Metric Learning for Ordered Labeled Trees with pq-grams

Hikaru Shindo[1] and **Masaaki Nishino**[2] and **Yasuaki Kobayashi**[3] and **Akihiro Yamamoto**[4]

Abstract. Computing the similarity between two data points plays a vital role in many machine learning algorithms. Metric learning has the aim of learning a good metric automatically from data. Most existing studies on metric learning for tree-structured data have adopted the approach of learning the tree edit distance. However, the edit distance is not amenable for big data analysis because it incurs high computation cost. In this paper, we propose a new metric learning approach for tree-structured data with pq-grams. The pq-gram distance is a distance for ordered labeled trees, and has much lower computation cost than the tree edit distance. In order to perform metric learning based on pq-grams, we propose a new differentiable parameterized distance, *weighted pq-gram distance*. We also propose a way to learn the proposed distance based on Large Margin Nearest Neighbors (LMNN), which is a well-studied and practical metric learning scheme. We formulate the metric learning problem as an optimization problem and use the gradient descent technique to perform metric learning. We empirically show that the proposed approach not only achieves competitive results with the state-of-the-art edit distance-based methods in various classification problems, but also solves the classification problems much more rapidly than the edit distance-based methods.

1 Introduction

The performance of many machine learning algorithms depends on the way in which the distance or similarity between data points are measured [19]. For instance, k-nearest neighbor classification [7] decides the class label of a data point from those of its neighbors, while Learning Vector Quantization (LVQ) [17] classifies each data point based on the closest prototype according to a given distance measure. Clustering algorithms, such as K-Means [20], rely on a given distance or similarity function for input data. In order for these algorithms to perform accurately and efficiently, a metric that suits the given problem is necessary. The metric should capture the characteristics of the datasets expected, i.e., a pair of data points from the same class is closer than a pair of points from different classes.

The objective of metric learning is to learn a good metric from data since handcrafting good metrics for specific problems is generally difficult [5]. Metric learning has been an active research topic because of its applicability, i.e., any algorithm using a distance measure internally can benefit from its results [5, 19]. In metric learning, a training set consists of sets of pairs; *positive pairs* and *negative pairs*. Metrics are learned by optimizing a loss function that makes
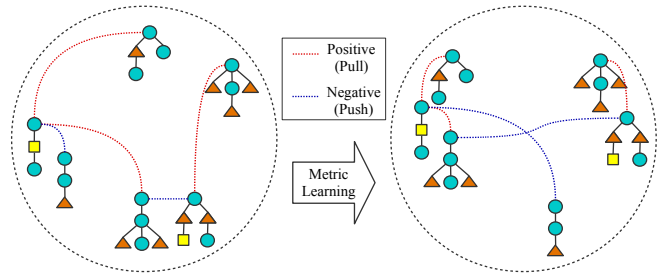
---
[1] Kyoto University, Japan, email: hikarushindo@iip.ist.i.kyoto-u.ac.jp
[2] NTT Communication Science Laboratories, Japan
[3] Kyoto University, Japan
[4] Kyoto University, Japan

Figure 1. An illustration of metric learning applied to tree-structured data. The training examples are given as pairwise constraints. The red dotted lines represent positive pairs. The blue dotted lines represent negative pairs. Metrics are optimized to pull positive pairs closer and push negative pairs father apart.

positive pairs closer while separating negative pairs. This enables us to improve the accuracy of various machine learning algorithms that depend on the metric. Figure 1 illustrates metric learning applied to tree-structured data, which is the focus of this paper.

Discrete structures, in particular tree structures, play a key role in several research domains, for instance, XML documents on the web, parse trees for computer programs and natural language, and glycan structures in bioinformatics. Distance measures that exploit such structures have been extensively studied. The tree edit distance [34] is one of the common choices for processing tree-structured data. Intuitively, the tree edit distance is measured by the number of operations needed to transform one input tree into another input tree. The edit operations consist of deletion, insertion, and replacement of the nodes in the trees. The tree edit distance is used in many research domains such as information extraction and bioinformatics [14, 30]. Computing the tree edit distance, however, is not scalable making it problematic for large-scale datasets. The current best algorithm runs in $\mathcal{O}(n^3)$ time where n is the number of nodes of the input trees [8]. To overcome this issue, Augsten [1] proposed the pq-gram distance; it can be computed faster than the tree edit distance. Computing the pq-gram distance of trees with n nodes can be done in time $\mathcal{O}(n \log n)$ for fixed $p, q \in \mathbb{N}$. Moreover, the pq-gram distance is known to approximate the fanout weighted tree edit distance, which is a weighted variant of the tree edit distance [1].

Most existing studies of metric learning for tree-structured data use the tree edit distance in learning, i.e., learned costs of the edit operations from examples [4, 22, 27]. The edit distance is, however, expensive to compute and hence not suitable for large-scale datasets.

In this paper, we propose a novel metric learning approach for tree-structured data that has the following features. First, we propose a differentiable parameterized distance based on pq-grams, the *weighted pq-gram distance*, to achieve practical metric learning even for large-scale tree-structured data sets. To make the distance func-

tion differentiable and always positive, we use the *softplus* function. It enables us to learn the distance function by gradient descent techniques and retain the triangle inequality during the learning process. Second, we also propose a way to learn the weighted pq-gram distance through Large Margin Nearest Neighbors (LMNN) [36], which is one of the most widely-used metric learning schemes. Our proposed approach not only achieves results competitive with those of state-of-the-art edit distance-based methods [4, 27] in various classification problems, but also solves classification problems much faster than edit distance-based methods. Third, our method is interpretable. Our approach shares some aspects with kernel methods [11], however, different from kernel methods, we do not implicitly cast data points into a high dimension space. Moreover, our weight parameter indicates which tree substructures are important for classifying input trees.

The remainder of this paper is structured as follows. We discuss related work on metric learning for structured data in Section 2. In Section 3, we revisit the basic concepts of tree-structured data, the pq-gram distance, and the scheme of distance metric learning as background. Section 4 describes our metric learning system in detail. Section 5 describes the experiments conducted on our methods, including accuracy and time comparisons. We conclude in Section 6.

2 Related Work

Our work is related to some machine learning research areas, especially metric learning and machine learning for structured data.

A pioneering study of metric learning learned Mahalanobis distance as an optimization problem [37]. Large Margin Nearest Neighbors (LMNN) [36] was proposed in order to learn the Mahalanobis distance from nearest neighbors. LMNN is often used because of its simplicity and efficiency [15, 19, 28, 35]. LMNN is also a well-studied metric-learning scheme. For instance, the relation between LMNN and Support Vector Machine has been pointed out in a unified view [9]. We apply the LMNN scheme to learn the distance between labeled ordered trees.

Almost all past studies on learning distances between trees employ edit distance, i.e., learning the costs of edit operations from examples. Early work in learning edit distance is the stochastic approach [21, 25, 31]. Good Edit Similarity Learning (GESL) [4] is a well-organized framework to learn the edit distance. GESL essentially optimizes (ϵ, γ, τ)-goodness [2, 3], which guarantees its generalization performance. Mokbel [22] proposed a novel approach to learn simultaneously the edit costs for sequences and the Generalized Learning Vector Quantization (GLVQ) [32] model. Paaßen [27] proposed to learn embeddings of the tree node labels while learning the GLVQ model. This approach is called Embedding Edit Distance Learning (BEDL) and succeeds in learning the edit distance flexibly from examples. These works blazed a trail in the field of metric learning for structured data. However, all of these methods incur high computation cost since they essentially compute the edit distance between trees. Our method uses the pq-gram distance rather than the edit distance in learning the parameters. This is a key difference between past studies and our approach.

