Tree Edit Distance Learning via Adaptive Symbol Embeddings

Benjamin Paaßen [1] **Claudio Gallicchio** [2] **Alessio Micheli** [2] **Barbara Hammer** [1]

Abstract

Metric learning has the aim to improve classification accuracy by learning a distance measure which brings data points from the same class closer together and pushes data points from different classes further apart. Recent research has demonstrated that metric learning approaches can also be applied to trees, such as molecular structures, abstract syntax trees of computer programs, or syntax trees of natural language, by learning the cost function of an edit distance, i.e. the costs of replacing, deleting, or inserting nodes in a tree. However, learning such costs directly may yield an edit distance which violates metric axioms, is challenging to interpret, and may not generalize well. In this contribution, we propose a novel metric learning approach for trees which learns an edit distance indirectly by embedding the tree nodes as vectors, such that the Euclidean distance between those vectors supports class discrimination. We learn such embeddings by reducing the distance to prototypical trees from the same class and increasing the distance to prototypical trees from different classes. In our experiments, we show that our proposed metric learning approach improves upon the state-of-the-art in metric learning for trees on six benchmark data sets, ranging from computer science over biomedical data to a natural-language processing data set containing over 300,000 nodes.

1. Introduction

Many classification approaches in machine learning explicitly or implicitly rely on some measure of *distance* (Kulis, 2013; Bellet et al., 2014; Mokbel et al., 2015). This is particularly apparent in case of the k-nearest neighbor classifier which classifies data points by assigning the label of the majority of the k *nearest neighbors* according to a given distance (Cover & Hart, 1967); or in case of *learning vector quantization* approaches which classify data points by assigning the label of the closest prototype according to a given distance (Kohonen, 1995). The success of such machine learning approaches hinges on the distance being *discriminative*, that is, data points from the same class being generally closer compared to data points from different classes. If the distance does not fulfill this criterion, one has to *adapt* or *learn* the distance measure with respect to the data, which is the topic of *metric learning* (Kulis, 2013; Bellet et al., 2014).

Most prior research in metric learning has focused on learning a generalization of the Euclidean distance according to some cost function (Kulis, 2013; Bellet et al., 2014). However, the Euclidean distance is not applicable to non-vectorial data, such as protein sequences, abstract syntax trees of computer programs, or syntax trees of natural language. To process these kinds of data, *edit distances* are a popular option, in particular the tree edit distance (Zhang & Shasha, 1989). In this contribution, we develop a novel metric learning scheme for the tree edit distance.

While past research on metric learning for trees does exist (Bellet et al., 2014), we go beyond this state-of-the-art in multiple aspects:

- Based on the work of Bellet et al. (2012), we provide a generalized re-formulation of the edit distance which lends itself to metric learning, and can be applied to any kind of edit distance which uses replacement, deletion, and insertion operations. Furthermore, we consider not only *one* optimal edit script for metric learning, but *all* co-optimal edit scripts via a novel forward-backward algorithm.
- Our approach requires only a linear number of data tuples for metric learning, as we represent classes by few *prototypes*, which are selected via median learning vector quantization (Nebel et al., 2015).
- Most importantly, we do not directly learn the operation costs for the string edit distance, but instead learn a vectorial embedding of the label alphabet for our data structures, which yields Euclidean operation costs.

[1]Cognitive Interaction Technology, Bielefeld University, Germany [2]Department of Computer Science, University of Pisa, Italy. Correspondence to: Benjamin Paaßen <bpaassen@techfak.uni-bielefeld.de>.

Proceedings of the 35^{th} International Conference on Machine Learning, Stockholm, Sweden, PMLR 80, 2018.

This re-formulation ensures that the resulting edit distance conforms to all metric axioms. Further, we can interpret the resulting embedding vectors via visualization, their pairwise distances and norms.

We begin by discussing related work, then we describe our proposed approach in more detail, and finally we evaluate our approach experimentally and discuss the results.

2. Related Work

Our work is related to multiple areas of machine learning, most notably distances on structured data, metric learning, and vector embeddings.

In the past decades, multiple distance measures for structured data - i.e. sequences, trees, and graphs - have been suggested. In particular, one could define a distance based on existing string, tree, and graph kernels (Da San Martino & Sperduti, 2010), such as Weisfeiler-Lehman Graph Kernels (Shervashidze et al., 2011), topological distance-based tree kernels (Aiolli et al., 2015), or deep graph kernels (Yanardag & Vishwanathan, 2015). Such kernels achieve state-of-the-art results on structured data and can be adapted to training data via multiple-kernel learning (Aiolli & Donini, 2015), or kernels based on Hidden-Markov-Model states (Bacciu et al., 2018). Kernels, however, have drawbacks in terms of interpretability, as a higher distance value does not necessarily relate to any kind of intuitive difference between the input trees. Further, kernel matrices are by definition limited to be positive semi-definite, which may be an undue restriction for certain data sets (Schleif & Tino, 2015).

