Distance Metric Learning for Graph Structured Data

Tomoki Yoshida[1], Ichiro Takeuchi[2], and Masayuki Karasuyama[3]

[1,2,3]Nagoya Institute of Technology

[3]Japan Science and Technology Agency

[2,3]National Institute for Material Science

[2]RIKEN Center for Advanced Intelligence Project

yoshida.t.mllab.nit@gmail.com, {takeuchi.ichiro,karasuyama}@nitech.ac.jp

Abstract

Graphs are versatile tools for representing structured data. Therefore, a variety of machine learning methods have been studied for graph data analysis. Although many of those learning methods depend on the measurement of differences between input graphs, defining an appropriate distance metric for a graph remains a controversial issue. Hence, we propose a supervised distance metric learning method for the graph classification problem. Our method, named *interpretable graph metric learning* (IGML), learns discriminative metrics in a subgraph-based feature space, which has a strong graph representation capability. By introducing a sparsity-inducing penalty on a weight of each subgraph, IGML can identify a small number of important subgraphs that can provide insight about the given classification task. Since our formulation has a large number of optimization variables, an efficient algorithm is also proposed by using pruning techniques based on *safe screening* and *working set selection* methods. An important property of IGML is that the optimality of the solution is guaranteed because the problem is formulated as a convex problem and our pruning strategies only discard unnecessary subgraphs. Further, we show that IGML is also applicable to other structured data such as item-set and sequence data, and that it can incorporate vertex-label similarity by using a transportation-based subgraph feature. We empirically evaluate the computational efficiency and classification performance on several benchmark datasets and show some illustrative examples demonstrating that IGML identifies important subgraphs from a given graph dataset.

1 Introduction

Because of the growing diversity of data-science applications, machine learning methods are required to adapt to a variety of complicated structured data, from which it is often difficult to obtain usual numerical vector representations of input objects. A standard approach to modeling structured data is to employ a *graph*. For example, in domains such as chemo- and bio- informatics, graph-based representations are prevalent. In this study, we particularly focus on the case in which a data instance is represented as a pair composed of a graph and an associated class label.

Although numerous machine learning methods explicitly/implicitly depend on how to measure differences between input objects, defining an appropriate distance metric on a graph remains a controversial issue in the community. A widely accepted approach is *graph kernel* (Gärtner et al., 2003; Vishwanathan et al., 2010), which enables to apply machine learning methods to graph data without having explicit vector representations. Another popular approach would be to use a neural network (Atwood and Towsley, 2016; Narayanan et al., 2017) by which a suitable representation is expected to be learned in the network, such that we can avoid explicitly defining the metric. However, in these approaches, it is difficult to explicitly extract significant sub-structures, i.e., subgraphs. For example, finding a subgraph in a molecular graph which has a strong effect on toxicity would be insightful. Further details of the existing studies are discussed in Section 2.

We propose a supervised method, which obtains a metric for the graph, achieving both high predictive performance and interpretability. Our method, named *interpretable graph metric learning* (IGML), combines the concept of *metric learning* (e.g., Weinberger and Saul, 2009; Davis et al., 2007) with a subgraph representation of the graph, where each graph is represented through a set of subgraphs contained in it. IGML optimizes a metric which has a weight $m_i(H) \geq 0$ for each subgraph H contained in a given graph G. Let $\phi_H(G)$ be a feature of the graph G, which is monotonically non-decreasing with respect to the frequency of subgraph H contained in G. Note that we assume that subgraphs are counted without overlapped vertices and edges throughout the study. We consider the following squared distance between two graphs G and G':

$$d_{\boldsymbol{m}}(G, G') \coloneqq \sum_{H \in \mathcal{G}} m_{i(H)} \left(\phi_H(G) - \phi_H(G')\right)^2, \tag{1}$$

where $\mathcal{G}$ is the set of all connected graphs. Although it is known that the subgraph approach has strong graph representation capability (e.g. Gärtner et al., 2003), naïve calculation is obviously infeasible except when the weight parameters have some special structure.

We formulate IGML as a supervised learning problem of the distance function (1) by using a pairwise loss function of metric learning (Davis et al., 2007) with a sparse penalty on $m_{i(H)}$. The resulting optimization problem is computationally infeasible at a glance, because the number of weight parameters is equal to the number of possible subgraphs, which is usually intractable. We overcome this difficulty by introducing *safe screening* (Ghaoui et al., 2010) and *working set selection* (Fan et al., 2008) approach. Both of these approaches can drastically reduce the number of variables, and further, they can be combined with a *pruning* strategy on the tree traverse of *graph mining*. These optimization tricks are inspired by two recent studies (Nakagawa et al., 2016) and (Morvan and Vert, 2018), which present safe screening- and working set- based pruning for a linear prediction model with the LASSO penalty. By combining these two techniques, we construct a path-wise optimization method that can obtain the sparse solution of the weight parameter $m_{i(H)}$ without directly enumerating all possible subgraphs.

To our knowledge, none of the existing studies can provide an interpretable subgraph-based metric in a supervised manner. The advantages of IGML can be summarized as follows:

- Since IGML is formulated as a convex optimization, the global optimal can be found by the standard gradient-based optimization.

- The safe screening- and working set- based optimization algorithms make our problem practically tractable without sacrificing the optimality.

- We can identify a small number of important subgraphs that discriminate different classes. This means that the resulting metric is easy to compute and highly interpretable, and thus useful for a variety of subsequent data analysis. For example, applying the nearest neighbor classification or decision tree on the learned space would be effective.

We further propose three extensions of IGML. First, we show that IGML is directly applicable to other structured data such as item-set and sequence data. Second, the application to a triplet based loss function is discussed. Third, we extend IGML, such that similarity information of vertex-labels can be incorporated. In our experiments, we empirically verify the superior or comparable prediction performance of IGML to other existing graph classification methods (most of which do not have interpretability). We also show some examples of extracted subgraphs and data analyses on the learned metric space.

This paper is organized as follows. In Section 2, we review existing studies on graph data analysis. In Section 3, we introduce a formulation of our proposed IGML. Section 4 discusses the strategies that reduce the size of the optimization problem of IGML. The detailed computational procedure of IGML is described in Section 5. The three extensions of IGML are presented in Section 6. Section 7 empirically evaluates the effectiveness of IGML on several benchmark datasets.

Note that this paper is an extended version of the preliminary conference paper (Yoshida et al., 2019a). The source code of the program used in experiments is available at `https://github.com/takeuchi-lab/Learning-Interpretable-Metric-between-Graphs`.

2 Related Work

Kernel-based approaches have been widely studied for graph data analysis, and they can provide a metric of graph data in a reproducing kernel Hilbert space. In particular, subgraph-based graph kernels are closely related to our study. The graphlet kernel (Shervashidze et al., 2009) creates a kernel through small subgraphs with only about 3-5 vertices, called a graphlet. The neighborhood subgraph pairwise distance kernel (Costa and Grave, 2010) selects pairs of subgraphs from a graph and counts the number of identical pairs with another graph. The subgraph matching kernel (Kriege and Mutzel, 2012) identifies common subgraphs based on cliques in the product graph of two graphs. Although these studies mainly focus on the computation of the kernel, the subgraph-based feature representation of the graph is also available. However, since these approaches are unsupervised, it is impossible to eliminate subgraphs that are unnecessary for classification. Consequently, all candidate subgraphs must be once enumerated before applying a learning method.

There are many other kernels including the shortest path (Borgwardt and Kriegel, 2005)-, random walk (Vishwanathan et al., 2010; Sugiyama and Borgwardt, 2015; Zhang et al., 2018b)-, and spectrum (Kondor and Borgwardt, 2008; Kondor et al., 2009; Kondor and Pan, 2016; Verma and Zhang, 2017)-based approaches. The Weisfeiler-Lehman (WL) kernel (Shervashidze and Borgwardt, 2009; Shervashidze et al., 2011), which is based on the graph isomorphism test, is a popular and empirically successful kernel that has been employed in many studies (Yanardag and Vishwanathan, 2015; Niepert et al., 2016; Narayanan et al., 2017; Zhang et al., 2018a). These approaches are once more unsupervised, and it is quite difficult to interpret results from the perspective of substructures of a graph. Although several kernels deal with continuous attributes on vertices (Feragen et al., 2013; Orsini et al., 2015; Su et al., 2016; Morris et al., 2016), we only focus on the cases where vertex-labels are discrete because of its high interpretability.

Since obtaining a good metric is an essential problem in data analysis, metric learning has been extensively studied so far, as reviewed in (Li and Tian, 2018). However, due to computational difficulty, metric learning for graph data has not been widely studied. A few studies considered the *edit distance* approaches. For example, Bellet et al. (2012) is a method of learning a similarity function through an edit distance in a supervised manner. Another approach is that Neuhaus and Bunke (2007) probabilistically formulates the editing process of the graph and estimates the parameters by using labeled data. However, these approaches cannot provide any clear interpretation of the resulting metric in a sense of the subgraph.

The deep neural network (DNN) likewise represents a standard approach to graph data analysis. The deep graph kernel (Yanardag and Vishwanathan, 2015) incorporates neural language modeling, where decomposed sub-structures of a graph are regarded as a sentence. The PATCHY-SAN (Niepert et al., 2016) and DGCNN (Niepert et al., 2016) convert the graph to a tensor by using the WL-Kernel and convolute it. Several other studies also have combined popular convolution techniques with graph data (Tixier et al., 2018; Atwood and Towsley, 2016; Simonovsky and Komodakis, 2017). These approaches are supervised, but the interpretability of these DNN is obviously quite low. *Attention* enhances interpretability of deep learning, yet extracting important subgraphs is difficult because the attention algorithm for graph (Lee et al., 2018) only provides the significance of the vertex transition on a graph. Another related DNN approach would be representation learning. For example, sub2vec (Adhikari et al., 2018) and graph2vec (Narayanan et al., 2017) can embed graph data into a continuous space, but they are unsupervised, and it is difficult to extract substructures that characterize different classes. There are other fingerprint learning methods for graphs by neural networks (e.g. Duvenaud et al., 2015) where the contribution from each node can be evaluated for each dimension of the fingerprint. Although it is possible to highlight sub-structures for the given input graph, this does not produce important common subgraphs for prediction.

Supervised pattern mining (Cheng et al., 2008; Novak et al., 2009; Thoma et al., 2010) can be used for identifying important subgraphs by enumerating patterns with some discriminative score. However, these approaches usually 1) employ the greedy strategy to add a pattern by which global optimality cannot be guaranteed, and 2) do not optimize a metric or a representation. A few other studies (Saigo et al., 2009; Nakagawa et al., 2016) considered optimizing a linear model on the subgraph feature with the LASSO penalty by employing graph mining, but these are specific to the parameter optimization of the specific linear model.

3 Formulation of Interpretable Graph Metric Learning

Suppose that the training dataset $\{(G_i, y_i)\}_{i \in [n]}$ consists of n pairs of a graph G_i and a class label y_i, where $[n] \coloneqq \{1, \ldots, n\}$. Let $\mathcal{G}$ be a set of all induced connected subgraphs of $\{G_i\}_{i \in [n]}$. In each graph, vertices and edges can be labeled. If $H \in \mathcal{G}$ is an induced connected subgraph of $G \in \mathcal{G}$, we write $H \sqsubseteq G$. Further, let $\#(H \sqsubseteq G)$ be the frequency of the subgraph H contained in G. Note that we adopt the definition of frequency which does not allow overlap of any vertices or edges among the counted subgraphs. As a representation of a graph G, we consider the following subgraph-based feature representation:

$$\phi_H(G) = g\big(\#(H \sqsubseteq G)\big), \text{ for } H \in \mathcal{G}, \tag{2}$$

where g is some monotonically non-decreasing and non-negative function, such as the identity function $g(x) = x$ or the indicator function $g(x) = 1_{x>0}$, which takes 1 if $x > 0$, and otherwise 0. It is widely known that the subgraph-based feature is an effective way to represent graphs. For example, $g(x) = x$ allows to distinguish all non-isomorphic graphs. A similar idea was shown in (Gärtner et al., 2003) in the case of the frequency which allows overlaps. However, this feature space is practically infeasible because the possible number of subgraphs is prohibitively large.

We focus on how to measure the distance between two graphs, which is essential for a variety of machine learning problems. We consider the following weighted squared distance between two graphs:

$$d_{\boldsymbol{m}}(G, G') \coloneqq \sum_{H \in \mathcal{G}} m_{i(H)} \left(\phi_H(G) - \phi_H(G')\right)^2,$$

where $i(H)$ is the index of the subgraph H for a weight parameter $m_{i(H)} \geq 0$. To obtain an effective and computable distance metric, we adaptively estimate $m_{i(H)}$, such that only a small number of important subgraphs have non-zero values of $m_{i(H)}$.

Let $\boldsymbol{x}_i \in \mathbb{R}^p$ be the feature vector defined by concatenating $\phi_H(G_i)$ for all $H \in \mathcal{G}$ included in the training dataset. Then, we see

$$d_{\boldsymbol{m}}(\boldsymbol{x}_i, \boldsymbol{x}_j) = (\boldsymbol{x}_i - \boldsymbol{x}_j)^\top \mathrm{diag}(\boldsymbol{m})(\boldsymbol{x}_i - \boldsymbol{x}_j) = \boldsymbol{m}^\top \boldsymbol{c}_{ij},$$

where $\boldsymbol{m} \in \mathbb{R}_+^p$ is a vector of $m_{i(H)}$, and $\boldsymbol{c}_{ij} \in \mathbb{R}^p$ is defined as $\boldsymbol{c}_{ij} := (\boldsymbol{x}_i - \boldsymbol{x}_j) \circ (\boldsymbol{x}_i - \boldsymbol{x}_j)$ using the element-wise product $\circ$.

Let $\mathcal{S}_i \subseteq [n]$ be a subset of indices that are in the same class as $\boldsymbol{x}_i$, and $\mathcal{D}_i \subseteq [n]$ be a subset of indices that are in different classes from $\boldsymbol{x}_i$. For both of these sets, we select K most similar inputs to $\boldsymbol{x}_i$ by using some default metric (e.g., a graph kernel). As a loss function for $\boldsymbol{x}_i$, we consider

$$\ell_i(\boldsymbol{m}; L, U) := \sum_{l \in \mathcal{D}_i} \ell_L(\boldsymbol{m}^\top \boldsymbol{c}_{il}) + \sum_{j \in \mathcal{S}_i} \ell_{-U}(-\boldsymbol{m}^\top \boldsymbol{c}_{ij}), \tag{3}$$

where $L, U \in \mathbb{R}_+$ are the constant parameters satisfying $U \le L$, and $\ell_t(x) = [t - x]_+^2$ is the standard squared hinge loss function with the threshold $t \in \mathbb{R}$. This loss function is a variant of the pairwise loss functions used in metric learning (Davis et al., 2007). The first term in the loss function yields a penalty if $\boldsymbol{x}_i$ and $\boldsymbol{x}_l$ are closer than L for $l \in \mathcal{D}_i$, and the second term yields a penalty if $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are more distant than U for $j \in \mathcal{S}_i$.

Let $R(\boldsymbol{m}) = \|\boldsymbol{m}\|_1 + \frac{\eta}{2}\|\boldsymbol{m}\|_2^2 = \boldsymbol{m}^\top \boldsymbol{1} + \frac{\eta}{2}\|\boldsymbol{m}\|_2^2$ be an elastic-net type sparsity inducing penalty, where $\eta \ge 0$ is a non-negative parameter. We define our proposed IGML (*interpretable graph metric learning*) as the following regularized loss minimization problem:

$$\min_{\boldsymbol{m} \ge \boldsymbol{0}} P_\lambda(\boldsymbol{m}) := \sum_{i \in [n]} \ell_i(\boldsymbol{m}; L, U) + \lambda R(\boldsymbol{m}), \tag{4}$$

where $\lambda > 0$ is the regularization parameter. The solution of this problem can provide not only a discriminative metric, but also insight into important subgraphs because the sparse penalty is expected to select only a small number of non-zero parameters.

Let $\boldsymbol{\alpha} \in \mathbb{R}_+^{2nK}$ be the vector of dual variables, where α_{il} and α_{ij} for $i \in [n], l \in \mathcal{D}_i$, and $j \in \mathcal{S}_i$ are concatenated. The dual problem of (4) is written as follows (see Appendix A for derivation):

$$\max_{\boldsymbol{\alpha} \ge \boldsymbol{0}} D_\lambda(\boldsymbol{\alpha}) := -\frac{1}{4}\|\boldsymbol{\alpha}\|_2^2 + \boldsymbol{t}^\top \boldsymbol{\alpha} - \frac{\lambda\eta}{2}\|\boldsymbol{m}_\lambda(\boldsymbol{\alpha})\|_2^2, \tag{5}$$

where

$$\boldsymbol{m}_\lambda(\boldsymbol{\alpha}) := \frac{1}{\lambda\eta}[\boldsymbol{C}\boldsymbol{\alpha} - \lambda\boldsymbol{1}]_+, \tag{6}$$

$\boldsymbol{t} := [L, \ldots, L, -U, \ldots, -U]^\top \in \mathbb{R}^{2nK}$ and $\boldsymbol{C} := [\ldots, \boldsymbol{c}_{il}, \ldots, -\boldsymbol{c}_{ij}, \ldots] \in \mathbb{R}^{p \times 2nK}$. Then, from the optimality condition, we obtain the following relationship between the primal and dual variables:

$$\alpha_{il} = -\ell'_L(\boldsymbol{m}^\top \boldsymbol{c}_{il}),\ \alpha_{ij} = -\ell'_{-U}(-\boldsymbol{m}^\top \boldsymbol{c}_{ij}), \tag{7}$$

where $\ell'_t(x) = -2[t - x]_+$ is the derivative of ℓ_t. When the regularization parameter λ is larger than certain $\lambda_{\max}$, the optimal solution is $\boldsymbol{m} = \boldsymbol{0}$. Then, the optimal dual variables are $\alpha_{il} = -\ell'_L(0) = 2L$ and $\alpha_{ij} = -\ell'_{-U}(0) = 0$. By substituting these equations into (6), we obtain $\lambda_{\max}$ as

$$\lambda_{\max} = \max_k \boldsymbol{C}_{k,:}\boldsymbol{\alpha}. \tag{8}$$

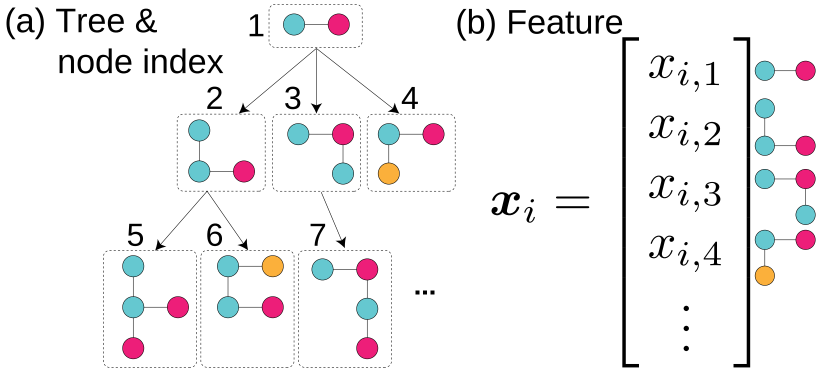


Figure 1: Schematic illustration of tree and feature.

4 Creating Tractable Sub-problem

Because the problems (4) and (5) are convex, the local solution is equivalent to the global optimal. However, naïvely solving these problems is computationally intractable because of the high dimensionality of $\boldsymbol{m}$. In this section, we introduce several useful rules for restricting candidate subgraphs while maintaining the optimality of the final solution. Note that the proofs for all the lemma and theorems are provided in the appendix.