In the field of machine learning for structured data, the kernel method is an active research topic [11]. For instance, Kuboyama [18] proposed a kernel function that is computed from q-grams of trees. Tree kernels have been applied in many domains, such as Natural Language Processing and Bioinformatics [23, 38]. Kernel methods, however, lack interpretability since they implicitly cast data points into a high dimension space. Moreover, the kernel matrix must be positive semidefinite, but this constraint does not suit some problems [33].

3 BACKGROUND

In this section, we define the basic concepts of tree-structured data and the pq-gram distance following [1]. We also review the general concept of metric learning following [5, 19] and a metric learning algorithm following [5, 36].

3.1 Preliminaries

Tree **T** is a directed, acyclic, connected, non-empty graph with node set $N(\mathbf{T})$ and edge set $E(\mathbf{T})$. An *edge* is an ordered pair (p, c), where $p, c \in N(\mathbf{T})$ are nodes, and p is the *parent* of c. A node can have at most one parent, and nodes with the same parent are *siblings*. Total order $<$ is defined on each group of sibling nodes. Two siblings s_1, s_2 are *contiguous* iff $s_1 < s_2$ and they have no sibling x such that $s_1 < x < s_2$. Node c is the i-th child of p iff $i = |\{x \in N(\mathbf{T})|(p, x) \in E(\mathbf{T}), x \leq c\}|$. The node with no parent is the root node, denoted by $root(\mathbf{T})$, and a node without children is a *leaf*. Each node v has a *label*, $\lambda(v) \in \Sigma$, where Σ is a finite alphabet. In what follows, such trees are called *ordered labeled* trees. We write, in recursive style, tree **T** as $x(c_1, \ldots, c_n)$ where $x = \lambda(root(\mathbf{T}))$ and $c_1, \ldots c_n$ is a list of subtrees whose root is a child of the root node.

3.2 pq-gram

Intuitively, the pq-grams of a tree are all subtrees with a specific shape. Parameters $p, q \in \mathbb{N}$ define pq-gram shape. To ensure that each node of the tree appears in at least one pq-gram, we extend the tree with dummy nodes.

Definition 1 (pq-Extended Tree)**.** Let **T** be a tree, and $p > 0$ and $q > 0$ be two integers. The *pq-extended tree*, $\mathbf{T}^{p,q}$, is constructed from **T** by (i) adding $p - 1$ ancestors to the root node, (ii) inserting $q - 1$ children before the first and after the last child of each non-leaf node, and (iii) adding q children to each leaf of **T**. All newly inserted nodes are dummy nodes that do not occur in **T** and have a special label $* \notin \Sigma$.

An example of a pq-extended tree is given in Figure 2-(b).

Definition 2 (pq-Gram)**.** Let **T** be a tree, $\mathbf{T}^{p,q}$ the corresponding extended tree, $p > 0, q > 0$. A subtree of $\mathbf{T}^{p,q}$ is a *pq-gram* **G** of **T** iff (i) **G** has q leaf nodes and p non-leaf nodes, (ii) all leaf nodes of **G** are children of a single node, and (iii) the leaf nodes of **G** are consecutive siblings in $\mathbf{T}^{p,q}$.

The set of pq-grams are extracted by sliding two windows horizontally and vertically over a pq-extended tree. Window sizes are p and q, respectively [1] (See Figure 2-(c)). The number of nodes of a pq-gram is always $p + q$. An example of extracted pq-grams is given in Figure 2-(d).

3.3 pq-gram distance

We can define a distance measure between trees based on pq-grams, which we call the *pq-gram distance*. Intuitively, the pq-gram distance is the number of pq-grams that are not shared by two trees. The pq-gram distance is computed as follows (1) extract all pq-grams of input trees, and (2) count the number of non-shared pq-grams. To save space, we use the tuple representation of pq-grams.
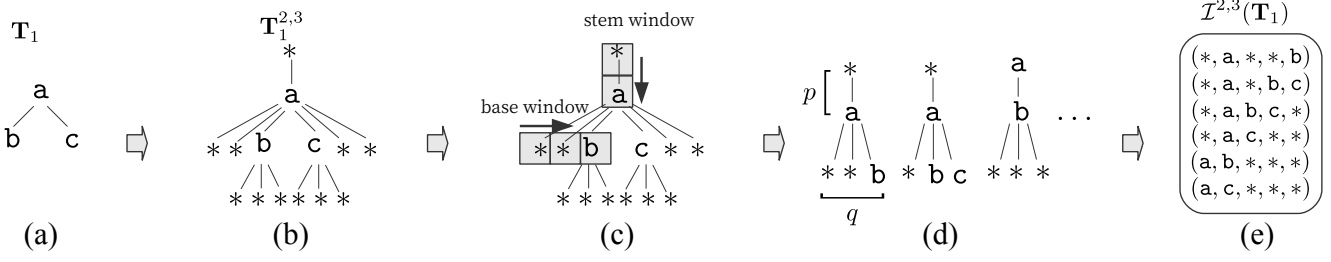
Figure 2. Overview of extracting pq-grams with $p = 2$ and $q = 3$ from tree $\mathbf{T}_1$. (a) A rooted labeled ordered tree $\mathbf{T}_1$. (b) The pq-extended tree of $\mathbf{T}_1$, is obtained by inserting dummy nodes as in Def. 1. The dummy nodes are represented by $*$. (c) Sliding two windows pq-extended tree. The size p window (stem window) moves vertically and the size q window (base window) moves horizontally. (d) The multiset of extracted pq-grams of $\mathbf{T}_1$. (e) The pq-gram index of $\mathbf{T}_1$. Since the shape of pq-grams depends only on the values of p and q, we can represent the multiset of pq-grams as a multiset of tuples of length $p + q$.

Definition 3 (Label Tuple). Let $\mathbf{G}$ be a pq-gram with nodes $\{v_1, \ldots, v_p, v_{p+1}, \ldots v_{p+q}\}$ where v_i is the i-th node in the preorder transversal of $\mathbf{G}$. Tuple $\lambda^*(\mathbf{G}) = (\lambda(v_1), \ldots, \lambda(v_p), \lambda(v_{p+1}), \ldots \lambda(v_{p+q}))$ is called the *label tuple* of $\mathbf{G}$ where $\lambda(v_i)$ is the label of node v_i.

Definition 4 (pq-Gram Index). Let $\mathcal{A}$ be the multiset of all pq-grams of a tree $\mathbf{T}$, $p, q \in \mathbb{N}$. The pq-gram index, $\mathcal{I}^{p,q}(\mathbf{T})$, of $\mathbf{T}$ is defined as the multiset of label tuples of all pq-grams of $\mathbf{T}$, i.e., $\mathcal{I}^{p,q}(\mathbf{T}) = \bigcup_{\mathbf{G} \in \mathcal{A}} \lambda^*(\mathbf{G})$.

