Nguyen, Dai Hai; Nguyen, Canh Hao; Mamitsuka, Hiroshi

Learning subtree pattern importance for Weisfeiler-Lehman based graph kernels

Published in:
Machine Learning

DOI:
10.1007/s10994-021-05991-y

Published: 01/07/2021

Document Version
Peer-reviewed accepted author manuscript, also known as Final accepted manuscript or Post-print

Please cite the original version:
Nguyen, D. H., Nguyen, C. H., & Mamitsuka, H. (2021). Learning subtree pattern importance for Weisfeiler-Lehman based graph kernels. *Machine Learning*, *110*(7), 1585-1607. https://doi.org/10.1007/s10994-021-05991-y

Noname manuscript No.
(will be inserted by the editor)

Learning subtree pattern importance for Weisfeiler-Lehman based graph kernels

Dai Hai Nguyen · Canh Hao Nguyen · Hiroshi Mamitsuka

Received: date / Accepted: date

Abstract Graph is an usual representation of relational data, which are ubiquitous in many domains such as molecules, biological and social networks. A popular approach to learning with graph structured data is to make use of graph kernels, which measure the similarity between graphs and are plugged into a kernel machine such as a support vector machine. Weisfeiler-Lehman (WL) based graph kernels, which employ WL labeling scheme to extract subtree patterns and perform node embedding, are demonstrated to achieve great performance while being efficiently computable. However, one of the main drawbacks of a general kernel is the decoupling of kernel construction and learning process. For molecular graphs, usual kernels such as WL subtree, based on substructures of the molecules, consider all available substructures having the same importance, which might not be suitable in practice. In this paper, we propose a method to learn the weights of subtree patterns in the framework of WWL kernels, the state of the art method for graph classification task [14]. To overcome the computational issue on large scale data sets, we present an efficient learning algorithm and also derive a generalization gap bound to show its convergence. Finally, through experiments on synthetic and real-world data sets, we demonstrate the effectiveness of our proposed method for learning the weights of subtree patterns.

Keywords Graph kernel · Optimal transport · Weisfeiler Lehman scheme

Dai Hai Nguyen
Graduate School of Frontier Sciences, The University of Tokyo, 5-1-5 Kashiwa-no-ha, Kashiwa, Chiba 277-8561, Japan
E-mail: hai@k.u-tokyo.ac.jp

Canh Hao Nguyen
Bioinformatics Center, Institute for Chemical Research, Kyoto University, Uji 611-0011, Japan
E-mail: canhhao@kuicr.kyoto-u.ac.jp

Hiroshi Mamitsuka
Bioinformatics Center, Institute for Chemical Research, Kyoto University, Uji 611-0011, Japan and Department of Computer Science, Alato University, Espoo 02150, Finland
E-mail: mami@kuicr.kyoto-u.ac.jp

1 Introduction

Graphs are natural data structures, which appear in various domains such as bioinformatics [12], cheminformatics [15], social network analysis [11] and so on, where nodes (vertices) represent objects and edges represent the relations between them. A popular approach to learning with graph structured data is to make use of graph kernels. Essentially, a graph kernel is a measure of the similarity between two graphs and must satisfy two fundamental requirements of being a valid kernel: 1) symmetric and 2) positive semi-definite (PSD). Furthermore, the requirements of designing a graph kernel are: it should capture the semantic inherent in the graph structures (e.g. substructures of different levels), and it must be efficiently computable [17].

A number of graph kernels have been proposed in literature such as random walk [5], shortest path [1], Weisfeiler-Lehman (WL) subtree [13] kernels, just to name a few. Most of them are based on $\mathscr{R}$-Convolution framework [4], which decomposes two graphs into substructures and adds up the similarities between their substructures to compute kernel values. Different graph kernels are defined under different ways of decomposition (types of substructures). For instance, the substructures can be random walks [5], shortest paths [1] or subtree patterns [17]. Among these, WL subtree kernels have been shown to achieve great prediction performance while being efficiently computable. The key point is that it simply employs a WL based color refinement scheme to embed each node in a given graph into a vector of WL labels, which correspond to *subtree patterns* of the graphs. Then, the kernel between two graphs is defined as the sum of all pairwise similarities between any two node embeddings of the two graphs.

Following a different approach, WL based optimal assignment (WL-OA) kernel [6] assigns one node embedding of one graph to one embedding of the other such that the total similarities between assigned node embeddings is maximized. This is also known as optimal assignment problem in combinatorial mathematics [9]. In a similar vein, Wasserstein WL (WWL, [14]) uses optimal transport (OT), also known as Wasserstein distance [16], for measuring the distance between two graphs based on their WL node embeddings. The distance is then converted into a similarity matrix through Laplacian kernel. Furthermore, both of these similarity matrices are shown to be valid kernels due to the hierachy property of WL labels (see [6] and [14] for more details).

One of the main drawbacks of these kernels is that they are predefined feature extraction without learning the importance of substructures to the problem. This results in the decoupling of data representation and learning process. In these kernels, substructures are given the same weights. However, for the problems such as molecule classification, it is known that only subparts of the molecules are responsible for their properties. Therefore, we wish to be able to give weights to their substructures to have higher classification performance and model interpretation. Based on this motivation, we propose a model to learn the weights of *subtree patterns* (extracted by WL labeling scheme). Our work extends WWL kernels [14] by formulating an OT based distance as a parametric function of subtree pattern weights before converting into kernels. We also propose an efficient stochastic learning algorithm to estimate the weights and derive a generalization gap bound for the algorithm. Finally, through experiments on synthetic and four real-world data sets, we show that learning important subtree patterns by our proposed method can lead to more accurate predictive performance and extract important patterns which enhance the classification results.

The remainder of the paper is organized as follows: in Section 2, we review graph kernels which are based on WL labeling scheme, including WL subtree, WL-OA and WWL kernels. In Section 3, we present our method that parameterizes the Wasserstein distance

between two graphs with WL labeling scheme as a function of subtree patterns and present the stochastic algorithm for learning parameters of the function. In Section 4, we derive a generalization bound for the learning algorithm. In Section 5, experimental results on the synthetic and real-world data sets are provided. Finally, we conclude by summarizing this work and discussing possible extensions in Section 6.

2 Related work

In this paper, we consider the binary classification problem for graph structured data: given a collection of labeled graphs $(g_i, y_i), i = 1,..,n$ (where n is the number of examples) drawn from an unknown joint distribution $\mathscr{P}$ over $\mathscr{G} \times \{-1,1\}$, where $\mathscr{G}$ is a space of graphs. We wish to learn a classifier $h : \mathscr{G} \to \{-1,1\}$, which is based on a similarity function $K : \mathscr{G} \times \mathscr{G} \to [-1,1]$. If K is symmetric and positive semi-definite (PSD), it is called a valid kernel.

There are a number of proposed kernels on graphs, see [5, 1, 13]. In general, they are defined based on $\mathscr{R}$-Convolution framework [4], that is, each graph $g \in \mathscr{G}$ is decomposed into substructures, and a kernel value $K(g,g')$ is defined as a sum of pairwise similarities between their substructures. In fact, many graph kernels can be considered as instances of the $\mathscr{R}$-Convolution framework under different decomposition into substructures. The substructures can be random walks [5], shortest paths [1] or circle subtrees [13]. Among these, Weisfeiler-Lehman (WL) subtree kernels [13] and its variants have been shown to achieve great performance for the graph classification tasks. In this work, we focus on WL based kernels and will review them in the following subsections.

2.1 Weisfeiler-Lehman (WL) scheme for node embeddings

Weisfeiler-Lehman (WL) subtree kernels [13] are based on an iterative colour refinement (also known as WL labeling scheme) and have been shown to achieve great performance for graph classification task. For each node of a given graph, the WL labeling scheme creates a sequence of ordered strings by the aggregation of the labels of the node and its neighbors; these strings are then hashed or indexed to produce compressed updated node labels or new indices. If the iteration of the scheme is increased, these obtained labels represent increasingly broader neighborhood of each node. More specifically, for a graph $G = (V,E)$ with initial labels $\ell_0(v)$ for $v \in V$ and let H be the number of WL iterations, we can define a sequence of refined labels $(\ell_0, \ell_1, ..., \ell_H)$, where ℓ_{h+1} is obtained from ℓ_h by the following procedure: $\ell_{h+1}(v) = \text{hash}(\ell_h(v), \mathscr{N}_h(v))$, where $\mathscr{N}_h(v)$ denotes a lexicographically sorted sequence of labels of v's neighbors at iteration h and the hash function is to create a updated compressed node label for v. We use perfect hashing for the hash function, as in [13], ensuring two nodes at iteration $h+1$ have the same label if and only if their label and those of their neighbors at iteration h are the same. Throughout the rest of paper, we denote the set of WL labels at iteration h as Σ^h, for $h = 1,...,H$, and the set of all WL labels as $\Sigma = \cup_{h=1}^{H} \Sigma^h$. The WL labeling scheme is illustrated in Figure 1 (a).

Based on WL labeling scheme, a WL node embedding scheme was proposed to generate node embeddings from node labels of the graphs [13].

Definition 1. (WL based node embedding scheme, [13]). Let $G = (V,E)$ and let H be the number of WL iterations. For every $h \in \{1,...,H\}$, we define the node embedding $x^h(v)$ of

a node $v \in V$ and graph embedding $\mathbf{X}^h(G)$ of G at iteration h as follows:

$$x^h(v) = \ell^h(v), \mathbf{X}^h(G) = [x^h(v_1), ..., x^h(v_{n_V})]^T \tag{1}$$

Then the graph embedding of G can be defined as:

$$f^H(G) : \mathscr{G} \mapsto \Sigma^{n_V \times H}$$
$$G \mapsto \left[\mathbf{X}^1(G)^T, ..., \mathbf{X}^H(G)^T\right]^T$$

where n_V is the number of nodes in G. With the WL node embedding scheme above, we are ready to introduce some notations which will be used throughout the rest of paper.

