis.

In this paper, we propose the topological Bayesian optimization, which is a Bayesian optimization algorithm using features extracted by persistent homology. More specifically, we first introduce the persistence weighted Gaussian kernel (PWGK) [12] and the persistence Fisher kernel (PFK) [14] for Gaussian processes, and derive a Bayesian optimization algorithm using topological information. Since the current persistence homology based approach considers only one type of topological information, it may not be able to capture various types of topological information. Therefore, we further propose a multiple kernel learning based algorithm and apply it to Bayesian optimization problems. Through experiments using synthetic and four real datasets about properties of molecules, we show that our method can search for the optimal structure more efficiently compared to the random search baseline and the state-of-the-art Bayesian optimization for graphs [5].

Contributions: The contributions of this paper are as follows:

- We propose a Bayesian optimization algorithm utilizing topological information extracted by persistent homology.
- We further propose a multiple kernel learning based algorithm to use various types of topological information.
- Through experiments, we show that our method can search for the optimal structure more efficiently compared to the random search baseline and the graph Bayesian optimization algorithms.

[1] Kyoto University, Japan, email: shiraishi.t@ml.ist.i.kyoto-u.ac.jp
[2] RIKEN AIP, Japan, email: tam.le@riken.jp
[3] Kyoto University/RIKEN AIP, Japan, email: kashima@i.kyoto-u.ac.jp
[4] Kyoto University/RIKEN AIP, Japan, email: myamada@i.kyoto-u.ac.jp

2 BACKGROUND

In this section, we briefly review the traditional Bayesian optimization algorithm based on Gaussian process and the topological data analysis based on persistent homology.

2.1 Bayesian optimization

Bayesian optimization (BO) is an effective optimization method for expensive-to-evaluate objective functions [1]. Let us denote the input vector $\boldsymbol{x} \in \mathbb{R}^d$ and a black box function $f : \mathbb{R}^d \to \mathbb{R}$. Bayesian optimization tries to find the optimal data point of the following optimization problem:

$$\boldsymbol{x}^* = \underset{\boldsymbol{x}\in\mathbb{R}^d}{\operatorname{argmin}} \ f(\boldsymbol{x}).$$

Since Bayesian optimization does not need derivatives of the objective function for finding the optimal data point, it is particularly effective when optimizing black-box objective functions. Bayesian optimization is an iterative method, and each step consists of two steps: (i) calculation of a predictive distribution of an objective function value by a Gaussian process and (ii) selection of the next search point based on an acquisition function.

Gaussian process: Gaussian process is a generalization of Gaussian probability distribution [20]. More specifically, Gaussian process describes the functions of random variables, while Gaussian probability distribution describes random scalars or vectors. In Bayesian optimization, the objective function $f : \mathbb{R}^d \to \mathbb{R}$ is modeled by a Gaussian process, which enables easy calculation of predictive distributions. Now, let $\{ (\boldsymbol{x}_1, y_1), \cdots, (\boldsymbol{x}_t, y_t) \}$ be pairs of the input and the corresponding output of the objective function observed up to a certain step, where $\boldsymbol{x}_i \in \mathbb{R}^d, y_i \in \mathbb{R}$ for $i = 1, \cdots, t$. We assume that the true value $f(\boldsymbol{x}_i)$ is not necessarily observed as y_i, but an independent additive Gaussian noise $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$ is included:

$$y_i = f(\boldsymbol{x}_i) + \epsilon_i.$$

According to the definition of Gaussian process, the joint probability distribution of $f(\boldsymbol{x}_1), \cdots, f(\boldsymbol{x}_t)$ is

$$(f(\boldsymbol{x}_1), \cdots, f(\boldsymbol{x}_t))^T \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{K}), \tag{1}$$

where $\boldsymbol{0} = (0, \cdots, 0)^T$, $\cdot^T$ denotes the transpose operator, and each element of $\boldsymbol{K} \in \mathbb{R}^{t\times t}$ is expressed by $K_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$ using the kernel function $k(\cdot, \cdot)$. Then, the predictive distribution of the function value $f(\boldsymbol{x}_{t+1})$ at the point $\boldsymbol{x}_{t+1}$, which is not included in the data, can be calculated. Since the joint probability distribution of $f(\boldsymbol{x}_1), \cdots, f(\boldsymbol{x}_t), f(\boldsymbol{x}_{t+1})$ is also expressed similar to the expression (i.e., Eq. (1)) and the additive noise is included in the observations, the predictive distribution of $f(\boldsymbol{x}_{t+1})$ is also a Gaussian distribution whose mean $\mu(\boldsymbol{x}_{t+1})$ and covariance $\sigma^2(\boldsymbol{x}_{t+1})$ can be calculated as follows:

$$\begin{aligned}\mu(\boldsymbol{x}_{t+1}) &= \boldsymbol{k}(\boldsymbol{K} + \sigma^2\boldsymbol{I})^{-1}\boldsymbol{y},\\ \sigma^2(\boldsymbol{x}_{t+1}) &= k(\boldsymbol{x}_{t+1}, \boldsymbol{x}_{t+1}) - \boldsymbol{k}(\boldsymbol{K} + \sigma^2\boldsymbol{I})^{-1}\boldsymbol{k}^T,\end{aligned}$$

where $\boldsymbol{k} = (k(\boldsymbol{x}_{t+1}, \boldsymbol{x}_1), \cdots, k(\boldsymbol{x}_{t+1}, \boldsymbol{x}_t)), \boldsymbol{y} = (y_1, \cdots, y_t)^T$. See [20] for the detailed derivation.