Figure 2-(e) shows an example of the label tuple and the pq-gram index. The size of the pq-gram index is linear in the number of tree nodes [1]. Hereafter, if the distinction is clear from the context, we use the term pq-gram for both the pq-gram itself and its representation as a label tuple.

Definition 5 (pq-Gram Distance). Let $\mathbf{T}_1$ and $\mathbf{T}_2$ be trees, $\mathcal{I}_1 = \mathcal{I}^{p,q}(\mathbf{T}_1), \mathcal{I}_2 = \mathcal{I}^{p,q}(\mathbf{T}_2)$, $p, q \in \mathbb{N}$. The pq-gram distance, $dist^{p,q}(\mathbf{T}_1, \mathbf{T}_2)$, between the trees $\mathbf{T}_1, \mathbf{T}_2$ is defined as the size of the symmetric difference between their indexes:

$$dist^{p,q}(\mathbf{T}_1, \mathbf{T}_2) = |\mathcal{I}_1 \cup \mathcal{I}_2| - 2|\mathcal{I}_1 \cap \mathcal{I}_2|, \qquad (1)$$

where $\cup$ is multiset union and $\cap$ is multiset intersection.

Example 1. Consider the $1, 2$-gram distance between input trees $\mathsf{a(b, c)}$ and $\mathsf{a(c, b)}$, namely $dist^{1,2}(\mathsf{a(b, c)}, \mathsf{a(c, b)})$. The pq-gram indexes for the input trees are $\mathcal{I}^{1,2}(\mathsf{a(b, c)}) = \{(\mathsf{a}, *, \mathsf{b}), (\mathsf{a}, \mathsf{b}, \mathsf{c}), (\mathsf{a}, \mathsf{c}, *), (\mathsf{b}, *, *), (\mathsf{c}, *, *)\}$, and $\mathcal{I}^{1,2}(\mathsf{a(c, b)}) = \{(\mathsf{a}, *, \mathsf{c}), (\mathsf{a}, \mathsf{c}, \mathsf{b}), (\mathsf{a}, \mathsf{b}, *), (\mathsf{b}, *, *), (\mathsf{c}, *, *)\}$, respectively. We have $|\mathcal{I}^{1,2}(\mathsf{a(b, c)}) \cup \mathcal{I}^{1,2}(\mathsf{a(c, b)})| = 5 + 5 = 10$, and $|\mathcal{I}^{1,2}(\mathsf{a(b, c)}) \cap \mathcal{I}^{1,2}(\mathsf{a(c, b)})| = 2$, then $dist^{1,2}(\mathsf{a(b, c)}, \mathsf{a(c, b)}) = 10 - 2 \times 2 = 6$.

Augsten [1] showed that the pq-gram distance is *pseudo-metric*, that is, the distance can be zero for distinct trees in contrast to a normal metric. The computation time of the pq-gram distance is $\mathcal{O}(n \log n)$ for fixed $p, q \in \mathbb{N}$, where n is the number of nodes of input trees. Moreover, the pq-gram distance approximates edit distance weighted by the number of each node, which is called fanout weighted tree edit distance [1].

3.4 Metric learning with nearest neighbors

The purpose of metric learning is to adapt the parameterized measure by given positive examples and negative examples. The metric learning problem is typically formulated as an optimization problem that has the following general form [5]:

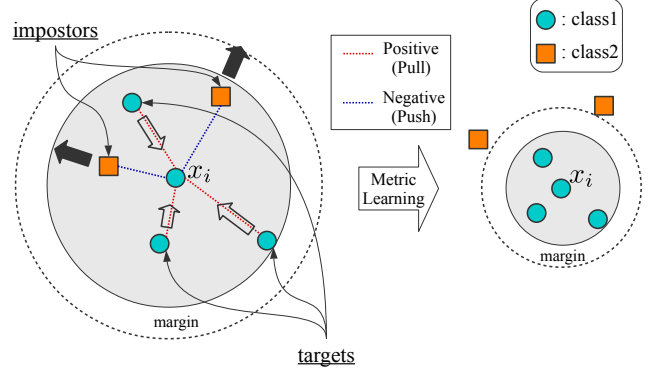$$\min_{\theta} L(d_{\theta}, \mathcal{P}, \mathcal{N}) \qquad (2)$$

Figure 3. An illustration of the LMNN distance learning scheme. For each data point, x_i, the neighbors with the same label, called "targets", are paired as positive. On the other hand, the neighbors with different labels, called "impostors", are paired as negative. After the learning step, targets are brought closer to x_i while impostors are moved further away from x_i.

where L is a loss function that incurs a penalty when training constraints are violated, d_{θ} is a distance function parameterized by θ, $\mathcal{P}$ is a set of positive pairs, and $\mathcal{N}$ is a set of negative pairs. More precisely.

$$\mathcal{P} = \{(x_i, x_j) : x_i \text{ and } x_j \text{ should be similar}\}, \qquad (3)$$

$$\mathcal{N} = \{(x_i, x_j) : x_i \text{ and } x_j \text{ should be dissimilar}\}. \qquad (4)$$

Large Margin Nearest Neighbors (LMNN) [36] is one of the most widely-used distance learning schemes. LMNN locally defines the training pairs: for each data point, same class neighbors, the *target neighbors*, are paired as positive, while different class neighbors, *impostors*, are paired as negative. A schematic illustration of LMNN is given in Figure 3.

4 Method

In this section, we introduce a novel approach of metric learning between trees based on pq-grams. The pq-gram distance is computed by simply counting the number of pq-grams not shared between input trees. Some pq-grams, however, can be important as discriminators for a given classification problem. We introduce a weight function for pq-grams and learn it from examples.

Our approach shares some aspects with edit distance-based approaches [4, 22, 27]. The edit distance is defined by the minimum number of edit operations needed to transform one tree into another. Edit distance-based methods primarily learn the cost of edit operations, i.e., they distinguish which edit operations are essential for a given classification problem. The edit operations are defined between two nodes, so they mostly learn the importance of the relations

3

of nodes. On the other hand, our approach learns the importance of subtrees of the tree structure.

4.0.1 *pq-gram distance with vector representations*

Augsten [1] represents the pq-gram index as a multiset and compute the distance by operations between multisets. However, if the set of tree node labels Σ is finite, the pq-gram index can be represented as a vector of fixed dimension. Vector representation allows us to compute the pq-gram distance efficiently.

Definition 6. (pq-Gram Vector) Let $\mathcal{J}$ be the set of all pq-grams in dataset $\mathcal{D}$. For tree $\mathbf{T} \in \mathcal{D}$ and its pq-gram index $\mathcal{I}^{p,q}(\mathbf{T})$, pq-*gram vector* is a $|\mathcal{J}|$-dim count vector $\mathbf{v}^{p,q}(\mathbf{T})$. Each dimension of $\mathbf{v}^{p,q}(\mathbf{T})$ corresponds to a pq-gram.

In order to compute the pq-gram distance from counting vectors, we introduce a function that computes the number of not shared pq-grams.