Notations: Let $D_h^{\text{Ham}}(f^h(G), f^h(G'))$ be the Hamming distance matrix where each entry is the *normalized Hamming distance* between the corresponding node embeddings of G and G' at iteration h, defined as:

$$d_h^{\text{Ham}}(u,v) = \frac{1}{h}\sum_{i=1}^{h} d_i^{\text{disc}}(u,v), d_i^{\text{disc}}(u,v) = \begin{cases} 0 & \text{if } u_i = v_i \\ 1 & \text{otherwise} \end{cases} \tag{2}$$

where u and v denote two node embeddings, $d_h^{\text{Ham}}(u,v)$ denotes the normalized Hamming distance between u and v at iteration h and $d_i^{\text{disc}}(u,v)$ denotes the *discrete distance* between u and v at iteration i. Similarly, let $D_h^{\text{Disc}}(\mathbf{X}^h(G), \mathbf{X}^h(G'))$ be the discrete distance matrix where each entry is the discrete distance between the corresponding node embeddings of G and G' at iteration h. It is easy to see that $[D_h^{\text{Ham}}]_{ij} \in [0,1]$ and $[D_h^{\text{Disc}}]_{ij} \in \{0,1\}$. We also define a base kernel which corresponds to the averaged number of feature shared by two node embeddings as:

$$k_h(u,v) = \frac{1}{h}\sum_{i=1}^{h} \mathbb{1}(u_i = v_i) \tag{3}$$

It is easy to see that $k_h(u,v) = 1 - d_h^{\text{Ham}}(u,v)$. Thanks to WL node embedding scheme, a graph can be represented as a point cloud or a set of node embeddings. Measuring the similarity (or dissimilarity) between two graphs boils down to measuring the similarity (or dissimilarity) between two sets of embeddings. Each node embedding captures information about the neighborhood of the corresponding node (or a rooted *subtree pattern*). In the following subsections, we will review different WL based graph kernels derived from different ways of comparing their sets of WL node embeddings.

2.2 WL subtree kernels

WL subtree kernels [13] simply employ the aforementioned WL labeling scheme to extract subtree patterns, which represent the neighborhood of each node in the graph up to a given distance (or number of hops H). Essentially WL subtree kernel counts the number of common WL labels. In the context of WL node embedding scheme, it can be computed by summing all pairwise similarities between node embeddings of two graphs. More formally, for two graphs G and G' with two sets of node embeddings $f^H(G)$ and $f^H(G')$, respectively, the WL subtree kernel value between them can be defined as:

$$\mathbf{K}^{\text{WL}}(G,G') = \sum_{\mathbf{x} \in f^H(G)} \sum_{\mathbf{y} \in f^H(G')} k_H(\mathbf{x},\mathbf{y}) \tag{4}$$

It is obvious that as the base kernel $k_H(\mathbf{x},\mathbf{y})$ (defined in Eq. (3)) is equal to the number of WL labels (subtree patterns) shared by two node embeddings, the kernel $\mathbf{K}^{\mathrm{WL}}(G,G')$ is equal to the total number of WL subtree patterns shared by two graphs.

2.3 WL-based optimal assignment kernels

Optimal assignments are natural measures of similarity between two sets of points. In particular, for two sets of points, the goal is to assign one point of one set to another point of the other set (one-to-one correspondence) such that the sum of similarities between assigned points is maximized. Finding such an optimal alignment or bijection is also known as the well-studied assignment problem in combinatorial optimization [9]. However, a challenge is how to design a valid kernel based on optimal assignments.

Kriege *et al* [6] introduced a restricted class of kernels, called *strong kernels*, that guarantees the construction of valid optimal assignment based kernels. An important result is that the strong kernels give rise to hierarchies defined on the domain of kernels. Based on this, the authors proposed WL-based optimal assignment (WL-OA) kernels with the WL node embedding scheme. WL-OA kernels employ the base kernel, defined in Eq. (3), which satisfies the requirement of being a strong kernel as the sequence of refined WL labels $(\ell_0,\ell_1,...,\ell_H)$ gives rise to a family of nested subsets, which can be represented by a hierarchy. Consequently, WL-OA kernels are valid. Formally, given two graphs G and G' with two sets of node embeddings $f^H(G)$ and $f^H(G')$, respectively, the WL-OA kernel value between them is defined as:

$$\mathbf{K}^{\mathrm{OA}}(G,G') = \max_{B\in\mathscr{B}(f^H(G),f^H(G'))} \sum_{(x,y)\in B} k_H(\mathbf{x},\mathbf{y}) \tag{5}$$

where $\mathscr{B}(f^H(G),f^H(G'))$ is the set of all bijections between two sets $f^H(G)$ and $f^H(G')$. To apply this kernel to graphs of different number of nodes, we can fill up the graph with smaller number of nodes, says $f^H(G')$, by new node embeddings z with $k(\mathbf{x},\mathbf{z}) = 0$ for all $\mathbf{x} \in f^H(G)$ without changing the result. It is worth noting that the WL-OA kernels take the similarities of aligned node embeddings into account, while the WL subtree kernels consider all pairwise similarities.

2.4 Wasserstein WL kernels

Optimal transport (OT), also known as Wasserstein distance function [16], has gained much attraction in machine learning community as a powerful tool for the comparsion of two probability distributions. The naive computation of this distance between two discrete measures, e.g. point clouds, involves solving transport problem. Formally, let $X = \{\mathbf{x}_1,...,\mathbf{x}_m\}$ and $Y = \{\mathbf{y}_1,...,\mathbf{y}_n\}$ be two sets of points, where m and n denote the size of two sets X and Y, respectively; $\mathbf{p} \in \mathbb{R}^m$ and $\mathbf{q} \in \mathbb{R}^n$ are two discrete probability distributions over X and Y, respectively. We use d_{ij} to denote the distance between $\mathbf{x}_i$ and $\mathbf{y}_j$, e.g. the squared Euclidean distance $d_{ij} = \left\|\mathbf{x}_i - \mathbf{y}_j\right\|_2^2$. The Wasserstein distance is formulated as a linear program over the *transportation matrix* (or joint probability) $\mathbf{P} \in \mathbb{R}^{m\times n}$:

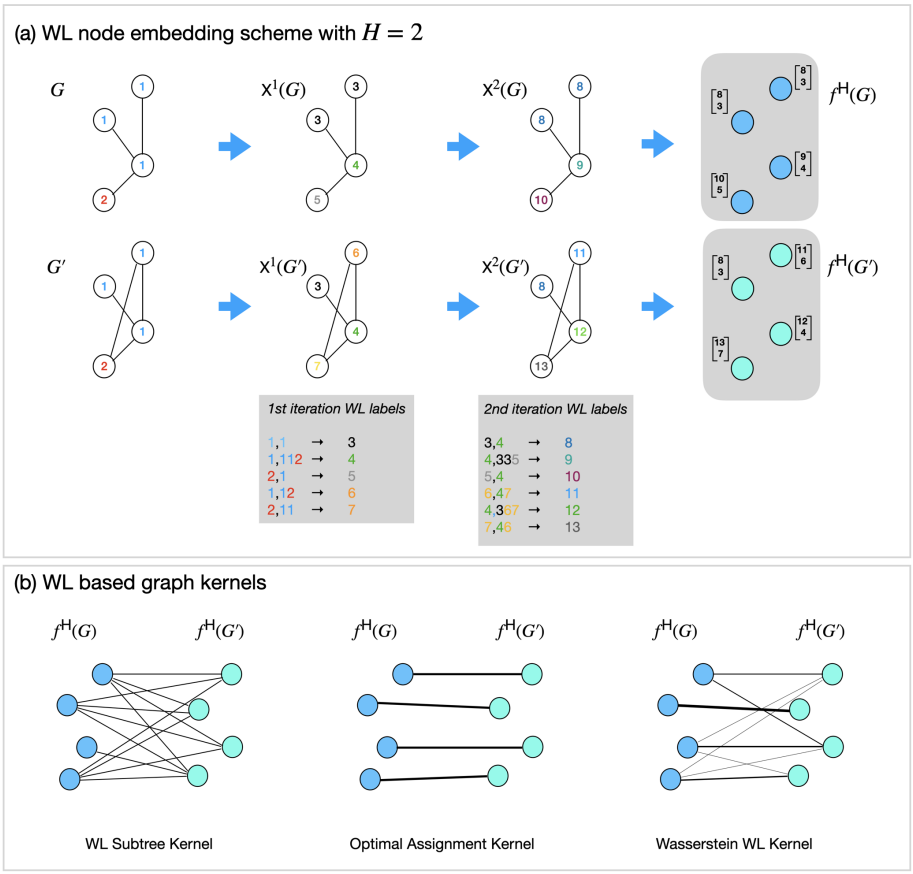

Fig. 1 (a) Illustration of Weisfeiler-Lehman (WL) node embedding scheme with two iterations ($H = 2$). (b) Illustration of different WL based graph kernels: WL Subtree kernel is computed by considering all pairwise similarities between node embeddings of two graphs; WL optimal assignment kernel takes the similarities of only aligned node embeddings into account; For Wasserstein WL kernel, one node embedding of one graph can be coupled with multiple node embeddings of the other graph for computing the kernel.

$$
\begin{aligned}
\mathscr{W}_1(X,Y,d) &= \min_{\mathbf{P}_{ij}} \sum_{i=1}^{m}\sum_{j=1}^{n} \mathbf{P}_{ij} d_{ij} \\
\text{subject to } & \sum_{i=1}^{m} \mathbf{P}_{ij} = \mathbf{q}_j,\ \forall j \in [1,n] \\
& \sum_{j=1}^{n} \mathbf{P}_{ij} = \mathbf{p}_i,\ \forall i \in [1,m]
\end{aligned}
\tag{6}
$$

With WL node embedding scheme to generate node embeddings for graphs, Togninalli *et al* [14] evaluated the pairwise Wasserstein distance between graphs with the normalised Hamming (2) as the ground distance. Then, Wasserstein WL (WWL) kernel is defined as an

instance of Laplacian kernels, see Eq. (7).

$$\begin{aligned}\mathbf{D}^{\mathrm{WWL}}(G,G') &= \mathscr{W}_1(f^H(G), f^H(G'), d_H^{\mathrm{Ham}}) \\ \mathbf{K}^{\mathrm{WWL}}(G,G') &= e^{-\gamma \mathbf{D}^{\mathrm{WWL}}(G,G')}\end{aligned} \tag{7}$$

where γ is a hyperparameter.