To make the optimization problem tractable, we work with only a small subset of features during the optimization process. Let $\mathcal{F} \subseteq [p]$ be a subset of features. By fixing $m_i = 0$ for $i \notin \mathcal{F}$, we define sub-problems of the original primal P_λ and dual D_λ problems as follows:

$$\min_{\boldsymbol{m}_{\mathcal{F}} \geq \boldsymbol{0}} P_\lambda^{\mathcal{F}}(\boldsymbol{m}_{\mathcal{F}}) \coloneqq \sum_{i \in [n]} [\sum_{l \in \mathcal{D}_i} \ell_L(\boldsymbol{m}_{\mathcal{F}}^\top \boldsymbol{c}_{il\,\mathcal{F}}) + \sum_{j \in \mathcal{S}_i} \ell_{-U}(-\boldsymbol{m}_{\mathcal{F}}^\top \boldsymbol{c}_{ij\,\mathcal{F}})] + \lambda R(\boldsymbol{m}_{\mathcal{F}}), \tag{9}$$

$$\max_{\boldsymbol{\alpha} \geq \boldsymbol{0}} D_\lambda^{\mathcal{F}}(\boldsymbol{\alpha}) \coloneqq -\frac{1}{4}\|\boldsymbol{\alpha}\|_2^2 + \boldsymbol{t}^\top \boldsymbol{\alpha} - \frac{\lambda\eta}{2}\|\boldsymbol{m}_\lambda(\boldsymbol{\alpha})_{\mathcal{F}}\|_2^2, \tag{10}$$

where $\boldsymbol{m}_{\mathcal{F}}$, $\boldsymbol{c}_{ij\mathcal{F}}$, and $\boldsymbol{m}_\lambda(\boldsymbol{\alpha})_{\mathcal{F}}$ are sub-vectors specified by $\mathcal{F}$. If the size of $\mathcal{F}$ is moderate, these sub-problems are computationally significantly easier to solve than the original problems.

We introduce several criteria that determine whether the feature k should be included in $\mathcal{F}$ by using techniques of *safe screening* (Ghaoui et al., 2010) and *working set selection* (Fan et al., 2008). A general form of our criteria can be written as

$$\boldsymbol{C}_{k,:}\boldsymbol{q} + r\|\boldsymbol{C}_{k,;}\|_2 \leq T, \tag{11}$$

where $\boldsymbol{q} \in \mathbb{R}_+^{2nK}$, $r \geq 0$, and $T \in \mathbb{R}$ are constants that assume different values depending on the criterion. If this inequality holds for k, we exclude the k-th feature from $\mathcal{F}$. An important property is that although our algorithm only solves these small sub-problems, we can guarantee the optimality of the final solution, as shown later in this study.

However, selecting $\mathcal{F}$ itself is computationally quite expensive since the evaluation of (11) requires $O(n)$ computations for each k. Thus, we exploit a tree structure of graphs for determining $\mathcal{F}$. Figure 1 shows an example of the tree, which can be constructed by graph mining algorithms such as gSpan (Yan and Han, 2002). Suppose that the k-th node corresponds to the k-th dimension of $\boldsymbol{x}$ (Note that here the node index is not the order of the visit). If a graph corresponding to the k-th node is a subgraph of the k'-th node, the node k' is a descendant of k, which is denoted as $k' \supseteq k$. Then, the following monotonic relation is immediately derived from the monotonicity of ϕ_H:

$$x_{i,k'} \leq x_{i,k} \text{ if } k' \supseteq k. \tag{12}$$

Based on this property, the following lemma enables us to prune a node during the tree traverse:

Lemma 4.1. *Let*

$$\mathrm{Prune}(k|\boldsymbol{q}, r) \coloneqq \sum_{i\in[n]} \sum_{l\in\mathcal{D}_i} q_{il} \max\{x_{i,k}, x_{l,k}\}^2 + r\sqrt{\sum_{i\in[n]} [\sum_{l\in\mathcal{D}_i} \max\{x_{i,k}, x_{l,k}\}^4 + \sum_{j\in\mathcal{S}_i} \max\{x_{i,k}, x_{j,k}\}^4]} \tag{13}$$

be a pruning criterion. Then, if the inequality

$$\mathrm{Prune}(k|\boldsymbol{q}, r) \leq T, \tag{14}$$

holds, for any descendant node $k' \supseteq k$, the following inequality holds

$$\boldsymbol{C}_{k',:}\boldsymbol{q} + r\|\boldsymbol{C}_{k',:}\|_2 \leq T,$$

where $\boldsymbol{q} \in \mathbb{R}_+^{2nK}$ and $r \geq 0$ are arbitrary constant vector and scalar variables.

This lemma indicates that if the condition (14) is satisfied, we can determine that none of the descendant nodes are included in $\mathcal{F}$. Assuming that the indicator function $g(x) = 1_{x>0}$ is used in (2), a tighter bound can be obtained as follows.

Lemma 4.2. *If $g(x) = 1_{x>0}$ is set in* (2)*, the pruning criterion* (13) *can be replaced with*

$$\mathrm{Prune}(k|\boldsymbol{q}, r) \coloneqq \sum_{i\in[n]} \max\{\sum_{l\in\mathcal{D}_i} q_{il}x_{l,k}, x_{i,k}[\sum_{l\in\mathcal{D}_i} q_{il} - \sum_{j\in\mathcal{S}_i} q_{ij}(1 - x_{j,k})]\} \\ + r\sqrt{\sum_{i\in[n]} [\sum_{l\in\mathcal{D}_i} \max\{x_{i,k}, x_{l,k}\} + \sum_{j\in\mathcal{S}_i} \max\{x_{i,k}, x_{j,k}\}]}.$$

By comparing the first terms of lemma 4.1 and lemma 4.2, we see that lemma 4.2 is tighter when $g(x) = 1_{x>0}$ as follows:

$$\begin{aligned}\sum_{i\in[n]} \max\{\sum_{l\in\mathcal{D}_i} q_{il}x_{l,k}, x_{i,k}[\sum_{l\in\mathcal{D}_i} q_{il} - \sum_{j\in\mathcal{S}_i} q_{ij}(1 - x_{j,k})]\} &\leq \sum_{i\in[n]} \max\{\sum_{l\in\mathcal{D}_i} q_{il}x_{l,k}, x_{i,k}\sum_{l\in\mathcal{D}_i} q_{il}\} \\ &= \sum_{i\in[n]} \max\{\sum_{l\in\mathcal{D}_i} q_{il}x_{l,k}, \sum_{l\in\mathcal{D}_i} q_{il}x_{i,k}\} \\ &\leq \sum_{i\in[n]} \sum_{l\in\mathcal{D}_i} \max\{q_{il}x_{l,k}, q_{il}x_{i,k}\} \\ &= \sum_{i\in[n]} \sum_{l\in\mathcal{D}_i} q_{il} \max\{x_{l,k}, x_{i,k}\}.\end{aligned}$$

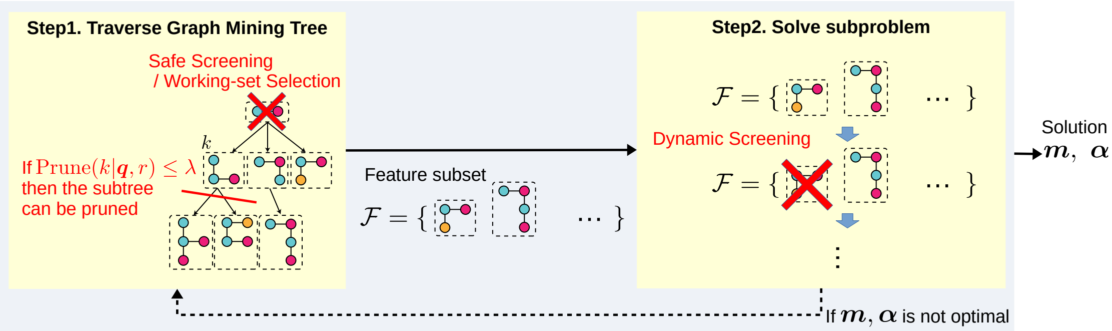

Step1. Traverse Graph Mining Tree
Safe Screening
/ Working-set Selection
k
If $\mathrm{Prune}(k|\boldsymbol{q}, r) \leq \lambda$
then the subtree
can be pruned
Feature subset
$\mathcal{F} = \{ \cdots \}$
Step2. Solve subproblem
$\mathcal{F} = \{ \cdots \}$
Dynamic Screening
$\mathcal{F} = \{ \cdots \}$
Solution
$\boldsymbol{m}, \ \boldsymbol{\alpha}$
If $\boldsymbol{m}, \boldsymbol{\alpha}$ is not optimal

Figure 2: Schematic illustration of our optimization algorithm for IGML.

A schematic illustration of our optimization algorithm for IGML is shown in Figure 2 (for the details, see Section 5). To generate a subset of features $\mathcal{F}$, we first traverse the graph mining tree during which the safe screening/working set selection procedure and their pruning extensions are performed (Step1). Next, we solve the subproblem (9) with the generated $\mathcal{F}$ by using a standard gradient-based algorithm (Step2). Safe screening is also performed during the optimization iteration in this Step2, which is referred to as *dynamic screening*. This further reduces the size of $\mathcal{F}$.

4.1 Safe Screening

Safe screening (Ghaoui et al., 2010) was first proposed to identify unnecessary features in LASSO-type problems. Typically, this approach considers a bounded region of dual variables where the optimal solution must exist. Then, we can eliminate dual inequality constraints which are never violated as far as the solution exists in that region. The well-known Karush-Kuhn-Tucker (KKT) conditions show that this is equivalent to the elimination of primal variables which take 0 at the optimal solution. In Section 4.1.1, we first derive a spherical bound of our optimal solution, and in Section 4.1.2, a rule for safe screening is shown. Section 4.1.3 shows an extension of rules that are specifically useful for the regularization path calculation.

4.1.1 Sphere Bound for Optimal Solution

The following theorem provides a hyper-sphere containing the optimal dual variable $\boldsymbol{\alpha}^\star$:

Theorem 4.1 (DGB)**.** *For any pair of* $\boldsymbol{m} \geq \boldsymbol{0}$ *and* $\boldsymbol{\alpha} \geq \boldsymbol{0}$*, the optimal dual variable* $\boldsymbol{\alpha}^\star$ *must satisfy*

$$\|\boldsymbol{\alpha} - \boldsymbol{\alpha}^\star\|_2^2 \leq 4(P_\lambda(\boldsymbol{m}) - D_\lambda(\boldsymbol{\alpha})).$$

This bound is called *duality gap bound* (DGB), and the parameters $\boldsymbol{m}$ and $\boldsymbol{\alpha}$ used to construct the bound are referred to as the *reference solution*.

If the optimal solution for λ_0 is available as a reference solution to construct the bound for λ_1, the following bound, called *regularization path bound* (RPB), can be obtained:

Theorem 4.2 (RPB)**.** *Let $\boldsymbol{\alpha}_0^\star$ be the optimal solution for λ_0, and $\boldsymbol{\alpha}_1^\star$ be the optimal solution for λ_1.*

$$\left\| \boldsymbol{\alpha}_1^\star - \frac{\lambda_0 + \lambda_1}{2\lambda_0} \boldsymbol{\alpha}_0^\star \right\|_2^2 \le \left\| \frac{\lambda_0 - \lambda_1}{2\lambda_0} \boldsymbol{\alpha}_0^\star \right\|_2^2 .$$

However, RPB requires the exact solution, which is difficult in practice because of numerical errors. *Relaxed RPB* (RRPB) extends RPB, such that it can incorporate the approximate solution as a reference solution:

Theorem 4.3 (RRPB)**.** *Assuming that $\boldsymbol{\alpha}_0$ satisfies $\|\boldsymbol{\alpha}_0 - \boldsymbol{\alpha}_0^\star\|_2 \le \epsilon$. The optimal solution $\boldsymbol{\alpha}_1^\star$ for λ_1 must satisfy*

$$\left\| \boldsymbol{\alpha}_1^\star - \frac{\lambda_0 + \lambda_1}{2\lambda_0} \boldsymbol{\alpha}_0 \right\|_2 \le \left\| \frac{\lambda_0 - \lambda_1}{2\lambda_0} \boldsymbol{\alpha}_0 \right\|_2 + \Big(\frac{\lambda_0 + \lambda_1}{2\lambda_0} + \frac{|\lambda_0 - \lambda_1|}{2\lambda_0} \Big) \epsilon.$$

For example, ϵ can be obtained by using the DGB (Theorem 4.1).

Similar bounds to those of which we derived here were previously considered for the triplet screening of metric learning on usual numerical data (Yoshida et al., 2018, 2019b). Here, we extend a similar idea to derive subgraph screening.

4.1.2 Safe Screening and Safe Pruning Rules

Theorem 4.1 and 4.3 identify the regions where optimal solution exists by using a current feasible solution $\boldsymbol{\alpha}$. Further, from (6), when $\boldsymbol{C}_{k,:}\boldsymbol{\alpha}^\star \le \lambda$, we see $m_k^\star = 0$. This indicates that

$$\max_{\boldsymbol{\alpha} \in \mathcal{B}} \boldsymbol{C}_{k,:}\boldsymbol{\alpha} \le \lambda \Rightarrow m_k^\star = 0, \tag{15}$$

where $\mathcal{B}$ is a region containing the optimal solution $\boldsymbol{\alpha}^\star$, i.e., $\boldsymbol{\alpha}^\star \in \mathcal{B}$. By solving this maximization problem, we obtain the following safe screening rule (SS Rule):

Theorem 4.4 (SS Rule)**.** *If the optimal solution $\boldsymbol{\alpha}^\star$ exists in the bound $\mathcal{B} = \{\boldsymbol{\alpha} \mid \|\boldsymbol{\alpha} - \boldsymbol{q}\|_2^2 \le r^2\}$, the following rule holds*

$$\boldsymbol{C}_{k,:}\boldsymbol{q} + r\|\boldsymbol{C}_{k,:}\|_2 \le \lambda \Rightarrow m_k^\star = 0. \tag{16}$$

Theorem 4.4 indicates that we can eliminate unnecessary features by evaluating the condition shown in (16). An important property of this rule is that it guarantees optimality, meaning that the sub-problems (9) and (10) have the exact same optimal solution to the original problem if $\mathcal{F}$ is defined through this rule. However, it is still necessary to evaluate the rule for all p features, which is currently intractable. To avoid this problem, we derive a pruning strategy on the graph mining tree, which we call safe pruning rule (SP Rule):

Theorem 4.5 (SP Rule)**.** *If the optimal solution $\boldsymbol{\alpha}^\star$ is in the bound $\mathcal{B} = \{\boldsymbol{\alpha} \mid \|\boldsymbol{\alpha} - \boldsymbol{q}\|_2^2 \le r^2, \boldsymbol{q} \ge \boldsymbol{0}\}$, the following rule holds*

$$\text{Prune}(k|\boldsymbol{q}, r) \le \lambda \Rightarrow m_{k'}^\star = 0. \ \textit{for} \ \forall k' \supseteq k. \tag{17}$$

This theorem is a direct consequence of lemma 4.1. If this condition holds for a node k during the tree traverse, a subtree below that node can be pruned. This means that we can safely eliminate unnecessary subgraphs even without enumerating them.

4.1.3 Range-based Safe Screening & Range-based Safe Pruning

SS and SP are rules for a fixed λ. The range-based extension identifies an interval of λ where the satisfaction of SS/SP is guaranteed. This is particularly useful for the *path-wise optimization* or *regularization path* calculation, where we need to solve the problem with a sequence of λ. We assume that the sequence is sorted in descending order as optimization algorithms typically start from the trivial solution $\boldsymbol{m} = \mathbf{0}$. Let $\lambda = \lambda_1 \leq \lambda_0$. By combining RRPB with the rule (16), we obtain the following theorem:

Theorem 4.6 (Range-based Safe Screening (RSS))**.** *For any k, the following rule holds*

$$\lambda_a \leq \lambda \leq \lambda_0 \Rightarrow m_k^\star = 0, \tag{18}$$

where

$$\lambda_a \coloneqq \frac{\lambda_0(2\epsilon\|\boldsymbol{C}_{k,:}\|_2 + \|\boldsymbol{\alpha}_0\|_2\|\boldsymbol{C}_{k,:}\|_2 + \boldsymbol{C}_{k,:}\boldsymbol{\alpha}_0)}{2\lambda_0 + \|\boldsymbol{\alpha}_0\|_2\|\boldsymbol{C}_{k,:}\|_2 - \boldsymbol{C}_{k,:}\boldsymbol{\alpha}_0}.$$

For SP, the range-based rule can also be derived from (17):

Theorem 4.7 (Range-based Safe Pruning (RSP))**.** *For any $k' \supseteq k$, the following pruning rule holds:*

$$\lambda'_a \coloneqq \frac{\lambda_0(2\epsilon b + \|\boldsymbol{\alpha}_0\|_2 b + a)}{2\lambda_0 + \|\boldsymbol{\alpha}_0\|_2 b - a} \leq \lambda \leq \lambda_0 \Rightarrow m_{k'}^\star = 0, \tag{19}$$

where

$$a \coloneqq \sum_{i\in[n]}\sum_{l\in\mathcal{D}_i} \alpha_{0il} \max\{x_{l,k}, x_{i,k}\}^2,$$
$$b \coloneqq \sqrt{\sum_{i\in[n]}[\sum_{l\in\mathcal{D}_i} \max\{x_{i,k}, x_{l,k}\}^4 + \sum_{j\in\mathcal{S}_i} \max\{x_{i,k}, x_{j,k}\}^4]}.$$

Here again, if the feature vector is generated from $g(x) = 1_{x>0}$ (i.e., binary), the following theorem holds:

Theorem 4.8 (Range-Based Safe Pruning (RSP) for binary feature)**.** *Assuming $g(x) = 1_{x>0}$ in* (2)*, a and b in theorem 4.7 can be replaced with*

$$a \coloneqq \sum_{i\in[n]} \max\{\sum_{l\in\mathcal{D}_i} \alpha_{0il} x_{l,k}, x_{i,k}[\sum_{l\in\mathcal{D}_i} \alpha_{0il} - \sum_{j\in\mathcal{S}_i} \alpha_{0ij}(1 - x_{j,k})]\},$$
$$b \coloneqq \sqrt{\sum_{i\in[n]}[\sum_{l\in\mathcal{D}_i} \max\{x_{i,k}, x_{l,k}\} + \sum_{j\in\mathcal{S}_i} \max\{x_{i,k}, x_{j,k}\}]}.$$

Since these constants a and b are derived from the tighter bound in lemma 4.2, the obtained range becomes wider than the range in theorem 4.7.

After once we calculate λ_a and λ'_a of (18) and (19) for some λ, they are stored at each node of the tree. Subsequently, those λ_a and λ'_a can be used for the next tree traverse with different λ'. If the condition (18) or (19) is satisfied, the node can be skipped (RSS) or pruned (RSP). Otherwise, we update λ_a and λ'_a by using the current reference solution.

4.2 Working Set Method

Safe rules are strong rules in a sense that they can completely remove features, and thus sometimes they are too conservative to fully accelerate the optimization. In contrast, the *working set selection* is a widely accepted heuristic approach to selecting a subset of features.

4.2.1 Working Set Selection & Working Set Pruning

Working set (WS) method optimizes the problem only with respect to selected working set features. Then, if the optimality condition for the original problem is not satisfied, the working set is selected again and the optimization on the new working set re-starts. This process iterates until the optimality on the original problem is achieved.

Besides the safe rules, we use the following WS selection criterion, which is obtained directly from the KKT conditions:

$$\boldsymbol{C}_{k,:}\boldsymbol{\alpha} \leq \lambda. \tag{20}$$

If this inequality is satisfied, the k-th dimension is predicted as $m_k^\star = 0$. Hence, the working set is defined by

$$\mathcal{W} \coloneqq \{k \mid \boldsymbol{C}_{k,:}\boldsymbol{\alpha} > \lambda\}.$$

Although $m_i^\star = 0$ for $i \notin \mathcal{W}$ is not guaranteed, the final convergence of the procedure is shown by the following theorem:

Theorem 4.9 (Convergence of WS)**.** *Assume that there is a solver for the sub-problem* (9) *(or equivalently* (10)*), which returns the optimal solution for given $\mathcal{F}$. Working set method, which iterates optimizating the sub-problem with $\mathcal{F} = \mathcal{W}$ and updating $\mathcal{W}$ alternately, returns the optimal solution of the original problem with finite steps.*

However, here again, the inequality (20) needs to be evaluated for all features, which is computationally intractable.