Definition 7. (pq-Gram Symmetric Difference Vector) Let $\mathbf{T}_1$ and $\mathbf{T}_2$ be input trees. We define the pq-*gram symmetric difference vector* between $\mathbf{T}_1$ and $\mathbf{T}_2$ as:

$$\mathbf{d}^{p,q}(\mathbf{T}_1, \mathbf{T}_2) = \mathbf{v}^{p,q}(\mathbf{T}_1) + \mathbf{v}^{p,q}(\mathbf{T}_2) \\ - 2\mathbf{min}(\mathbf{v}^{p,q}(\mathbf{T}_1), \mathbf{v}^{p,q}(\mathbf{T}_2)), \quad (5)$$

where $\mathbf{min}$ is the element-wise minimum function, i.e., for n-dimensional vector $\mathbf{x}$ and $\mathbf{y}$,

$$\mathbf{min}(\mathbf{x}, \mathbf{y}) = (\min(x_i, y_i))_{i=1\dots n}. \quad (6)$$

Theorem 1. Let $\mathbf{T}_1$ and $\mathbf{T}_2$ be trees. The pq-gram distance equals the sum of the elements of the pq-gram symmetric difference vector, i.e.,

$$dist^{p,q}(\mathbf{T}_1, \mathbf{T}_2) = \mathbf{1}^{\mathrm{T}} \cdot \mathbf{d}^{p,q}(\mathbf{T}_1, \mathbf{T}_2), \quad (7)$$

where $\mathbf{1}$ is the all-one vector and $\mathbf{d}^{p,q}(\mathbf{T}_1, \mathbf{T}_2)$ is the pq-gram symmetric difference vector.

4.1 Computing the weighted pq-gram distance

In this section, we introduce the *weighted pq-gram distance* to perform metric learning based on pq-grams. The weight reflects the "importance" of each pq-gram, and allows the pq-gram distance to yield highly granular classification. In order to make the distance function differentiable and always positive, we use the softplus function. It enables us to learn the distance function by gradient descent techniques and retain the triangle inequality during the learning process.

4.1.1 *Weighted pq-gram distance*

Definition 8 (Softplus function)**.** The softplus function is defined as:

$$softplus(x) = \ln(1 + e^x). \quad (8)$$

The softplus function always returns positive values. In order to prevent weight parameters from being negative, we apply the softplus function to the parameters. The softplus function is differentiable with respect to the input variables, and enables us to learn distance parameters by gradient descent techniques.

Definition 9 (Weighted pq-Gram Distance)**.** Let $\mathbf{T}_1$ and $\mathbf{T}_2$ be input trees. The *weighted pq-gram distance*, $dist_{\mathbf{w}}^{p,q}(\mathbf{T}_1, \mathbf{T}_2)$ is defined as follows:

$$dist_{\mathbf{w}}^{p,q}(\mathbf{T}_1, \mathbf{T}_2) = \mathbf{a}(\mathbf{w})^{\mathrm{T}} \cdot \mathbf{d}^{p,q}(\mathbf{T}_1, \mathbf{T}_2), \quad (9)$$

where $\mathbf{a}(\mathbf{w}) = (softplus(w_i))_{i=1\dots|\mathbf{w}|} = (\ln(1 + e^{w_i}))_{i=1\dots|\mathbf{w}|}$. $\mathbf{w}$ is a parameter that we learn. $\mathbf{d}^{p,q}(\mathbf{T}_1, \mathbf{T}_2)$ is the pq-gram symmetric difference vector between $\mathbf{T}_1$ and $\mathbf{T}_2$.

Theorem 2. The weighted pq-gram distance is pseudo-metric, i.e., satisfies the following conditions:
(i) non-negativity: $dist_{\mathbf{w}}^{p,q}(x, y) \geq 0$
(ii) reflexivity: $x = y \Rightarrow dist_{\mathbf{w}}^{p,q}(x, y) = 0$
(iii) symmetry: $dist_{\mathbf{w}}^{p,q}(x, y) = dist_{\mathbf{w}}^{p,q}(y, x)$
(iv) triangle inequality: $dist_{\mathbf{w}}^{p,q}(x, y) + dist_{\mathbf{w}}^{p,q}(y, z) \geq dist_{\mathbf{w}}^{p,q}(x, z)$

Proof. (i), (ii), and (iii) are clear by definition.
(iv) Let $\mathcal{A}_x$, $\mathcal{A}_y$, and $\mathcal{A}_z$ be the multisets of extracted pq-grams of trees x, y, and z, respectively. Let g be a pq-gram with $g \in \mathcal{A}_x \cup \mathcal{A}_y \cup \mathcal{A}_z$, and i, j, and k be the numbers of occurrences of g in $\mathcal{A}_x$, $\mathcal{A}_y$, and $\mathcal{A}_z$, respectively. Let $d_g(x, y)$, $d_g(y, z)$, $d_g(x, z)$ be the contributions of g to the distance $dist_{\mathbf{w}}^{p,q}(x, y)$, $dist_{\mathbf{w}}^{p,q}(y, z)$, and $dist_{\mathbf{w}}^{p,q}(x, z)$, respectively. The distance is computed as the sum of the contributions: $dist_{\mathbf{w}}^{p,q}(x, y) = \sum_{g \in \mathcal{A}_x \cup \mathcal{A}_y \cup \mathcal{A}_z} d_g(x, y)$. Here, $d_g(x, y) = a_g(i + j - 2\min(i, j)) = a_g|i - j|$, $d_g(y, z) = a_g|j - k|$, and $d_g(x, z) = a_g|i - k|$, where a_g is a weight parameter for g. Note that a_g is positive since it is output by the softplus function. Therefore, we have $dist_{\mathbf{w}}^{p,q}(x, y) + dist_{\mathbf{w}}^{p,q}(y, z) = \sum_{g \in \mathcal{A}_x \cup \mathcal{A}_y \cup \mathcal{A}_z}(d_g(x, y) + d_g(y, z)) = \sum_{g \in \mathcal{A}_x \cup \mathcal{A}_y \cup \mathcal{A}_z} a_g(|i - j| + |j - k|) \geq \sum_{g \in \mathcal{A}_x \cup \mathcal{A}_y \cup \mathcal{A}_z} a_g|i - k| = dist^{p,q}(x, z)$. $\square$

4.2 Learning the weighted pq-gram distance

The steps to perform metric learning are (1) create training pairs from a given dataset, (2) set an appropriate loss function for the training pairs, and (3) optimize the loss function with respect to the parameter of the target distance function.

We generate training pairs following the LMNN scheme. For every data point x_i with class label y_i in given dataset $\mathcal{D}$, we create positive and negative pairs as follows:

$$\mathcal{P}_i = \left\{(x_i, x_j) : x_j \in N_k^+(x_i)\right\}, \quad (10)$$

$$\mathcal{N}_i = \left\{(x_i, x_j) : x_j \in N_k^-(x_i)\right\}, \quad (11)$$

where $N_k^+(x_i)$ is the "target" function that returns the set of k-nearest neighbors with the same label y_i, and $N_k^-(x_i)$ is an "impostor" function that returns the set of data points with different labels and are closer than the farthest (k-th) target. We now define the set of positive pairs as $\mathcal{P} = \bigcup_{i=1\dots m} \mathcal{P}_i$, and the set of negative pairs as $\mathcal{N} = \bigcup_{i=1\dots m} \mathcal{N}_i$, where m is the number of training data points. We note that the training pairs are created with the default distance metric before the learning step.