In general cases, it is not necessarily possible to derive a valid kernel from the Wasserstein distance. However, thanks to the special property of WL labels and normalized Hamming distance (2), $\mathbf{D}^{\mathrm{WWL}}$ was shown to be conditionally negative definite (CND), resulting in the validity of $\mathbf{K}^{\mathrm{WWL}}$, by proving the following lemma (see [14] for its proof):

Lemma 1. *If a transportation matrix P^H is optimal solution of (6) with the ground distance d_H^{Ham} (2) between node embeddings at iteration H, then we have the two following claims:*

1. *P^H is also optimal solution of (6) with the discrete distance d_H^{Disc} between H-iteration values.*
2. *P^H is also optimal solution of (6) with the normalised Hamming distance d_{H-1}^{Ham} between node embeddings at iteration $H-1$.*

Let P^* be the optimal solution of (6) for D_H^{Ham}. From the above lemma, it is also the optimal solution for D_h^{Disc}, $h = 1,..,H$. The Wasserstein distance between two graphs G and G' in (7) can be simplified as follows:

$$\mathbf{D}^{WWL}(G,G') = \frac{1}{H}\sum_{h=1}^{H} \mathscr{W}_1(\mathbf{X}^h(G), \mathbf{X}^h(G'), d_h^{\mathrm{Disc}}) \tag{8}$$

The Eq. (8) is a sum of OT distances with the discrete distances as ground metrics, which are CND. Therefore, the sum is also CND, leading to the validity of the similarity matrix $\mathbf{K}^{\mathrm{WWL}}$.

3 Incorporating subtree pattern importance into WL based graph kernels

One of the main limitations of kernels is the decoupling of data representation and learning process, that is, the kernel must be predefined prior to learning, leading to limited predictive performance. Furthermore, in prediction tasks for molecular data, the output might be determined by the presence of a few important substructures, while these kernels contain all substructures with equal weights. Motivated by this drawback, in this paper we address the problem of incorporating subtree pattern weights for WWL kernel [14]. To this end, we aim to learn new kernels from a parametric form of Wasserstein distance taking into account subtree pattern weights (8), and learn these weights from data optimally for the task.

3.1 Parametric form of Wasserstein distance with subtree pattern weights

To derive a parametric form of the distance function (8), we rely on the following simple observation:

Lemma 2. *Let S be a set of elements, $X = \{\boldsymbol{x}_1,...,\boldsymbol{x}_m\}$ and $Y = \{\boldsymbol{y}_1,...,\boldsymbol{y}_n\}$ $(X,Y \subseteq S)$ be two multiset of S of m and n samples, respectively, the Wasserstein distance between them with the discrete distance as the ground metric is determined by:*

$$\mathscr{W}_1(X,Y) = 1 - \sum_{v \in S} \min(\mu_X(v), \mu_Y(v)) \tag{9}$$

where $\mu_X(v)$ denotes the mass density function of the multiset X with $v \in S$.

Applying this lemma to Eq. (8), we have:

$$\mathbf{D}^{\mathrm{WWL}}(G,G') = 1 - \frac{1}{H}\sum_{h=1}^{H}\sum_{v \in \Sigma^h} \min(\mu_{\mathbf{X}_h(G)}(v), \mu_{\mathbf{X}_h(G')}(v)) \tag{10}$$

Our idea is to give each substructure or WL label v a nonnegative weight $w_v \in \mathbb{R}_{\geq 0}$ for its importance to the problem, so the parametric form of Eq. (10) is defined as follows:

$$b - \frac{1}{H}\sum_{h=1}^{H} \langle \mathbf{w}_h, \mathbf{z}_h\left(G,G'\right)\rangle \tag{11}$$

where $\mathbf{w}_h$, $\mathbf{z}_h\left(G,G'\right) \in \mathbb{R}^{|\Sigma^h|}$ are the vectors of entries w_v and $\min(\mu_{\mathbf{X}_h(G)}(v), \mu_{\mathbf{X}_h(G')}(v))$, respectively, for $v \in \Sigma^h$; b is a constant to ensure that the value of parametric function is nonnegative. In vector form, this can be expressed as:

$$d_{\mathbf{W}}(G,G') = b - \langle \mathbf{W}, \mathbf{Z}\left(G,G'\right)\rangle \tag{12}$$

where $\begin{cases} \mathbf{Z}(G,G') = \frac{1}{H}\left[\mathbf{z}_1(G,G')^T,...,\mathbf{z}_H(G,G')^T\right]^T \\ \mathbf{W} = \left[\mathbf{w}_1^T,...,\mathbf{w}_H^T\right]^T \end{cases}$

The parametric form (12) is a linear function with respect to the parameter vector $\mathbf{W} \in \mathbb{R}^d$ $(d = |\Sigma^1| + ... + |\Sigma^H|)$ and $\mathbf{Z}(G,G')$ is considered as a feature vector of a pair of graphs G and G'. Once the parameters are estimated, we can derive a similarity matrix through the Laplacian kernel as in Eq. (7). More importantly, as $\mathbf{W}$ is nonnegative, it is easy to see that $d_{\mathbf{W}}$ is a CND function, and thus the derived similarity matrix is valid.

3.2 Formulation of learning subtree pattern weights $\mathbf{W}$

We aim to learn the parameters W in Eq. (12) using the notions of metric learning [7]. That is two input graphs with the same labels are encouraged to be closer while the two with different labels become far away from each other. In other words, within class distances should be small, while between class distances should be large. For this purpose, as a loss function for a graph pair g and g', we can use the following two hinge loss function: $\max(0, \alpha_1 - d_{\mathbf{W}}(g,g'))$ if g and g' are with different labels and $\max(0, d_{\mathbf{W}}(g,g') - \alpha_2)$ otherwise, for learning subtree pattern weights, where α_1 and α_2 are constants $(\alpha_1 \geq \alpha_2)$. The former yields a penalty if g and g' of different labels are closer than α_1 while the latter yields a penalty when g and g' of the same label are more distant than α_2. Instead of using these functions in the optimization problem, we use their smooth versions: V_1 and V_2 (see Figure 2), which offer useful properties for deriving a generalization bound for the problem in Section 4. The derivation of these functions is based on the connection between the

strong convexity of a function and Lipschitz continuous gradient of its Fenchel dual (see more details in [10]).

More concretely, let $D_n = \{z_1 = (g_1, y_1), ..., z_n = (g_n, y_n)\}$ where $g_i \in \mathscr{G}$ and $y_i \in \mathscr{Y} = \{-1, 1\}$, for $i = 1, ..., n$, we formulate a constrained minimization problem for learning subtree pattern weights as follows:

$$\begin{aligned} \underset{\mathbf{W}}{\text{minimize}} \quad & \frac{1}{n^2} \sum_{i=1}^{n} \sum_{j=1}^{n} \ell(\mathbf{W}, z_i, z_j) \\ \text{subject to} \quad & \mathbf{W} \in \mathscr{C} \end{aligned} \tag{13}$$

where $\mathscr{C} = \{\mathbf{W} \in \mathbb{R}^d : \mathbf{W} = \left[\mathbf{w}_1^T, ..., \mathbf{w}_H^T\right]^T, ||\mathbf{w}_h - \mathbf{c}_h||_2 \leq \varepsilon_h, 1 \leq h \leq H\}$ ($\mathbf{c}_h$ and ε_h are constant vectors and scalars); ℓ is a continuously differentiable function and defined as follows:

$$\ell(\mathbf{W}, z_i, z_j) = \begin{cases} V_1(\mathbf{W}, g_i, g_j) & \text{if } y_i \neq y_j \\ V_2(\mathbf{W}, g_i, g_j) & \text{otherwise} \end{cases}$$

$$V_1(\mathbf{W}, g_i, g_j) = \begin{cases} 0 & \text{if } d_{\mathbf{W}}(g_i, g_j) \geq \alpha_1 \\ \alpha_1 - \frac{\sigma}{2} - d_{\mathbf{W}}(g_i, g_j) & \text{if } d_{\mathbf{W}}(g_i, g_j) \leq \alpha_1 - \sigma \\ \frac{1}{2\sigma}\left(d_{\mathbf{W}}(g_i, g_j) - \alpha_1\right)^2 & \text{if } \alpha_1 - \sigma < d_{\mathbf{W}}(g_i, g_j) < \alpha_1 \end{cases} \tag{14}$$

$$V_2(\mathbf{W}, g_i, g_j) = \begin{cases} 0 & \text{if } d_{\mathbf{W}}(g_i, g_j) \leq \alpha_2 \\ d_{\mathbf{W}}(g_i, g_j) - \alpha_2 - \frac{\sigma}{2} & \text{if } d_{\mathbf{W}}(g_i, g_j) \geq \alpha_2 + \sigma \\ \frac{1}{2\sigma}\left(d_{\mathbf{W}}(g_i, g_j) - \alpha_2\right)^2 & \text{if } \alpha_2 < d_{\mathbf{W}}(g_i, g_j) < \alpha_2 + \sigma \end{cases} \tag{15}$$

and α_1, α_2 ($\alpha_1 \geq \alpha_2$) and σ are constants; $d_{\mathbf{W}}(g_i, g_j)$ is calculated from $\mathbf{Z}(g_i, g_j)$ as in Eq. (12). To reduce notation, we use $\mathbf{Z}_{i,j}$ rather than $\mathbf{Z}(g_i, g_j)$ in the rest of the paper.