The same pruning strategy as SS/SP can be incorporated into working set selection. The criterion (20) is also a special case of (11), and lemma 4.1 indicates that if the following inequality

$$\mathrm{Prune}_{\mathrm{WP}}(k) \coloneqq \mathrm{Prune}(k|\boldsymbol{\alpha}, 0) \leq \lambda,$$

holds, then any $k' \supseteq k$ is not included in the working set. We refer to this criterion as working set pruning (WP).

4.2.2 Relation with Safe Rules

Note that for working set method, we may need to update $\mathcal{W}$ multiple times unlike safe screening approaches as we see in theorem 4.9. Instead, working set method can usually exclude a larger number of features compared with safe screening approaches. In fact, when the condition of the SS rule (16) is satisfied, the WS criterion (20) must be likewise satisfied. Since all the spheres (DGB, RPB and RRPB) contain the reference solution $\boldsymbol{\alpha}$ (which is usually the current solution), the inequality

$$\boldsymbol{C}_{k,:}\boldsymbol{\alpha} \leq \max_{\boldsymbol{\alpha}' \in \mathcal{B}} \boldsymbol{C}_{k,:}\boldsymbol{\alpha}' \tag{21}$$

holds, where $\mathcal{B}$ is a sphere created by DGB, RPB or RRPB. This indicates that when the SS rule excludes the k-th feature, the WS also excludes the k-th feature. However, to guarantee convergence, WS needs to be fixed until the sub-problem (9)-(10) is solved (theorem 4.9). In contrast, the SS rule is applicable anytime during the optimization procedure without affecting the final optimality. This enables us to apply the SS rule even to the sub-problem (9)-(10) in which $\mathcal{F}$ is defined by WS as shown in step 2 of Figure 2 (dynamic screening).

For the pruning rules, we first confirm the following two properties:

$$\begin{aligned}\mathrm{Prune}(k|\boldsymbol{q}, r) &\geq \mathrm{Prune}(k|\boldsymbol{q}, 0),\\ \mathrm{Prune}(k|C\boldsymbol{q}, 0) &= C\ \mathrm{Prune}(k|\boldsymbol{q}, 0),\end{aligned}$$

where $\boldsymbol{q} \in \mathbb{R}_+^{2nK}$ is the center of the sphere, $r \geq 0$ is the radius, and $C \in \mathbb{R}$ is a constant. In the case of DGB, the center of the sphere is the reference solution $\boldsymbol{\alpha}$ itself, i.e., $\boldsymbol{q} = \boldsymbol{\alpha}$. Then, the following relation holds between the SP criterion $\mathrm{Prune}(k|\boldsymbol{q}, r)$ and the WP criterion $\mathrm{Prune}_{\mathrm{WP}}(k)$:

$$\mathrm{Prune}(k|\boldsymbol{q}, r) = \mathrm{Prune}(k|\boldsymbol{\alpha}_0, r) \geq \mathrm{Prune}(k|\boldsymbol{\alpha}_0, 0) = \mathrm{Prune}_{\mathrm{WP}}(k).$$

This once more indicates that when the SP rule is satisfied, the WP rule must be satisfied as well. When the RPB or RRPB sphere is used, the center of sphere is $\boldsymbol{q} = \frac{\lambda_0+\lambda_1}{2\lambda_0}\boldsymbol{\alpha}_0$. Assuming that the solution for λ_0 is used as the reference solution, i.e., $\boldsymbol{\alpha} = \boldsymbol{\alpha}_0$, we obtain

$$\begin{aligned}\mathrm{Prune}(k|\boldsymbol{q}, r) &= \mathrm{Prune}(k|\frac{\lambda_0+\lambda_1}{2\lambda_0}\boldsymbol{\alpha}, r)\\ &\geq \mathrm{Prune}(k|\frac{\lambda_0+\lambda_1}{2\lambda_0}\boldsymbol{\alpha}, 0)\\ &= \frac{\lambda_0+\lambda_1}{2\lambda_0}\mathrm{Prune}(k|\boldsymbol{\alpha}, 0)\\ &= \frac{\lambda_0+\lambda_1}{2\lambda_0}\mathrm{Prune}_{\mathrm{WP}}(k).\end{aligned}$$

Using this inequality, we obtain

$$\mathrm{Prune}(k|\boldsymbol{q}, r) - \mathrm{Prune}_{\mathrm{WP}}(k) \geq \frac{\lambda_1-\lambda_0}{2\lambda_0}\mathrm{Prune}_{\mathrm{WP}}(k).$$

From this inequality, if $\lambda_1 > \lambda_0$, then $\mathrm{Prune}(k|\boldsymbol{q}, r) > \mathrm{Prune}_{\mathrm{WP}}(k)$ (note that $\mathrm{Prune}_{\mathrm{WP}}(k) \geq 0$ because $\boldsymbol{\alpha} \geq \boldsymbol{0}$), indicating that the pruning of WS is always tighter than that of the safe rule. However, in our algorithm shown in Section 5, $\lambda_1 < \lambda_0$ holds because we start from a larger value of λ and gradually decrease it. Then, this inequality does not hold, and $\mathrm{Prune}(k|\boldsymbol{q}, r) < \mathrm{Prune}_{\mathrm{WP}}(k)$ becomes a possibility.

When the WS and WP rules are strictly tighter than the SS and SP rules, respectively, using both of WS/WP and SS/SP rules is equivalent to using WS/WP only (except for dynamic screening). Even in this case, the range-based safe approaches (the RSS and RSP rules) can still be effective. When the range-based rules are evaluated, we obtain the range of λ where the SS or SP rule is satisfied. Thus, as long as λ is in that range, we do not need to evaluate any safe or working set rules.

5 Algorithm and Computations

5.1 Training with Path-wise Optimization

We employ *path-wise optimization* (Friedman et al., 2007), where the optimization starts from $\lambda = \lambda_{\max}$ and gradually decreases λ while optimizing $\boldsymbol{m}$. As we see in (8), $\lambda_{\max}$ is defined by the maximum of the inner product $\boldsymbol{C}_{k,:}\boldsymbol{\alpha}$. This value can also be found by the tree search with pruning. Suppose that we calculate $\boldsymbol{C}_{k,:}\boldsymbol{\alpha}$ while traversing the tree, and $\hat{\lambda}_{\max}$ is the current maximum value during the traverse. Using lemma 4.1, we can derive the pruning rule

$$\mathrm{Prune}(k|\boldsymbol{\alpha}, 0) \leq \hat{\lambda}_{\max}.$$

If this condition holds, the descendant nodes of k cannot be the maximum, and thus we can identify $\lambda_{\max}$ without calculating $\boldsymbol{C}_{k,:}\boldsymbol{\alpha}$ for all candidate features.

Algorithm 1 shows the outer loop of our path-wise optimization. The `TRAVERSE` and `SOLVE` functions in Algorithm 1 are shown in Algorithm 2 and 3, respectively. Algorithm 1 first calculates $\lambda_{\max}$ which is the minimum λ at which the optimal solution is $\boldsymbol{m}^\star = \boldsymbol{0}$ (line 3). The outer loop in line 5-14 is the process of decreasing λ with the decreasing rate R. The `TRAVERSE` function in line 7 determines the subset of features $\mathcal{F}$ by the traversing tree with safe screening and working set selection. The inner loop (line 9-13) alternately solves the optimization problem with the current $\mathcal{F}$ and updates $\mathcal{F}$, until the duality gap becomes less than the given threshold eps.

Algorithm 2 shows the `TRAVERSE` function, which recursively visits the tree node to determine $\mathcal{F}$. The variable `node.pruning` contains λ'_a of RSP, and if the RSP condition (19) is satisfied (line 3), the function returns the current $\mathcal{F}$ (the node is pruned). The variable `node.screening` contains λ_a of RSS, and if the RSS condition (18) is satisfied (line 5), this node can be skipped, and the function proceeds to the next node. If these two conditions are not satisfied, the function performs 1) updating `node.pruning` and `node.screening` if `update` is true, and 2) evaluating the conditions of RSP and WP (line 10), and RSS and WS (line 14). At line 17-18, gSpan expands children of the current node, and for each child node, the `TRAVERSE` function is called recursively.

Algorithm 3 shows a solver for the primal problem with the subset of features $\mathcal{F}$. Although we employ a simple projected gradient algorithm, any optimization algorithm can be used in this process. In line 7-10, the SS rule is evaluated at every freq iterations. Note that this SS is only for sub-problems (9) and (10)

Algorithm 1: Path-wise Optimization

```
function PathwiseOptimization($R, T, \mathrm{freq}, \mathrm{MaxIter}, \mathrm{eps}$)
    $\boldsymbol{m}_0 = \mathbf{0}, \boldsymbol{\alpha}_0 = [2L, \dots, 2L, 0, \dots, 0], \epsilon = 0$
    $\lambda_0 = \lambda_{\max} = \max_k \boldsymbol{C}_{k,:} \boldsymbol{\alpha}_0$                    ▷ Compute $\lambda_{\max}$
    Initialize root node as root.children = empty, root.screening = $\infty$, and root.pruning = $\infty$
    for $t = 1, 2, \dots, T$ do
        $\lambda_t = R\lambda_{t-1}$
        $\mathcal{F}$ = Traverse($\lambda_{t-1}, \lambda_t, \boldsymbol{\alpha}_{t-1}, \epsilon$, root, true)    ▷ get working set & update range of $\lambda$
        $\boldsymbol{m}_t = \boldsymbol{m}_{t-1}$
        repeat
            $\boldsymbol{m}_t, \boldsymbol{\alpha}_t, P$ = Solve($\lambda_t, \boldsymbol{m}_t, \mathcal{F}$, freq, MaxIter, eps)
            $\mathcal{F}$ = Traverse(null, $\lambda_t, \boldsymbol{\alpha}_t$, null, root, false)          ▷ update working set
            gap = $P - D_{\lambda_t}^{\mathcal{F}}(\boldsymbol{\alpha}_t)$
        until $\frac{\mathrm{gap}}{P} \leq$ eps                                                  ▷ check optimality
        $\epsilon = 2\sqrt{\mathrm{gap}}$
    return $\{\boldsymbol{m}_t\}_{t=0}^{t=T}$
```

created by current $\mathcal{F}$ (not for the original problems).

5.2 Enumerating Subgraphs for Test Data

To obtain a feature vector for test data, we only need to enumerate subgraphs that have $m_k \neq 0$. When gSpan is used as a mining algorithm, a unique code, called *minimum DFS code*, is assigned to each node. If a DFS code for a node is $(a_1, a_2, \dots, a_n)$, a child node is represented by $(a_1, a_2, \dots, a_n, a_{n+1})$. This enables us to prune a node which does not generate subgraphs with $m_k \neq 0$. Suppose that a subgraph $(a_1, a_2, a_3) = (x, y, z)$ has to be enumerated, and that currently we are in the node $(a_1) = (x)$. Then, a child with $(a_1, a_2) = (x, y)$ should be traversed, but a child with $(a_1, a_2) = (x, w)$ cannot generate (x, y, z), and consequently we can stop the traverse of this node.

5.3 Post-processing

5.3.1 Learning Mahalanobis Distance for Selected Subgraphs

Instead of $\boldsymbol{m}$, the following Mahalanobis distance can be considered

$$d_{\boldsymbol{M}}(\boldsymbol{x}_i, \boldsymbol{x}_j) = (\boldsymbol{x}_i - \boldsymbol{x}_j)^\top \boldsymbol{M} (\boldsymbol{x}_i - \boldsymbol{x}_j), \tag{22}$$

where $\boldsymbol{M}$ is a positive definite matrix. Directly optimizing $\boldsymbol{M}$ requires $O(p^2)$ primal variables and semi-definite constraint, making the problem computationally quite expensive even for relatively small p. Thus,

Algorithm 2: Traverse gSpan with RSS&RSP+WS&WP

```
function Traverse($\lambda_0, \lambda, \boldsymbol{\alpha}_0, \epsilon$, node, update)
    $\mathcal{F} = \{\}, k =$ node.feature
    if node.pruning $\leq \lambda$ then                                ▷ RSP Rule
        return $\mathcal{F}$
    else if node.screening $\leq \lambda$ then                         ▷ RSS Rule
        do nothing
    else                              ▷ Update the range of $\lambda$ if update $=$ true
        if update $=$ true then
            node.pruning $= \frac{\lambda_0(2\epsilon b+\|\boldsymbol{\alpha}_0\|_2 b+a)}{2\lambda_0+\|\boldsymbol{\alpha}_0\|_2 b-a}$                ▷ Eq.(19)
        if node.pruning $\leq \lambda$ or Prune$_{\mathrm{WP}}(k) \leq \lambda$ then
            return $\mathcal{F}$
        if update $=$ true then
            node.screening $= \frac{\lambda_0(2\epsilon\|\boldsymbol{C}_{k,:}\|_2+\|\boldsymbol{\alpha}_0\|_2\|\boldsymbol{C}_{k,:}\|_2+\boldsymbol{C}_{k,:}\boldsymbol{\alpha}_0)}{2\lambda_0+\|\boldsymbol{\alpha}_0\|_2\|\boldsymbol{C}_{k,:}\|_2-\boldsymbol{C}_{k,:}\boldsymbol{\alpha}_0}$    ▷ Eq.(18)
        if node.screening $> \lambda$ and $\boldsymbol{C}_{k,:}\boldsymbol{\alpha}_0 > \lambda$ then
            $\mathcal{F} = \mathcal{F} \cup \{k\}$
        createChildren(node)
        for child $=$ node.children do
            $\mathcal{F} = \mathcal{F} \cup$ Traverse($\lambda_0, \lambda, \boldsymbol{\alpha}_0, \epsilon$, child, update)
    return $\mathcal{F}$
function createChildren(node)
    if node.children $=$ empty then
        Set node.children by gSpan
        for child $=$ node.children do
            child.children $=$ empty
            child.screening $= \infty$, child.pruning $= \infty$
```

Algorithm 3: Gradient descent with dynamic screening

```
function SOLVE($\lambda, \boldsymbol{m}, \mathcal{F}$, freq, MaxIter, eps)   ▷ Solve primal problem $P_\lambda^{\mathcal{F}}$, which is considered only for feature set $\mathcal{F}$
    for iter $= 0, 1, \ldots,$ MaxIter do
        Compute $\boldsymbol{\alpha}$ by (7)
        gap $= P_\lambda^{\mathcal{F}}(\boldsymbol{m}) - D_\lambda^{\mathcal{F}}(\boldsymbol{\alpha})$
        if $\frac{\text{gap}}{P_\lambda^{\mathcal{F}}(\boldsymbol{m})} \leq$ eps then   ▷ convergence
            return $\boldsymbol{m}, \boldsymbol{\alpha}, P_\lambda^{\mathcal{F}}(\boldsymbol{m})$
        if mod(iter, freq) $= 0$ then
            for $k \in \mathcal{F}$ do   ▷ perform dynamic screening
                if $\boldsymbol{C}_{k,:}\boldsymbol{\alpha} + 2\sqrt{\text{gap}}\|\boldsymbol{C}_{k,:}\|_2 \leq \lambda$ then   ▷ SS by DGB
                    $\mathcal{F} = \mathcal{F} - \{k\}$
        $\boldsymbol{m} = [\boldsymbol{m} - \gamma\nabla P_\lambda^{\mathcal{F}}(\boldsymbol{m})]_+$   ▷ update $\boldsymbol{m}$ ($\gamma$: step-size)
    return $\boldsymbol{m}, \boldsymbol{\alpha}, P_\lambda^{\mathcal{F}}(\boldsymbol{m})$
```

in this paper, as optional post-processing, we consider optimizing the Mahalanobis distance (22) for a small number of subgraphs selected by the optimized $\boldsymbol{m}$. Let $\mathcal{H} \subseteq \mathcal{G}$ be a set of subgraphs $m_{i(H)} > 0$ for $H \in \mathcal{H}$, and $\boldsymbol{z}_i$ be a $h \coloneqq |\mathcal{H}|$ dimensional feature vector consisting of $\phi_H(G_i)$ for $H \in \mathcal{H}$. For $\boldsymbol{M} \in \mathbb{R}^{h \times h}$, we consider the following metric learning problem:

$$\min_{\boldsymbol{M} \succeq \boldsymbol{O}} \sum_{i \in [n]} [\sum_{l \in \mathcal{D}_i} \ell_L(d_{\boldsymbol{M}}(\boldsymbol{z}_i, \boldsymbol{z}_l)) + \sum_{j \in \mathcal{S}_i} \ell_{-U}(-d_{\boldsymbol{M}}(\boldsymbol{z}_i, \boldsymbol{z}_j))] + \lambda R(\boldsymbol{M}),$$

where $R : \mathbb{R}^{h \times h} \to \mathbb{R}$ is a regularization term for $\boldsymbol{M}$, where a typical setting is $R(\boldsymbol{M}) = \text{tr}\boldsymbol{M} + \frac{\eta}{2}\|\boldsymbol{M}\|_F^2$, where tr is the trace of a matrix. This metric can be more discriminative, because it is optimized to the training data with a higher degree of freedom.

5.3.2 Vector Representation of Graph

An explicit vector representation of an input graph can be obtained by using optimized $\boldsymbol{m}$:

$$\boldsymbol{x}_i' = \sqrt{\boldsymbol{m}} \circ \boldsymbol{x}_i \tag{23}$$

Unlike the original $\boldsymbol{x}_i$, the new representation $\boldsymbol{x}_i'$ is computationally tractable because of the sparsity of $\boldsymbol{m}$, and simultaneously, this space should be highly discriminative. This property is beneficial for further analysis of the graph data. We show an example of applying the decision tree on the learned space in our later experiment.

In the case of the general Mahalanobis distance shown in Section 5.3.1, we can obtain further transformation. Let $\boldsymbol{M} = \boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^\top$ be the eigenvalue decomposition of the learned $\boldsymbol{M}$. Employing the regularization term $R(\boldsymbol{M}) = \mathrm{tr}\boldsymbol{M} + \frac{\eta}{2}\|\boldsymbol{M}\|_F^2$, part of the eigenvalues of $\boldsymbol{M}$ can be shrunk to 0 because $\mathrm{tr}\boldsymbol{M}$ is equal to the sum of the eigenvalues. If $\boldsymbol{M}$ has $h' < h$ non-zero eigenvalues, $\boldsymbol{\Lambda}$ can be written as a $h' \times h'$ diagonal matrix, and $\boldsymbol{V}$ is a $h \times h'$ matrix where each column is the eigenvector of the non-zero eigenvalue. Then, a representation of graph is

$$\sqrt{\boldsymbol{\Lambda}}\boldsymbol{V}^\top \boldsymbol{z}_i. \tag{24}$$

This can be considered as a supervised dimensionality reduction from h- to h'-dimensional space. Although each dimension is no longer corresponding to a subgraph in this representation, the interpretation remains clear because each dimension of the transformed vector is just a linear combination of $\boldsymbol{z}_i$.

6 Extensions

In this section, we consider the three extensions of IGML: applications to other data types, employing a triplet loss function, and introducing vertex-label similarity.