Now we introduce the loss function based on the hinge loss:

$$L(dist_{\mathbf{w}}^{p,q}, \mathcal{P}, \mathcal{N}) = \beta||\mathbf{w}||^2 + \sum_{(\mathbf{T}_1, \mathbf{T}_2) \in \mathcal{P}} [dist_{\mathbf{w}}^{p,q}(\mathbf{T}_1, \mathbf{T}_2) - \mu_1]_+$$

$$+ \sum_{(\mathbf{T}_1, \mathbf{T}_2) \in \mathcal{N}} [\mu_2 - dist_{\mathbf{w}}^{p,q}(\mathbf{T}_1, \mathbf{T}_2)]_+, \quad (12)$$

where β is a regularization coefficient, $[x]_+ = \max(0, x)$, and μ_1 and μ_2 are constants that represent margins. The first term is the

L2 regularization term, the second term is a loss that makes positive pairs closer, and the third term is a loss that makes negative pairs farther apart. The hinge loss-based formulation is widely-used in margin-based methods such as soft-margin SVM [6], LMNN [36], and GESL [4]. If a positive (resp. negative) pair satisfies a certain criterion, i.e., it is close (resp. far) enough, then it does not contribute to the loss function.

We minimize the loss function by gradient descent. In order to perform gradient descent, we need the gradient $\nabla_{\mathbf{w}} dist_{\mathbf{w}}^{p,q}(\mathbf{T}_1, \mathbf{T}_2)$ for input trees $\mathbf{T}_1$ and $\mathbf{T}_2$. The gradient of weighted pq-gram distance function with respect to $\mathbf{w}$ is computed as follows:

$$\nabla_{\mathbf{w}} dist_{\mathbf{w}}^{p,q}(\mathbf{T}_1, \mathbf{T}_2) = \left(\frac{e^{w_i}}{1 + e^{w_i}} d_i^{p,q}(\mathbf{T}_1, \mathbf{T}_2) \right)_{i=1\ldots|\mathbf{w}|}, \quad (13)$$

where w_i is the i-th element of $\mathbf{w}$ and $d_i^{p,q}(\mathbf{T}_1, \mathbf{T}_2)$ is the i-th element of the pq-gram symmetric difference vector $\mathbf{d}^{p,q}(\mathbf{T}_1, \mathbf{T}_2)$.

The learning procedure is summarized as follows: First, the pq-gram vector representations are computed for all input trees. We note that the computed pq-gram vector representations are used internally by the distance function $dist_{\mathbf{w}}^{p,q}$ in the following steps. Second, the set of positive pairs and the set of negative pairs are created by Eq. (10) and Eq. (11), respectively, from a given training dataset. Finally we minimize the loss function in Eq. (12) with respect to $\mathbf{w}$ by gradient descent. In practice, we update impostors during the learning process to improve model performance.

5 EXPERIMENTS

In this section, we discuss our experiments on artificial and real-world datasets. The experiments are designed to show that the proposed approach not only achieves competitive results with state-of-the-art edit distance-based methods in various classification problems, but also solves the classification problems much faster than edit distance-based methods. All experiments were performed on a desktop computer with Intel(R) Xeon(R) E5-2680 v2 @ 2.80GHz CPU, 126GB RAM, and CentOS Linux 7.4.

We used our Python 3.6 and PyTorch [29] implementation for LMNN learning with the weighted pq-gram distance. The tree edit distance algorithm is implemented following [26]. We adopted Adam optimizer [16] to optimize parameters in our training phase. We also used a Java implementation[5] for BEDL and GESL.

5.1 Dataset

We evaluated our approach on one artificial dataset and several real-world datasets.

Strings This dataset consists of two classes of strings with a length of exactly nine. The first class is drawn randomly from the set of strings expressed by $((\texttt{A}|\texttt{B})(\texttt{C}|\texttt{D})(\texttt{A}|\texttt{B}))^3$ in the regular expression. For the second class, we do the same from the set of strings expressed by $(\texttt{A}|\texttt{B}|\texttt{C}|\texttt{D})^9$ in the regular expression. We can observe that substring $\texttt{DAD}$ never appears in the first class, and every string in the first class is generated in a "periodic way" unlike those in the second class. These facts are important for classifying these strings. We can regard a string as a tree (without branching nodes) in a natural way. We used 100 strings for each class.

Glycan We used two datasets from KEGG database [12] as used in [39]. Glycans are defined as the third major class of biomolecules following DNA and proteins. Each monosaccharide in a glycan structure is connected to one or more monosaccharides, and we can regard a glycan structure as a labeled tree. CarbBank/CCSD [10] gives the class labels for glycans. The trees have node labels and edge labels. We put edge labels into corresponding child nodes in the same way as [39]. For instance, subtree $\overset{\texttt{1b4}}{}\texttt{Gal}\overset{\texttt{1a3}}{}\texttt{Fuc}$ is represented as $\texttt{Gal.1b4} - \texttt{Fuc.1a3}$. Every leaf node is represented by the special label \$. Each glycan structure is assigned to a blood component class among Erythrocyte, Leukemic, Serum, and Plasma. We created two binary classification problems (i) Erythrocyte/Leukemic (Glycan_EL) and (ii) Serum/Plasma (Glycan_SP). These problems have 138 trees and 267 trees, respectively. For evaluating multi-label classification, we also used the dataset that contains all instances (Glycan_MULTI) in accuracy comparison. Glycan_MULTI contains 405 trees and 4 class labels.

Words The words dataset used in [4] contains English words and French words extracted from Wiktionary[6]. It consists of basic English/French words in order of frequency of use. We can regard a word as a tree (without branching nodes) in a natural way. Every leaf node is represented by the special label \$. We considered only words of length at least 4 to remove articles and prepositions. We used the top 500 words for each class.

Sentiment Treebank Sentiment Treebank dataset contains movie reviews with their parse trees. The internal nodes have one of the 5-class labels from highly negative to highly positive. We set the class label as the sentence is positive or negative as a whole, i.e., the root node's label. Every root node is replaced by a unique node whose label is $\texttt{root}$. Every leaf node represents a specific word in the review sentence. We replaced each word with its POS tag for scaling. We randomly chose 100 trees for each class.

The datasets we used are summarized in Table 1.

Dataset	trees	node labels	mean tree size	class
Strings	200	4	9.0	2
Glycan_SP	138	48	10.9	2
Glycan_EL	267	37	13.4	2
Words	1000	28	6.8	2
Sentiment	200	39	39.2	2
Glycan_MULTI	405	51	12.5	4

Table 1. The description of datasets used in our experiments. The first column shows the number of trees in each dataset. The second column shows the number of node labels of trees in each dataset. The third column shows the mean size of trees (number of nodes in a tree) for each dataset. The fourth column shows the number of classes.

5.2 Accuracy comparison

We evaluated several classification problems using different models with different distance measures. The problem setting has 3 parts: (i) the distance measure used by the classification model, (ii) the metric learning algorithm, and (iii) the distance-based classification model. We compared the following 5 settings: (E1) the pq-gram distance and the k-nearest neighbor classifier, (E2) the weighted pq-gram distance with LMNN and the k-nearest neighbor classifier (proposed), (E3) the edit distance with LMNN and the k-nearest neighbor classifier, (E4) the edit distance with GESL and the MRGLVQ classifier [4, 24], and (E5) the edit distance with BEDL and the MRGLVQ classifier [24, 27].