Lemma 3. *Let $\beta = \max_{1 \leq i < j \leq n} \left\|\mathbf{Z}_{ij}\right\|_2$, $L = \frac{\beta^2}{\sigma}$, and $M = \max(b - \frac{\sigma}{2} - \alpha_2, \alpha_1 - \frac{\sigma}{2})$, the loss function ℓ defined in problem (13) is L-lipschitz, β-smooth and upper bounded by M.*

Proof. First, it is obvious to see that as $0 \leq d_{\mathbf{W}} \leq b$, we have the following bounds: $0 \leq V_1 \leq \alpha_1 - \frac{\sigma}{2}$ and $0 \leq V_2 \leq b - \frac{\sigma}{2} - \alpha_2$. Therefore, ℓ is upper bounded by M.

We derive the first and second order derivatives of the smooth function V_1 as follows:

$$\nabla V_1(\mathbf{W}, g_i, g_j) = \begin{cases} 0 & \text{if } d_{\mathbf{W}}(g_i, g_j) \geq \alpha_1 \\ \mathbf{Z}_{ij} & \text{if } d_{\mathbf{W}}(g_i, g_j) \leq \alpha_1 - \sigma \\ \frac{1}{\sigma}\left(\alpha_1 - d_{\mathbf{W}}(g_i, g_j)\right)\mathbf{Z}_{ij} & \text{if } \alpha_1 - \sigma < d_{\mathbf{W}}(g_i, g_j) < \alpha_1 \end{cases} \tag{16}$$

$$\nabla^2 V_1(\mathbf{W}, g_i, g_j) = \begin{cases} 0 & \text{if } d_{\mathbf{W}}(g_i, g_j) \geq \alpha_1 \\ 0 & \text{if } d_{\mathbf{W}}(g_i, g_j) \leq \alpha_1 - \sigma \\ \frac{1}{\sigma}\mathbf{Z}_{ij}\mathbf{Z}_{ij}^T & \text{if } \alpha_1 - \sigma < d_{\mathbf{W}}(g_i, g_j) < \alpha_1 \end{cases} \tag{17}$$

In order to prove the function V_1 is L-Lipschitz and β-smooth, it is sufficient to show that the norm of its derivative is always less than L: $\left\|\nabla V_1(\mathbf{W}, g_i, g_j)\right\|_2 \leq L$ and the spectral norm (or the maximum eigen value) of its second order derivative is always less than β: $||\nabla^2 V_1(\mathbf{W}, g_i, g_j)|| \leq \beta$, $\forall g_i, g_j \in \mathscr{G}$. Indeed, from Eq. (16) and (17), we have: $\left\|V_1(\mathbf{W}, g_i, g_j)\right\|_2 \leq \left\|\mathbf{Z}_{ij}\right\|_2$ and $\left\|\nabla^2 V_1(\mathbf{W}, g_i, g_j)\right\| \leq \frac{1}{\sigma}\left\|\mathbf{Z}_{ij}\mathbf{Z}_{ij}^T\right\| = \frac{1}{\sigma}\left\|\mathbf{Z}_{ij}\right\|_2^2$. Also, we can bound $\left\|\mathbf{Z}_{ij}\right\|_2$ by the inequality $\left\|\mathbf{Z}_{ij}\right\|_2 \leq \beta$. Thus V_1 is β-smooth and L-Lipschitz. Similarly, we can also show that V_2 is β-smooth and L-Lipschitz. The lemma is proven. □

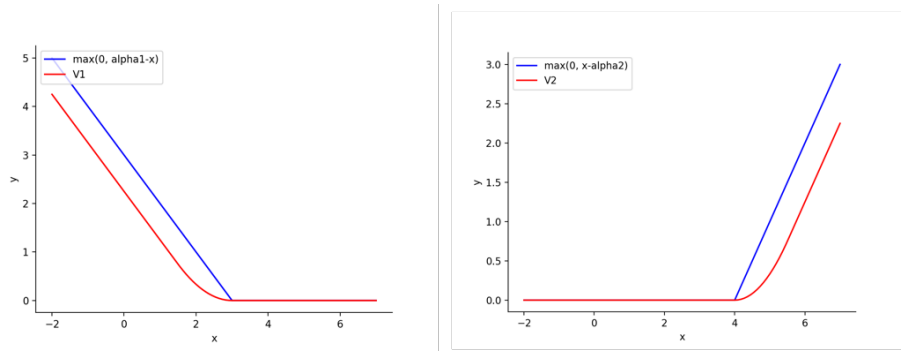

Fig. 2 Illustration of Hinge loss $\max(0, \alpha_1 - x)$ and $\max(0, x - \alpha_2)$; and their smooth versions: V_1 (left) and V_2 (right), respectively, with $\alpha_1 = 3, \alpha_2 = 4$ and $\sigma = 1.5$.

3.3 A stochastic learning algorithm for constrained optimization

The constrained optimization problem (13) is convex and thus guarantees to find its global optimum. Standard methods such as projected gradient descent can be used to solve the problem (13). However, for large scale data sets, solving the problem (13), involving n^2 terms with d parameters, might be computationally expensive. For instance, the data set PROTEIN (see Table 1) has $n^2 > 10^6$ pairs of examples and the weight vector size of $d > 10^5$ with the number of WL iterations $H = 5$. In this subsection, we present an efficient stochastic learning algorithm for dealing with this issue.

Algorithm 1: A stochastic algorithm for learning $\mathbf{W}$

Input: $D_n = \{z_1, ..., z_n\}$, c, μ: learning rate and T:#maxIters
Output: solution $\mathbf{W}^\star$
$\mathbf{W}^{(0)} = \mathbf{1}_d$, $t = 0$;
while $t \leq T$ **do**
  randomly pick two examples $z = (g, y)$ and $z' = (g', y')$ from D_n;
  if $y = y'$ **then**
    $\text{grad}^{(t)} = \nabla V_1(\mathbf{W}^{(t)}, g, g')$;
  else
    $\text{grad}^{(t)} = \nabla V_2(\mathbf{W}^{(t)}, g, g')$;
  end
  $\mathbf{W}^{(t+1)} = \text{proj}_{\mathscr{C}}[\mathbf{W}^{(t)} - \mu \text{grad}^{(t)}]$;
end
$\mathbf{W}^\star = \mathbf{W}^{(T)}$;

Let $\mathbf{W}^{(t)}$ denote the weight at iteration t. The weight is initialized by a vector of ones: $\mathbf{W}^{(0)} = \mathbf{1}_d$, which is also the case of WWL kernel without learning subtree pattern importance. At each iteration t, we randomly pick up a pair of examples $(z = (g, y), z' = (g', y'))$ from the training data set D_n and compute the gradient $\text{grad}^{(t)}$ corresponding to this pair. In fact this step can be done efficiently due to the sparsity of the feature vector $\mathbf{Z}(g, g')$ in Eq. (12). Then, we update the current solution $\mathbf{W}^{(t)}$ to $\mathbf{W}^{(t+1)}$ by the following rule:

$$\mathbf{W}^{(t+1)} = \text{proj}_{\mathscr{C}}[\mathbf{W}^{(t)} - \mu \text{grad}^{(t)}]$$

where $\text{proj}_{\mathscr{C}}[\mathbf{W}]_h = \begin{cases} \mathbf{w}_h & \text{if } ||\mathbf{w}_h - \mathbf{c}_h||_2 \leq \varepsilon_h \\ \frac{\varepsilon_h}{||\mathbf{w}_h - \mathbf{c}_h||_2}(\mathbf{w}_h - \mathbf{c}_h) + \mathbf{c}_h & \text{otherwise} \end{cases}$ maps a point $\mathbf{w}_h$ ($1 \leq h \leq H$) back to the bounded feasible region. The procedure is illustrated in Algorithm 1.

4 Theoretical Guarantees: A Bound on Generalization Gap

In this section, we provide a bound on the generalization gap of the proposed stochastic learning algorithm for solving the problem (13). The gap is defined as the expected difference between the generalization error $R(.)$ and empirical error $R_{D_n}(\cdot)$. In order to derive the generalization bound, we first provide basic setup and notations; then prove that our learning algorithm has a uniform stability, which is established in Theorem 5 using Lemma 3 and Lemma 4; finally derive our generalization bound, which is established in Theorem 9 using the McDiarmid inequality (see Theorem 6).

4.1 Basic setup and notations

Generalization Error. Let $\mathbf{W}_n$ be the parameters of the parametric function (12) obtained by training on the data set D_n using Algorithm 1. Then the generalization error (or risk) $R(\mathbf{W}_n)$ with a loss function ℓ is defined as:

$$R(\mathbf{W}_n) = \mathbf{E}_{z,z'}[\ell(\mathbf{W}_n, z, z')]$$

where $\mathbf{E}_{z,z'}[\ell(\cdot,\cdot,\cdot)]$ denotes the expectation of function ℓ when z and z' are sampled according the distribution $\mathscr{P}$.

Empirical Error. The empirical error $R_{D_n}(\mathbf{W}_n)$ is defined on the training data set D_n as :

$$R_{D_n}(\mathbf{W}_n) = \frac{1}{n^2}\sum_{z_i \in D_n}\sum_{z_j \in D_n} \ell(\mathbf{W}_n, z_i, z_j)$$

Expected Generalization Gap. As Algorithm 1 is based on a randomized procedure, we use the definition of the expected generalization gap as follows:

$$\mathbb{K}_n = \mathbf{E}_{\text{SGD}}[R(\mathbf{W}_n) - R_{D_n}(\mathbf{W}_n)]$$

where $\mathbf{E}_{\text{SGD}}$ denotes the expectation taken over the inherent randomness of the stochastic algorithm.