Acquisition function: The acquisition function $\mathrm{acq}(\boldsymbol{x})$ expresses the degree to which we should evaluate the input point $\boldsymbol{x}$ based on the predictive distribution calculated utilizing a Gaussian process. In Bayesian optimization, the point that maximizes the acquisition function is selected as the next evaluation point:

$$\boldsymbol{x}_{t+1} = \underset{\boldsymbol{x}\in\mathbb{R}^d}{\operatorname{argmax}} \ \mathrm{acq}(\boldsymbol{x}).$$

There are many acquisition functions including probability of improvement (PI) [13], expected improvement (EI) [16], and lower confidence bound (LCB) [22]. The balance between exploitation and exploration is important for acquisition functions. Exploitation involves evaluation of points in the surroundings of the point observed with the best objective function value, while exploration involves evaluation of points with high uncertainty. EI, which we use in the experiments, is the expected value of the difference between the best observation value y_{best} obtained up to a certain step and the predicted objective function value $f(\boldsymbol{x})$.

$$\begin{aligned}\mathrm{acq}_{EI}(\boldsymbol{x}) &= \mathbb{E}[\max\{0, f(\boldsymbol{x}) - y_{best}\}]\\ &= \begin{cases}\sigma(\boldsymbol{x})(Z\Phi(Z) + \phi(Z)) & \sigma(\boldsymbol{x}) \neq 0\\ 0 & \sigma(\boldsymbol{x}) = 0\end{cases},\end{aligned}$$

where $Z = \frac{\mu(\boldsymbol{x}) - y_{best}}{\sigma(\boldsymbol{x})}$, and Φ and ϕ are the cumulative density function and probability density function of a standard normal distribution, respectively.

2.2 TDA based on persistent homology

In TDA, we focus on the shapes of a complex data from the viewpoint of topology. Here, we give an intuitive explanation of one of the TDA methods, namely persistent homology [3, 6]. We analyze a point cloud $\{ \boldsymbol{x}_1, \cdots, \boldsymbol{x}_N \}$ on a metric space (M, c). In case of analysis of a compound, the point cloud may be the set of 3D coordinates of atoms forming the compound. In order to analyze this by persistent homology, we consider the union of balls centered on each point with radius r:

$$S_r = \bigcup_{i=1}^{N} \{ \boldsymbol{x} \in M \mid c(\boldsymbol{x}, \boldsymbol{x}_i) \leq r \}.$$

Figure 1 shows examples of S_r. We can observe that topological structures like connected components and rings appear and disappear while increasing r. In persistent homology, we focus on when each topological structure appears and how long it persists.

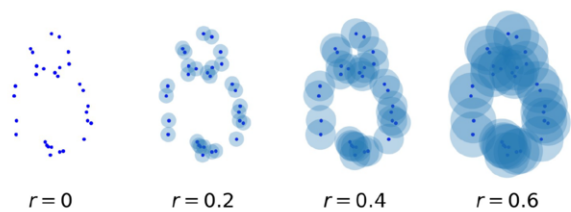

Figure 1. Examples of S_r. We can observe that topological structures like connected components and rings appear and disappear while increasing r.

The topological features extracted by persistent homology can be expressed as a point cloud on $\mathbb{R}^2$ called a persistence diagram (PD). A point (b, d) on a PD shows the corresponding topological structure that appears at radius b and disappears at radius d. b and d are

called birth and death of the structure, respectively, and $d-b$ is called persistence of the structure. Since $b < d$, all the points on a PD are distributed above the diagonal. We can consider multiple PDs for the same point cloud depending on the structure of interest. It is called the 0th PD when we focus on the connected components, the 1st PD when we focus on the rings and so on. Figure 2 shows the 0th PD and the 1st PD for the point cloud of Figure 1. The births of all points in the 0th PD are 0 because all connected components already exist at $r = 0$ and new one does not appear in the middle of increasing r. Two points corresponding to the small ring and the large ring in the point cloud can be seen in the 1st PD. The small ring corresponds to the point closer to the diagonal because it has smaller persistence than the large one. A point with small persistence is likely to be caused by noise. Thus, the points distributed near the diagonal may represent noisy structures, while the points distributed far from the diagonal may represent more important structures.

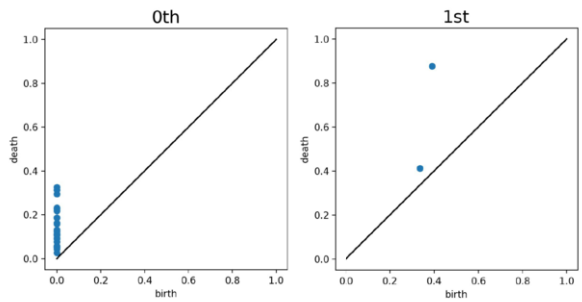

Figure 2. 0th and 1st PDs for the point cloud of Figure 1. The births of all points in the 0th PD are 0 because all connected components already exist at $r = 0$. Two points corresponding to the small ring and the large ring are in the 1st PD. The smaller ring corresponds to the point closer to the diagonal because it has smaller persistence than the large one. These points are often treated as noisy structures since small persistences are likely to be caused by noise in data (i.e. the point cloud of Figure 1).

In this section, we gave the explanation of persistent homology based on a fattened ball model which is applicable only to point cloud data, since it is easy to intuitively understand persistent homology while associating it with the concept of holes. However, we can also apply persistent homology to images and graphs and extract persistence diagrams from them [10]. There is more general explanation based on sublevel sets which is applicable to these various data (e.g. images and graphs) [6].

3 PROPOSED METHOD

In order to handle structured data by Bayesian optimization, it is necessary to design a similarity that captures the topological features of a structure. Although TDA has attracted considerable attention as techniques that can extract such features from complex data, there has been no Bayesian optimization method utilizing TDA to design the similarity. Therefore, in this paper, we propose Bayesian optimization utilizing features extracted by persistent homology.