6.1 Application to Other Structured Data

The proposed method can be applied to item-set/sequence data in addition to graph data. For the item set, the Jaccard index, defined as the size of the intersection of two sets divided by the size of the union, is the most popular similarity measure. Although a few studies consider kernels for the item set (Zhang et al., 2007), to our knowledge, it remains difficult to adapt a metric on the given labeled dataset in an interpretable manner. In contrast, there are many kernel approaches for sequence data. The spectrum kernel (Leslie et al., 2001) creates a kernel matrix by enumerating all k-length subsequences in the given sequence. The mismatch kernel (Leslie et al., 2004) enumerates subsequences allowing m-discrepancies in a pattern of length k. The gappy kernel (Leslie and Kuang, 2004; Kuksa et al., 2008) counts the number of k-mers (subsequences) with a certain number of gaps g that appear in the sequence. These kernels require the value of hyperparameter k, although various lengths may be in fact related. The motif kernel (Zhang and Zaki, 2006; Pissis et al., 2013; Pissis, 2014) counts the number of “motifs” appearing in the input sequences. The “motif” must be decided by the user. Since these approaches are based on the idea of the ‘kernel’, they are unsupervised unlike our approach.

By employing a similar approach to the graph input, we can construct the feature representation $\phi_H(X_i)$ for both item-set and sequence data. For the item-set data, the i-th input is a set of items $X_i \subseteq \mathcal{I}$, where $\mathcal{I}$ is a set of all items. For example, $X_1 = \{a, b\}, X_2 = \{b, c, e\}, \ldots$ with the candidate items $\mathcal{I} = \{a, b, c, d, e\}$. The feature $\phi_H(X_i)$ is defined by $1_{H \subseteq X_i}$ for $\forall H \subseteq \mathcal{I}$. This feature $\phi_H(X_i)$ also has the monotonicity $\phi_{H'}(X_i) \leq \phi_H(X_i)$ for $H' \supseteq H$. In the sequence data, the i-th input X_i is a sequence of items. Thus, the feature $\phi_H(X_i)$ is defined from the frequency of a sub-sequence H in the given X_i. For example, if we have $X_i = \langle b, b, a, b, a, c, d \rangle$ and $H = \langle b, a \rangle$, then H occurs two times in X_i. For the sequence data, again, the monotonicity property is guaranteed since $\phi_{H'}(X_i) \leq \phi_H(X_i)$ where H is a sub-sequence of H'. Because

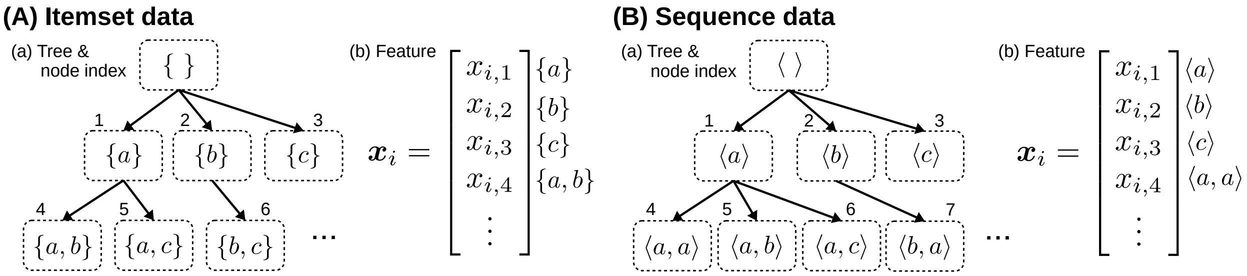


Figure 3: Schematic illustration of trees and features for item-set/sequence data.

of these monotonicity properties, we can apply the same pruning procedures to both of the item-set and sequence data. Figure 3 shows an example of the tree, which can be constructed by item-set/sequence mining algorithms (Agrawal et al., 1994; Pei et al., 2001).

6.2 Triplet Loss

We formulated the loss function of IGML as the pair-wise loss (3). A triplet loss function is likewise widely used in metric learning (e.g., Weinberger and Saul, 2009):

$$\sum_{(i,j,l)\in\mathcal{T}} \ell_1(\boldsymbol{m}^\top \boldsymbol{c}_{il} - \boldsymbol{m}^\top \boldsymbol{c}_{ij}),$$

where $\mathcal{T}$ is an index set of triplets, consisting of (i, j, l), which satisfies $y_i = y_j, y_i \neq y_l$. This loss incurs a penalty when the distance between samples in the same class is larger than the distance between samples in different classes. Because the loss is defined by a 'triplet' of samples, this approach can be more time-consuming than the pair-wise approach. In contrast, the relative evaluation such as $d_{\boldsymbol{m}}(\boldsymbol{x}_i, \boldsymbol{x}_j) < d_{\boldsymbol{m}}(\boldsymbol{x}_i, \boldsymbol{x}_l)$ (the j-th sample must be closer to the i-th sample compared with the l-th sample) can capture the higher order relation of input objects rather than penalizing the pair-wise distance.

A pruning rule can be derived even for the case of the triplet loss function. By defining $\boldsymbol{c}_{ijl} \coloneqq \boldsymbol{c}_{il} - \boldsymbol{c}_{ij}$, the loss function is written as

$$\sum_{(i,j,l)\in\mathcal{T}} \ell_1(\boldsymbol{m}^\top \boldsymbol{c}_{ijl}).$$

Because this is the same form as pair-wise loss with $L = 1, U = 0$, the optimization problem is reduced to the same form as the pairwise case. We need a slight modification on lemma 4.1 because of the change of the constant coefficients (i.e., from $\boldsymbol{c}_{ij}$ to $\boldsymbol{c}_{ijl}$). The equation (13) is changed to

$$\text{Prune}(k|\boldsymbol{q}, r) \coloneqq \sum_{(i,j,l)\in\mathcal{T}} q_{ijl} \max\{x_{i,k}, x_{l,k}\}^2 + r\sqrt{\sum_{ijl} \max\{x_{i,k}, x_{l,k}\}^4}. \tag{25}$$

This is easily proven using

$$c_{ijl,k'} = (x_{i,k'} - x_{l,k'})^2 - (x_{i,k'} - x_{j,k'})^2 \leq \max\{x_{i,k}, x_{l,k}\}^2, \forall k' \supseteq k,$$

which is an immediate consequence of the monotonicity inequality (12).

$$
\begin{array}{cc}
 & \begin{array}{ccc} {\color{red}\bullet} & {\color{orange}\bullet} & {\color{teal}\bullet} \end{array} \\
\begin{array}{c} {\color{red}\bullet} \\ {\color{orange}\bullet} \\ {\color{teal}\bullet} \end{array} & \begin{bmatrix} 0 & 1.2 & 0.1 \\ 1.2 & 0 & 0.6 \\ 0.1 & 0.6 & 0 \end{bmatrix}
\end{array}
$$

Figure 4: A dissimilarity matrix among vertex-labels.

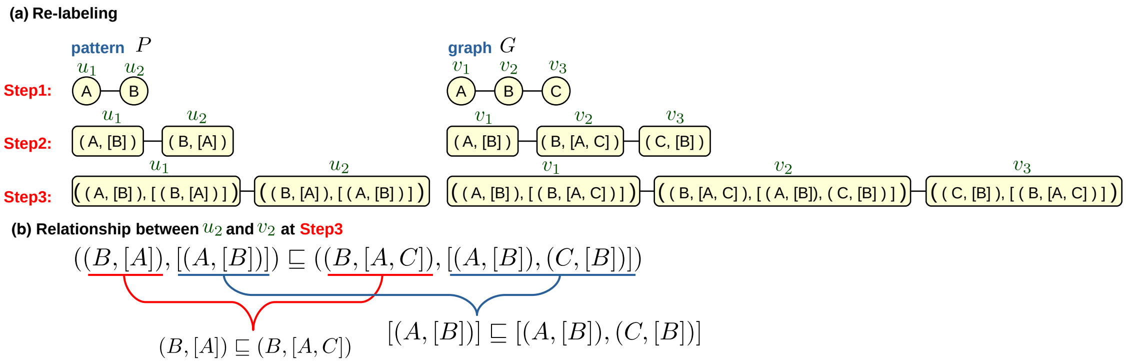

(a) Re-labeling
pattern P
graph G
Step1:
Step2:
Step3:
u_1 u_2
v_1 v_2 v_3
A B C
(A, [B]) (B, [A])
(A, [B]) (B, [A, C]) (C, [B])
((A, [B]), [(B, [A])]) ((B, [A]), [(A, [B])])
((A, [B]), [(B, [A, C])]) ((B, [A, C]), [(A, [B]), (C, [B])]) ((C, [B]), [(B, [A, C])])
(b) Relationship between u_2 and v_2 at Step3
$((B,[A]),[(A,[B])]) \sqsubseteq ((B,[A,C]),[(A,[B]),(C,[B])])$
$(B,[A]) \sqsubseteq (B,[A,C])$
$[(A,[B])] \sqsubseteq [(A,[B]),(C,[B])]$

Figure 5: Re-labeling and inclusion relationship. $(X, [\,])$ is abbreviated as X, where $X \in \{A, B, C\}$. (a) In each step, all vertices are re-labeled by combining a vertex-label and neighboring labels at the previous step. (b) Example of inclusion relationship, defined by (26) and (27). The relation $L_P(u_2, 3) \sqsubseteq L_G(v_2, 3)$ is satisfied between u_2 and v_2 at Step3.

6.3 Considering Vertex-Label Similarity

Since IGML is based on the exact matching of subgraphs to create the feature $\phi_H(G)$, it is difficult to provide a prediction to a graph that does not exactly match many of the selected subgraphs. Typically, this happens when the test dataset has a different distribution of vertex-labels. For example, in the case of the prediction on a chemical compound group whose atomic compositions are largely different from those of the training dataset, the exact match may not be expected as in the case of the training dataset. We consider incorporating similarity/dissimilarity information of graph vertex-labels for relaxing this exact matching constraint. A toy example of vertex-label dissimilarity is shown in Figure 4. In this case, the 'red' vertex is similar to the 'green' vertex, while it is dissimilar to the 'yellow' vertex. For example, we can create this type of table by using prior domain knowledge (e.g., chemical properties of atoms). Even when no prior information is available, a similarity matrix can be inferred by using any embedding methods (e.g., Huang et al., 2017).

Because it is difficult to directly incorporate similarity information into our subgraph-isomorphism based feature $\phi_H(G)$, we first introduce a relaxed evaluation of inclusion of a graph P in a given graph G. We assume that P is obtained from the gSpan tree of training data. Our approach is based on the idea of 're-labeling' of graph vertex-labels in the Weisfeiler-Lehman (WL) kernel (Shervashidze et al., 2011), which is a well-known graph kernel with the approximate graph-isomorphism test. Figure 5 (a) shows an example of the re-labeling procedure, which is performed by the fixed number of recursive steps. The number of steps is denoted as T ($T = 3$ in the figure), and is assumed to be pre-specified. In step h, each graph vertex v has a *level h hierarchical label* $L_G(v, h) \coloneqq (F^{(h)}, S^{(h)} = [S_1^{(h)}, \dots, S_n^{(h)}])$, where $F^{(h)}$ is recursively defined by the level $h-1$ hierarchical label of the same vertex, i.e., $F^{(h)} = L_G(v, h-1)$, and $S^{(h)}$ is a multiset created by the level $h-1$ hierarchical labels $L_G(v', h-1)$ from all the neighboring vertices v' connected to v. Note that a multiset, denoted by '$[,]$', is a set where duplicate elements are allowed. For example, in the graph G shown in the right side of Figure 5 (a), the hierarchical label of the vertex v_1 on the level $h = 3$ is $L_G(v_1, 3) = ((A, [B]), [(B, [A, C])])$. In this case, $F^{(3)} = (A, [B])$, which is equal to $L_G(v_1, 2)$, and $S_1^{(3)} = (B, [A, C])$, which is equal to $L_G(v_2, 2)$. The original label A can also be regarded as a hierarchical label $(A, [\,])$ on the level $h = 1$, but it is shown as 'A' for simplicity.

We define a relation of the inclusion '$\sqsubseteq$' between two hierarchical labels $L_P(u, h) = (F^{(h)}, S^{(h)} = [S_1^{(h)}, \dots, S_m^{(h)}])$ and $L_G(v, h) = (F'^{(h)}, S'^{(h)} = [S_1'^{(h)}, \dots, S_n'^{(h)}])$, which originate from the two vertices u and v in graphs P and G, respectively. We say that $L_P(v, h)$ is included in $L_G(u, h)$, and it is denoted by

$$L_P(v, h) \sqsubseteq L_G(u, h), \tag{26}$$

when the following recursive condition is satisfied

$$\begin{cases} F^{(h)} = F'^{(h)}, & \text{if } S^{(h)} = S'^{(h)} = [\,], \qquad (27a) \\ F^{(h)} \sqsubseteq F'^{(h)} \land \exists\sigma(\land_{i\in[m]} S_i^{(h)} \sqsubseteq S'^{(h)}_{\sigma(i)}), & \text{otherwise}, \qquad (27b) \end{cases}$$

where $\sigma : [m] \to [n]$ is an injection from $[m]$ to $[n]$ (i.e., $\sigma(i) \neq \sigma(j)$ when $i \neq j$), and $\exists\sigma(\land_{i\in[m]} S_i^{(h)} \sqsubseteq S'^{(h)}_{\sigma(i)})$ indicates that there exists an injection σ which satisfies $S_i^{(h)} \sqsubseteq S'^{(h)}_{\sigma(i)}$ for $\forall i \in [m]$. The first condition (27a) is for the case of $S^{(h)} = S'^{(h)} = [\,]$, which occurs at the first level $h = 1$, and in this case, it simply evaluates whether the two hierarchical labels are equal $F^{(h)} = F'^{(h)}$. Note that when $h = 1$, the hierarchical label is simply $(X, [])$, where X is one of the original vertex-labels. In the other case (27b), both of the two conditions $F^{(h)} \sqsubseteq F'^{(h)}$ and $\exists\sigma(\land_{i\in[m]} S_i^{(h)} \sqsubseteq S'^{(h)}_{\sigma(i)})$ are recursively defined. Suppose that we already evaluated the level $h-1$ relation $L_P(u, h-1) \sqsubseteq L_G(v, h-1)$ for all the pairs $\forall(u, v)$ from P and G. Because $F^{(h)} = L_P(u, h-1)$ and $F'^{(h)} = L_G(v, h-1)$, the condition $F^{(h)} \sqsubseteq F'^{(h)}$ is equivalent to $L_P(u, h-1) \sqsubseteq L_G(v, h-1)$, which is assumed to be already obtained on the level $h-1$ computation. Because $S_i^{(h)}$ and $S_i'^{(h)}$ are also from hierarchical labels on the level $h-1$, the condition $\exists\sigma(\land_{i\in[m]} S_i^{(h)} \sqsubseteq S'^{(h)}_{\sigma(i)})$ is also recursive. From the result of the level $h-1$ evaluations, we can obtain whether $S_i^{(h)} \sqsubseteq S_j'^{(h)}$ holds for $\forall(i, j)$. Then, the evaluation of the condition $\exists\sigma(\land_{i\in[n]} S_i^{(h)} \sqsubseteq S'^{(h)}_{\sigma(i)})$ is reduced to a matching problem from $i \in [m]$ to $j \in [n]$. This problem can be simply transformed into a *maximum bipartite matching* problem for a pair of $\{S_1^{(h)}, \dots, S_n^{(h)}\}$ and $\{S_1'^{(h)}, \dots, S_m'^{(h)}\}$, where edges exist on a set of pairs $\{(i, j) \mid S_i^{(h)} \sqsubseteq S_j'^{(h)}\}$. When the number of the maximum matching is equal to m, this means that there exists an injection $\sigma(i)$ that satisfies $\land_{i\in[m]} S_i^{(h)} \sqsubseteq S'^{(h)}_{\sigma(i)}$. It is well known that the maximum bipartite matching can be reduced to *maximum-flow problem*, and that it can be solved by the polynomial time (Goldberg and Tarjan, 1988). An example of the inclusion relationship is shown in Figure 5 (b).

Let $|P|$ and $|G|$ be the numbers of vertices in P and G. Then, multisets of the level T hierarchical labels of all the vertices in P and G are written as $[L_P(u_i, T)]_{i \in [|P|]} \coloneqq [L_P(u_1, T), L_P(u_2, T), \dots, L_P(u_{|P|}, T)]$ and $[L_G(v_i, T)]_{i \in [|G|]} \coloneqq [L_G(v_1, T), L_G(v_2, T), \dots, L_G(v_{|G|}, T)]$, respectively. For a feature of a given input graph G, we define the *approximate subgraph isomorphism feature (ASIF)* as follows

$$x_{P \sqsubseteq G} \coloneqq \begin{cases} 1, & \text{if } \exists \sigma (\wedge_{i \in [|P|]} L_P(u_i, T) \sqsubseteq L_G(v_{\sigma(i)}, T)), \\ 0, & \text{otherwise.} \end{cases} \tag{28}$$

This feature approximately evaluates the existence of a subgraph P in G using the level T hierarchical labels. ASIF satisfies the monotone decreasing property (12), i.e., $x_{P' \sqsubseteq G} \leq x_{P \sqsubseteq G}$ if $P' \sqsupseteq P$, because the number of conditions in (27) only increases when P grows.

To incorporate a label dissimilarity information (like Figure 4) into ASIF, we first extend the label inclusion relation (26) by using a concept of *optimal transportation cost.* As a label-similarity based relaxed evaluation of $L_P(v, h) \sqsubseteq L_G(u, h)$, we define an asymmetric cost between $L_P(u, h)$ and $L_G(v, h)$ as follows

$$\text{cost}_h(L_P(u, h) \to L_G(v, h)) \coloneqq \begin{cases} \text{dissimilarity}(F^{(h)}, F'^{(h)}), & \text{if } S^{(h)} = S'^{(h)} = [\,], \quad \text{(29a)} \\ \text{cost}_{h-1}(F^{(h)} \to F'^{(h)}) + & \\ \qquad \text{LTC}(S^{(h)} \to S'^{(h)}, \text{cost}_{h-1}), & \text{otherwise}, \quad \text{(29b)} \end{cases}$$

where the second term of (29b) is

$$\text{LTC}(S^{(h)} \to S'^{(h)}, \text{cost}_{h-1}) \coloneqq \min_{\sigma \in \mathcal{I}} \sum_{i \in [m]} \text{cost}_{h-1}(S_i^{(h)} \to S'^{(h)}_{\sigma(i)}), \tag{30}$$

which we refer to as the *label transportation cost* (LTC) representing the optimal transportation from the multiset $S^{(h)}$ to another multiset $S'^{(h)}$ among the set of all injections $\mathcal{I} \coloneqq \{\forall \sigma : [m] \to [n] \mid \sigma(i) \neq \sigma(j) \text{ for } i \neq j\}$. The equation (29) has a recursive structure that is similar to (26). The first case (29a) occurs when $S^{(h)} = S'^{(h)} = [\,]$, which takes place at the first level $h = 1$. In this case, cost_1 is defined by $\text{dissimilarity}(F^{(1)}, F'^{(1)})$, which is directly obtained as a dissimilarity between original labels since $F^{(1)}$ and $F'^{(1)}$ stem from the original vertex-labels. In the other case (29b), the cost is recursively defined as the sum of cost from $F^{(h)}$ to $F'^{(h)}$ and the optimal-transport cost from $S^{(h)}$ to $S'^{(h)}$. Although the definition is recursive, as in the case of ASIF, the evaluation can be performed by computing sequentially from $h = 1$ to $h = T$. Because $F^{(h)} = L_P(v, h-1)$ and $F'^{(h)} = L_G(u, h-1)$, the first term $\text{cost}_{h-1}(F^{(h)} \to F'^{(h)})$ represents the cost between hierarchical labels on the level $h - 1$, which is assumed to be already obtained. The second term $\text{LTC}(S^{(h)} \to S'^{(h)}, \text{cost}_{h-1})$ evaluates the best match between $[S_1^{(h)}, \dots, S_m^{(h)}]$ and $[S_1'^{(h)}, \dots, S_n'^{(h)}]$, as defined in (30). This matching problem can be seen as an optimal transportation problem, which minimizes the cost of the transportation of m items to n warehouses under the given cost matrix specified by cost_{h-1}. The values of cost_{h-1} for all the pairs in $[m]$ and $[n]$ are also available from the computation of the level $h-1$.