---

[5] https://pub.uni-bielefeld.de/data/2919994

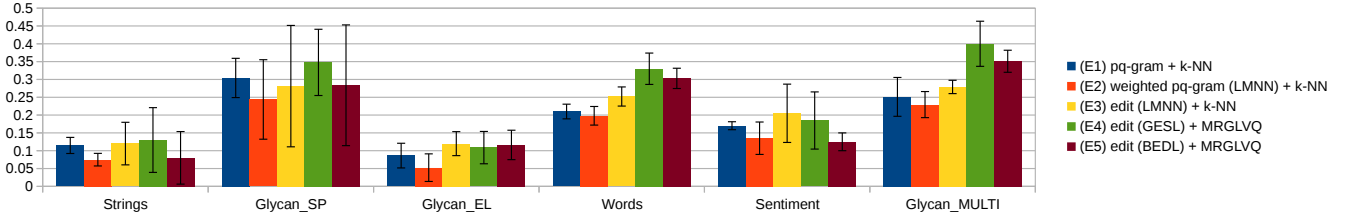[6] http://en.wiktionary.org/wiki/Wiktionary:Frequency_lists

Figure 4. The figure indicates error rates for each real-world datasets. The bottom labels specify the datasets. The bar represents the error rate for each method. The vertical black line represents the standard deviation of the results of the folds. For each dataset, the bar corresponds to each combination of distance and classifier: (E1) the pq-gram distance and the k-nearest neighbor classifier, (E2) the weighted pq-gram distance with LMNN and the k-nearest neighbor classifier (proposed), (E3) the edit distance with LMNN and the k-nearest neighbor classifier, (E4) the edit distance with GESL and the MRGLVQ classifier, (E5) the edit distance with BEDL and the MRGLVQ classifier.
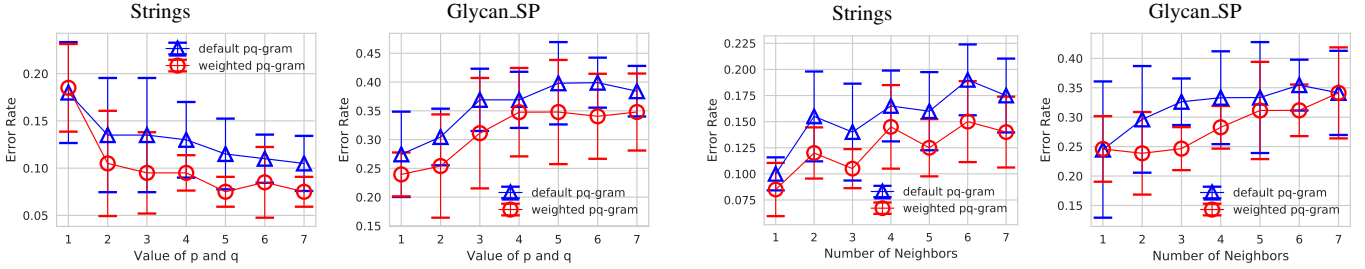

Figure 5. The figure indicates error rates with standard deviations for $1 \leq p = q \leq 7$. The horizontal axis represents the value of p and q. The vertical axis represents the error rate of k-nearest neighbor classifier. The blue line represents the error rate with the pq-gram distance, and the red line represents the error rate with the weighted pq-gram distance.
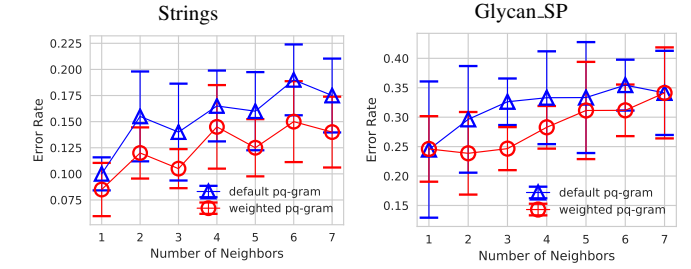

Figure 6. The figure indicates error rates with standard deviations for $1 \leq k \leq 7$. The horizontal axis represents the number of neighbors for the k-nearest neighbor classifier and the LMNN learning algorithm. The vertical axis represents the error rate of the k-nearest neighbor classifier. The blue line represents the error rate with the pq-gram distance, and the red line represents the error rate with the weighted pq-gram distance.

On each data set, we performed 5-fold cross-validation and compared the mean test error across the folds. In the setting (E1) and (E2), we set $p = 2$, $q = 2$ as pq-gram size. We set $k = 1$ for Strings and Glycan_MULTI dataset, and $k = 3$ for others, where k is the number of neighbors for the k-nearest neighbor classifier and the number of "targets" of LMNN learning. As these parameters affect the classification results, we analyze their impact in the next subsection. Other parameters were set as follows: $\mu_1 = \mu_2 = 5.0$ for margin parameters, $\eta = 10^{-2}$ for the initial learning rate of the Adam optimizer [16], $\beta = 10^{-4}$ for the L2 regularization. We trained the model for 600 epochs. We updated impostors for LMNN every 50 epochs. In the LMNN learning step, we randomly chose 200 training data points if the number of input training data points is more than 200. In the setting (E3), we fixed the optimal edit operations that are computed for the first time of the learning algorithm, in the same way, as [5] and performed metric learning using the LMNN scheme with gradient descent with respect to the edit costs. The parameters for the settings (E4) and (E5) are selected by nested cross-validation following [27].

Figure 4 shows the results of our experiments. On each dataset, the k-nearest neighbor classifier with the weighted pq-gram distance (E2) achieves a lower error rate than the pq-gram distance (E1) . Moreover, for all datasets, the k-nearest neighbor classification with the weighted pq-gram distance (E2) achieves better results than that with the tree edit distance (E3). Also, our approach achieves competitive results with the state-of-the-art edit distance-based methods such as GESL and BEDL with the MRGLVQ classifier (E4) (E5).

5.2.1 Effects of parameters

Since the values of p and q determine the shape of pq-grams, they can affect the error rates of classification. We analyze the effect on our proposed method of changing the values of p and q (E2). In particu-

lar, we investigate the error rate of the 3-nearest neighbor classifiers with the conventional pq-gram distance and with the weighted pq-gram distance. The values p and q are chosen as $1 \leq p = q \leq 7$. Figure 5 shows the transition in error rates with respect to p and q. For the Strings dataset, the case $p = q = 1$ is worst among all, which follows our intuition that every string in the first class is composed of substrings of length three, and these substrings can be seen only in the case where $p = q \geq 2$. For the Glycan_SP dataset, the error rates gradually increase as both p and q become large. In both datasets, for all p and q values except for the case $p = q = 1$ in the Strings dataset, the k-nearest neighbor classifier with the weighted pq-gram distance outperformed that with the conventional pq-gram distance.

We also analyze the effect of the number of neighbors k on the k-nearest neighbor classifier and the LMNN metric learning scheme. In the training step, we created training pairs with k targets for each tree. In the classification step, we performed the k-nearest neighbor classification. Figure 6 shows the transition in error rate with respect to k. Interestingly, in the Strings dataset, the case of $k = 1$ achieves the highest accuracy among all. With regard to the number of neighbors k for the k-nearest neighbor classifier, Hastie [13] pointed out that the best k value is situation dependent. We highlight the fact that our proposed distance outperformed the conventional pq-gram distance regardless of the k values in both datasets except for the case $k = 1, 7$ in the Glycan_SP dataset.