Moreover, most studies on the applications of persistent homology to machine learning, especially studies on kernels for PDs, consider one type of PD extracted from one data to calculate the kernel. However, it is possible to extract multiple types of PD from one data by using persistent homology. We further propose methods to handle multiple topological features extracted by persistent homology by constructing a kernel using kernels calculated from each type of PD.

In this section, we first formulate the *topological Bayesian optimization* problem using persistence diagrams. Then, we propose the kernel based Bayesian optimization algorithms. We first introduce kernels for PDs in Section 3.2, and then explain methods for constructing a kernel from multiple kernels in Section 3.3.

3.1 Problem formulation

Let us denote the dataset of PDs by $\mathcal{D} = \{ D_i \}_{i \in I}$, where I is the set of *oracle* indices that we cannot observe in the beginning. In addition, we assume that evaluating a PD D_i is expensive. Since TDA is highly used in material science, this assumption is rather reasonable. In this paper, we consider searching for the point that minimizes the objective function from the dataset $\mathcal{D}$:

$$D^* = \underset{D \in \mathcal{D}}{\operatorname{argmin}} \ f(D), \tag{2}$$

where $f : \mathcal{D} \to \mathbb{R}$ is a black box function. This problem can be solved easily if we can examine all possible cases. However, since the objective function is expensive to evaluate, we need to find the optimal data point with a small number of evaluations. We also assume that the objective function value can be observed only in a state of including the independent additive Gaussian noise $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$. Note that we search for the optimal PD from the given dataset $\mathcal{D}$ which we generate from the given dataset of structured data. Since we know all possible PDs and the corresponding structured data, we do not consider about the inverse problem. That is, we need not to reconstruct a structured data from a PD. The final goal of this paper is to develop a Bayesian optimization algorithm to solve Eq. (2).

3.2 Kernels for persistence diagrams

There have been several kernels for persistence diagrams. We use the persistence weighted Gaussian kernel proposed by Kusano et al. and the persistence Fisher kernel proposed by Le and Yamada. In this section, we introduce these kernels.

Persistence weighted Gaussian kernel [12]: Persistence weighted Gaussian kernel (PWGK) considers a PD as a weighted measure. It first vectorizes the measure on an RKHS by kernel mean embedding, and then uses conventional vectorial kernels such as linear kernel and Gaussian kernel on the RKHS. More specifically, it considers the following weighted measure for a persistent diagram D:

$$\mu_D = \sum_{\boldsymbol{x} \in D} w(\boldsymbol{x}) \delta_{\boldsymbol{x}}, \tag{3}$$

where $\delta_{\boldsymbol{x}}$ is a Dirac measure, which takes 1 for $\boldsymbol{x}$ and 0 for other points. Additionally, Dirac measures are weighted by the weight function $w(\boldsymbol{x}) : \mathbb{R}^2 \to \mathbb{R}$ based on the idea that the points close to the diagonal in the PD may represent noisy features, while the points far from the diagonal may represent relatively important features. Let $E(\mu_D) = \sum_{\boldsymbol{x} \in D} w(\boldsymbol{x}) k(\cdot, \boldsymbol{x})$ be the vector representation of μ_D embedded by kernel mean embedding into the RKHS $\mathcal{H}$ using Gaussian kernel $k(\boldsymbol{x}, \boldsymbol{y}) = \exp\left(-\frac{\|\boldsymbol{x}-\boldsymbol{y}\|^2}{2\nu^2}\right)$. Then, the linear kernel of persistence diagrams D_i, D_j on the RKHS is

$$\begin{aligned} k_L(D_i, D_j) &= \langle E(\mu_{D_i}), E(\mu_{D_j}) \rangle_{\mathcal{H}} \\ &= \sum_{\boldsymbol{x} \in D_i} \sum_{\boldsymbol{y} \in D_j} w(\boldsymbol{x}) w(\boldsymbol{y}) \exp\left(-\frac{\|\boldsymbol{x}-\boldsymbol{y}\|^2}{2\nu^2}\right), \end{aligned}$$

where $\nu > 0$ is the kernel bandwidth. In addition, the Gaussian kernel on the RKHS is

$$k_G(D_i, D_j) = \exp\left(-\frac{\|E(\mu_{D_i}) - E(\mu_{D_j})\|^2_{\mathcal{H}}}{2\tau^2}\right).$$

Here, $\tau > 0$ and $\|E(\mu_{D_i}) - E(\mu_{D_j})\|^2_{\mathcal{H}} = k_L(D_i, D_i) + k_L(D_j, D_j) - 2k_L(D_i, D_j)$. We will refer to them as PWGK-Linear and PWGK-Gaussian, respectively.

Note that PWGK can be efficiently approximated using random Fourier features [18]. This method uses the random variables $\boldsymbol{z}_1, \cdots, \boldsymbol{z}_M$ from the normal distribution $\mathcal{N}((0,0), \nu^{-2}\boldsymbol{I})$ to approximate $k_L(D_i, D_j)$ by

$$k_L(D_i, D_j) \approx \frac{1}{M}\sum_{m=1}^{M} B_i^m (B_j^m)^*,$$

where $B_i^m = \sum_{\boldsymbol{x}\in D_i} w(\boldsymbol{x}) \exp\left(\sqrt{-1}\boldsymbol{z}_m \boldsymbol{x}\right)$ and * denotes the complex conjugate.