If one strives for an interpretable measure of distance, *edit distances* are a popular choice, for example for the comparison of protein sequences in bioinformatics (Smith & Waterman, 1981), or abstract syntax trees for intelligent tutoring systems (Paaßen et al., 2018). Here, we focus on the tree edit distance, which permits deletions, insertions, and replacements of single nodes to transform an ordered tree $\bar{x}$ into another ordered tree $\bar{y}$ (Zhang & Shasha, 1989). Such ordered trees are the most general data structures which can still be treated efficiently via edit distances, as edit distances on unordered trees and general graphs are provably NP-hard (Zhang et al., 1992; Zeng et al., 2009). Furthermore, the tree edit distance includes the edit distance on sequences as a special case, such that it can be seen as a representative for edit distances as such.

Metric learning for the tree edit distance corresponds to adapting the costs of edit operations in order to bring trees from the same class closer and push trees from different classes further apart. Almost all of the existing approaches, however, only bring trees from the same class closer together (Bellet et al., 2014). For example, Boyer et al. (2007) have proposed to replace the tree edit distance by the negative log probability of all tree edit scripts which transform the left input tree $\bar{x}$ into the right input tree $\bar{y}$. Accordingly, the costs of edit operations change to probabilities of replacing, deleting, or inserting a certain label. These edit probabilities are adapted to maximize the probability that trees from the same class are edited into each other (Boyer et al., 2007). To replace generative models by discriminative ones, Bellet et al. (2012; 2016), have proposed to learn an edit distance d, such that the corresponding similarity $2 \cdot \exp[-d(\bar{x}, \bar{y})] - 1$ is "good" as defined by the goodness-framework of (Balcan et al., 2008). Goodness according to this framework means that a linear separator with low error between the classes exists in the space of similarities (Balcan et al., 2008; Bellet et al., 2012). Bellet et al. (2012) have experimentally shown that this approach outperforms generative edit distance metric learning and have also established generalization guarantees based on the goodness framework. Therefore, this *good edit similarity learning* (GESL) approach of Bellet et al. (2012) is our main reference method.

Our novel approach is strongly inspired by GESL. However, our approach goes beyond GESL in key aspects. First, we utilize a different cost function, namely the generalized learning vector quantization (GLVQ) cost function, which quantifies how much closer every data point is to the closest prototypical example from the same class compared to the closest prototypical example from another class (Sato & Yamada, 1995). Just as GESL, LVQ methods are theoretically well-justified because it yields a maximum-margin classifier (Schneider et al., 2009), and have been successfully applied for metric learning on the string edit distance in the past (Mokbel et al., 2015; Paaßen et al., 2016). However, in contrast to GESL, GLVQ also provides a principled way to select prototypical examples for metric learning, and is flexible enough to not only learn a cost matrix, but also a vectorial embedding of the tree labels, such that the Euclidean distance on these embeddings provides a discriminative cost function.

While embedding approaches are common in the literature, prior work has focused mostly on embedding trees *as a whole*, for example via graph kernel approaches (Aiolli et al., 2015; Bacciu et al., 2018; Da San Martino & Sperduti, 2010; Shervashidze et al., 2011; Yanardag & Vishwanathan, 2015), recursive neural networks (Gallicchio & Micheli, 2013; Hammer et al., 2007; Socher et al., 2013), or dimensionality reduction approaches (Gisbrecht et al., 2015). In this contribution, we wish to obtain an embedding for the single elements of a tree and maintain the tree structure. As of yet, such approaches only exist for sequences, namely in the form of recurrent neural network for natural language processing tasks (Cho et al., 2014; Sutskever et al., 2014). In addition to word embeddings for trees, our approach also provides a corresponding tree edit distance, which is op-

timized for classification, and offers an intuitive view on the data, supporting applications like intelligent tutoring systems (Paaßen et al., 2018).

3. Background

In this section, we revisit the basic problem of tree edit distance learning by first introducing the tree edit distance of Zhang & Shasha (1989), as well as the metric learning formalization suggested by Bellet et al. (2012).