For the given cost values, the problem $\mathrm{LTC}(S^{(h)} \to S'^{(h)}, \mathrm{cost}_{h-1})$ can be reduced to *minimum-cost-flow problem* on a bipartite graph with a weight $\mathrm{cost}_{h-1}(S_i^{(h)} \to S_j'^{(h)}, \mathrm{cost}_{h-1})$ between $S_i^{(h)}$ and $S_j'^{(h)}$, and it can be solved in polynomial time (Goldberg and Tarjan, 1988).

We define an asymmetric transport cost for two graphs P and G, which we call the *graph transportation cost* (GTC), as LTC from all the level T hierarchical labels of P to those of G:

$$\mathrm{GTC}(P \to G) \coloneqq \mathrm{LTC}([L_P(u_i, T)]_{i \in [|P|]} \to [L_G(v_i, T)]_{i \in [|G|]}, \mathrm{cost}_T).$$

Then, as a feature of the input graph G, we define the following *sim-ASIF*:

$$x_{P \to G} \coloneqq \exp\{-\rho\, \mathrm{GTC}(P \to G)\}, \tag{31}$$

where $\rho > 0$ is a hyperparameter. This sim-ASIF can be regarded as a generalization of (28) based on the vertex-label similarity. When $\mathrm{dissimilarity}(F^{(1)}, F'^{(1)}) \coloneqq \infty \times 1_{F^{(1)} \neq F'^{(1)}}$, the feature (31) is equivalent to (28). Similarly to ASIF, $\mathrm{GTC}(P \to G)$ satisfies the monotonicity property

$$\mathrm{GTC}(P \to G) \leq \mathrm{GTC}(P' \to G) \text{ for } P' \sqsupseteq P$$

because the number of vertices to transport increases as P grows. Therefore, sim-ASIF (31) satisfies the monotonicity property: $x_{P' \to G} \leq x_{P \to G}$ if $P' \sqsupseteq P$.

From the definition (31), sim-ASIF always has a positive value $x_{P \to G} > 0$ except for the case $\mathrm{GTC}(P \to G) = \infty$, which may not be suitable for identifying a small number of important subgraphs. Further, in sim-ASIF, the bipartite graph in the minimum-cost-flow calculation $\mathrm{LTC}(S \to S', \mathrm{cost}_{h-1})$ is always a complete bipartite graph, where all the vertices in S are connected to all the vertices in S'. Because the efficiency of most of standard minimum-cost-flow algorithms depends on the number of edges, this may require a large computational cost. As an extension that mitigates these issues, a threshold can be introduced into sim-ASIF as follows:

$$x \coloneqq \begin{cases} \exp\{-\rho\, \mathrm{GTC}(P \to G)\}, & \exp\{-\rho\, \mathrm{GTC}(P \to G)\} > t \\ 0, & \exp\{-\rho\, \mathrm{GTC}(P \to G)\} \leq t \end{cases}, \tag{32}$$

where $t > 0$ is a threshold parameter. In this definition, x is 0 when $\exp\{-\rho\, \mathrm{GTC}(P \to G)\} \leq t$, i.e., $\mathrm{GTC}(P \to G) \geq -(\log t)/\rho$. This indicates that if a cost is larger than $-(\log t)/\rho$, we can regard the cost as ∞. Therefore, at any h, if the cost between $S_i^{(h)}$ and $S_j'^{(h)}$ is larger than $-(\log t)/\rho$, the edge between $S_i^{(h)}$ and $S_j'^{(h)}$ is not necessary. Then, the number of matching pairs can be less than m in $\mathrm{LTC}(\cdot)$ because of the lack of edges, and in this case, the cost is regarded as ∞. Furthermore, if $\mathrm{cost}_h(F^{(h)} \to F'^{(h)})$ is larger than $-(\log t)/\rho$ in (29b), the computation of $\mathrm{LTC}(S^{(h)} \to S'^{(h)}, \mathrm{cost}_{h-1})$ is not needed because $x = 0$ is determined.

Note that the transportation-based graph metric has been studied (e.g., Titouan et al., 2019), but the purpose of those studies is to evaluate the similarity between two graphs (not inclusion). Our (sim-)ASIF provides a feature with the monotonicity property as a natural relaxation of subgraph isomorphism, by which the optimality of our pruning strategy can be guaranteed. In contrast, there are many studies for the inexact graph matching (Yan et al., 2016) such as eigenvector (Leordeanu et al., 2012; Kang et al., 2013)-, edit distance (Gao et al., 2010)-, and random walk (Gori et al., 2005; Cho et al., 2010)- based methods. Some of them provide a score of the matching which can be seen as a similarity score between a searched graph pattern and a matched graph. However, those studies did not guarantee monotonicity of the similarity score for pattern growth. If the similarity score satisfies monotonicity, it can be combined with IGML. Although we only describe the vertex-label, the edge-label can also be incorporated into (sim-)ASIF. A simple approach is to transform a labeled-edge into a labeled-node with two unlabeled edges, such that (sim-)ASIF is directly applicable.

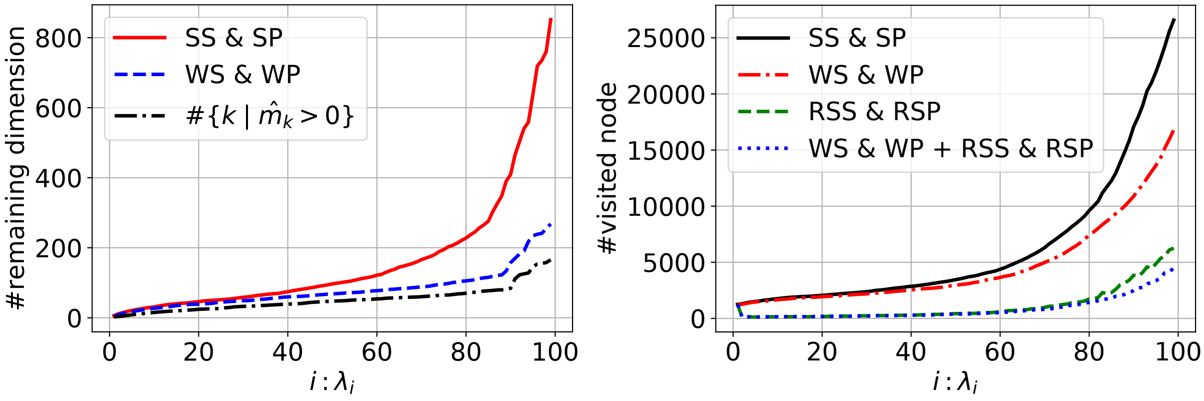
7 Experiments

We evaluate the performance of IGML using the benchmark datasets shown in Table 2. These datasets can be obtained from Kersting et al. (2016). We did not use the edge label because implementations of compared methods cannot deal with the edge label, and the maximum connected graph is used if the graph is not connected. #maxvertices in the table is the size (number of vertices) of the maximum subgraph considered in IGML. The sets $\mathcal{S}_i$ and $\mathcal{D}_i$ are respectively selected as the ten nearest neighborhoods of $\boldsymbol{x}_i$ ($K = |\mathcal{S}_i| = |\mathcal{D}_i| = 10$) by using the WL-Kernel. A sequence of the regularization coefficients is created by equally spaced 100 grid points on the logarithmic scale between $\lambda_{\max}$ and $0.01\lambda_{\max}$. The gSpan search tree (where the minimum support is set as 0) is usually traversed only just after λ changes. In the working-set method, after convergence, it is necessary to traverse the tree again in order to confirm the overall optimality. If the termination condition is not satisfied, optimization with a new working set must be performed. The termination condition for the optimization is that the relative duality gap is less than 10^{-6}. In this experiment, we used $g(x) = 1_{x>0}$ in $\phi_H(G)$ with Lemma 4.2 unless otherwise noted. The dataset is randomly divided in such a way that the ratio of partitioning is train : validation : test = 0.6 : 0.2 : 0.2, and our experimental result has an average value of 10 times.

7.1 Evaluating Computational Efficiency

In this section, we confirm the effect of proposed pruning methods. We evaluated four settings, i.e., safe feature screening "SS&SP", its range based extention "RSS&RSP", working set selection "WS&WP", and the combination "WS&WP+RSS&RSP". Each method performs dynamic screening with DGB at every update of $\boldsymbol{m}$. We here used the AIDS dataset, where #maxvertices=30. In this dataset, when we fully traverse the gSpan tree without safe screening/working set selection, the number of tree nodes was more than 9.126×10^7 (where our implementation with gSpan stopped because we ran out of memory).

Figure 6 (a) shows the size of $\mathcal{F}$ after the first traverse at each λ, and the number of non-zero m_k after the optimization is also shown as a baseline. We first observe that both approaches drastically reduced the



(a)

(b)

Figure 6: (a): Size of $\mathcal{F}$, and (b): number of visited nodes. Both are evaluated at the first traverse of each λ, where the index is shown as the horizontal axis. The dataset employed here is AIDS.

number of features. Even for the largest case, where about 200 of features were finally selected by m_k, only less than a thousand of features remained. We observe that WS&WP exhibits significantly smaller values than SS&SP. Instead, WS&WP may need to perform the entire tree search again because it cannot guarantee the sufficiency of current $\mathcal{F}$ while SS&SP does not need to search the tree again because it guarantees that $\mathcal{F}$ must contain all $m_k \neq 0$.

The number of visited nodes in the first traverse at each λ is shown in Figure 6 (b). We observe that the #visited nodes of SS&SP is the largest, but it is less than about 27000 ($27000/9.126 \times 10^7 \approx 0.0003$). Comparing SS&SP and WS&WP, we see that WS&WP pruned a larger number of nodes. In contrast, the #visited nodes of RSS&RSP is less than 6000. The difference between SS&SP and RSS&RSP indicates that a larger number of nodes can be skipped by the range based method. Therefore, by combining the node skip by RSS&RSP with stronger pruning of WS&WP, the #visited nodes was further reduced.

The total time in the path-wise optimization is shown in Table 1. RSS&RSP were fast with regard to the traversing time, and WS&WP were fast with regard to the solving time. In total, WS&WP+RSS&RSP was fastest. The result indicates that our method only took about 1 minute to solve the optimization problem with more than 9.126×10^7 variables. We also show the computational cost evaluation for other datasets in the Appendix I.

7.2 Predictive Accuracy Comparison

In this section, we compare the prediction accuracy of IGML with the Graphlet-Kernel (GK)(Shervashidze et al., 2009), Shortest-Path Kernel (SPK)(Borgwardt and Kriegel, 2005), Random-Walk Kernel (RW)(Vishwanathan et al., 2010), Weisfeiler-Lehman Kernel (WL)(Shervashidze et al., 2011), and Deep Graph Convolutional Neu-

Table 1: Total time in path-wise optimization (sec) on AIDS dataset.

Method \ Process	Traverse	Solve	Total
SS&SP	25.9		112.7
	±4.0	86.7	±16.5
RSS&RSP	7.7	±14.1	94.4
	±1.6		±15.1
WS&WP	39.1		94.1
	±3.7	**55.0**	±11.6
WS&WP+	**7.4**	±12.1	**62.5**
RSS&RSP	±1.1		±12.3

ral Network (DGCNN)(Zhang et al., 2018a). We use the implementation available at URLs [1] for the graph kernels and DGCNN, respectively. The prediction of the kernel method and IGML is made by the k-nearest neighbor (k-nn). The values of k in the k-nn is $k = 1, 3, 5, 7, ..., 49$ and hyperparameters of each method are selected by using the validation data, and the prediction accuracy is evaluated on the test data. The graphlet size in GK is fixed to three. The parameter λ_{RW} in RW is set by the recommended $\lambda_{\text{RW}} = \max_{i \in \mathbb{Z}: 10^i < 1/d^2} 10^i$, where d denotes the maximum degree. The loop parameter h of WL is selected from $0, 1, 2, ..., 10$ by using the validation data. In DGCNN, the number of hidden units and their sort-pooling is also selected by the validation data, each ranging from $64, 128, 256$ and from $40\%, 60\%, 80\%$, respectively.

The micro-F1 score for each dataset is shown in Table 2. “IGML (Diag)” is IGML with the weighted squared distance (1), and “IGML (Diag→Full)” indicates that post-processing with the Mahalanobis distance (22) was performed. We first focus on “IGML (Diag)”, which yields the best score on six out of nine datasets except for “IGML (Diag→Full)”. Among those six datasets, “IGML (Diag→Full)” further improves the accuracy for the four datasets. The second best would be WL, which exhibits superior performance compared to the other methods except for the proposed method on six out of nine datasets. DGCNN shows high accuracy with DBLP_v1, which has a large number of samples, while for the other data, the accuracy was low.

7.3 Illustrative Examples of Selected Subgraphs

Figure 7 shows an illustrative example of IGML on the Mutagenicity dataset, where mutagenicity is predicted from a graph representation of molecules. Figure 7 (a) is a graphical representation of subgraphs, each of which has a weight shown in (b). For example, we can clearly see that the subgraph #2 is estimated as an important sub-structure to discriminate different classes. Figure 7 (c) shows a heatmap of the transformation matrix $\sqrt{\boldsymbol{\Lambda}} \boldsymbol{V}^\top$ optimized for these thirteen features, containing three non-zero eigenvalues. For example, we see that two subgraphs #10 and #12 have similar columns in the heatmap. This indicates that these

[1] `http://mlcb.is.tuebingen.mpg.de/Mitarbeiter/Nino/Graphkernels/`, and `https://github.com/muhanzhang/pytorch_DGCNN`

Table 2: Comparison of micro-F1 score. OOM means out of memory. ">1week" indicates that the algorithm was running for more than a week. "±" is the standard deviation. Every dataset has two classes.

Method \ Dataset	AIDS	BZR	DD	DHFR	FRANKENSTEIN	Mutagenicity	NCI1	COX2	DBLP_v1
#samples	2000	405	1178	467	4337	4337	4110	467	19456
#maxvertices	30	15	30	15	15	10	15	15	30
GK	0.985	0.815	0.632	0.688	0.603	0.747	0.703	0.782	OOM
	±0.006	±0.034	±0.021	±0.037	±0.012	±0.017	±0.011	±0.045	
SPK	0.994	0.842	>1week	0.737	0.640	0.719	0.722	0.774	0.784
	±0.003	±0.039		±0.040	±0.012	±0.014	±0.012	±0.034	±0.012
RW	**0.998**	0.811	OOM	0.659	0.616	0.679	0.649	0.770	OOM
	±0.002	±0.025		±0.032	±0.013	±0.018	±0.017	±0.038	
WL	0.995	0.854	0.769	0.780	0.694	0.768	0.772	**0.790**	0.814
	±0.003	±0.039	±0.027	±0.045	±0.017	±0.012	±0.015	±0.040	±0.014
DGCNN	0.985	0.791	0.773	0.678	0.615	0.705	0.706	0.764	**0.927**
	±0.005	±0.020	±0.023	±0.030	±0.016	±0.018	±0.016	±0.039	±0.003
IGML (Diag)	0.976	**0.860**	0.778	**0.797**	0.696	0.783	0.775	0.777	0.860
	±0.006	±0.030	±0.026	±0.035	±0.014	±0.016	±0.012	±0.037	±0.005
IGML (Diag→Full)	0.977	0.830	**0.783**	0.794	**0.699**	**0.790**	**0.782**	0.773	0.856
	±0.008	±0.029	±0.022	±0.042	±0.013	±0.023	±0.014	±0.038	±0.005

two similar subgraphs (#10 contains #12) are shrunk to almost same representation by the effect of the regularization term $R(\boldsymbol{M})$.

As another example of graph data analysis on the learned representation, we applied the decision tree algorithm to the obtained feature (23) on the Mutagenicity dataset. Although there is a study constructing the decision tree directly for graph data (Nguyen et al., 2006), it requires a severe restriction on the pattern to be considered for computational reasons. In contrast, since (23) is a simple vector representation with a reasonable dimension, it is quite easy to apply the decision tree algorithm. Because of space limitation, we select two paths from the obtained decision tree as shown in Figure 8. For example, in the path (a), if a given graph contains "$\boldsymbol{O} = \boldsymbol{N}$", and does not contain "$\boldsymbol{H} - \boldsymbol{O} - \boldsymbol{C} - \boldsymbol{C} = \boldsymbol{C} - \boldsymbol{C} - \boldsymbol{H}$", and contains "$\boldsymbol{N} - \boldsymbol{C} = \boldsymbol{C} - \boldsymbol{C} = \boldsymbol{C} <^{\boldsymbol{C}}_{\boldsymbol{C}}$", the given graph is predicted as $y = 0$ with probability 140/146. Both rules clearly separate the two classes, which is highly insightful as we can trace the process of the decision based on the subgraphs.

7.4 Experiments for Three Extensions

In this section, we evaluates the performance of the three extensions of IGML described in Section 6.

First, we evaluate the performance of IGML for item-set/sequence data by using the benchmark datasets shown in the first two rows in Table 3-4, respectively. These datasets can be obtained from (Dua and Graff, 2017) and (Chang and Lin, 2011). We set the maximum-pattern size considered in IGML as 30. Table 3

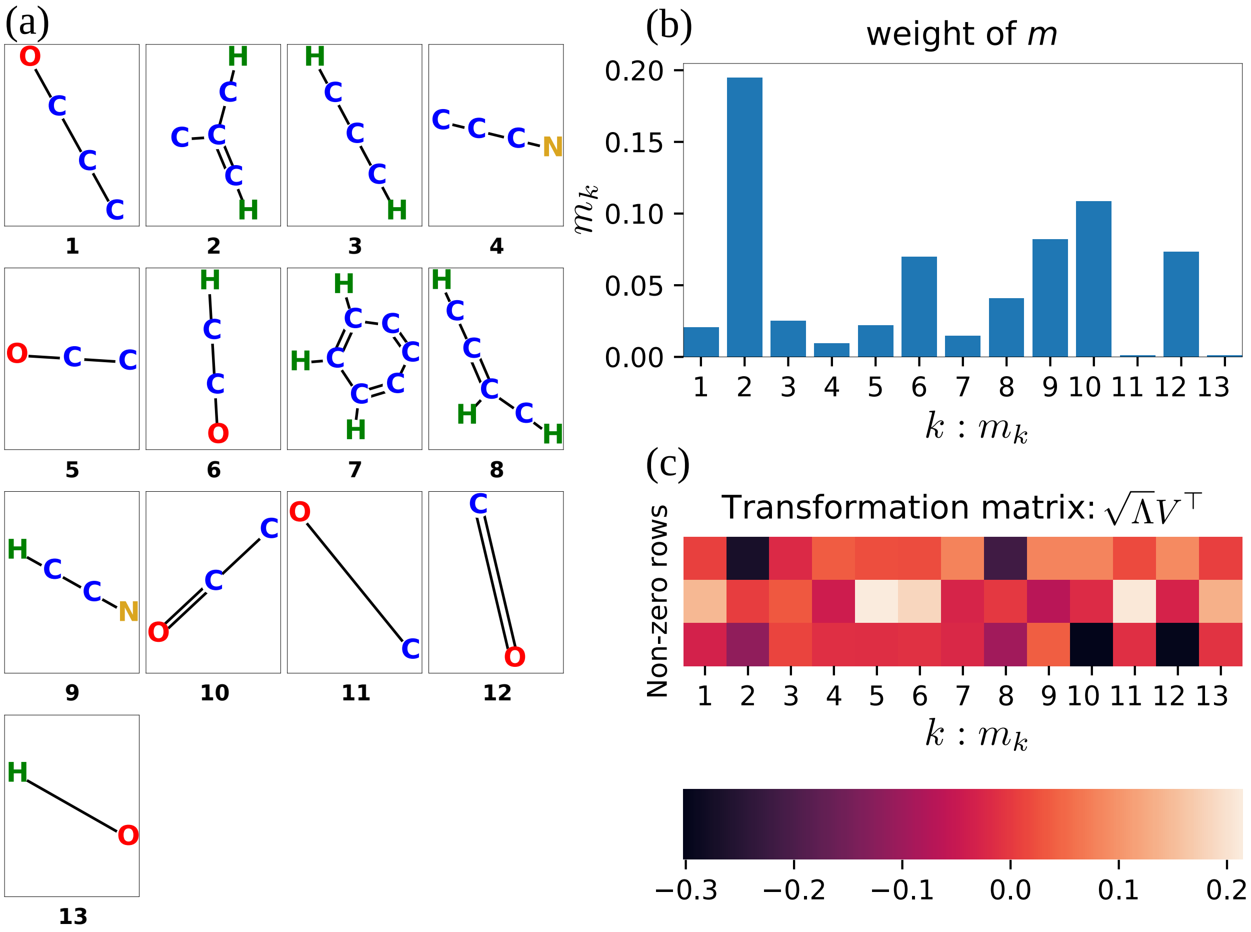


Figure 7: An example of selected subgraphs. (a): Illustration of subgraphs. (b): Learned weight of subgraph. (c): A transformation matrix (24).