Persistence Fisher kernel [14]: Persistence Fisher kernel (PFK) considers a PD as the sum of normal distributions and measures the similarity between the distributions by using the Fisher information metric. Let $D_{i\Delta}$ and $D_{j\Delta}$ be the point sets obtained by projecting persistence diagrams D_i and D_j on the diagonal, respectively. PFK compares $D_i' = D_i \cup D_{j\Delta}$ and $D_j' = D_j \cup D_{i\Delta}$ instead of comparing D_i and D_j. It makes the sizes of each point cloud equal, which makes it easy to apply various similarities. Then, it considers the following summation of normal distributions for D_i':

$$\rho_{D_i'} = \frac{1}{Z}\sum_{\boldsymbol{\mu}\in D_i'} \mathcal{N}(\boldsymbol{\mu}, \nu\boldsymbol{I}),$$

where $\nu > 0$ and $Z = \int \sum_{\boldsymbol{\mu}\in D_i'} \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}, \nu\boldsymbol{I}) d\boldsymbol{x}$ is the normalization constant. The Fisher information metric of the probability distributions $\rho_{D_i'}$ and $\rho_{D_j'}$ is as follows:

$$d_{FIM}(D_i, D_j) = \arccos\left(\int \sqrt{\rho_{D_i'}(\boldsymbol{x})\rho_{D_j'}(\boldsymbol{x})} d\boldsymbol{x}\right).$$

The integral appearing in Z and d_{FIM} is calculated using the function value at $\Theta = D_i \cup D_{j\Delta} \cup D_j \cup D_{i\Delta}$. Finally, PFK is expressed as follows using the Fisher information metric:

$$k_{PF}(D_i, D_j) = \exp(-t d_{FIM}(D_i, D_j)),$$

where $t > 0$ is the tuning parameter.

PFK can also be efficiently computed by approximating ρ by fast Gauss transform [17]. We use their implementation[5].

3.3 Multiple kernel learning

In order to handle multiple topological features, we construct an additive kernel calculated from each feature. In particular, we consider a linear combination of k Gram matrices $\boldsymbol{K}_1, \cdots, \boldsymbol{K}_k$:

$$\boldsymbol{K} = \alpha_1 \boldsymbol{K}_1 + \cdots + \alpha_k \boldsymbol{K}_k, \tag{4}$$

where $\alpha_i \geq 0$ for all i. This construction makes it possible to maintain the positive definiteness of each kernel. We consider two methods to learn the coefficient parameter $\boldsymbol{\alpha} = (\alpha_1, \cdots, \alpha_k)^T$.

[5] http://users.umiacs.umd.edu/~morariu/figtree/

Kernel target alignment: A method of maximizing a value called alignment was proposed to learn $\boldsymbol{\alpha}$ [4]. It first considers the centered Gram matrix $\boldsymbol{K}_c$ for the Gram matrix $\boldsymbol{K}$:

$$(K_c)_{ij} = K_{ij} - \mathbb{E}_i[K_{ij}] - \mathbb{E}_j[K_{ij}] + \mathbb{E}_{i,j}[K_{ij}].$$

Then, the alignment of two Gram matrices $\boldsymbol{K}, \boldsymbol{K}'$ is as follows:

$$\kappa(\boldsymbol{K}, \boldsymbol{K}') = \frac{\langle \boldsymbol{K}_c, \boldsymbol{K}_c' \rangle_F}{\|\boldsymbol{K}_c\|_F \|\boldsymbol{K}_c'\|_F},$$

where $\langle \cdot, \cdot \rangle_F$ is the Frobenius inner product and $\| \cdot \|_F$ is the Frobenius norm. In the alignment-based method [4], we maximize the alignment of $\boldsymbol{K} = \sum_i \alpha_i \boldsymbol{K}_i$ and $\boldsymbol{Y} = \boldsymbol{y}\boldsymbol{y}^T$. Maximization of the alignment results in the following quadratic programming problem:

$$\min_{\boldsymbol{v}\geq \boldsymbol{0}} \boldsymbol{v}^T \boldsymbol{M} \boldsymbol{v} - 2\boldsymbol{v}^T \boldsymbol{a},$$

where

$$M_{ij} = \langle \boldsymbol{K}_{ic}, \boldsymbol{K}_{jc} \rangle_F, \quad \boldsymbol{a} = (\langle \boldsymbol{K}_{1c}, \boldsymbol{Y} \rangle_F, \cdots, \langle \boldsymbol{K}_{kc}, \boldsymbol{Y} \rangle_F)^T.$$

Let $\boldsymbol{v}^*$ be the solution of this problem. Then, the coefficients are calculated by $\boldsymbol{\alpha} = \boldsymbol{v}^*/\|\boldsymbol{v}^*\|$. Since $\boldsymbol{y}$ is updated at each step in Bayesian optimization, learning is performed when a new observation is obtained at each step.

Marginal likelihood maximization: In Bayesian optimization, the objective function is modeled by a Gaussian process. Therefore, given the outputs of the function obtained up to a certain step $\boldsymbol{y} = (y_1, \cdots, y_t)^T$, the log marginal likelihood of $\boldsymbol{y}$ can be calculated by:

$$\log p(\boldsymbol{y}|\boldsymbol{\alpha}) \propto -\frac{1}{2}\log|\boldsymbol{K} + \sigma^2\boldsymbol{I}| - \frac{1}{2}\boldsymbol{y}^T(\boldsymbol{K} + \sigma^2\boldsymbol{I})^{-1}\boldsymbol{y}.$$

We consider the use of marginal likelihood maximization to learn $\boldsymbol{\alpha}$. This can be performed by a gradient-based optimization method [2]. As in the case of kernel target alignment, we learn the coefficients when a new observation is obtained.

4 RELATED WORK

Bayesian optimization is widely used for optimizing expensive-to-evaluate, black-box, and noisy objective functions [1]. For example, it is used for automated tuning of hyperparameters in machine learning models [21], path planning of mobile robots [15] and finding the optimal set of sensors [8]. Although most studies on Bayesian optimization including these studies consider vectorial data as input, there are few studies that consider structured data such as point clouds and graphs.