3.1. Tree Edit Distance

We define a tree $\bar{x}$ over some set $\mathcal{X}$ as $x(\bar{x}_1, \ldots, \bar{x}_R)$ where $x \in \mathcal{X}$ and $\bar{x}_1, \ldots, \bar{x}_R$ is a (potentially empty) list of trees over $\mathcal{X}$. We denote the set of all possible trees over $\mathcal{X}$ as $\mathcal{T}(\mathcal{X})$. Further, we call x the *label* of the tree. We define the *size* of a tree $x(\bar{x}_1, \ldots, \bar{x}_R)$ as $|\bar{x}| := 1 + \sum_{r=1}^{R} |\bar{x}_r|$. Finally, we call a list of trees $\bar{x}_1, \ldots, \bar{x}_R$ a *forest*. Note that every tree is also a forest.

Next, we introduce *edits* over trees. In general, a tree edit δ is a function which transforms a forest into a forest (Paaßen et al., 2018). In this particular case, we are only concerned with three kinds of edits, namely deletions, which remove a certain label from a forest; insertions, which insert a certain label into a forest; and replacements which remove a certain label from a forest and put another label in its place. For example, deleting $\mathtt{x}$ from a tree $\mathtt{x}(\mathtt{y}, \mathtt{z})$ results in the forest $\mathtt{y}, \mathtt{z}$. Inserting $\mathtt{x}$ into this forest as parent of $\mathtt{y}$ results in the forest $\mathtt{x}(\mathtt{y}), \mathtt{z}$. Finally, replacing $\mathtt{z}$ with $\mathtt{q}$ in this forest results in the forest $\mathtt{x}(\mathtt{y}), \mathtt{q}$.

We associate each edit with a *cost* via a function $c : (\mathcal{X} \cup \{-\})^2 \to \mathbb{R}$. In particular, we define the cost of a deletion of label x as $c(x, -)$, the cost of an insertion of label y as $c(-, y)$, and the cost of a replacement of label x with label y as $c(x, y)$. We define the cost of a *sequence* of edits $\delta_1, \ldots, \delta_T$ as the sum over the costs of all edits.

Finally, we define the tree edit distance $d_c(\bar{x}, \bar{y})$ between any two trees $\bar{x}$ and $\bar{y}$ according to c as the cost of the cheapest sequence of edits that transforms $\bar{x}$ to $\bar{y}$. For example, if all edits have a cost of 1, the edit distance $d_c(\mathtt{x}(\mathtt{y}, \mathtt{z}), \mathtt{q}(\mathtt{z}(\mathtt{q})))$ between the trees $\mathtt{x}(\mathtt{y}, \mathtt{z})$ and $\mathtt{q}(\mathtt{z}(\mathtt{q}))$ is 3 because the cheapest sequence of edits is to replace $\mathtt{x}$ with $\mathtt{q}$, delete $\mathtt{y}$, and insert $\mathtt{q}$.

Zhang & Shasha (1989) showed that the tree edit distance can be computed efficiently using a dynamic programming algorithm if c is a pseudo-metric, meaning that c is a non-negative and symmetric function, such that $c(x, x) = 0$ for any $x \in \mathcal{X}$, and such that the triangular inequality is fulfilled.

Theorem 1. *Let c be a pseudo-metric on $\mathcal{X} \cup \{-\}$. Then, the corresponding tree edit distance $d_c(\bar{x}, \bar{y})$ can be computed in $\mathcal{O}(|\bar{x}|^2 \cdot |\bar{y}|^2)$ using a dynamic programming scheme.*

Conversely, if c violates the triangular inequality, the dynamic programming scheme overestimates the tree edit distance.

Proof. Refer to Zhang & Shasha (1989) for a proof of the first claim, and refer to the supplementary material (Paaßen, 2018b) for a proof of the second claim. □

Beyond enabling us to compute the tree edit distance efficiently, pseudo-metric cost functions c also ensure that the resulting tree edit distance d_c is a pseudo-metric itself.

Theorem 2. *Let c be a pseudo-metric on $\mathcal{X} \cup \{-\}$. Then, the corresponding tree edit distance d_c is a pseudo-metric on the set of possible trees over $\mathcal{X}$.*

However, if c violates any of the pseudo-metric properties (except for the triangular inequality), we can construct examples such that d_c violates the same pseudo-metric properties.

Proof. Refer to the supplementary material (Paaßen, 2018b). □

Both of these theorems make a pseudo-metric cost function c desirable. However, ensuring pseudo-metric properties on c may be challenging in metric learning, which is one of our key motivations for vectorial embeddings.