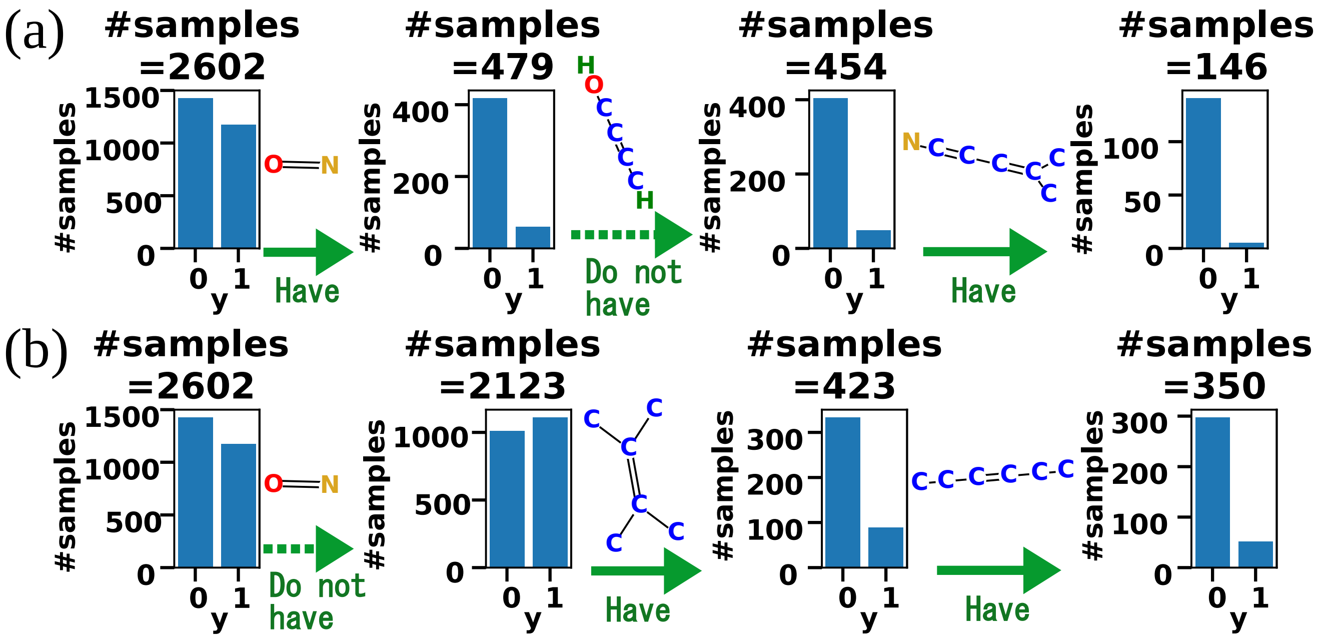


Figure 8: Examples of paths on decision tree constructed by selected subgraphs. #samples indicates number of samples satisfying all preceding conditions.

Table 3: Micro-F1 score on item-set dataset.

Method \ Dataset	dna	car	nursery
#samples	2000	1728	12960
Jaccard Similarity	0.860±0.017	0.888±0.020	0.961±0.006
IGML (Diag)	0.908±0.014	0.936±0.011	0.982±0.005
IGML (Diag→Full)	**0.931**±0.009	**0.948**±0.014	**0.993**±0.002

shows the micro-F1 score on the item-set dataset. We used k-nn with the Jaccard similarity as a baseline, in which k was selected by using the validation set as we performed in Section 7.2. The scores of both of IGML (Diag) and (Diag→Full) were superior to those of the Jaccard similarity on all the datasets. Table 4 shows the micro-F1 score on the sequence dataset. Although IGML (Diag) did not outperform the mismatch kernel (Leslie et al., 2004) for the promoters dataset, IGML (Diag→Full) achieved a higher F1-score than the kernel on all the datasets. Figure 9 shows an illustrative example of identified sequences by IGML on the promoters dataset, where the task is to predict whether an input DNA sequence stem from a promoter region. Figure 9 (a) is a graphical representation of the sequence, with the corresponding weights shown in (b). For example, the sub-sequence #1 in (a) can be considered as an important sub-sequence to discriminate different classes.

Second, we show results of the triplet formulation described in Section 6.2. To create the triplet set $\mathcal{T}$, we followed the approach by Shen et al. (2014), where k neighborhoods in the same class $\boldsymbol{x}_j$ and k neighborhoods in different classes $\boldsymbol{x}_l$ are sampled for each $\boldsymbol{x}_i$ ($k = 4$). Here, IGML with the pairwise-loss is referred to as

Table 4: Micro-F1 score on sequence dataset.

Method \ Dataset	promoters	splice
#samples	106	3190
Mismatch Kernel	0.832±0.081	0.596±0.017
IGML (Diag)	0.800±0.104	0.651±0.015
IGML (Diag→Full)	**0.886**±0.068	**0.694**±0.017

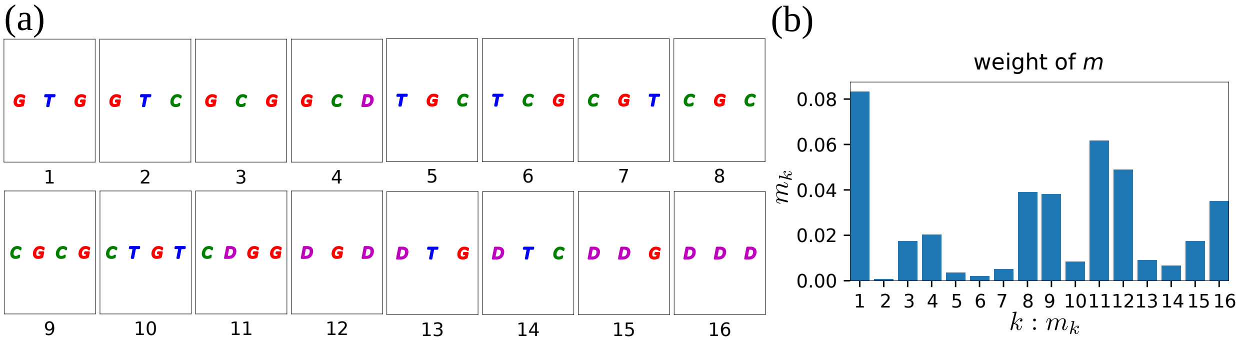


Figure 9: Examples of (a) selected sequences and (b) their weights (promoters dataset).

'IGML (Pairwise)', and IGML with the triplet-loss is referred to as 'IGML (Triplet)'. Table 5 compares the micro-F1 of IGML (Pairwise) and IGML (Triplet). The IGML (Triplet) showed higher F1-scores than IGML (Pairwise) in three among nine datasets, but it was not computable on the two datasets due to having ran out of memory (OOM). This is because the pruning rule of the triplet case (25) was looser than the pair-wise case.

Finally, we evaluate sim-ASIF (32). We set the scaling factor of the exponential function as $\rho = 1$, the threshold of the feature as $t = 0.7$, and the number of re-labeling steps as $T = 3$. Here, we employed a simple heuristic approach to create a dissimilarity matrix among vertex-labels by using labeled graphs in the given dataset. Suppose that a set of possible vertex-labels is $\mathcal{L}$, and $f(\ell, \ell')$ is the frequency that $\ell \in \mathcal{L}$ and $\ell' \in \mathcal{L}$ are adjacent in all graphs of the dataset. Subsequently, by concatenating $f(\ell, \ell')$ for all $\ell' \in \mathcal{L}$, we obtain a vector representation of a label ℓ. We normalize this vector representation, such that the vector has the unit L2 norm. By calculating the Euclidean distance of this normalized representations, we obtain the dissimilarity matrix of vertex-labels. We are particularly interested in the case where the distribution of vertex-label frequency is largely different between the training and test datasets, as in this case the exact

Table 5: Comparison of pairwise with triplet on micro-F1 score.

Method \ Dataset	AIDS	BZR	DD	DHFR	FRANKENSTEIN	Mutagenicity	NCI1	COX2	DBLP_v1
IGML (Pairwise)	**0.976**	**0.860**	**0.778**	0.797	**0.696**	0.783	0.775	**0.777**	**0.860**
from Table 2	±0.006	±0.030	±0.026	±0.035	±0.014	±0.016	±0.012	±0.037	±0.005
IGML (Triplet)	0.968	0.844	OOM	**0.811**	0.693	**0.808**	**0.782**	0.765	OOM
#maxvertices=10	±0.012	±0.032		±0.033	±0.013	±0.012	±0.013	±0.042	

Table 6: Evaluating sim-ASIF with micro-F1 score. The training and test sets of these datasets were split using a clustering algorithm such that the distribution of vertex-labels can be largely different.

#maxvertices	Feature\Dataset	AIDS	Mutagenicity	NCI1
According to Table 2	Normal	0.574±0.039	**0.720**±0.014	0.735±0.025
8	Normal	0.572±0.038	0.705±0.017	0.726±0.019
8	sim-ASIF (32)	**0.663**±0.033	0.702±0.016	**0.755**±0.017

matching of IGML may not be suitable to provide the prediction. We synthetically emulate this setting by splitting training and test datasets through a clustering algorithm. Each input graph is transformed into a vector created by the frequencies of each one of the vertex-label $\ell \in \mathcal{L}$ contained in that graph. Subsequently, we apply the k-means clustering, by which the dataset is split into two clusters $\mathcal{C}_1$ and $\mathcal{C}_2$. We used $\mathcal{C}_1$ as the training and validation datasets, and $\mathcal{C}_2$ is used as the test dataset. Table 6 shows the comparison of the micro-F1 scores on the AIDS, Mutagenicity, and NCI1 datasets (we did not employ other datasets because the sizes of the training sets created from the clustering procedure were too small). We fixed the #maxvertices of sim-ASIF by 8, which is less than the value in our original IGML evaluation Table 2, because sim-ASIF takes more time than the feature without vertex-label similarity. For the original IGML, we show the result of the previous setting in Table 2 and the results with #maxvertices 8. IGML with sim-ASIF was superior to the original IGML for the both #maxvertices settings on the AIDS and NCI1 datasets, although it has smaller #maxvertices settings, as shown in Table 6. For the Mutagenicity dataset, sim-ASIF was inferior to the original IGML of Table 2, but in the comparison under the same #maxvertices value, their performance was comparable. These results suggest that when the exact matching of the subgraph is not appropriate, sim-ASIF can improve the prediction performance of IGML.

7.5 Performance on Frequency Feature

In this section, we evaluate IGML with $g(x) = \log(1+x)$ instead of $g(x) = 1_{x>0}$. Note that because computing the frequency without overlap $\#(H \sqsubseteq G)$ is NP-complete (Schreiber and Schwöbbermeyer, 2005), in addition to the exact count, we evaluated the feature defined by an upper bound of $\#(H \sqsubseteq G)$ (see Appendix J for the details). We employed log, because the scale of the frequency x is highly diversified. From the result in Section 7.1, we use WS&WP+RSS&RSP in this section. The #maxvertices for each dataset follows Table 2.

The comparison of micro-F1 scores for the exact $\#(H \sqsubseteq G)$ and the approximation of $\#(H \sqsubseteq G)$ is shown in Table 7. The exact $\#(H \sqsubseteq G)$ did not complete the five datasets mainly due to the computational difficulty of the frequency counting. In contrast, the approximate $\#(H \sqsubseteq G)$ completed on all datasets. Overall, for both the exact and approximate frequency features, the micro-F1 scores were comparable with the case of $g(x) = 1_{x>0}$ shown in Table 2.

Table 8 shows the total time in the path-wise optimization for the exact $\#(H \sqsubseteq G)$ and the approximation

Table 7: Micro-F1 score in case of $g(x) = \log(1 + x)$.

Method \ Dataset	AIDS	BZR	DD	DHFR	FRANKENSTEIN	Mutagenicity	NCI1	COX2	DBLP_v1
exact	-	0.833	-	0.802	-	-	-	0.769	0.858
$\#(H \sqsubseteq G)$		±0.045		±0.031				±0.030	±0.005
approximation	0.982	0.842	0.772	0.791	0.690	0.779	0.762	0.769	0.858
of $\#(H \sqsubseteq G)$	±0.005	±0.049	±0.026	±0.046	±0.013	±0.010	±0.015	±0.042	±0.005

Table 8: Total time in path-wise optimization (sec) in case of $g(x) = \log(1 + x)$.

Dataset	AIDS			BZR		
Method \ Process	Traverse	Solve	Total	Traverse	Solve	Total
exact $\#(H \sqsubseteq G)$	> a day			1662.2±93.0	93.0±19.4	1755.2±213.5
approximation of $\#(H \sqsubseteq G)$	8.6±1.4	14.5±1.4	23.1±1.9	236.0±26.1	13.0± 3.1	249.0± 28.9

of $\#(H \sqsubseteq G)$. On the AIDS dataset, the experiment using exact $\#(H \sqsubseteq G)$ did not complete within a day while the traverse time using approximate $\#(H \sqsubseteq G)$ was only 8.6 sec. On the BZR dataset, the traverse time using the exact $\#(H \sqsubseteq G)$ was seven times that using the approximate $\#(H \sqsubseteq G)$. The solving time for the approximation was lower, because the $|\mathcal{F}|$ after traversing of the approximation was significantly less than that of the exact $\#(H \sqsubseteq G)$ in this case. Since the approximate $\#(H \sqsubseteq G)$ is an upper bound of the exact $\#(H \sqsubseteq G)$, the variation of the values of the exact $\#(H \sqsubseteq G)$ was smaller than the approximate $\#(H \sqsubseteq G)$. This resulted in higher correlations among features created by the exact $\#(H \sqsubseteq G)$. It is known that the elastic-net regularization tends to select correlated features simultaneously (Zou and Hastie, 2005), and therefore, $|\mathcal{F}|$ in the case of the exact $\#(H \sqsubseteq G)$ becomes larger than in the approximate case.

Figure 10 shows the number of visited nodes, size of the feature subset $|\mathcal{F}|$ after the traverse, and the number of selected features on the AIDS dataset with the approximate $\#(H \sqsubseteq G)$. This indicates that IGML keeps the number of subgraphs tractable even if $g(x) = \log(1+x)$ is used as the feature. The #visited nodes is less than only about 3500, and $|\mathcal{F}|$ after traversing is sufficiently close to $|\{k \mid \hat{m}_k > 0\}|$.

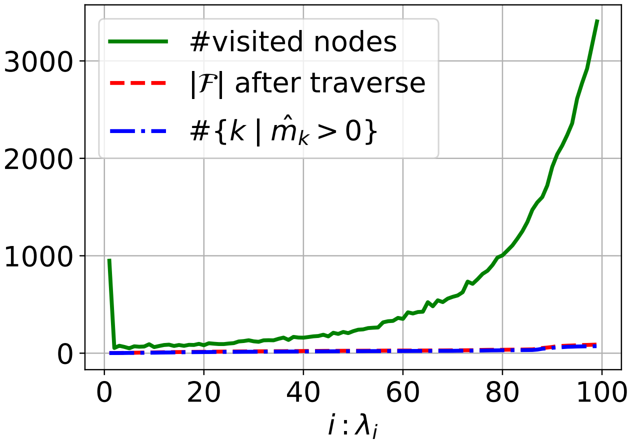


Figure 10: The result of IGML with $g(x) = \log(1 + x)$ on AIDS dataset.

8 Conclusions

We proposed an interpretable metric learning method for graph data, named *interpretable graph metric learning* (IGML). To avoid computational difficulty, we build an optimization algorithm that combines safe screening, working set selection, and their pruning extensions. We also discussed the three extensions of IGML including (a) applications to other structured data, (b) triplet loss-based formulation, and (c) incorporating vertex-label similarity into the feature. We empirically evaluated the performance of IGML compared with the existing graph classification methods. Although IGML was the only method that has clear interpretability, it showed superior or comparable prediction performance compared to other state-of-the-art methods. Further, the practicality of IGML was also shown through some illustrative examples of identified subgraphs.

Acknowledgments

This work was supported by MEXT KAKENHI to I.T. (16H06538, 17H00758) and M.K. (16H06538, 17H04694); from JST CREST awarded to I.T. (JPMJCR1302, JPMJCR1502) and PRESTO awarded to M.K. (JPMJPR15N2); from the Collaborative Research Program of Institute for Chemical Research, Kyoto University to M.K. (grant #2018-33); from the MI2I project of the Support Program for Starting Up Innovation Hub from JST awarded to I.T., and M.K.; and from RIKEN Center for AIP awarded to I.T.

References

Adhikari, B., Zhang, Y., Ramakrishnan, N., and Prakash, B. A. (2018). Sub2vec: Feature learning for subgraphs. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 170–182. Springer.

Agrawal, R., Srikant, R., et al. (1994). Fast algorithms for mining association rules. In *Proc. 20th int. conf. very large data bases, VLDB*, volume 1215, pages 487–499.

Atwood, J. and Towsley, D. (2016). Diffusion-convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1993–2001.

Bellet, A., Habrard, A., and Sebban, M. (2012). Good edit similarity learning by loss minimization. *Machine Learning*, 89(1-2):5–35.

Borgwardt, K. M. and Kriegel, H.-P. (2005). Shortest-path kernels on graphs. In *Proceedings of the Fifth IEEE International Conference on Data Mining (ICDM 2005)*, pages 74–81. IEEE Computer Society.

Chang, C.-C. and Lin, C.-J. (2011). LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

Cheng, H., Yan, X., Han, J., and Philip, S. Y. (2008). Direct discriminative pattern mining for effective classification. In *2008 IEEE 24th International Conference on Data Engineering*, pages 169–178. IEEE.

Cho, M., Lee, J., and Lee, K. M. (2010). Reweighted random walks for graph matching. In *European conference on Computer vision*, pages 492–505. Springer.

Costa, F. and Grave, K. D. (2010). Fast neighborhood subgraph pairwise distance kernel. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, pages 255–262. Omnipress.

Davis, J. V., Kulis, B., Jain, P., Sra, S., and Dhillon, I. S. (2007). Information-theoretic metric learning. In *Proceedings of the 24th international conference on Machine learning*, pages 209–216. ACM.

Dua, D. and Graff, C. (2017). UCI machine learning repository. `http://archive.ics.uci.edu/ml`.

Duvenaud, D. K., Maclaurin, D., Iparraguirre, J., Bombarell, R., Hirzel, T., Aspuru-Guzik, A., and Adams, R. P. (2015). Convolutional networks on graphs for learning molecular fingerprints. In *Advances in neural information processing systems*, pages 2224–2232. Curran Associates, Inc.

Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R., and Lin, C.-J. (2008). LIBLINEAR: A library for large linear classification. *Journal of machine learning research*, 9:1871–1874.

Feragen, A., Kasenburg, N., Petersen, J., de Bruijne, M., and Borgwardt, K. (2013). Scalable kernels for graphs with continuous attributes. In *Advances in Neural Information Processing Systems*, pages 216–224.

Friedman, J., Hastie, T., Höfling, H., and Tibshirani, R. (2007). Pathwise coordinate optimization. *The Annals of Applied Statistics*, 1(2):302–332.