The graph Bayesian optimization (GBO) was proposed as a framework of Bayesian optimization for graph data in particular for tree structured data [19]. Then, it was recently extended to an arbitrary graph structure [5]. GBO proposed by [5] uses a linear combination of two kernels. One is a conventional vectorial kernel such as linear kernel and Gaussian kernel for the explicit feature vector including the number of nodes, average degree centrality, and average betweenness centrality. The other one is a graph kernel, which may capture the implicit topological features that cannot be expressed by explicit features. The coefficients of the linear combination is learned through the Bayesian optimization process. After that, we can analyze which features expressed by the vectorial kernel or the graph kernel were effective as a result. Specifically, they used the automatic relevance determination squared exponential (SEARD) kernel as a vectorial kernel and the deep graph kernel based on subgraphs [25] as a graph kernel. However, to the best of our knowledge, there is no Bayesian optimization framework that *explicitly* uses topological information.

5 EXPERIMENTS

In this section, we evaluate our proposed algorithms using synthetic and four real datasets about properties of molecules.

5.1 Setup

For the proposed method, we use marginal likelihood maximization like as described in Section 3.3 for estimating the noise parameter σ in Bayesian optimization.

We set the hyperparameters of PWGK and PFK according to the original papers [12] and [14], respectively. Let $\{ D_1, \cdots, D_n \}$ be the PDs for each point cloud in a dataset. In PWGK, we use the weight function:

$$w(\boldsymbol{x}) = \arctan(C\mathrm{pers}(\boldsymbol{x})^p),$$

where $\mathrm{pers}(\boldsymbol{x}) = d - b$ for $\boldsymbol{x} = (b, d)$. Therefore, the hyperparameters of PWGK-Linear are C and p in the weight function and the kernel bandwidth ν. PWGK-Gaussian includes τ in addition. We fix the hyperparameters with the following values:

- $C = \mathrm{median}\{ \mathrm{pers}(D_i) \mid i = 1, \cdots, n \}$,
- $p = 5$,
- $\nu = \mathrm{median}\{ \nu(D_i) \mid i = 1, \cdots, n \}$,
- $\tau = \mathrm{median}\left\{ \left|\left| E(\mu_{D_i}) - E(\mu_{D_j}) \right|\right|_{\mathcal{H}} \mid i < j \right\}$,

where

$$\mathrm{pers}(D_i) = \mathrm{median}\{ \mathrm{pers}(\boldsymbol{x}_j) \mid \boldsymbol{x}_j \in D_i \},$$
$$\nu(D_i) = \mathrm{median}\{ ||\boldsymbol{x}_j - \boldsymbol{x}_k|| \mid \boldsymbol{x}_j, \boldsymbol{x}_k \in D_i, j < k \}.$$

The hyperparameters of PFK are ν and t. We search these parameters from $\nu \in \{ 10^{-3}, 10, 10^3 \}$ and $1/t \in \{ q_1, q_2, q_5, q_{10}, q_{20}, q_{50} \}$, where q_s is the s% quantile of $\{ d_{FIM}(D_i, D_j) \mid i < j \}$.

We compare our proposed algorithm with the random search baseline and GBO [5]. For GBO, since the synthetic data is given as a point cloud, we first compute a 5 nearest-neighbor graph and then feed the graph into GBO. We use the same kernels as used in the original paper. We extract 5 features from a graph (the number of nodes, the number of edges, average degree centrality, average betweenness centrality, and average clustering coefficient). Each element x is normalized by $\tilde{x} = (x - x_{min})/(x_{max} - x_{min})$. The window size and embedding dimension for the deep graph kernel are chosen from $\{ 2, 5, 10, 25, 50 \}$. The kernel bandwidths in the SEARD kernel and the coefficients of the linear combination are estimated by marginal likelihood maximization. We also compared with the method only using the SEARD kernel.

In Bayesian optimization, we randomly choose 10 data points to calculate the predictive distribution for the first search point. We use PWGK-Linear, PWGK-Gaussian and PFK as the kernel for PDs and EI as an acquisition function. We first calculate the 1st PDs for synthetic dataset, and the 0th PDs for real datasets. We calculate these kernels using approximation methods (random Fourier features for PWGK and fast Gauss transform for PFK, respectively). We conduct Bayesian optimization 30 times.

5.2 Synthetic dataset

To generate the synthetic dataset, we used the method proposed in [9]. This method generates a point cloud on $[0, 1] \times [0, 1]$. We generate $M = 1000$ point clouds consisting of $N = 1000$ points as the dataset. The specific procedure is as follows.

1. Randomly choose $(x_0, y_0) \in [0, 1] \times [0, 1]$.
2. Iterate the following procedure M times.
   (a) Randomly choose $r \in [2.0, 4.3]$.
   (b) Generate a point cloud $\{ (x_1, y_1), \cdots, (x_N, y_N) \}$ according to the following recurrence relations:

$$x_{n+1} = x_n + r y_n (1 - y_n) \mod 1,$$
$$y_{n+1} = y_n + r x_{n+1} (1 - x_{n+1}) \mod 1.$$

The point clouds generated for $r = 2.0$ and $r = 4.3$ are shown in Figure 3. We use the value of r, which was used to generate a point cloud, as the label of the point cloud. In this experiment, we find the PD of the point cloud with minimum r by using Bayesian optimization algorithms.