3.2. Tree Edit Distance Learning

Tree edit distance learning essentially means to adapt the cost function c, such that the resulting tree edit distance d_c is better suited for the task at hand. Following Bellet et al. (2012; 2014), we frame tree edit distance learning as minimizing some loss function over a set of *positive* pairs of trees $P \subset \mathcal{T}(\mathcal{X})^2$ and *negative* pairs of trees $N \subset \mathcal{T}(\mathcal{X})^2$, that is, trees which should be close and far away respectively. In particular, given a loss function E we wish to solve the optimization problem:

$$\min_c E(d_c, P, N) \tag{1}$$

In our contribution, we build upon the *good edit similarity learning* (GESL) approach of Bellet et al. (2012), who propose the loss function

$$\begin{aligned} E(d_c, P, N) =& \beta \cdot \|c\|^2 + \sum_{(\bar{x},\bar{y}) \in P} [d_c(\bar{x}, \bar{y}) - \eta]_+ \\ &+ \sum_{(\bar{x},\bar{y}) \in N} [\log(2) + \eta - d_c(\bar{x}, \bar{y})]_+ \end{aligned} \tag{2}$$

where $[\mu]_+ = \max\{0, \mu\}$ denotes the hinge loss, $\eta \in [0, \log(2)]$ is a slack variable permitting higher distances

between positive pairs if negative pairs are further apart, β is a scalar regularization constant, and $\|c\|^2$ denotes $\sum_{x\in\mathcal{X}\cup\{-\}}\sum_{y\in\mathcal{X}\cup\{-\}} c(x,y)^2$. As positive and negative pairs, Bellet et al. (2012) propose to use the K closest neighbors in the same class and the K furthest data points from a different class respectively, where "closeness" refers to the tree edit distance obtained via some initial, default cost function c_0 (Bellet et al., 2012).

Note that minimizing the loss function 2 is infeasible because changing the parameters c may change other edit sequences to become the cheapest option which in turn induces non-differentiable points in the loss function (Mokbel et al., 2015). Therefore, Bellet et al. (2012) suggest to compute the cheapest edit scripts according to a default cost function c_0 and keep them fixed for the optimization. More precisely, let $\boldsymbol{P}_{c_0}(\bar{x},\bar{y})$ be a matrix of size $|\bar{x}|+1\times|\bar{y}|+1$, such that $\boldsymbol{P}_{c_0}(\bar{x},\bar{y})_{i,j}$ is 1 if and only if the cheapest edit script that transforms $\bar{x}$ to $\bar{y}$ according to the cost function c_0 contains a replacement of node x_i with node y_j, where x_i is the ith node in tree $\bar{x}$ according to pre-order and y_j is the jth node in tree $\bar{y}$ according to pre-order. Further, if the cheapest edit script deletes x_i, we define $\boldsymbol{P}_{c_0}(\bar{x},\bar{y})_{i,|\bar{y}|+1}=1$ and if the cheapest edit script inserts y_j, we define $\boldsymbol{P}_{c_0}(\bar{x},\bar{y})_{|\bar{x}|+1,j}=1$. We define all other entries of $\boldsymbol{P}_{c_0}(\bar{x},\bar{y})$ as zero. We can compute the matrix $\boldsymbol{P}_{c_0}(\bar{x},\bar{y})$ in $\mathcal{O}(|\bar{x}|^2\cdot|\bar{y}|+|\bar{x}|\cdot|\bar{y}|^2)$ via backtracing (refer to the supplementary material (Paaßen, 2018a) for details).

Using this matrix, we can define the *pseudo edit distance* $\tilde{d}_c(\bar{x},\bar{y})$, which we define as follows.

$$\tilde{d}_c(\bar{x},\bar{y}) = \sum_{i=1}^{|\bar{x}|+1}\sum_{j=1}^{|\bar{y}|+1} \boldsymbol{P}_{c_0}(\bar{x},\bar{y})_{i,j}\cdot c(x_i,y_j) \qquad (3)$$

where we define $x_{|\bar{x}|+1}=y_{|\bar{y}|+1}:=-$. GESL now minimizes the loss function 2 with respect to the pseudo-edit distance, which is a quadratic optimization problem.