4.2 Uniform stability of the stochastic learning algorithm

Intuitively, a learning algorithm is said to have a uniform stability if its output is stable under a small modification of the training data set. For a randomized learning algorithm, the uniform stability property is defined as follows:

Definition 2 (Uniform Stability of the randomized algorithm)**.** A randomized algorithm $\mathbb{A}$ is β_n-uniformly stable with respect to a loss function ℓ, if the following inequality holds:

$$\forall (D_n, k),\ \sup_{z,z'} |\mathbf{E}_{\text{SGD}}[\ell(\mathbf{W}_n, z, z')] - \mathbf{E}_{\text{SGD}}[\ell(\mathbf{W}_{n,k}, z, z')]| \leq \beta_n$$

where $D_{n,k}$ is the new data set obtained from D_n by replacing $z_k \in D_n$ with a new example $\hat{z}_k$ sampled from $\mathscr{P}$; $\mathbf{W}_n$ and $\mathbf{W}_{n,k}$ are the outputs of $\mathbb{A}$ trained on two data sets D_n and $D_{n,k}$, respectively.

In order to prove that Algorithm 1 has the uniform stability property, we need the following lemma (its proof is placed in the appendix section):

Lemma 4. *Let the loss function ℓ defined in the problem (13) be β-smooth and L-Lipschitz; $\boldsymbol{W}_n^{(T)}$ and $\boldsymbol{W}_{n,k}^{(T)}$ be the parameters of the parametric form (12) trained on D_n and $D_{n,k}$, respectively, using Algorithm 1 with the number of iterations T and learning rate μ. Then, the expected difference in the model parameters is upper bounded by:*

$$\boldsymbol{E}_{SGD}\left[\left\|\boldsymbol{W}_n^{(T)} - \boldsymbol{W}_{n,k}^{(T)}\right\|_2\right] \leq \frac{4}{n}\mu TL \tag{18}$$

Using Lemma 4 and L-Lipschitz property of function ℓ (see Lemma 3), we can now prove the stability of Algorithm 1.

Theorem 5. *[Uniform Stability of Algorithm 1] Let the loss function ℓ defined in the problem (13) be β-smooth and L-Lipschitz. Then Algorithm 1 with the fixed learning rate μ is k_n-uniformly stable where $k_n = \frac{4}{n}\mu TL^2$.*

Proof. We have the following inequalities:

$$|\mathbf{E}_{\text{SGD}}[\ell(\mathbf{W}_n, z, z')] - \mathbf{E}_{\text{SGD}}[\ell(\mathbf{W}_{n,k}, z, z')]| \leq L\mathbf{E}_{\text{SGD}}\left\|\mathbf{W}_n - \mathbf{W}_{n,k}\right\|_2 \leq \frac{4}{n}\mu TL^2 \tag{19}$$

where the first and second inequalities are obtained by the L-Lipschitz property of ℓ and Lemma 4, respectively. This completes the proof. □

4.3 Bound on generalization gap

Using the property of uniform stability in the previous subsection, we can derive the generalization bound which is done by the McDiarmid inequality [8].

Theorem 6. *[McDiarmid inequality [8]] Let $X_1, X_2, ..., X_n$ be n independent random variables taking values in $\mathscr{X}$ and let $Z = f(X_1, ..., X_n)$. If, for each $1 \leq i \leq n$, there exists a constant c_i such that*

$\sup_{x_1,...,x_n,x_i'} |f(x_1, ..., x_n) - f(x_1, ..., x_i', ..., x_n)| \leq c_i, \forall 1 \leq i \leq n,$

then for any $\varepsilon > 0$, $Pr[|Z - \boldsymbol{E}[Z]| \geq \varepsilon] \leq 2\exp\left(\frac{-2\varepsilon^2}{\sum_{i=1}^n c_i^2}\right)$

To derive the bound on $R(W_n)$, we replace Z by K_n in Theorem 6 and bound $\mathbf{E}_{\text{SGD}}[\mathbb{K}_n]$ and $|\mathbb{K}_n - \mathbb{K}_{n,k}|$ which are established by the following lemmas (see their proofs in the appendix section).

Lemma 7. *For the loss function satisfying a uniform stability in k_n, we have the following inequality:*

$$\boldsymbol{E}_{D_n}[\mathbb{K}_n] \leq 2k_n \tag{20}$$

Lemma 8. *For the loss function satisfying a uniform stability in k_n and upper bounded by M, we have the following inequality:*

$$|\mathbb{K}_n - \mathbb{K}_{n,k}| \leq 2k_n + \frac{2M}{n} \tag{21}$$

Now we can derive the generalization bound for $R(\mathbf{W}_n)$ in the following theorem:

Theorem 9. *Let D_n be a training data set with n samples, $\boldsymbol{W}_n$ be the solution obtained by minimizing the optimization problem (13) using Algorithm 1 with uniform stability k_n. Then the following inequality holds for probability of at least $1-\delta$ $(0 \leq \delta \leq 1)$:*

$$\boldsymbol{E}_{SGD}\left[R(\boldsymbol{W}_n) - R_{D_n}(\boldsymbol{W}_n)\right] \leq 2k_n + (nk_n + M)\sqrt{\frac{2}{n}\log\frac{2}{\delta}} \tag{22}$$

Proof. Applying McDiarmid's concentration inequality (6) by replacing Z with $\mathbb{K}_n$, we have:

$$Pr(\mathbb{K}_n - \mathbf{E}_{D_n}\left[\mathbb{K}_n\right] \geq \varepsilon) \leq 2\exp\left(\frac{-2\varepsilon^2}{n\left(2k_n + \frac{2M}{n}\right)^2}\right)$$

By fixing $\delta = 2\exp\left(\frac{-2\varepsilon^2}{n\left(2k_n+\frac{2M}{n}\right)^2}\right)$, we get $\varepsilon = (nk_n + M)\sqrt{\frac{2}{n}\log\frac{1}{\delta}}$ which completes the proof of Theorem 9. □

The generalization bound is meaningful if the bound converge to 0 as $n \to \infty$. Our derived generalization bound converges to 0 as k_n decays with $O(\frac{1}{n})$. This confirms Algorithm 1 converges.

5 Experiments

In this section, we demonstrate the benefit of learning subtree pattern weights by experiments on both synthetic and real-world data. We performed classification experiments using the C-SVM implementation LIBSVM [2]. The necessary parameters of SVM were selected by cross-validation on the training set. These are the regularization parameter $C \in \{10^{-3}, 10^{-2}, ..., 10^2, 10^3\}$ and kernel parameter $\gamma \in \{0.0001, 0.001, 0.01\}$. For learning the weights $\mathbf{W}$ of subtree patterns (or WL labels) in Algorithm 1, the learning rate μ and maximum number of iterations T were set as 0.0001 and 500, respectively; $\varepsilon_h \in \{0.1, 0.5, 1.0\}$ were selected by cross validation based on the training set and $\mathbf{c}_h$ was fixed as a vector of ones, i.e. $\mathbf{c}_h = \mathbf{1}_{|\Sigma^h|}$ for $h = 1, .., H$; the hyperparameters α_1, α_2 and σ were empirically determined as 1.0, 0.5 and 0.1, respectively. All kernels were implemented in Python 3.0 and experiments were conducted on an Intel Core i9 at 2.3 Ghz with 64GB of RAM using a single processor only. The source code can be accessed through `https://github.com/haidnguyen0909/weightedWWL`.

5.1 Synthetic data

We designed eight substructures, shown in Figure 3, in which substructures indexed by 1, 2, 5 and 6 are assumed to be indicative to positive class (+1) as they have a pattern '1-0(-2)-0' in common. The others are indicative to negative class (-1). Our synthetic data set consists of *eight groups of graphs, each corresponds to one of these eight substructures* by randomly adding noisy nodes and edges. We used groups corresponding substructures 1, 2, 3 and 4 as training data and the others as testing data. We constructed two kernels for graphs: WWL and the proposed method with number of WL iterations $H = 2$, then used SVM for classification. We reported mean accuracy obtained by ten synthetic data sets generated in this way.

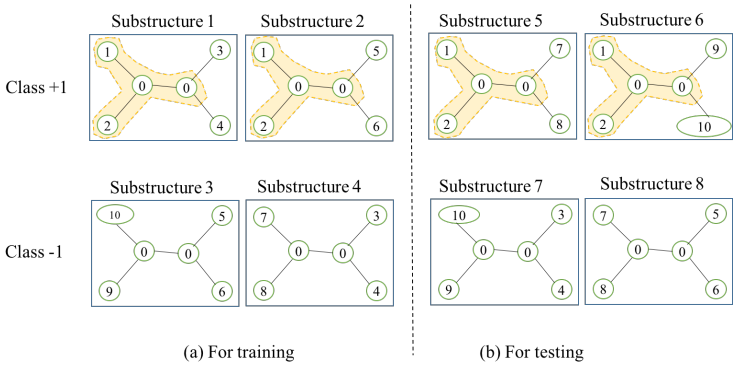

Fig. 3 Designed substructures 1-8: substructures 1,2,5 and 6 are indicative to positive class as they contain pattern '1-0(-2)-0' (emphasized in yellow). The rest are indicative to negative class. Graphs are generated from substructures by adding random nodes and noisy edges (20 examples for each substructure). Graphs generated from the substructures 1, 2, 3 and 4 are used for training. Graph generated from the substructures 5,6,7 and 8 are used for testing.

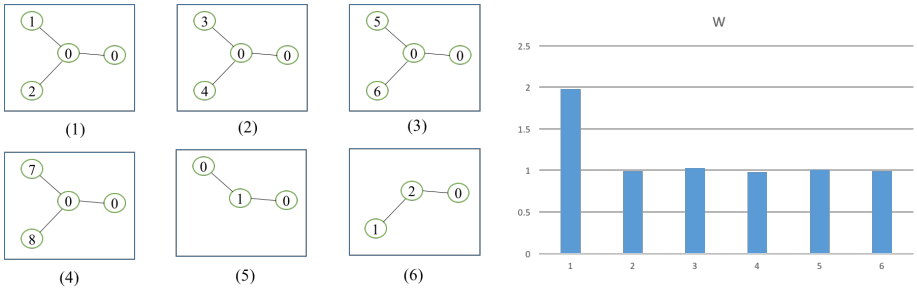

Fig. 4 Examples of selected subtree patterns of the first level $h = 1$ (left) and their weights learned by the proposed learning algorithm (right).