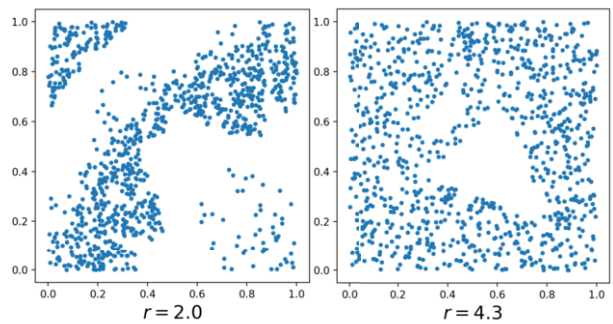

Figure 3. Illustrative examples of synthesized data.

Figure 4 shows averages of the minimum observation obtained at each step for the synthetic dataset over 30 BO runs. As we expected, the topological Bayesian optimization methods outperformed random search baseline and the GBO algorithm.

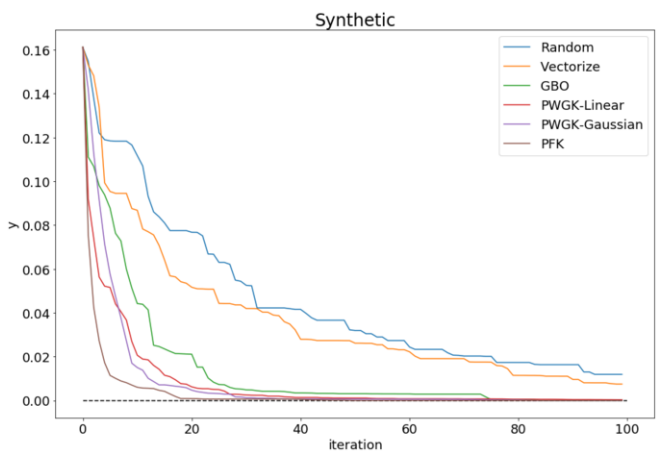

Figure 4. Comparison between random search baseline, GBO and PD kernels using the synthetic dataset. The black dotted line shows the objective function value of the target data that we want to search for.

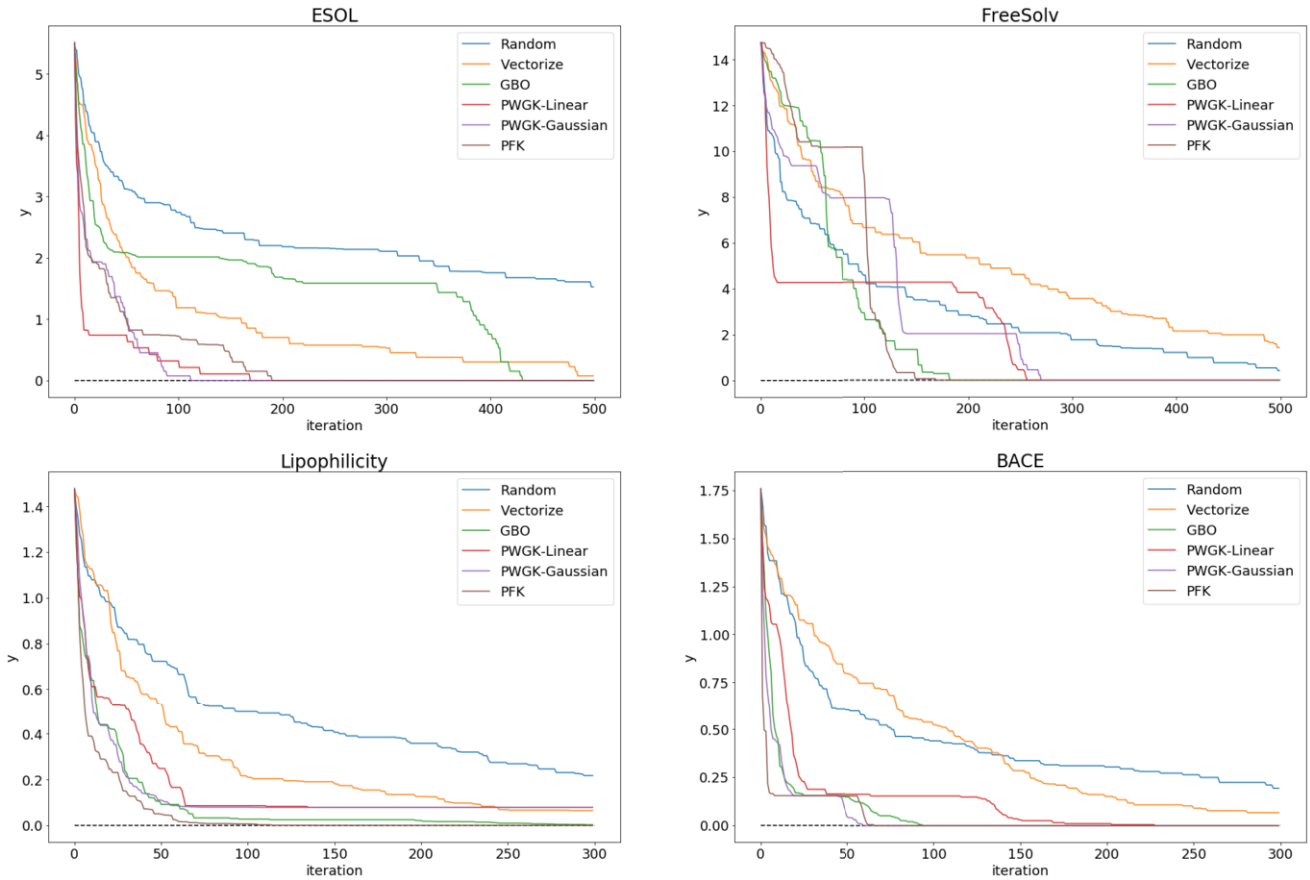

Figure 5. Comparison between random search baseline, GBO and PD kernels using four real datasets. The black dotted line shows the objective function value of the target data that we want to search for.