5.2.2 Interpretability

In this subsection, we discuss the interpretability of our method. We can consider that pq-grams with substantial weights are important discriminators for classification problems since they essentially determine the classification results.

We exhibit some pq-grams receiving substantial weights and their occurrences in each class in Strings, Glycan_EL, and Words dataset

as in Table 2. We can observe that high-weight pq-grams tend to appear many times in one of the classes, but not so much in the other classes. This fact implies that pq-grams with substantial weights are important features for classifying trees. In the Glycan_EL dataset, for example, $(\texttt{GlcNAc.1b4}, \texttt{Gal.1b3}, \$, *)$ appears only with class 1. It means that subtree $\xrightarrow{\text{1b4}}\texttt{GlcNAc}\xrightarrow{\text{1b3}}\texttt{Gal}$ which contains the leaf node, is a key feature for class 1. In the Words dataset, pq-gram $(\texttt{e}, \texttt{u}, \texttt{r}, *)$ appears on French words 14 times, but never on English words, which means French words in the dataset often have substring "eur", but English words do not.

Dataset	pq-gram	class1	class2
Strings	$(\texttt{B}, \texttt{B}, \texttt{A}, *)$	0	9
	$(\texttt{D}, \texttt{A}, *, *)$	21	8
	$(\texttt{A}, \texttt{D}, \texttt{B}, *)$	45	12
	$(\texttt{C}, \texttt{A}, *, *)$	26	2
	$(\texttt{D}, \texttt{A}, \texttt{D}, *)$	0	11
Glycan_EL	$(*, \texttt{Xyl.}, \texttt{Gal.1b4}, *)$	3	0
	$(\texttt{GlcNAc.1b4}, \texttt{Gal.1b3}, \$, *)$	2	0
	$(\texttt{Gal.1b4}, \texttt{GlcNAc.1b3}, \$, *)$	4	0
	$(\texttt{Gal.}, \texttt{GlcNAc.1b3}, \$, *)$	0	1
	$(\texttt{GalNAc.1b3}, \texttt{GalNAc.1a3}, *, \$)$	2	0
Words	$(\texttt{c}, \texttt{h}, *, \$)$	6	0
	$(\texttt{o}, \texttt{n}, *, \texttt{e})$	9	0
	$(\texttt{s}, \texttt{e}, \texttt{u}, *)$	0	4
	$(\texttt{e}, \texttt{u}, \texttt{r}, *)$	0	14
	$(\texttt{o}, \texttt{i}, \texttt{s}, *)$	0	6

Table 2. Extracted high-weight pq-grams and appearance number in each class. Right 2 columns show the number of occurrences of pq-grams in each class.

5.3 Running time comparison

In order to show the practical performance of our method, we compare the running times of classification algorithms based on the weighted pq-gram distance and the tree edit distance. In this experiment, we run the standard k-nearest neighbor algorithm with two distinct distance functions, the weighted pq-gram distance and the tree edit distance. We measured the time for executing whole process: (i) encoding into count vectors, (ii) computing the distances between the test data and the training data to identify neighbors, (iii) and making an inference by the majority vote. The first encoding step is executed only for the weighted pq-gram distance. We first note that the theoretical running time $\mathcal{O}(n \log n)$ of computing the pq-gram distance between trees is much faster than the fastest known tree edit distance algorithm, that of [8] which runs in $\mathcal{O}(n^3)$ time. Thus, the running time of the k-nearest neighbor inference with the weighted pq-gram distance is $\mathcal{O}(m_1 m_2(n \log n + k))$, and that with the tree edit distance is $\mathcal{O}(m_1 m_2(n^3 + k))$, where m_1 is the number of training data points and m_2 is the number of test data points.

Table 3 shows the mean running time of 3-nearest neighbor inference for 5-fold cross-validation. In all datasets, the weighted pq-gram distance yields much shorter inference times than the tree edit distance as a distance function. Especially in the Sentiment dataset, whose mean tree size is the largest among all datasets, the k-nearest neighbor classifier using our proposed method yields over 5000 times shorter inference time than that using the tree edit distance.

We note that edit distance-based methods such as GESL do not directly learn the edit cost of tree edit distance. They first compute an optimal sequence of edit operations for training data which is then fixed (held unchanged). Then, they learn an appropriate edit cost along with this sequence of edit operations, which makes their learning process much easier and faster than directly learning the edit cost. However, when making an inference for test data using the learned

distance, the costly computation for the edit distance is still a major drawback.

Dataset	proposed	edit distance
Strings	**1.29 ± 0.098** sec	23.7 ± 0.09 sec
Glycan_SP	**0.665 ± 0.059** sec	28.4 ± 2.21 sec
Glycan_EL	**0.249 ± 0.121** sec	182.9 ± 8.86 sec
Words	**41.3 ± 0.31** sec	283.9 ± 5.18 sec
Sentiment	**1.79 ± 0.105** sec	9013 ± 770 sec

Table 3. Mean inference time of the 3-nearest neighbor classifier in each dataset. The first column specifies the dataset. The second column shows the inference time of the 3-nearest neighbor classifier using the weighted pq-gram distance. The third column shows that the inference time of the 3-nearest neighbor classifier using the tree edit distance.

6 Conclusion

This contribution has proposed a novel metric learning approach for tree-structured data that has the following features. The differentiable parameterized distance based on pq-grams (proposed herein), called the *weighted pq-gram distance*, achieves fast metric learning for tree-structured data. The time complexity of the weighted pq-gram distance is $\mathcal{O}(n \log n)$, while that of the tree edit distance is more than $\mathcal{O}(n^3)$, where n is the number of nodes of the input trees. Moreover, computation of the proposed distance involves only basic vector operations with the softplus function. It enables the distance function to be learned by the gradient descent techniques while retaining the triangle inequality during the learning process. Second, a way of learning the weighted pq-gram distance through LMNN, which is one of the most widely-used metric learning schemes, was also proposed. Moreover, the metric learning problem was formulated as an optimization problem based on the hinge loss-based formulation. Third, the results of our proposal are interpretable. Our weight parameter indicates which substructures in trees are important for classifying input trees.

We have empirically shown that for various classification problems our proposed method reduces error rates compared to the conventional pq-gram distance using the k-nearest neighbor classification. Moreover, our approach achieved competitive results with the state-of-the-art edit distance-based methods such as GESL and BEDL. We have also shown that our approach solves classification problems much faster than the edit distance-based methods. In our experiments, the k-nearest neighbor classifier using our proposed method solved the various classification problems at most 5000 times faster than that using the tree edit distance.

References

[1] Nikolaus Augsten, Michael Böhlen, and Johann Gamper, 'The pq-gram distance between ordered labeled trees', *ACM Trans. Database Syst.*, **35**(1), 4:1–4:36, (2008).

[2] Maria-Florina Balcan, Avrim Blum, and Nathan Srebro, 'Improved guarantees for learning via similarity functions', in *21st Annual Conference on Learning Theory (COLT)*, pp. 287–298, (2008).