Bellet et al. (2012) show that GESL optimizes the "goodness" of the similarity measure $k(\bar{x},\bar{y}) = 2\cdot\exp(-\tilde{d}_c(\bar{x},\bar{y}))-1$. The concept of goodness has been introduced by Balcan et al. (2008) and quantifies how well a given similarity measure lends itself for binary classification. In particular, assume trees $\bar{x}_1,\dots,\bar{x}_m$ with class assignments $\ell(\bar{x}_1),\dots,\ell(\bar{x}_m)$. Then, we wish to learn parameters $\vec{\alpha}\in\mathbb{R}^m$, such that we can classify a new tree $\bar{x}$ via the predictive function $f(\bar{x})=\operatorname{sign}(\sum_{i=1}^m \alpha_i\cdot k(\bar{x},\bar{x}_i))$. We can learn the parameters $\vec{\alpha}$ by solving the linear minimization problem:

$$\min_{\vec{\alpha}} \sum_{i=1}^m \left[1-\ell(\bar{x}_i)\cdot\sum_{j=1}^m \alpha_j\cdot k(\bar{x}_i,\bar{x}_j)\right]_+ + \lambda\cdot\|\vec{\alpha}\|_1$$

where λ is a hyper-parameter regulating the L1 regularization, and hence the sparsity, of $\vec{\alpha}$.

Recall that GESL optimizes the pseudo edit distance $\tilde{d}_c(\bar{x},\bar{y})$ instead of the edit distance $d_c(\bar{x},\bar{y})$, and that the theory provided by Bellet et al. (2012) does not guarantee the goodness of the actual edit distance $d_c(\bar{x},\bar{y})$. Indeed, it may occur that the loss $E(d_c,P,N)$ for the actual tree edit distance d_c is considerably larger than the loss $E(\tilde{d}_c,P,N)$.

Theorem 3. *There exists combinations of an alphabet $\mathcal{X}$, positive pairs P, negative pairs N, a default cost function c_0, and a regularization constant β, such that the cost function c_1 learned by GESL is not a pseudo-metric, and yields a loss $E(d_{c_1},P,N) > E(\tilde{d}_{c_1},P,N)$, as well as $E(d_{c_1},P,N) > E(d_{c_0},P,N)$.*

Proof. Refer to the supplementary material (Paaßen, 2018b). □

Overall, we identify three key points in GESL we would like to address. First, we would like to select positive and negative pairs in a principled fashion, in contrast to the ad-hoc scheme of choosing the closest trees from the same class and the furthest trees from another class. Second, we would like to enhance the coupling between the pseudo tree edit distance $\tilde{d}_c$ to the actual tree edit distance d_c. Third, we would like to ensure pseudo-metric properties on c.

4. Method

In this section we introduce a novel method for tree edit distance learning. We start by selecting positive and negative pairs for metric learning via median learning vector quantization. Then, we introduce metric learning using the generalized learning vector quantization cost function. Finally, we propose a novel parameterization of the edit cost function c via symbol embeddings.

4.1. Median Learning Vector Quantization

To facilitate fast metric learning, we would like to limit ourselves to as few positive and negative pairs as possible. We propose to select positive and negative pairs via *prototypical* data points which represent the classes well. In particular, assume we have data points $\bar{x}_1,\dots,\bar{x}_m$ with class assignments $\ell(\bar{x}_1),\dots,\ell(\bar{x}_m)$. We propose to select a small sample of prototypes $\bar{w}_1,\dots,\bar{w}_K\subset\{\bar{x}_1,\dots,\bar{x}_m\}$ with $K\ll m$, and to construct positive pairs for all $\bar{x}_i$ as $(\bar{x}_i,\bar{w}_i^+)$, where $\bar{w}_i^+$ is the closest prototype to $\bar{x}_i$ according to d_{c_0} from the same class; and negative pairs as $(\bar{x}_i,\bar{w}_i^-)$, where $\bar{w}_i^-$ is the closest prototype to $\bar{x}_i$ according to d_{c_0} from a different class.

In our approach, we select these prototypical data points such that they help us to discriminate between the classes. More precisely, we aim for prototypes $\bar{w}_1,\dots,\bar{w}_K$ which can classify as many data points correctly as possible by

assigning the class of the closest prototype. One way to obtain such prototypes is to optimize the generalized learning vector quantization (GLVQ) cost function (Sato & Yamada, 1995):

$$E = \sum_{i=1}^{m} \Phi\Big(\frac{d_i^+ - d_i^-}{d_i^+ + d_i^-}\Big) \tag{4}$$

where d_i^+ is the distance of $\bar{x}_i$ to $\bar{w}_i^+$, d_i^- is the distance of $\bar{x}_i$ to $\bar{w}_i^-$, and $\Phi(\mu) = \log(4 + \mu)$ (Nebel et al., 2015). Note that the fraction $(d_i^+ - d_i^-)/(d_i^+ + d_i^-)$ is positive if and only if $\bar{x}_i$ is misclassified, such that the cost function is related to the classification error.