We observed that WWL obtained mean accuracies of 82.4%, while the proposed method achieved significantly higher accuracy of 95%. It is noted that the testing examples were confusing the classifier. For instance, the substructure 5 has the same similarity with substructures 2 (indicative to positive class) and 4 (indicative to negative class) according to the WWL kernel, making it hard for the classifier to distinguish graphs containing the substructure 5. In contrast, this confusion can be alleviated by learning subtree pattern weights. In particular, the pattern '1-0(-2)-0' present in the substructure 5 is assigned a high weight by the proposed method (see Figure 4). Therefore, graphs generated from the substructure 5 are more likely to be classified as positive.

5.2 Real-world data

In this subsection we present an experimental evaluation of the proposed method on real-world data. We report experimental results on four benchmark bioinformatics data sets, involving node-labeled graphs, particularly, MUTAG, PTC-MR, PROTEIN and NCI1. The MUTAG dataset consists of graph structures of 188 chemical compounds which are either mutagenic aromatic or heteroromatic nitro compounds and nodes can have 7 discrete labels.

Table 1 Statistics of datasets used in experiments

Datasets	#Graphs	#Classes	Avg. card(V)	Avg. card(E)	#labels
MUTAG	188	2 (125 vs. 63)	17.9	39.6	7
PTC-MR	344	2 (192 vs. 152)	25.6	51.9	19
PROTEIN	1113	2 (663 vs. 450)	39.1	145.63	3
NCI1	4110	2 (2053 vs. 2057)	N/A	N/A	N/A

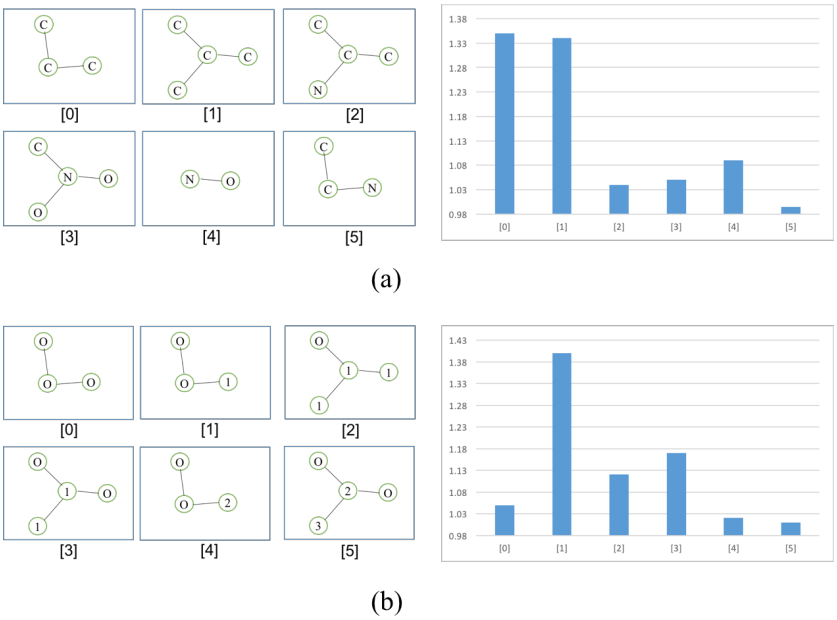

Fig. 5 Examples of selected subtree patterns and their weights of the first level $h = 1$ (a) and the second level $h = 2$ (b) extracted from MUTAG by Algorithm 1.

The PTC-MR dataset consists of 344 chemical compounds which are known to cause or not cause cancer in rats and mice, and nodes can have 19 discrete labels. The PROTEIN dataset consists of relations between secondary structure elements represented by nodes and neighborhood in the amino-acid sequence or in 3D space by edges, and nodes can have 3 discrete labels. The NCI1 dataset is a balanced subset of chemical compounds screened for their ability to inhibit the growth of a panel of human tumor cell lines, and nodes can have 37 discrete labels. Some statistics of these data sets are shown in Table 1.

We compared the proposed method to several state-of-the-art graph kernels. Due to the large number of graph kernels in the literature, we selected representatives of the major families of graph kernels. In particular, for the family of walk based kernels, we compared the proposed method to the fast random walk kernel [5] that essentially counts the common labeled walks. For the family of path based graph kernels, we compared to the shortest path kernel [1]. For the family of WL based graph kernels, we compared to WL subtree [17], WL-OA [6] and WWL kernels [14]. The WL based kernels have been shown to be superior to previous approaches.

We report mean predictive accuracies and standard deviations obtained by 10-fold cross-validation repeated 10 times with random fold assignments. Within each fold, the number

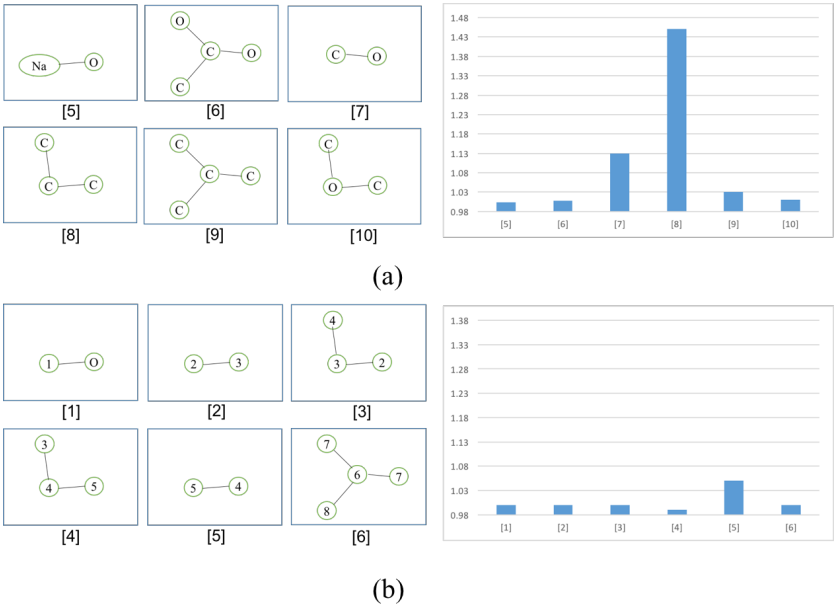

Fig. 6 Examples of selected subtree patterns and their weights of the first level $h = 1$ (a) and the second level $h = 2$ (b) extracted from PTC-MR by Algorithm 1.

of hops $H \in \{1, 2, ..., 6\}$ was selected by cross validation based on the training set. The results evaluated by classification accuracy are summarised in Table 2. We used one-sided paired *t-test* to verify if the accuracy differences between two methods on data sets are statistically significant. We empirically observed that random walk and shortest path kernels were less competitive to WL-based kernels on four data sets. On three datasets MUTAG, PROTEIN and NCI1, the proposed method was comparable with WL-OA while it improved its unweighted version WWL by 1.4%, 1.5% and 0.8%, respectively (the calculated p-values were 0.061, 0.0087 and 0.055, respectively, smaller than the significance level of $\alpha = 0.1$). On PTC-MR, the proposed method improved WWL by 0.6% while outperforming WL-OA by nearly 5% (the calculated p-values were 0.0035 and < 0.001, respectively). In all these data sets, random walk, shortest path and WL subtree kernels were dominated by the rest in large margins. Furthermore, we also investigated some selected subtree patterns at the first and second levels ($h = 1, 2$) along with their weights learned by the proposed algorithm from two data sets: MUTAG and PTC-MR (see Figures 5 and 6, respectively). Interestingly, the weights were found different over substructures, indicating their different degrees of importance in the prediction task. These experimental results confirmed the effectiveness of learning important subtree patterns for WWL kernels.

5.3 Computational efficiency of the proposed stochastic algorithm

In this subsection, we evaluate the computational efficiency of the proposed Algorithm 1. We empirically compared two variants: batch and stochastic (Algorithm 1), for solving the minimization problem (13) in terms of running time. The first variant considers all pairs of

Table 2 Classification accuracies and standard deviation on real-world graph data sets: MUTAG, PTC-MR, PROTEIN and NCI1.

Kernels	MUTAG	PTC-MR	PROTEIN	NCI1
Random Walk [5]	85.06 ± 0.18	55.74 ± 3.64	71.11 ± 0.83	62.88 ± 0.22
Shortest path [1]	85.49 ± 0.59	53.29 ± 0.92	73.03 ± 1.13	61.36 ± 0.19
WL Subtree [13]	85.61 ± 0.85	61.89 ± 1.97	72.5 ± 0.32	85.61 ± 0.13
WL-OA [6]	**88.17** ± 1.98	60.49 ± 1.39	**75.89** ± 0.41	**86.17** ± 0.35
WWL [14]	86.95 ± 1.35	64.86 ± 1.57	74.25 ± 0.74	85.69 ± 0.28
Proposed	**88.37** ± 1.82	**65.44** ± 0.97	**75.73** ± 0.57	**86.45** ± 0.11

Table 3 Running time in seconds of two variants: full batch and stochastic on real-world data sets: MUTAG, PTC-MR, PROTEIN and NCI1.

variants/ data sets	MUTAG	PTC-MR	PROTEIN	NCI1
Full batch	4h 29'	8h 50'	$>$ 3 days	$>$ 1 week
Stochastic	1' 32"	3' 48"	1h 20'40"	5h 5'

graphs for every step of projected gradient descent. The second variant considers one pair of graphs at a time to take a single step.