Gao, X., Xiao, B., Tao, D., and Li, X. (2010). A survey of graph edit distance. *Pattern Analysis and applications*, 13(1):113–129.

Gärtner, T., Flach, P., and Wrobel, S. (2003). On graph kernels: Hardness results and efficient alternatives. In *Learning Theory and Kernel Machines*, pages 129–143. Springer.

Ghaoui, L. E., Viallon, V., and Rabbani, T. (2010). Safe feature elimination for the lasso and sparse supervised learning problems. *arXiv:1009.4219*.

Goldberg, A. V. and Tarjan, R. E. (1988). A new approach to the maximum-flow problem. *Journal of the ACM (JACM)*, 35(4):921–940.

Gori, M., Maggini, M., and Sarti, L. (2005). Exact and approximate graph matching using random walks. *IEEE transactions on pattern analysis and machine intelligence*, 27(7):1100–1111.

Huang, X., Li, J., and Hu, X. (2017). Label informed attributed network embedding. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, pages 731–739.

Kang, U., Hebert, M., and Park, S. (2013). Fast and scalable approximate spectral graph matching for correspondence problems. *Information Sciences*, 220:306–318.

Kersting, K., Kriege, N. M., Morris, C., Mutzel, P., and Neumann, M. (2016). Benchmark data sets for graph kernels. `http://graphkernels.cs.tu-dortmund.de`.

Kondor, R. and Borgwardt, K. M. (2008). The skew spectrum of graphs. In *Proceedings of the 25th international conference on Machine learning*, pages 496–503. ACM.

Kondor, R. and Pan, H. (2016). The multiscale laplacian graph kernel. In *Advances in Neural Information Processing Systems*, pages 2990–2998.

Kondor, R., Shervashidze, N., and Borgwardt, K. M. (2009). The graphlet spectrum. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 529–536. ACM.

Kriege, N. and Mutzel, P. (2012). Subgraph matching kernels for attributed graphs. *arXiv preprint arXiv:1206.6483*.

Kuksa, P., Huang, P.-H., and Pavlovic, V. (2008). A fast, large-scale learning method for protein sequence classification. In *8th Int. Workshop on Data Mining in Bioinformatics*, pages 29–37.

Lee, J. B., Rossi, R., and Kong, X. (2018). Graph classification using structural attention. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1666–1674. ACM.

Leordeanu, M., Sukthankar, R., and Hebert, M. (2012). Unsupervised learning for graph matching. *International journal of computer vision*, 96(1):28–45.

Leslie, C., Eskin, E., and Noble, W. S. (2001). The spectrum kernel: A string kernel for svm protein classification. In *Biocomputing 2002*, pages 564–575. World Scientific.

Leslie, C. and Kuang, R. (2004). Fast string kernels using inexact matching for protein sequences. *Journal of Machine Learning Research*, 5(Nov):1435–1455.

Leslie, C. S., Eskin, E., Cohen, A., Weston, J., and Noble, W. S. (2004). Mismatch string kernels for discriminative protein classification. *Bioinformatics*, 20(4):467–476.

Li, D. and Tian, Y. (2018). Survey and experimental study on metric learning methods. *Neural Networks*, 105:447–462.

Morris, C., Kriege, N. M., Kersting, K., and Mutzel, P. (2016). Faster kernels for graphs with continuous attributes via hashing. In *Data Mining (ICDM), 2016 IEEE 16th International Conference on*, pages 1095–1100. IEEE.

Morvan, M. L. and Vert, J.-P. (2018). WHInter: A working set algorithm for high-dimensional sparse second order interaction models. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pages 3635–3644. PMLR.

Nakagawa, K., Suzumura, S., Karasuyama, M., Tsuda, K., and Takeuchi, I. (2016). Safe pattern pruning: An efficient approach for predictive pattern mining. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1785–1794. ACM.

Narayanan, A., Chandramohan, M., Venkatesan, R., Chen, L., Liu, Y., and Jaiswal, S. (2017). graph2vec: Learning distributed representations of graphs. *CoRR*, abs/1707.05005.

Neuhaus, M. and Bunke, H. (2007). Automatic learning of cost functions for graph edit distance. *Information Sciences*, 177(1):239–247.

Nguyen, P. C., Ohara, K., Mogi, A., Motoda, H., and Washio, T. (2006). Constructing decision trees for graph-structured data by chunkingless graph-based induction. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 390–399. Springer.

Niepert, M., Ahmed, M., and Kutzkov, K. (2016). Learning convolutional neural networks for graphs. In *International conference on machine learning*, pages 2014–2023.

Novak, P. K., Lavrač, N., and Webb, G. I. (2009). Supervised descriptive rule discovery: A unifying survey of contrast set, emerging pattern and subgroup mining. *Journal of Machine Learning Research*, 10:377–403.

Orsini, F., Frasconi, P., and De Raedt, L. (2015). Graph invariant kernels. In *Proceedings of the Twenty-fourth International Joint Conference on Artificial Intelligence*, pages 3756–3762.

Pei, J., Han, J., Mortazavi-Asl, B., Pinto, H., Chen, Q., Dayal, U., and Hsu, M.-C. (2001). Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth. In *Proceedings 17th international conference on data engineering*, pages 215–224. IEEE.

Pissis, S. P. (2014). Motex-ii: structured motif extraction from large-scale datasets. *BMC bioinformatics*, 15(1):235.

Pissis, S. P., Stamatakis, A., and Pavlidis, P. (2013). Motex: A word-based hpc tool for motif extraction. In *Proceedings of the International Conference on Bioinformatics, Computational Biology and Biomedical Informatics*, page 13. ACM.

Saigo, H., Nowozin, S., Kadowaki, T., Kudo, T., and Tsuda, K. (2009). gboost: a mathematical programming approach to graph classification and regression. *Machine Learning*, 75(1):69–89.

Schreiber, F. and Schwöbbermeyer, H. (2005). Frequency concepts and pattern detection for the analysis of motifs in networks. In *Transactions on computational systems biology III*, pages 89–104. Springer.

Shen, C., Kim, J., Liu, F., Wang, L., and Van Den Hengel, A. (2014). Efficient dual approach to distance metric learning. *IEEE transactions on neural networks and learning systems*, 25(2):394–406.

Shervashidze, N. and Borgwardt, K. M. (2009). Fast subtree kernels on graphs. In *Advances in neural information processing systems*, pages 1660–1668.

Shervashidze, N., Schweitzer, P., Leeuwen, E. J. v., Mehlhorn, K., and Borgwardt, K. M. (2011). Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research*, 12(Sep):2539–2561.

Shervashidze, N., Vishwanathan, S., Petri, T., Mehlhorn, K., and Borgwardt, K. (2009). Efficient graphlet kernels for large graph comparison. In *Artificial Intelligence and Statistics*, pages 488–495.

Simonovsky, M. and Komodakis, N. (2017). Dynamic edge-conditioned filters in convolutional neural networks on graphs. In *Proc. CVPR*.

Su, Y., Han, F., Harang, R. E., and Yan, X. (2016). A fast kernel for attributed graphs. In *Proceedings of the 2016 SIAM International Conference on Data Mining*, pages 486–494. SIAM.

Sugiyama, M. and Borgwardt, K. (2015). Halting in random walk kernels. In *Advances in neural information processing systems*, pages 1639–1647.

Thoma, M., Cheng, H., Gretton, A., Han, J., Kriegel, H.-P., Smola, A., Song, L., Yu, P. S., Yan, X., and Borgwardt, K. M. (2010). Discriminative frequent subgraph mining with optimality guarantees. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 3(5):302–318.

Titouan, V., Courty, N., Tavenard, R., Laetitia, C., and Flamary, R. (2019). Optimal transport for structured data with application on graphs. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97, pages 6275–6284. PMLR.

Tixier, A. J.-P., Nikolentzos, G., Meladianos, P., and Vazirgiannis, M. (2018). Graph classification with 2d convolutional neural networks.

Verma, S. and Zhang, Z.-L. (2017). Hunt for the unique, stable, sparse and fast feature learning on graphs. In *Advances in Neural Information Processing Systems*, pages 88–98.

Vishwanathan, S. V. N., Schraudolph, N. N., Kondor, R., and Borgwardt, K. M. (2010). Graph kernels. *Journal of Machine Learning Research*, 11(Apr):1201–1242.

Weinberger, K. Q. and Saul, L. K. (2009). Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research*, 10(Feb):207–244.

Yan, J., Yin, X.-C., Lin, W., Deng, C., Zha, H., and Yang, X. (2016). A short survey of recent advances in graph matching. In *Proceedings of the 2016 ACM on International Conference on Multimedia Retrieval*, pages 167–174.

Yan, X. and Han, J. (2002). gspan: Graph-based substructure pattern mining. In *Data Mining, 2002. ICDM 2003. Proceedings. 2002 IEEE International Conference on*, pages 721–724. IEEE.

Yanardag, P. and Vishwanathan, S. (2015). Deep graph kernels. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1365–1374. ACM.

Yoshida, T., Takeuchi, I., and Karasuyama, M. (2018). Safe triplet screening for distance metric learning. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 2653–2662.

Yoshida, T., Takeuchi, I., and Karasuyama, M. (2019a). Learning interpretable metric between graphs: Convex formulation and computation with graph mining. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1026–1036.

Yoshida, T., Takeuchi, I., and Karasuyama, M. (2019b). Safe triplet screening for distance metric learning. *Neural Computation*, 31(12):2432–2491.

Zhang, M., Cui, Z., Neumann, M., and Chen, Y. (2018a). An end-to-end deep learning architecture for graph classification. In *Proceedings of AAAI Conference on Artificial Inteligence*.

Zhang, Y., Liu, Y., Jing, X., and Yan, J. (2007). ACIK: association classifier based on itemset kernel. In *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems*, pages 865–875. Springer.

Zhang, Y. and Zaki, M. J. (2006). Exmotif: efficient structured motif extraction. *Algorithms for Molecular Biology*, 1(1):21.

Zhang, Z., Wang, M., Xiang, Y., Huang, Y., and Nehorai, A. (2018b). Retgk: Graph kernels based on return probabilities of random walks. In *Advances in Neural Information Processing Systems*, pages 3968–3978.

Zou, H. and Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the royal statistical society: series B (statistical methodology)*, 67(2):301–320.

Appendix

A Dual Problem

The primal problem (4) can be re-written as

$$\min_{\boldsymbol{m},\boldsymbol{z}} \sum_{i\in[n]}[\sum_{l\in\mathcal{D}_i} \ell_L(z_{il}) + \sum_{j\in\mathcal{S}_i} \ell_{-U}(z_{ij})] + \lambda R(\boldsymbol{m})$$

$$\text{s.t. } \boldsymbol{m} \geq \boldsymbol{0}, z_{il} = \boldsymbol{m}^\top \boldsymbol{c}_{il},\ z_{ij} = -\boldsymbol{m}^\top \boldsymbol{c}_{ij}.$$

The Lagrange function $\mathcal{L}$ is

$$\mathcal{L}(\boldsymbol{m},\boldsymbol{z},\boldsymbol{\alpha},\boldsymbol{\beta}) \coloneqq \sum_{i\in[n]}[\sum_{l\in\mathcal{D}_i} \ell_L(z_{il}) + \sum_{j\in\mathcal{S}_i} \ell_{-U}(z_{ij})] + \lambda R(\boldsymbol{m}) + \sum_{i\in[n]}[\sum_{l\in\mathcal{D}_i} \alpha_{il}(z_{il} - \boldsymbol{m}^\top \boldsymbol{c}_{il}) + \sum_{j\in\mathcal{S}_i} \alpha_{ij}(z_{ij} + \boldsymbol{m}^\top \boldsymbol{c}_{ij})] - \boldsymbol{\beta}^\top \boldsymbol{m},$$

where $\boldsymbol{\alpha} \in \mathbb{R}^{2nK}$ and $\boldsymbol{\beta} \in \mathbb{R}^p_+$ are Lagrange multipliers. The dual function D_λ is then

$$D_\lambda(\boldsymbol{\alpha},\boldsymbol{\beta}) \coloneqq \inf_{\boldsymbol{m},\boldsymbol{z}} \mathcal{L}(\boldsymbol{m},\boldsymbol{z},\boldsymbol{\alpha},\boldsymbol{\beta}). \tag{33}$$

In the definition of dual function (33), in order to minimize $\mathcal{L}$ with respect to $\boldsymbol{m}$, by partially differentiating $\mathcal{L}$, we obtain

$$\nabla_{\boldsymbol{m}}\mathcal{L} = \lambda(\boldsymbol{1} + \eta \boldsymbol{m}) + \sum_{i\in[n]}[-\sum_{l\in\mathcal{D}_i} \alpha_{il}\boldsymbol{c}_{il} + \sum_{j\in\mathcal{S}_i} \alpha_{ij}\boldsymbol{c}_{ij}] - \boldsymbol{\beta} = \boldsymbol{0}. \tag{34}$$

The convex conjugate function of ℓ_t is

$$\ell_t^*(-\alpha_{ij}) = \sup_{z_{ij}}\{(-\alpha_{ij})z_{ij} - \ell_t(z_{ij})\}, \tag{35}$$

which can be written as

$$\ell_t^*(x_*) = \frac{1}{4}x_*^2 + tx_*, (x_* \leq 0). \tag{36}$$

From equations (34), (35), and (36), the dual function can be written as

$$\begin{aligned}
&D_\lambda(\boldsymbol{\alpha},\boldsymbol{\beta}) \\
&= -\sum_{i\in[n]}[\sum_{l\in\mathcal{D}_i} \ell_L^*(-\alpha_{il}) + \sum_{j\in\mathcal{S}_i} \ell_{-U}^*(-\alpha_{ij})] - \frac{\lambda\eta}{2}\|\boldsymbol{m}_\lambda(\boldsymbol{\alpha},\boldsymbol{\beta})\|_2^2 \\
&= -\frac{1}{4}\|\boldsymbol{\alpha}\|_2^2 + \boldsymbol{t}^\top\boldsymbol{\alpha} - \frac{\lambda\eta}{2}\|\boldsymbol{m}_\lambda(\boldsymbol{\alpha},\boldsymbol{\beta})\|_2^2.
\end{aligned}$$

where

$$\begin{aligned}
\boldsymbol{m}_\lambda(\boldsymbol{\alpha},\boldsymbol{\beta}) &\coloneqq \frac{1}{\lambda\eta}\left[\boldsymbol{\beta} + \sum_{i\in[n]}(\sum_{l\in\mathcal{D}_i} \alpha_{il}\boldsymbol{c}_{il} - \sum_{j\in\mathcal{S}_i} \alpha_{ij}\boldsymbol{c}_{ij}) - \lambda\boldsymbol{1}\right] \\
&= \frac{1}{\lambda\eta}[\boldsymbol{\beta} + \boldsymbol{C}\boldsymbol{\alpha} - \lambda\boldsymbol{1}].
\end{aligned}$$

Therefore, although the dual problem can be written as

$$\max_{\boldsymbol{\alpha}\geq\mathbf{0},\boldsymbol{\beta}\geq\mathbf{0}} D(\boldsymbol{\alpha},\boldsymbol{\beta}),$$

by maximizing $D(\boldsymbol{\alpha},\boldsymbol{\beta})$ with respect to $\boldsymbol{\beta}$, we obtain a more straightforward dual problem (5).

We obtain $\alpha_{ij} = -\ell_t'(z_{ij})$, used in (7), from the derivative of $\mathcal{L}$ with respect to z_{ij}.

B Proof of Lemma 4.1

From equation (12), the value of $(x_{i,k'} - x_{j,k'})^2$ is bounded as follows:

$$\begin{aligned}(x_{i,k'} - x_{j,k'})^2 &\leq \max_{0\leq x_{i,k'}\leq x_{i,k},0\leq x_{j,k'}\leq x_{j,k}} (x_{i,k'} - x_{j,k'})^2 \\ &= \max\{x_{i,k}, x_{j,k}\}^2.\end{aligned}$$

Using this inequality, the inner product $\boldsymbol{C}_{k',:}\boldsymbol{q}$ is likewise bounded:

$$\begin{aligned}\boldsymbol{C}_{k',:}\boldsymbol{q} &= \sum_{i\in[n]}\Big[\sum_{l\in\mathcal{D}_i} q_{il}(x_{i,k'} - x_{l,k'})^2 - \sum_{j\in\mathcal{S}_i} q_{ij}(x_{i,k'} - x_{j,k'})^2\Big] \\ &\leq \sum_{i\in[n]}\sum_{l\in\mathcal{D}_i} q_{il}\max\{x_{i,k}, x_{l,k}\}^2.\end{aligned}$$

Similarly, the norm $\|\boldsymbol{C}_{k',:}\|_2$ is bounded:

$$\begin{aligned}\|\boldsymbol{C}_{k',:}\|_2 &= \sqrt{\sum_{i\in[n]}[\sum_{l\in\mathcal{D}_i}(x_{i,k'} - x_{l,k'})^4 + \sum_{j\in\mathcal{S}_i}(x_{i,k'} - x_{j,k'})^4]} \\ &\leq \sqrt{\sum_{i\in[n]}[\sum_{l\in\mathcal{D}_i}\max\{x_{i,k}, x_{l,k}\}^4 + \sum_{j\in\mathcal{S}_i}\max\{x_{i,k}, x_{j,k}\}^4]}.\end{aligned}$$

Therefore, $\boldsymbol{C}_{k',:}\boldsymbol{q} + r\|\boldsymbol{C}_{k',;}\|_2$ is bounded by $\text{Prune}(k|\boldsymbol{q}, r)$.

C Proof of Lemma 4.2

First, we consider the first term of $\boldsymbol{C}_{k',:}\boldsymbol{q} + r\|\boldsymbol{C}_{k',:}\|_2$:

$$\boldsymbol{C}_{k',:}\boldsymbol{q} = \sum_{i\in[n]}\underbrace{\Big[\sum_{l\in\mathcal{D}_i} q_{il}(x_{i,k'} - x_{l,k'})^2 - \sum_{j\in\mathcal{S}_i} q_{ij}(x_{i,k'} - x_{j,k'})^2\Big]}_{:=\text{diff}}.$$

Now, $x_{i,k'} \in \{0,1\}$ is assumed. Then, if $x_{i,k'} = 0$, we obtain

$$\text{diff} = \sum_{l\in\mathcal{D}_i} q_{il}x_{l,k'} - \sum_{j\in\mathcal{S}_i} q_{ij}x_{j,k'} \leq \sum_{l\in\mathcal{D}_i} q_{il}x_{l,k}.$$

On the other hand, if $x_{i,k'} = 1$, we see $x_{i,k} = 1$ from the monotonicity, and subsequently

$$\text{diff} = \sum_{l\in\mathcal{D}_i} q_{il}(1 - x_{l,k'}) - \sum_{j\in\mathcal{S}_i} q_{ij}(1 - x_{j,k'}) \leq \sum_{l\in\mathcal{D}_i} q_{il} - \sum_{j\in\mathcal{S}_i} q_{ij}(1 - x_{j,k}).$$

By using “max”, we can unify these two upper bounds into

$$\boldsymbol{C}_{k',:}\boldsymbol{q} \le \sum_{i\in[n]} \max\Big\{\sum_{l\in\mathcal{D}_i} q_{il}x_{l,k} \ , \ x_{i,k}[\sum_{l\in\mathcal{D}_i} q_{il} - \sum_{j\in\mathcal{S}_i} q_{ij}(1-x_{j,k})]\Big\}.$$

Employing a similar concept, the norm of $\boldsymbol{C}_{k',:}$ can also be bounded by

$$\begin{aligned}\|\boldsymbol{C}_{k',:}\|_2 &= \sqrt{\sum_{i\in[n]}[\sum_{l\in\mathcal{D}_i}(x_{i,k'}-x_{l,k'})^4 + \sum_{j\in\mathcal{S}_i}(x_{i,k'}-x_{j,k'})^4]} \\ &\le \sqrt{\sum_{i\in[n]}[\sum_{l\in\mathcal{D}_i}\max\{x_{i,k},x_{l,k}\} + \sum_{j\in\mathcal{S}_i}\max\{x_{i,k},x_{j,k}\}]}.\end{aligned}$$

Thus, we obtain

$$\begin{aligned}\mathrm{Prune}(k) \coloneqq \sum_{i\in[n]} \max\{\sum_{l\in\mathcal{D}_i} q_{il}x_{l,k}, x_{i,k}[\sum_{l\in\mathcal{D}_i} q_{il} - \sum_{j\in\mathcal{S}_i} q_{ij}(1-x_{j,k})]\} \\ + r\sqrt{\sum_{i\in[n]}[\sum_{l\in\mathcal{D}_i}\max\{x_{i,k},x_{l,k}\} + \sum_{j\in\mathcal{S}_i}\max\{x_{i,k},x_{j,k}\}]}.\end{aligned}$$

D Proof of Theorem 4.1 (DGB)

From 1/2-strong convexity of $-D_\lambda(\boldsymbol{\alpha})$, for any $\boldsymbol{\alpha} \ge \boldsymbol{0}$ and $\boldsymbol{\alpha}^\star \ge \boldsymbol{0}$, we obtain

$$D_\lambda(\boldsymbol{\alpha}) \le D_\lambda(\boldsymbol{\alpha}^\star) + \nabla D_\lambda(\boldsymbol{\alpha}^\star)^\top(\boldsymbol{\alpha}-\boldsymbol{\alpha}^\star) - \frac{1}{4}\|\boldsymbol{\alpha}-\boldsymbol{\alpha}^\star\|_2^2. \tag{37}$$

Applying weak duality $P_\lambda(\boldsymbol{m}) \ge D_\lambda(\boldsymbol{\alpha}^\star)$ and the optimality condition of the dual problem $\nabla D_\lambda(\boldsymbol{\alpha}^\star)^\top(\boldsymbol{\alpha} - \boldsymbol{\alpha}^\star) \le 0$ to (37), we obtain DGB.