Note that optimizing the GLVQ cost function in this case requires a discrete optimization scheme because the prototypes $\bar{w}_k$ are limited to be training data points, which is called *median* learning vector quantization (Nebel et al., 2015). We follow the suggestion of Nebel et al. (2015) and apply a generalized expectation maximization (EM) scheme to maximize $\sum_{i=1}^{m} \log(g_i^- + g_i^+)$, where $g_i^- = 2 + d_i^-/(d_i^+ + d_i^-)$, and $g_i^+ = 2 - d_i^+/(d_i^+ + d_i^-)$. The expectation step of the EM scheme consists of computing the quantities $\gamma_i^+ = g_i^+/(g_i^+ + g_i^-)$ as well as $\gamma_i^- = g_i^-/(g_i^+ + g_i^-)$ for all i, and the maximization step consists of finding a prototype $\bar{w}_k$ which can be set to a different data point $\bar{x}_i$ such that the likelihood $\mathcal{L} = \sum_{i=1}^{m} \gamma_i^+ \cdot \log(g_i^+/\gamma_i^+) + \gamma_i^- \cdot \log(g_i^-/\gamma_i^-)$ is improved, assuming fixed γ_i^+ and γ_i^-. The EM scheme stops if it is not possible to improve $\mathcal{L}$ for any prototype anymore.

4.2. Metric Learning via Learning Vector Quantization

Recall that the GLVQ cost function in Equation 4 quantifies how well our prototypes classify the training data. Following the recommendation of Mokbel et al. (2015), we can not only use this cost function for learning the prototypes, but also for learning the metric. In particular, we can minimize the GLVQ loss with respect to the cost function c using any unconstrained gradient-based solver, such as the limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) algorithm (Liu & Nocedal, 1989). For the gradient $\nabla_c E$ we obtain:

$$\sum_{i=1}^{m} \Phi'\Big(\frac{d_i^+ - d_i^-}{d_i^+ + d_i^-}\Big) \cdot \frac{d_i^- \cdot \nabla_c d_i^+ - d_i^+ \cdot \nabla_c d_i^-}{(d_i^+ + d_i^-)^2} \tag{5}$$

where $\Phi'(\mu) = 1/(4 + \mu)$.

Following the GESL approach of Bellet et al. (2012), we optimize the pseudo tree edit distance $\tilde{d}_c$ instead of the tree edit distance itself, which yields the gradient $\nabla_c \tilde{d}(\bar{x}_i, \bar{w}_k) = \boldsymbol{P}_{c_0}(\bar{x}_i, \bar{w}_k)$. However, we improve upon GESL by not only considering only the cheapest edit script, but instead the average over all cheapest edit scripts. In particular, we consider $\boldsymbol{P}_{c_0}(\bar{x}_i, \bar{w}_k)$ to be the average over the matrices for all cheapest edit scripts.

Considering all co-optimal scripts permits us to exploit additional information, with which we can prevent many degenerate cases in which $\tilde{d}_c$ underestimates d_c. In particular, the counter example in the proof of Theorem 3 does not hold in this case. Computing this average over all cheapest edit scripts is possible efficiently via a novel forward-backward algorithm which we developed for this contribution (refer to the supplementary material (Paaßen, 2018a)).

We further note that changes to the metric may also enable us to optimize the prototype locations further. Therefore, we employ an alternating optimization scheme where we first learn the prototype positions according to median relational GLVQ, then adapt the metric, and repeat until either the prototype positions do not change anymore or the solver is not able to improve the metric anymore.

Until now, we have addressed the selection of positive and negative pairs, as well as a closer coupling between pseudo edit distance and edit distance. However, we still have to ensure pseudo-metric properties on the learned cost function. For this purpose, we introduce vectorial embeddings.

4.3. Tree Label Embeddings

Let $\mathcal{X}$ be a finite set with U elements. Then, a vector embedding of $\mathcal{X}$ is a matrix $\boldsymbol{A} \in \mathbb{R}^{V \times U}$ with $V \leq U$, where each column is a vector $\vec{a}(x)$ for one $x \in \mathcal{X}$. Further, we define $\vec{a}(-) := \vec{0}$, i.e. the origin of the V-dimensional Euclidean space. We define the cost function corresponding to an embedding as the Euclidean distance between the embedding vectors, that is: $c_{\boldsymbol{A}}(x, y) := \|\vec{a}(x) - \vec{a}(y)\|$.