[3] Maria-Florina Balcan, Avrim Blum, and Nathan Srebro, 'A theory of learning with similarity functions', *Machine Learning*, **72**(1), 89–112, (2008).

[4] Aurélien Bellet, Amaury Habrard, and Marc Sebban, 'Good edit similarity learning by loss minimization', *Machine Learning*, **89**(1), 5–35, (2012).

[5] Aurélien Bellet, Amaury Habrard, and Marc Sebban, 'A survey on metric learning for feature vectors and structured data', *CoRR*, **abs/1306.6709**, (2013).

[6] Di-Rong Chen, Qiang Wu, Yiming Ying, and Ding-Xuan Zhou, 'Support vector machine soft margin classifiers: Error analysis.', *Journal of Machine Learning Research*, **5**, 1143–1175, (2004).

[7] Thomas M. Cover and Peter E. Hart, 'Nearest neighbor pattern classification', *IEEE Trans. Information Theory*, **13**, 21–27, (1967).

[8] Erik D. Demaine, Shay Mozes, Benjamin Rossman, and Oren Weimann, 'An optimal decomposition algorithm for tree edit distance', *ACM Trans. Algorithms*, **6**(1), 2:1–2:19, (2009).

[9] Huyen Do, Alexandros Kalousis, Jun Wang, and Adam Woznica, 'A metric learning perspective of svm: on the relation of lmnn and svm', in *15th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pp. 308–317, (2012).

[10] Scott Doubet and Peter Albersheim, 'Letter to the Glyco-Forum: CarbBank', *Glycobiology*, **2**(6), 505–505, (1992).

[11] Thomas Gärtner, 'A survey of kernels for structured data', *SIGKDD Explorations*, **5**(1), 49–58, (2003).

[12] Kosuke Hashimoto, Susumu Goto, Shin Kawano, Kiyoko F. Aoki-Kinoshita, Nobuhisa Ueda, Masami Hamajima, Toshisuke Kawasaki, and Minoru Kanehisa, 'KEGG as a glycome informatics resource', *Glycobiology*, **16**(5), 63R–70R, (2006).

[13] Trevor Hastie, Robert Tibshirani, and Jerome Friedman, *The Elements of Statistical Learning*, Springer New York Inc., (2001).

[14] Tao Jiang, Guohui Lin, Bin ma, and Kaizhong Zhang, 'A general edit distance between rna structures', *Journal of computational biology : a journal of computational molecular cell biology*, **9**, 371–88, (2002).

[15] Dor Kedem, Stephen Tyree, Fei Sha, Gert R. Lanckriet, and Kilian Q Weinberger, 'Non-linear metric learning', in *Advances in Neural Information Processing Systems 25 (NeurIPS)*, 2573–2581, (2012).

[16] Diederik Kingma and Jimmy Ba, 'Adam: A method for stochastic optimization', *International Conference on Learning Representations (ICLR)*, (2014).

[17] Teuvo Kohonen, *Learning Vector Quantization*, 175–189, Springer Berlin Heidelberg, (1995).

[18] Tetsuji Kuboyama, Kouichi Hirata, Hisashi Kashima, Kiyoko F. Aoki-Kinoshita, and Hiroshi Yasuda, 'A spectrum tree kernel', *Information and Media Technologies*, **2**(1), 292–299, (2007).

[19] Brian Kulis, 'Metric learning: A survey', *Foundations and Trends® in Machine Learning*, **5**(4), 287–364, (2013).

[20] Stuart P. Lloyd, 'Least squares quantization in pcm', *IEEE Transactions on Information Theory*, **28**, 129–137, (1982).

[21] Andrew McCallum, Kedar Bellare, and Fernando Pereira, 'A conditional random field for discriminatively-trained finite-state string edit distance', in *Conference on Uncertainty in Artificial Intelligence (UAI)*, (2005).

[22] Bassam Mokbel, Benjamin Paassen, Frank-Michael Schleif, and Barbara Hammer, 'Metric learning for sequences in relational lvq', *Neurocomputing*, **169**, 306–322, (2015).

[23] Alessandro Moschitti, 'Making tree kernels practical for nat-

[24] ural language learning', in *11th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, (2006).

[24] David Nebel, Barbara Hammer, Kathleen Frohberg, and Thomas Villmann, 'Median variants of learning vector quantization for learning of dissimilarity data', *Neurocomputing*, **169**, 295–305, (2015).

[25] Jose Oncina and Marc Sebban, 'Learning stochastic edit distance: application in handwritten character recognition', *Pattern Recognition*, **39**, 1575–1587, (2006).

[26] Benjamin Paaßen, 'Revisiting the tree edit distance and its backtracing: A tutorial', *CoRR*, **abs/1805.06869**, (2018).

[27] Benjamin Paaßen, Claudio Gallicchio, Alessio Micheli, and Barbara Hammer, 'Tree edit distance learning via adaptive symbol embeddings', in *35th International Conference on Machine Learning (ICML)*, pp. 3976–3985, (2018).

[28] Shibin Parameswaran and Kilian Q Weinberger, 'Large margin multi-task metric learning', in *Advances in Neural Information Processing Systems 23 (NeurIPS)*, 1867–1875, (2010).

[29] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer, 'Automatic differentiation in PyTorch', in *NIPS Autodiff Workshop*, (2017).

[30] Davi Reis, Paulo Golgher, Altigran Silva, and Alberto Laender, 'Automatic web news extraction using tree edit distance', in *13th International World Wide Web Conference (WWW)*, pp. 502–511, (2004).

[31] Eric Ristad and Peter Yianilos, 'Learning string edit distance', *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, **20**, 522–532, (1998).

[32] Atsushi Sato and Keiji Yamada, 'Generalized learning vector quantization', in *Advances in Neural Information Processing Systems 8 (NeurIPS)*, 423–429, (1996).

[33] Frank-Michael Schleif and Peter Tino, 'Indefinite proximity learning: A review', *Neural Computation*, **27**(10), 2039–2096, (2015).

[34] Kuo-Chung Tai, 'The tree-to-tree correction problem', *J. ACM*, **26**(3), 422–433, (1979).

[35] Kilian Q. Weinberger and Lawrence K. Saul, 'Fast solvers and efficient implementations for distance metric learning', in *25th International Conference on Machine Learning (ICML)*, pp. 1160–1167, (2008).

[36] Kilian Q. Weinberger and Lawrence K. Saul, 'Distance metric learning for large margin nearest neighbor classification', *Machine Learning*, 207–244, (2009).

[37] Eric P. Xing, Michael I. Jordan, Stuart J Russell, and Andrew Y. Ng, 'Distance metric learning with application to clustering with side-information', in *Advances in Neural Information Processing Systems 15 (NeurIPS)*, 521–528, (2003).

[38] Yoshihiro Yamanishi, Francis Bach, and Jean-Philippe Vert, 'Glycan classification with tree kernels', *Bioinformatics*, **23**(10), 1211–1216, (2007).

[39] Tomoya Yamazaki, Akihiro Yamamoto, and Tetsuji Kuboyama, 'Tree PCA for extracting dominant substructures from labeled rooted trees', in *18th International Conference on Discovery Science (DS)*, pp. 316–323, (2015).

8