5.3 Real datasets

We use four real datasets about the properties of relatively small compounds from MoleculeNet [24]. ESOL is a dataset about the water solubility of 1128 compounds. FreeSolv is a dataset about the hydration free energy of 642 compounds in water. Lipophilicity is a dataset about the octanol/water distribution coefficient of 4200 compounds. BACE is a dataset about the binding results for a set of inhibitors of human β-secretase 1 of 1513 compounds. The average numbers of atoms in each dataset are 25.6, 18.1, 48.5, 64.7, respectively. For our method, we treat a compound as a point cloud using only the 3D coordinates of each atom forming the compound without considering any other information about atoms or bonds. We find the PD of the compound with the minimum property such as water solubility and hydration free energy by using the Bayesian optimization algorithms.

Figure 5 shows the averages of the minimum observation obtained at each step for real datasets. In each case, we can see that the information of PDs contributes to efficient search for the optimal structure. Our method shows comparable or better results than existing methods. Our method outperforms especially in the case of the ESOL dataset, which may show that molecular structure reflects some factors associated with water solubility.

5.4 Effectiveness of multiple kernel learning

We compare Bayesian optimization using only one type of PD and that using combined multiple types of PD. Here, we consider combining the information of the 0th PD and the 1st PD (i.e., $k = 2$ in Eq. (4)). We compare the cases of using only the 0th PD, using only the 1st PD and combining both information using kernel target alignment (KTA) and marginal likelihood maximization (MLM) as methods for learning the coefficients. When combining the PFKs, we first conduct experiments similar to those in the previous section using only one type of PD and optimize the hyperparameters in each PFK, and then we learn the coefficients of a linear combination.

The results are summarized in Table 1. We evaluate the performances according to the area under the convergence curve:

$$\sum_{t=1}^{T} \left(y_{best}^{(t)} - y_{best} \right),$$

where T is the number of iteration, $y_{best}^{(t)}$ is the minimum value observed before the tth iteration and y_{best} is the minimum function value. That is, we calculate the area between the convergence curve and the black dotted line in Figure 4 and 5. The values in the table are scaled so that the case of random search baseline becomes 1.

The performances of PWGK-Linear and PWGK-Gaussian are improved by combining the information of both PDs in many datasets. However, PFK get worse in many cases. Perhaps, this is because we perform optimization of kernel parameters and coefficients separately. In addition, when we apply PWGK-Linear to the ESOL dataset as example, the performance is better when combining by marginal likelihood maximization than when using only the 1st PD.

If there is no prior knowledge about which type of PD is effective, this shows that it may be better to combine both PDs than to choose one type of PD for intuition.

For example, when analyzing the water solubility of compounds, the situation we want to search the compound with some specific water solubility not only one with the minimum water solubility may be considered. Thus, we conduct another experiment of finding the maximizer of an objective function as another simple target compound. The results are summarized in Table 2. As in the case of finding the minimizer, it is shown that PWGK-Linear and PWGK-Gaussian tend to be improved by multiple kernel learning while PFK often not.

Table 1. Comparison between cases of using only one type of PD and of using multiple kernel learning methods to find the minimizer of a function.

		Synthetic	ESOL	Free	Lipo	BACE
Random		1.000	1.000	1.000	1.000	1.000
Vectorize		0.747	0.299	1.668	0.371	0.636
GBO		0.252	0.479	0.646	0.088	0.092
PWGK -Linear	0th	0.161	**0.056**	0.663	0.279	**0.184**
	1st	0.153	0.377	1.409	**0.261**	0.954
	KTA	0.166	0.301	1.061	0.336	0.211
	MLM	**0.090**	0.172	**0.456**	0.263	0.237
PWGK -Gaussian	0th	**0.151**	0.081	0.895	0.162	0.070
	1st	0.158	0.464	1.235	0.179	0.132
	KTA	0.167	0.246	0.878	**0.150**	**0.067**
	MLM	0.338	**0.054**	**0.544**	0.264	0.113
PFK	0th	0.117	0.115	0.770	**0.050**	**0.056**
	1st	**0.073**	0.254	**0.646**	0.123	0.068
	KTA	0.092	0.120	0.882	0.062	0.080
	MLM	0.147	**0.090**	0.745	0.080	0.252

Table 2. Comparison between cases of using only one type of PD and of using multiple kernel learning methods to find the maximizer of a function.

		Synthetic	ESOL	Free	Lipo	BACE
Random		1.000	1.000	1.000	1.000	1.000
Vectorize		0.774	0.396	1.029	0.688	0.510
GBO		0.466	0.195	0.356	0.512	0.026
PWGK -Linear	0th	1.153	**0.254**	1.339	0.606	0.021
	1st	0.276	0.284	**0.769**	0.386	0.112
	KTA	0.273	0.366	1.359	0.378	0.023
	MLM	**0.197**	0.272	0.988	**0.295**	**0.004**
PWGK -Gaussian	0th	0.473	**0.052**	0.835	0.357	**0.026**
	1st	0.303	0.199	0.880	0.304	0.047
	KTA	**0.296**	0.056	1.062	**0.255**	0.034
	MLM	0.675	0.117	**0.289**	0.418	0.080
PFK	0th	0.604	**0.075**	0.747	**0.330**	**0.030**
	1st	**0.438**	0.172	0.574	0.655	0.095
	KTA	0.608	0.142	0.795	0.577	0.058
	MLM	1.132	0.186	**0.524**	0.695	0.037

6 Conclusion

In this paper, we proposed the topological Bayesian optimization, which is a Bayesian optimization method using features extracted by persistent homology. In addition, we proposed a method to combine the kernels computed from multiple types of PDs by a linear combination, so that we can use the multiple topological features extracted from one source of data. Through experiments, we confirmed that our method can search for the optimal structure from complex structured data more efficiently than the random search baseline and the state-of-the-art graph Bayesian optimization algorithm by combining multiple kernels.