Because the cost function is the Euclidean distance, it is guaranteed to be a pseudo-metric, irrespective of the choice of the embedding $\boldsymbol{A}$. Furthermore, $c_{\boldsymbol{A}}$ is differentiable with respect to the embedding vectors with the gradient $\nabla_{\vec{a}(x)} c_{\boldsymbol{A}}(x, y) = (\vec{a}(x) - \vec{a}(y))/\|\vec{a}(x) - \vec{a}(y)\|$. Using this gradient and Equation 3, we can also obtain a gradient of the pseudo-edit distance $\tilde{d}_c(\bar{x}, \bar{y})$ with respect to $\vec{a}(x)$:

$$\begin{aligned} &\nabla_{\vec{a}(x)} \tilde{d}_{c_{\boldsymbol{A}}}(\bar{x}, \bar{y}) = \\ &\sum_{i=1}^{|\bar{x}|} \delta(x, x_i) \cdot \Bigg[\sum_{j=1}^{|\bar{y}|+1} \boldsymbol{P}_{c_0}(\bar{x}, \bar{y})_{i,j} \cdot \frac{\vec{a}(x) - \vec{a}(y_j)}{\|\vec{a}(x) - \vec{a}(y_j)\|}\Bigg] \\ &+ \sum_{j=1}^{|\bar{y}|} \delta(x, y_j) \cdot \Bigg[\sum_{i=1}^{|\bar{x}|+1} \boldsymbol{P}_{c_0}(\bar{x}, \bar{y})_{i,j} \cdot \frac{\vec{a}(x) - \vec{a}(x_i)}{\|\vec{a}(x) - \vec{a}(x_i)\|}\Bigg] \end{aligned} \tag{6}$$

where δ is the Kronecker-Delta, i.e.: $\delta(x, y) = 1$ if $x = y$ and 0 otherwise.

Finally, we can plug this result into Equation 5, which yields a gradient $\nabla_{\vec{a}(x)} E$, such that we can learn the vectorial embedding of $\mathcal{X}$ via gradient techniques. Note that prior theory on metric learning on the GLVQ cost function suggests that the learned embedding will degenerate to a very low rank-

matrix such that the model may become overly simplistic (Biehl et al., 2015). To prevent such a degeneration, we follow the regularization recommendation of Schneider et al. (2010) and add the term $\beta \cdot \log(\det(\boldsymbol{A}^T \cdot \boldsymbol{A}))$ to the GLVQ loss 4, which adds the gradient $\beta \cdot 2 \cdot \boldsymbol{A}^{\dagger T}$ where $\boldsymbol{A}^\dagger$ is the Moore-Penrose-Pseudoinverse of $\boldsymbol{A}$. Additionally, we follow the regularization approach of good edit similarity learning (Bellet et al., 2012) and add the Frobenius-norm $\beta \cdot \|\boldsymbol{A}\|_F^2$ to the loss, which adds the gradient $\beta \cdot 2 \cdot \boldsymbol{A}$.

As initialization of the vectorial embedding we use a U-dimensional simplex with side length 1, which leads to $c_0(x, y) = 0$ if $x = y$ and 1 otherwise (refer to the supplementary material for a more detailed look into this initialization (Paaßen, 2018b)).

Regarding computational complexity, we can analyze the gradient computation. To compute a gradient, we first need to select the closest correct and closest wrong prototype for every data point, which is possible in $\mathcal{O}(m \cdot K)$. Then, we need to compute the gradient for each data point via Equation 6, which is possible in $\mathcal{O}(m \cdot n^2 \cdot V)$ where n is the largest tree size in the data set. Computing the regularization requires $\mathcal{O}(V^3)$ due to the Pseudoinverse, resulting in $\mathcal{O}(m \cdot K + m \cdot n^2 \cdot V + V^3)$ overall. How often the gradient needs to be computed depends on the optimizer, but can typically be regarded as a constant. In our experiments, we limit the number of gradient computations to 200.

5. Experiments

In our experiment, we investigate whether our proposed metric learning scheme is able to improve classification accuracy beyond the default initialization, whether our proposed metric learning scheme improves upon the accuracy obtained by good edit similarity learning (Bellet et al., 2012), and whether the resulting embedding yields insight regarding the classification task in question. In particular, we evaluate on the following data sets, including a variety of domains and data set sizes.