First we assessed the running time of two variants on randomly generated graphs (as described in Subsection 5.1) with respect to two parameters: number of graphs N and number of WL iterations H. We varied N in range $\{50, 100, 200, 400, 600, 800, 1000\}$ and H in range $\{1, 2, 3, 4, 5, 6, 7\}$. For each individual experiment, we fixed one parameter at its default value and varied the other. The default values were 100 for N and 2 for H. We report CPU running times in seconds in Figure 7. Empirically, we observed that the running time of full batch variant increased quickly when increasing the number of graphs N and the number of WL iterations H. In contrast, the stochastic variant scaled much better with much lower running times, indicating that Algorithm 1 has high scalability for large scale data sets. The computational efficiency of Algorithm 1 can be explained by the fact that computing gradient of the loss function for a graph pair g and g' involves a few substructures (or WL labels) v shared by g and g' in Eq. (12), i.e., sparsity of the feature vector $\mathbf{Z}(g, g')$.

Second we assessed the running time of two variants on real-world data sets: MUTAG, PTC-MR, PROTEIN and NCI1. We reported the running time of two variants to finish the entire classification tasks, including learning subtree pattern weights, computing kernels and doing classification, in Table 3. The running time were taken average by 10-fold cross validation. We empirically observed that the full batch variant was slow when running on even small data sets MUTAG and PTC-MR, taking in approximately 4 hours and 9 hours, respectively. But the stochastic variant was much faster, taking only less than 4 minutes on the two data sets. We also observed that the stochastic variant could easily scale up to data sets with thousands of graphs. Particularly, on data sets PROTEIN and NCI1, the tasks were performed in nearly 1h 30' and 5h, respectively. However, it was impossible for the full batch variant to finish the tasks in less than 3 days. These evidence showed that the proposed stochastic variant is highly scalable.

6 Conclusion and Discussion

In this work, we proposed to learn the weights of substructures of graphs, particularly, *subtree patterns* (extracted by WL labeling scheme), to overcome the limitations of current graph kernels. We considered the problem of incorporating subtree pattern weights for

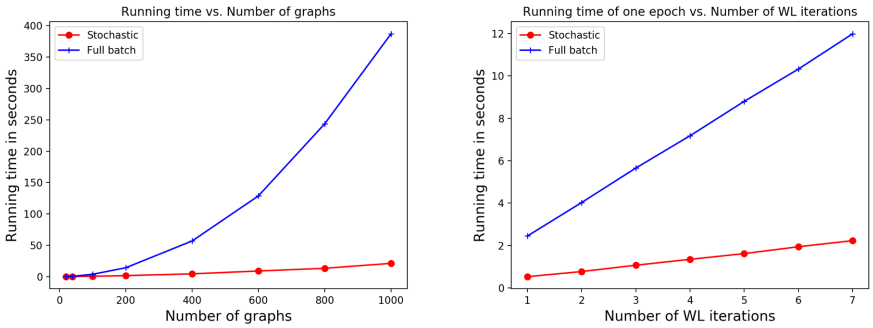

Fig. 7 Running time in seconds on synthetic data sets of two variants: full batch and stochastic algorithms, for learning the subtree pattern importance W (in the optimization problem (13)) (Default values: dataset size $N = 100$, WL iteration $H = 2$).

WWL kernel [14] by formulating the parametric form of Wasserstein distance taking into account subtree pattern weights, and learning these weights from data optimally for the tasks.

Our proposed method has several advantages. First, it can learn the importance of subtree patterns specifically for the tasks through their weights in the parametric distance function. Second, the kernels converted from the learned parametric function of subtree pattern weights are valid. Third, the efficient stochastic algorithm for learning the weights has high scalability for large scale data sets, and its theoretical guarantees are provided.

Although we considered WWL kernel for extracting important subtree patterns, an interesting and worthwhile extension of our work would be to apply this idea to other WL based graph kernels such as WL subtree and WL-OA kernels. The improvements of the optimization algorithm for learning subtree pattern weights in terms of both convergence and efficiency would also be our future work.

References

1. Borgwardt, K.M., Kriegel, H.P.: Shortest-path kernels on graphs. In: Fifth IEEE international conference on data mining (ICDM'05), pp. 8–pp. IEEE (2005)
2. Chang, C.C., Lin, C.J.: Libsvm: A library for support vector machines. ACM transactions on intelligent systems and technology (TIST) **2**(3), 1–27 (2011)
3. Hardt, M., Recht, B., Singer, Y.: Train faster, generalize better: Stability of stochastic gradient descent. In: International Conference on Machine Learning, pp. 1225–1234. PMLR (2016)
4. Haussler, D.: Convolution kernels on discrete structures. Tech. rep., Technical report, Department of Computer Science, University of California ... (1999)
5. Kashima, H., Tsuda, K., Inokuchi, A.: Marginalized kernels between labeled graphs. In: Proceedings of the 20th international conference on machine learning (ICML-03), pp. 321–328 (2003)
6. Kriege, N.M., Giscard, P.L., Wilson, R.: On valid optimal assignment kernels and applications to graph classification. In: Advances in Neural Information Processing Systems, pp. 1623–1631 (2016)
7. Kulis, B., et al.: Metric learning: A survey. Foundations and trends in machine learning **5**(4), 287–364 (2012)
8. McDiarmid, C.: On the method of bounded differences. Surveys in combinatorics **141**(1), 148–188 (1989)
9. Munkres, J.: Algorithms for the assignment and transportation problems. Journal of the society for industrial and applied mathematics **5**(1), 32–38 (1957)
10. Nesterov, Y.: Smooth minimization of non-smooth functions. Mathematical programming **103**(1), 127–152 (2005)

11. Scott, J.: Social network analysis: developments, advances, and prospects. Social network analysis and mining **1**(1), 21–26 (2011)
12. Sharan, R., Ideker, T.: Modeling cellular machinery through biological network comparison. Nature biotechnology **24**(4), 427–433 (2006)
13. Shervashidze, N., Borgwardt, K.: Fast subtree kernels on graphs. In: Advances in neural information processing systems, pp. 1660–1668 (2009)
14. Togninalli, M., Ghisu, E., Llinares-López, F., Rieck, B., Borgwardt, K.: Wasserstein weisfeiler-lehman graph kernels. In: Advances in Neural Information Processing Systems, pp. 6439–6449 (2019)
15. Trinajstic, N.: Chemical graph theory. Routledge (2018)
16. Villani, C.: Optimal transport: old and new, vol. 338. Springer Science & Business Media (2008)
17. Vishwanathan, S.V.N., Schraudolph, N.N., Kondor, R., Borgwardt, K.M.: Graph kernels. The Journal of Machine Learning Research **11**, 1201–1242 (2010)

A Appendices

A.1 Proof of Lemma 4

.

Proof. We prove the lemma by following the notion of using the same randomness for two data set D_n and $D_{n,k}$ as in [3]. Particularly, we supply the sample sequences $S = \{p_1 = (z_{i_1}, z_{j_1}), ..., p_T = (z_{i_T}, z_{j_T})\}$ to two identical learning algorithms except that for some t ($1 \le t \le T$), if p_t contains z_k ($p_t = (z_k, z_{j_t})$ or (z_{i_t}, z_k)), we replace it with $\hat{p}_t = (\hat{z}_k, z_{j_t})$ or $(z_{i_t}, \hat{z}_k)$. So, there are two cases to consider:
Case 1: At step t, Algorithm 1 picks a pair of samples (z, z') that contain no z_k ($z \neq z_k$ and $z' \neq z_k$) and this case occurs with probability $(1 - \frac{1}{n})^2$. Then, we have:

$$\begin{aligned}
&\left\|\mathbf{W}_n^{(t+1)} - \mathbf{W}_{n,k}^{(t+1)}\right\|_2^2 = \left\|\text{proj}_{\mathscr{C}}\left[\mathbf{W}_n^{(t)} - \mu\nabla\ell\left(\mathbf{W}_n^{(t)}, z, z'\right)\right] - \text{proj}_{\mathscr{C}}\left[\mathbf{W}_{n,k}^{(t)} - \mu\nabla\ell\left(\mathbf{W}_{n,k}^{(t)}, z, z'\right)\right]\right\|_2^2 \\
&\le \left\|\mathbf{W}_n^{(t)} - \mu\nabla\ell\left(\mathbf{W}_n^{(t)}, z, z'\right) - \mathbf{W}_{n,k}^{t} + \mu\nabla\ell\left(\mathbf{W}_{n,k}^{(t)}, z, z'\right)\right\|_2^2 \\
&= \left\|\mathbf{W}_n^{(t)} - \mathbf{W}_{n,k}^{(t)}\right\|_2^2 - 2\mu\left(\mathbf{W}_n^{(t)} - \mathbf{W}_{n,k}^{(t)}\right)\left(\nabla\ell\left(\mathbf{W}_n^{(t)}, z, z'\right) - \nabla\ell\left(\mathbf{W}_{n,k}^{(t)}, z, z'\right)\right) \\
&+ \mu^2\left\|\nabla\ell(\mathbf{W}_n^{(t)}, z, z') - \nabla\ell\left(\mathbf{W}_{n,k}^{(t)}, z, z'\right)\right\|_2^2 \\
&\le \left\|\mathbf{W}_n^{(t)} - \mathbf{W}_{n,k}^{(t)}\right\|_2^2 - \mu\left(\frac{2}{\beta} - \mu\right)\left\|\nabla\ell\left(\mathbf{W}_n^{(t)}, z, z'\right) - \nabla\ell\left(\mathbf{W}_{n,k}^{(t)}, z, z'\right)\right\|_2^2
\end{aligned}$$

The second line is obtained by the fact that $\|\text{proj}_{\mathscr{C}}[\mathbf{u}] - \text{proj}_{\mathscr{C}}[\mathbf{v}]\|_2 \le \|\mathbf{u} - \mathbf{v}\|_2$ for $\mathbf{u}, \mathbf{v}$ in the domain. The last line is obtained by the β-smoothness of function ℓ (see Lemma 3). So we have the following inequality (by selecting $\mu \le \frac{2}{\beta}$):

$$\left\|\mathbf{W}_n^{(t+1)} - \mathbf{W}_{n,k}^{(t+1)}\right\|_2 \le \left\|\mathbf{W}_n^{(t)} - \mathbf{W}_{n,k}^{(t)}\right\|_2 \tag{23}$$