In the experiments, we treated a compound as a point cloud. For future work, we consider to extend our method to utilize information about atoms and bonds as well. For example, we can treat a compound as a graph and directly apply persistent homology using graph filtration. However, we need to appropriately define a function on nodes or bonds to use graph filtration, which will be main challenge.

ACKNOWLEDGEMENTS

H.K. was supported by JSPS KAKENHI 15H01704. M.Y. was supported by the JST PRESTO program JPMJPR165A and partly supported by MEXT KAKENHI 16H06299 and the RIKEN engineering network funding.

REFERENCES

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

[2] Richard H. Byrd, Peihuang Lu, Jorge Nocedal, and Ciyou Zhu, 'A limited memory algorithm for bound constrained optimization', *SIAM Journal on Scientific Computing*, **16**(5), 1190–1208, (1995).

[3] Gunnar Carlsson, 'Topology and data', *Bulletin of The American Mathematical Society*, **46**, 255–308, (2009).

[4] Corinna Cortes, Mehryar Mohri, and Afshin Rostamizadeh, 'Algorithms for learning kernels based on centered alignment', *Journal of Machine Learning Research*, **13**(Mar), 795–828, (2012).

[5] Jiaxu Cui and Bo Yang, 'Graph bayesian optimization: Algorithms, evaluations and applications', *arXiv preprint arXiv:1805.01157*, (2018).

[6] Herbert Edelsbrunner and John L. Harer, *Computational Topology: An Introduction*, American Mathematical Society, 2010.

[7] Reza Zanjirani Farahani, Elnaz Miandoabchi, W.Y. Szeto, and Hannaneh Rashidi, 'A review of urban transportation network design problems', *European Journal of Operational Research*, **229**(2), 281–302, (2013).

[8] Roman Garnett, Michael A Osborne, and Stephen J. Roberts, 'Bayesian optimization for sensor set selection', *In Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks*, 209–219, (2010).

[9] Jan-Martin Hertzsch, Rob Sturman, and Stephen Wiggins, 'Dna microarrays: design principles for maximizing ergodic, chaotic mixing', *Small*, **3**, 202–218, (2007).

[10] Christoph Hofer, Roland Kwitt, Marc Niethammer, and Andreas Uhl, 'Deep learning with topological signatures', *Advances in Neural Information Processing Systems*, 1634–1644, (2017).

[11] Kirthevasan Kandasamy, Willie Neiswanger, Jeff Schneider, Barnabas Poczos, and Eric P Xing, 'Neural architecture search with bayesian optimisation and optimal transport', *Advances in Neural Information Processing Systems*, 2020–2029, (2018).

[12] Genki Kusano, Kenji Fukumizu, and Yasuaki Hiraoka, 'Kernel method for persistence diagrams via kernel embedding and weight factor', *Journal of Machine Learning Research*, **18**(189), 1–41, (2018).

[13] Harold 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**, 97–106, (1964).

[14] Tam Le and Makoto Yamada, 'Persistence fisher kernel: A riemannian manifold kernel for persistence diagrams', *Advances in Neural Information Processing Systems*, 10027–10038, (2018).

[15] Ruben Martinez-Cantin, Nando de Freitas, Eric Brochu, Jose Castellanos, and Arnaud Doucet, 'A bayesian exploration-exploitation approach for optimal online sensing and planning with a visually guided mobile robot', *Autonomous Robots*, **27**(2), 93–103, (2009).

[16] J Mockus, Vytautas Tiesis, and Antanas Zilinskas, 'The application of bayesian methods for seeking the extremum', *Towards Global Optimization*, **2**, 117–129, (1978).

[17] Vlad I. Morariu, Balaji V. Srinivasan, Vikas C. Raykar, Ramani Duraiswami, and Larry S. Davis, 'Automatic online tuning for fast gaussian summation', *Advances in Neural Information Processing Systems*, 1113–1120, (2009).

[18] Ali Rahimi and Ben Recht, 'Random features for large-scale kernel machines', *Advances in Neural Information Processing Systems*, 1177–1184, (2008).
[19] Dhanesh Ramachandram, Michal Lisicki, Timothy J Shields, Mohamed R Amer, and Graham W Taylor, 'Bayesian optimization on graph-structured search spaces: Optimizing deep multimodal fusion architectures', *Neurocomputing*, **298**, 80–89, (2018).
[20] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*, the MIT Press, 2006.
[21] Jasper Snoek, Hugo Larochelle, and Ryan P. Adams, 'Practical bayesian optimization of machine learning algorithms', *Advances in Neural Information Processing Systems*, 2951–2959, (2012).
[22] Niranjan Srinivas, Andreas Krause, Sham Kakade, and Matthias Seeger, 'Gaussian process optimization in the bandit setting: No regret and experimental design', *In Proceedings of the 27th International Conference on Machine Learning*, 1015–1022, (2010).
[23] Hui Wang, Yanchao Wang, Jian Lv, Quan Li, Lijun Zhang, and Yanming Ma, 'Calypso structure prediction method and its wide application', *Computational Materials Science*, **112**, 406–415, (2016).
[24] Zhenqin Wu, Bharath Ramsundar, Evan N. Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S. Pappu, Karl Leswing, and Vijay Pande, 'Moleculenet: A benchmark for molecular machine learning', *arXiv preprint arXiv:1703.00564*, (2017).
[25] Pinar Yanardag and S. V. N. Vishwanathan, 'Deep graph kernels', *In Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1365–1374, (2015).