E Proof of Theorem 4.2 (RPB)

From the optimality condition of the dual problem (5),

$$\nabla_{\boldsymbol{\alpha}} D_{\lambda_0}(\boldsymbol{\alpha}_0^\star)^\top(\frac{\lambda_0}{\lambda_1}\boldsymbol{\alpha}_1^\star - \boldsymbol{\alpha}_0^\star) \le 0, \tag{38}$$

$$\nabla_{\boldsymbol{\alpha}} D_{\lambda_1}(\boldsymbol{\alpha}_1^\star)^\top(\frac{\lambda_1}{\lambda_0}\boldsymbol{\alpha}_0^\star - \boldsymbol{\alpha}_1^\star) \le 0. \tag{39}$$

Here, the gradient vector at the optimal solution is

$$\begin{aligned}\nabla D_{\lambda_i}(\boldsymbol{\alpha}_i^\star) &= -\frac{1}{2}\boldsymbol{\alpha}_i^\star + \boldsymbol{t} - \boldsymbol{C}^\top \boldsymbol{m}_{\lambda_i}(\boldsymbol{\alpha}_i^\star) \\ &= -\frac{1}{2}\boldsymbol{\alpha}_i^\star + \boldsymbol{t} - \boldsymbol{C}^\top \boldsymbol{m}_i^\star,\end{aligned}$$

thus, by substituting this equation into (38) and (39),

$$(-\frac{1}{2}\boldsymbol{\alpha}_0^\star + \boldsymbol{t} - \boldsymbol{C}^\top\boldsymbol{m}_0^\star)^\top(\frac{\lambda_0}{\lambda_1}\boldsymbol{\alpha}_1^\star - \boldsymbol{\alpha}_0^\star) \le 0, \tag{40}$$

$$(-\frac{1}{2}\boldsymbol{\alpha}_1^\star + \boldsymbol{t} - \boldsymbol{C}^\top\boldsymbol{m}_1^\star)^\top(\frac{\lambda_1}{\lambda_0}\boldsymbol{\alpha}_0^\star - \boldsymbol{\alpha}_1^\star) \le 0. \tag{41}$$

From $\lambda_1 \times (40) + \lambda_0 \times (41)$,

$$(-\frac{1}{2}[\boldsymbol{\alpha}_0^\star - \boldsymbol{\alpha}_1^\star] - \boldsymbol{C}^\top[\boldsymbol{m}_0^\star - \boldsymbol{m}_1^\star])^\top(\lambda_0\boldsymbol{\alpha}_1^\star - \lambda_1\boldsymbol{\alpha}_0^\star) \leq 0. \tag{42}$$

From equation (34),

$$\boldsymbol{C}\boldsymbol{\alpha}_i = \lambda_i\eta\boldsymbol{m}_i + \lambda_i\boldsymbol{1} - \boldsymbol{\beta}_i. \tag{43}$$

By substituting equation (43) into equation (42),

$$-\frac{1}{2}[\boldsymbol{\alpha}_0^\star - \boldsymbol{\alpha}_1^\star]^\top(\lambda_0\boldsymbol{\alpha}_1^\star - \lambda_1\boldsymbol{\alpha}_0^\star) - [\boldsymbol{m}_0^\star - \boldsymbol{m}_1^\star]^\top(\lambda_0\lambda_1\eta[\boldsymbol{m}_1 - \boldsymbol{m}_0] - \lambda_0\boldsymbol{\beta}_1^\star + \lambda_1\boldsymbol{\beta}_0^\star) \leq 0.$$

Transforming this inequality based on completing the square with the complementary condition $\boldsymbol{m}_i^{\star\top}\boldsymbol{\beta}_i^\star = 0$ and $\boldsymbol{m}_1^{\star\top}\boldsymbol{\beta}_0^\star, \boldsymbol{m}_0^{\star\top}\boldsymbol{\beta}_1^\star \geq 0$, we obtain

$$\left\|\boldsymbol{\alpha}_1^\star - \frac{\lambda_0 + \lambda_1}{2\lambda_0}\boldsymbol{\alpha}_0^\star\right\|_2^2 + 2\lambda_1\eta\|\boldsymbol{m}_0^\star - \boldsymbol{m}_1^\star\|_2^2 \leq \left\|\frac{\lambda_0 - \lambda_1}{2\lambda_0}\boldsymbol{\alpha}_0^\star\right\|_2^2.$$

By using $\|\boldsymbol{m}_0^\star - \boldsymbol{m}_1^\star\|_2^2 \geq 0$ to this inequality, we obtain RPB.

F Proof of Theorem 4.3 (RRPB)

Considering a hypersphere that expands the RPB radius by $\frac{\lambda_0+\lambda_1}{2\lambda_0}\epsilon$ and replaces the RPB center with $\frac{\lambda_0+\lambda_1}{2\lambda_0}\boldsymbol{\alpha}_0$, we obtain

$$\left\|\boldsymbol{\alpha}_1^\star - \frac{\lambda_0 + \lambda_1}{2\lambda_0}\boldsymbol{\alpha}_0\right\|_2 \leq \frac{|\lambda_0 - \lambda_1|}{2\lambda_0}\|\boldsymbol{\alpha}_0^\star\|_2 + \frac{\lambda_0 + \lambda_1}{2\lambda_0}\epsilon.$$

Since ϵ is defined by $\|\boldsymbol{\alpha}_0^\star - \boldsymbol{\alpha}_0\|_2 \leq \epsilon$, this sphere covers any RPB made by $\boldsymbol{\alpha}_0^\star$ which satisfies $\|\boldsymbol{\alpha}_0^\star - \boldsymbol{\alpha}_0\|_2 \leq \epsilon$. Using the reverse triangle inequality

$$\|\boldsymbol{\alpha}_0^\star\|_2 - \|\boldsymbol{\alpha}_0\|_2 \leq \|\boldsymbol{\alpha}_0^\star - \boldsymbol{\alpha}_0\|_2 \leq \epsilon,$$

the following is obtained.

$$\left\|\boldsymbol{\alpha}_1^\star - \frac{\lambda_0 + \lambda_1}{2\lambda_0}\boldsymbol{\alpha}_0\right\|_2 \leq \frac{|\lambda_0 - \lambda_1|}{2\lambda_0}(\|\boldsymbol{\alpha}_0\|_2 + \epsilon) + \frac{\lambda_0 + \lambda_1}{2\lambda_0}\epsilon.$$

By arranging this, RRPB is obtained.

G Proof for Theorem 4.6 (RSS), 4.7 (RSP) and 4.8 (RSP for binary feature)

We consider theorem 4.6 and 4.7 because theorem 4.8 can be derived in almost the same way as theorem 4.7. When $\lambda_1 = \lambda$ is set in RRPB, the center and the radius of the bound $\mathcal{B} = \{\boldsymbol{\alpha} \mid \|\boldsymbol{\alpha} - \boldsymbol{q}\|_2^2 \leq r^2\}$ are $\boldsymbol{q} = \frac{\lambda_0+\lambda}{2\lambda_0}\boldsymbol{\alpha}_0$ and $r = \left\|\frac{\lambda_0-\lambda}{2\lambda_0}\boldsymbol{\alpha}_0\right\|_2 + \left(\frac{\lambda_0+\lambda}{2\lambda_0} + \frac{|\lambda_0-\lambda|}{2\lambda_0}\right)\epsilon$. Substituting these $\boldsymbol{q}$ and r into (16) and (17), respectively, and arranging them, we can obtain the range in which screening and pruning conditions hold.

Algorithm 4: General Working-Set Method

```
initialize $\boldsymbol{x}_0 \in \mathcal{D}$
for $t = 1, 2, ...$ until converged do
    $\mathcal{W}_t = \{j \mid h_j(\boldsymbol{x}_{t-1}) \geq 0\}$
    $\boldsymbol{x}_t = \arg\min_{\boldsymbol{x} \in \mathcal{D}} f(\boldsymbol{x})$ s.t. $h_j(\boldsymbol{x}) \leq 0, \forall j \in \mathcal{W}_t$
```

H Proof of Theorem 4.9 (Convergence of WS)

By introducing a new variable $\boldsymbol{s}$, the dual problem (5) can be written as

$$\max_{\boldsymbol{\alpha} \geq \boldsymbol{0}, \boldsymbol{s} \geq \boldsymbol{0}} \; -\frac{1}{4}\|\boldsymbol{\alpha}\|^2 + \boldsymbol{t}^\top \boldsymbol{\alpha} - \frac{1}{2\lambda\eta}\|\boldsymbol{s}\|^2$$
$$\text{s.t. } \boldsymbol{C}\boldsymbol{\alpha} - \lambda\boldsymbol{1} - \boldsymbol{s} \leq \boldsymbol{0}.$$

We demonstrate the convergence of working-set method on a more general convex problem as follows:

$$\boldsymbol{x}^\star := \arg\min_{\boldsymbol{x} \in \mathcal{D}} f(\boldsymbol{x}) \text{ s.t. } h_i(\boldsymbol{x}) \leq 0, \forall i \in [n], \tag{44}$$

where $f(\boldsymbol{x})$ is a γ-strong convex function ($\gamma > 0$). Here, as shown in Algorithm 4, the working set is defined by $\mathcal{W}_t = \{j \mid h_j(\boldsymbol{x}_{t-1}) \geq 0\}$ at every iteration. Then, the updated working set includes all the violated constraints and the constraints on the boundary. We show that Algorithm 4 finishes with finite T-steps and returns the optimal solution $\boldsymbol{x}_T = \boldsymbol{x}^\star$.

Proof. Since f is γ-strong convex from the assumption, the following inequality holds.

$$f(\boldsymbol{x}_{t+1}) \geq f(\boldsymbol{x}_t) + \nabla f(\boldsymbol{x}_t)^\top (\boldsymbol{x}_{t+1} - \boldsymbol{x}_t) + \frac{\gamma}{2}\|\boldsymbol{x}_{t+1} - \boldsymbol{x}_t\|^2. \tag{45}$$

At step t, the problem can be written as follows, using only the active constraint at the optimal solution $\boldsymbol{x}_t$.

$$\begin{aligned} \boldsymbol{x}_t &= \arg\min_{\boldsymbol{x} \in \mathcal{D}} f(\boldsymbol{x}) \text{ s.t. } h_i(\boldsymbol{x}) \leq 0, \forall i \in \mathcal{W}_t \\ &= \arg\min_{\boldsymbol{x} \in \mathcal{D}} f(\boldsymbol{x}) \text{ s.t. } h_i(\boldsymbol{x}) \leq 0, \forall i \in \{j \in \mathcal{W}_t \mid h_j(\boldsymbol{x}_t) = 0\} \end{aligned} \tag{46}$$

From the definition of $\mathcal{W}_t$, the working set $\mathcal{W}_{t+1}$ must contain all active constraints $\{j \in \mathcal{W}_t \mid h_j(\boldsymbol{x}_t) = 0\}$ at the step t and can contain other constraints that are not included in $\mathcal{W}_t$. This means that $\boldsymbol{x}_{t+1}$ must be in the feasible region of the optimization problem at the step t (46):

$$\mathcal{F} := \{\boldsymbol{x} \in \mathcal{D} \mid h_i(\boldsymbol{x}) \leq 0, \forall i \in \{j \in \mathcal{W}_t \mid h_j(\boldsymbol{x}_t) = 0\}\}$$

Therefore, from the optimality condition of the optimization problem (46),

$$\nabla f(\boldsymbol{x}_t)^\top (\boldsymbol{x}_{t+1} - \boldsymbol{x}_t) \geq 0, \boldsymbol{x}_{t+1} \in \mathcal{F}. \tag{47}$$

From the inequality (45) and the inequality (47), we obtain

$$f(\boldsymbol{x}_{t+1}) \geq f(\boldsymbol{x}_t) + \frac{\gamma}{2}\|\boldsymbol{x}_{t+1} - \boldsymbol{x}_t\|^2.$$

Table 9: Total time in the path-wise optimization (sec).

Dataset	BZR			DD			FRANKENSTEIN		
Method \ Process	Traverse	Solve	Total	Traverse	Solve	Total	Traverse	Solve	Total
SS&SP	1397.1		4281.9	4292.9		13961.3	249.1		5013.0
	±91.7	2884.8	±964.1	±388.3	9668.4	±1580.6	±9.3	4763.9	±442.4
RSS&RSP	**539.2**	±934.5	3424.0	1132.2	±1267.5	10800.6	**189.7**	±441.5	4953.6
	±47.2		±956.9	±118.0		±1354.9	±8.4		±439.1
WS&WP	2448.5		2724.3	5888.3		7652.8	380.1		938.3
	±170.8	**275.8**	±184.9	±465.6	**1764.5**	±622.6	±12.4	**558.2**	±57.5
WS&WP+	565.5	±68.5	**841.3**	**946.1**	±195.6	**2710.6**	233.0	±56.5	**791.1**
RSS&RSP	±49.7		±97.3	±83.1		±258.6	±11.7		±55.7

If $\boldsymbol{x}_t$ is not optimal, there exists at least one violated constraint $h_{j'}(\boldsymbol{x}_t) > 0$ for some j' because otherwise $\boldsymbol{x}_t$ is optimal. Then, we see $\boldsymbol{x}_{t+1} \neq \boldsymbol{x}_t$ because $\boldsymbol{x}_{t+1}$ should satisfy the constraint $h_{j'}(\boldsymbol{x}_{t+1}) \leq 0$. If $\boldsymbol{x}_t \neq \boldsymbol{x}_{t+1}$, by using $\|\boldsymbol{x}_{t+1} - \boldsymbol{x}_t\|^2 > 0$,

$$f(\boldsymbol{x}_{t+1}) \geq f(\boldsymbol{x}_t) + \frac{\gamma}{2}\|\boldsymbol{x}_{t+1} - \boldsymbol{x}_t\|^2 > f(\boldsymbol{x}_t).$$

Thus, the objective function always strictly increases ($f(\boldsymbol{x}_t) < f(\boldsymbol{x}_{t+1})$). This indicates that the algorithm never encounters the same working set $\mathcal{W}_t$ as the set of other iterations $t' \neq t$. For any step t, the optimal value $f(\boldsymbol{x}_t)$ with a subset of the original constraints $\mathcal{W}_t$ must be smaller than or equal to the optimal value $f(\boldsymbol{x}^\star)$ of original problem (44) with all constraints. Therefore, $f(\boldsymbol{x}_t) \leq f(\boldsymbol{x}^\star)$ is satisfied, and we obtain $f(\boldsymbol{x}_T) = f(\boldsymbol{x}^\star)$ at some finite step T. □

I CPU Time for Other Dataset

Table 9 shows computational time for the BZR, DD, and FRANKENSTEIN datasets. We first note that RSS&RSP was about 2-4 times faster in terms of the Traverse time compared with SS&SP. Next, comparing RSS&RSP and WS&WP, we see that RSS&RSP was faster for Traverse, and WS&WP was faster for Solve, as we observe in Table 1. Thus, the combination of WS&SP and RSS&RSP were the fastest for all three datasets in total.

J Approximating Frequency Without Overlap

Let "frequency without overlap" be the frequency of a subgraph in a given graph, where any shared vertices and edges are disallowed for counting. Under the condition that we know where all the subgraphs H appear in graph G, calculating the frequency without overlap is equivalent to the problem of finding the maximum independent set and is NP-complete (Schreiber and Schwöbbermeyer, 2005). In this section, using information obtained in the process of generating gSpan tree, we approximate the frequency without overlap by its upper bound.

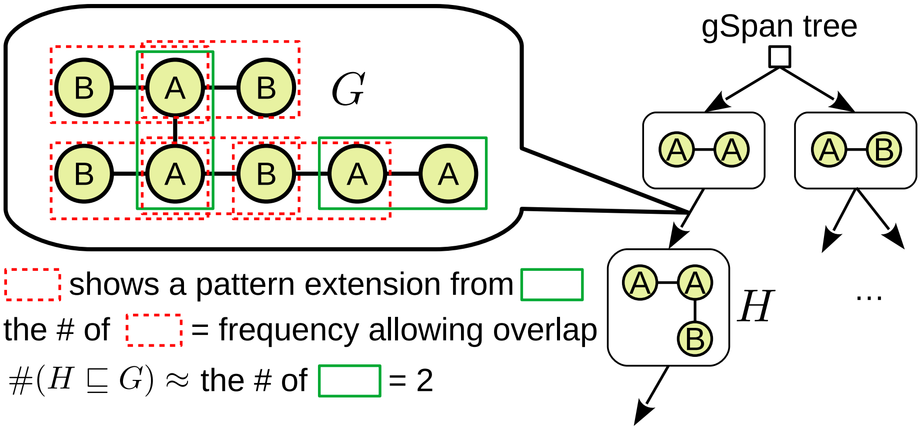

gSpan tree
B
A
B
G
B
A
B
A
A
A
A
A
B
A
A
B
H
…
shows a pattern extension from
the # of = frequency allowing overlap
#(H ⊑ G) ≈ the # of = 2

Figure 11: Approximation of $\#(H \sqsubseteq G)$.

Figure 11 shows the process of generating the gSpan tree and the frequency. In the figure, we consider the frequency of the subgraph H (Ⓐ-Ⓐ-Ⓑ) contained in the graph G. The graph H is obtained as a pattern extension of graph Ⓐ-Ⓐ (green frame) by Ⓐ-Ⓑ (red frame). Because gSpan stores these all pattern extensions, the frequency allowing overlap is obtained by counting the red frames (e.g. the frequency is five in the figure). Going back from each red frame to the green frame, we reach a start edge that was generating H. Then, the number of unique edges, i.e., the number of green frames, can be regarded as an approximation of $\#(H \sqsubseteq G)$ (e.g., the number is two in the figure). The number obtained by this approximation is less than or equals to the frequency allowing overlap because it is decreased from the number of red frames. Further, since only one edge (green frame) of overlap is considered instead of the entire overlap, the number obtained by this approximation is more than or equals to $\#(H \sqsubseteq G)$. As the graph H grows, the approximation method satisfies a monotonicity because the number of green frames is non-increasing.