Case 2: At step t, Algorithm 1 picks a pair of samples (z, z') that contain z_k ($z = z_k$ and $z' = z_k$) and this case occurs with probability $1 - (1 - \frac{1}{n})^2$. Then we have:

$$\left\|\mathbf{W}_n^{(t+1)} - \mathbf{W}_{n,k}^{(t+1)}\right\|_2 \le \left\|\mathbf{W}_n^{(t)} - \mathbf{W}_{n,k}^{(t)}\right\|_2 + 2\mu L \tag{24}$$

The above inequality holds as the norm of gradient of loss function ℓ is upper bounded by L. From two inequalities (23) and (24), and considering the probabilities of two cases, we have the following inequality:

$$\mathbf{E}_{\text{SGD}}\left[\left\|\mathbf{W}_n^{(t+1)} - \mathbf{W}_{n,k}^{(t+1)}\right\|_2\right] \le \left(1 - \frac{1}{n}\right)^2 \mathbf{E}_{\text{SGD}}\left[\left\|\mathbf{W}_n^{(t)} - \mathbf{W}_{n,k}^{(t)}\right\|_2\right] \tag{25}$$

$$+ \left(1 - \left(1 - \frac{1}{n}\right)^2\right)\left(\mathbf{E}_{\text{SGD}}\left[\left\|\mathbf{W}_n^{(t)} - \mathbf{W}_{n,k}^{(t)}\right\|_2\right] + 2\mu L\right) \tag{26}$$

$$= \mathbf{E}_{\text{SGD}}\left[\left\|\mathbf{W}_n^{(t)} - \mathbf{W}_{n,k}^{(t)}\right\|_2\right] + 2\mu L\left(1 - \left(1 - \frac{1}{n}\right)^2\right) \tag{27}$$

By the above recursive formula, we obtain the following:

$$\mathbf{E}_{\mathrm{SGD}}\left[\left\|\mathbf{W}_n^{(T)}-\mathbf{W}_{n,k}^{(T)}\right\|_2\right]\leq 2\left(1-\left(1-\frac{1}{n}\right)^2\right)\mu TL\leq\frac{4}{n}\mu TL \tag{28}$$

which completes the proof. □

A.2 Proof of Lemma 7

Proof. By the definition of $\mathbb{K}_n$ as in (4.1), we have the following:

$$\begin{aligned}
&\mathbf{E}_{D_n}[\mathbb{K}_n]=\mathbf{E}_{D_n}\mathbf{E}_{\mathrm{SGD}}[R(\mathbf{W}_n)-R_{D_n}(\mathbf{W}_n)]\\
&=\mathbf{E}_{D_n}\mathbf{E}_{\mathrm{SGD}}\mathbf{E}_{z,z'}\ell(\mathbf{W}_n,z,z')-\mathbf{E}_{D_n}\mathbf{E}_{\mathrm{SGD}}\frac{1}{n^2}\sum_{i,j=1}^{n}\ell(\mathbf{W}_n,z_i,z_j)\\
&=\mathbf{E}_{D_n,z,z'}\mathbf{E}_{\mathrm{SGD}}\left[\frac{1}{n^2}\sum_{k=1}^{n}\sum_{j=1}^{n}\left[\ell(\mathbf{W}_n,z,z')-\ell(\mathbf{W}_n,z_k,z')+\ell(\mathbf{W}_n,z_k,z')-\ell(\mathbf{W}_n,z_k,z_j)\right]\right]\\
&=\mathbf{E}_{D_n,z,z'}\mathbf{E}_{\mathrm{SGD}}\left[\frac{1}{n^2}\sum_{k=1}^{n}\sum_{j=1}^{n}\left[\ell(\mathbf{W}_n,z,z')-\ell(\mathbf{W}_n,z_k,z')+\ell(\mathbf{W}_n,z_k,z')-\ell(\mathbf{W}_n,z_k,z_j)\right]\right]\\
&=\underbrace{\mathbf{E}_{D_n,z,z'}\mathbf{E}_{\mathrm{SGD}}\left[\frac{1}{n^2}\sum_{k=1}^{n}\sum_{j=1}^{n}\left[\ell(\mathbf{W}_n,z,z')-\ell(\mathbf{W}_n,z_k,z')\right]\right]}_{(a)}\\
&+\underbrace{\mathbf{E}_{D_n,z,z'}\mathbf{E}_{\mathrm{SGD}}\left[\frac{1}{n^2}\sum_{k=1}^{n}\sum_{j=1}^{n}\left[\ell(\mathbf{W}_n,z_k,z')-\ell(\mathbf{W}_n,z_k,z_j)\right]\right]}_{(b)}
\end{aligned}$$

We first process the part (a) which is equivalent to the following:

$$\begin{aligned}
(\mathrm{a})&=\frac{1}{n}\sum_{k=1}^{n}\mathbf{E}_{D_n,z,z'}\mathbf{E}_{\mathrm{SGD}}\left[\ell(\mathbf{W}_n,z,z')\right]-\frac{1}{n}\sum_{k=1}^{n}\mathbf{E}_{D_n,z,z'}\mathbf{E}_{\mathrm{SGD}}\left[\ell(\mathbf{W}_n,z_k,z')\right]\\
&=\frac{1}{n}\sum_{k=1}^{n}\mathbf{E}_{D_n,z,z'}\mathbf{E}_{\mathrm{SGD}}\left[\ell(\mathbf{W}_n,z,z')\right]-\frac{1}{n}\sum_{k=1}^{n}\mathbf{E}_{D_n,\hat{z}_k,z'}\mathbf{E}_{\mathrm{SGD}}\left[\ell(\mathbf{W}_n,z_k,z')\right]\\
&=\frac{1}{n}\sum_{k=1}^{n}\mathbf{E}_{D_n,z,z'}\mathbf{E}_{\mathrm{SGD}}\left[\ell(\mathbf{W}_n,z,z')\right]-\frac{1}{n}\sum_{k=1}^{n}\mathbf{E}_{D_n,z,z'}\mathbf{E}_{\mathrm{SGD}}\left[\ell(\mathbf{W}_{n,k},z,z')\right]\\
&\leq\frac{1}{n}\sum_{k=1}^{n}\mathbf{E}_{D_n,z,z'}\mathbf{E}_{\mathrm{SGD}}\left[\ell(\mathbf{W}_n,z,z')-\ell(\mathbf{W}_{n,k},z,z')\right]\leq k_n
\end{aligned}$$

Similarly, we also prove that $(b)\leq k_n$ which completes the proof. □

A.3 Proof of Lemma 8

Proof. By the definition of $\mathbb{K}_n$ as in (4.1), we have:

$$\begin{aligned}
|\mathbb{K}_n-\mathbb{K}_{n,k}|&=|\mathbf{E}_{\mathrm{SGD}}[R(\mathbf{W}_n)-R_{D_n}(\mathbf{W}_n)]-\mathbf{E}_{\mathrm{SGD}}[R(\mathbf{W}_{n,k})-R_{D_{n,k}}(\mathbf{W}_{n,k})]|\\
&\leq\underbrace{|\mathbf{E}_{\mathrm{SGD}}R(\mathbf{W}_n)-\mathbf{E}_{\mathrm{SGD}}R(\mathbf{W}_{n,k})|}_{(\mathrm{c})}+\underbrace{|\mathbf{E}_{\mathrm{SGD}}R_{D_n}(\mathbf{W}_n)-\mathbf{E}_{\mathrm{SGD}}R_{D_{n,k}}(\mathbf{W}_{n,k})|}_{(\mathrm{d})}
\end{aligned}$$

We bound the two terms (c) and (d) as follows:

$$\begin{aligned}(\mathrm{c}) &= |\mathbf{E}_{\mathrm{SGD}}\mathbf{E}_{z,z'}[\ell(\mathbf{W}_n, z, z') - \ell(\mathbf{W}_{n,k}, z, z')]| \\ &\leq \mathbf{E}_{z,z'}|\mathbf{E}_{\mathrm{SGD}}[\ell(\mathbf{W}_n, z, z') - \ell(\mathbf{W}_{n,k}, z, z')]| \leq k_n\end{aligned}$$

$$\begin{aligned}(\mathrm{d}) &= |\mathbf{E}_{\mathrm{SGD}}\frac{1}{n^2}\sum_{z_i\in D_n}\sum_{z_j\in D_n}\ell(\mathbf{W}_n, z_i, z_j) - \mathbf{E}_{\mathrm{SGD}}\frac{1}{n^2}\sum_{z_i\in D_{n,k}}\sum_{z_j\in D_{n,k}}\ell(\mathbf{W}_{n,k}, z_i, z_j)| \\ &= |\frac{1}{n^2}\sum_{i\neq k, j\neq k}\mathbf{E}_{\mathrm{SGD}}[\ell(\mathbf{W}_n, z_i, z_j) - \ell(\mathbf{W}_{n,k}, z_i, z_j)] \\ &+ \frac{1}{n^2}\sum_{i\neq k}\mathbf{E}_{\mathrm{SGD}}[\ell(\mathbf{W}_n, z_i, z_k) - \ell(\mathbf{W}_{n,k}, z_i, z_k)] \\ &+ \frac{1}{n^2}\sum_{j\neq k}\mathbf{E}_{\mathrm{SGD}}[\ell(\mathbf{W}_n, z_k, z_j) - \ell(\mathbf{W}_{n,k}, z_k, z_j)] \\ &\leq \frac{(n-1)^2}{n^2}k_n + \frac{2(n-1)}{n^2}M < k_n + \frac{2M}{n}\end{aligned}$$

The two above inequalities complete the proof. □