Strings: A two-class data set of 200 strings of length 12, adapted from Mokbel et al. (2015). Strings in class 1 consist of 6 `a` or `b` symbols, followed by a `c` or `d`, followed by another 5 `a` or `b` symbols. Which of the two respective symbols is selected is chosen uniformly at random. Strings in class 2 are constructed in much the same way, except that they consist of 5 `a` or `b` symbols, followed by a `c` or `d`, followed by another 6 `a` or `b` symbols. Note that the classes can be neither discriminated via length nor via symbol frequency features. The decisive discriminative feature is where a `c` or `d` is located in the string.

MiniPalindrome and **Sorting:** Two data sets of Java programs, where classes represent different strategies to solve a programming task. The MiniPalindrome data set contains 48 programs implementing one of eight strategies to detect whether an input string contains only palindromes (Paaßen, 2016a), and the Sorting data set contains 64 programs implementing either a BubbleSort or an InsertionSort strategy (Paaßen, 2016b). The programs are represented by their abstract syntax tree where the label corresponds to one of 24 programming concepts (e.g. class declaration, function declaration, method call, etc.).

Cystic and **Leukemia:** Two data sets from KEGG/Glycan data base (Hashimoto et al., 2006) adapted from Gallicchio & Micheli (2013), where one class corresponds to benign molecules and the other class corresponds to molecules associated with cystic fibrosis or leukemia respectively. The molecules are represented as trees, where the label corresponds to mono-saccharide identifiers (one of 29 and one of 57 for Cystic and Leukemia, respectively), and the roots are chosen according to biological meaning (Hashimoto et al., 2006). The cystic data set contains 160, the Leukemia data set 442 molecules.

Sentiment: A large-scale two-class data set of 9613 sentences from movie reviews, where one class (4650 trees) corresponds to negative and the other class (4963 trees) to positive reviews. The sentences are represented by their syntax trees, where inner nodes are unlabeled and leaves are labeled with one of over $30,000$ words (Socher et al., 2013). Note that GESL is not practically applicable for this data set, as the number of parameters to learn scales quadratically with the number of words, i.e. $> 30,000^2$. We adapted our proposed method by not learning a vectorial embedding from scratch, but starting off with the already existing, 300-dimensional Common Crawl GloVe embedding (Pennington et al., 2014), which we reduce via PCA, retaining 95% of the data variance and $V = 16.4 \pm 2.3$ dimensions on average $\pm$ standard deviation. We adapt this initial embedding via a linear transformation $\Omega \in \mathbb{R}^{V \times V}$ which we learn using our proposed metric learning scheme. Further, we replaced the cost function with the cosine distance $c_\Omega(\vec{x}, \vec{y}) = \frac{1}{2} - \frac{1}{2} \cdot ((\Omega \cdot \vec{x})^T \cdot \Omega \cdot \vec{y}) / (\|\Omega \cdot \vec{x}\| \cdot \|\Omega \cdot \vec{y}\|)$, which is well established in the analysis of word embeddings (Pennington et al., 2014) (refer to the supplementary material for the gradient (Paaßen, 2018b)).

On each data set, we perform a crossvalidation[1] and compare the average test error across folds. In particular, we compare the error when using the initial tree edit distance with the error when using the pseudo-edit distance learned

[1] We used 20 folds for Strings and Sentiment, 10 for Cystic and Leukemia, 8 for Sorting and 6 for MiniPalindrome. For the programming data sets, the number of folds had to be reduced to ensure that each fold still contained a meaningful number of data points. For the Cystic and Leukemia data set, our ten folds were consistent with the paper of Gallicchio & Micheli (2013). In all cases, folds were generated such that the label distribution of the overall data set was maintained.

via good edit similarity learning (GESL), and the tree edit distance learned via our proposed embedding via learning vector quantization (LVQ) approach.

In general, we would expect that a discriminative metric learned for one classifier also facilitates classification using other classifiers. Therefore, we report the classification error for four classifiers, namely the median relational generalized learning vector quantization classifier (MRGLVQ), the K-nearest neighbor (KNN) classifier, the goodness classifier proposed by Bellet et al. (2012), and the support vector machine (SVM) based on the radial basis function kernel. To ensure positive semi-definiteness of the resulting similarity matrix for SVM, we set negative eigenvalues to zero (clip Eigenvalue correction). Note that this eigenvalue correction requires cubic runtime in terms of the number of data points and is thus prohibitively slow for data set sizes far beyond > 1000 data points. Therefore, for the Sentiment data set, we trained the classifiers on a randomly selected sample of 300 points from the training data.

We optimized all hyper-parameters in a nested 5-fold cross-validation, namely the number of prototypes K for MRGLVQ and LVQ metric learning in the range $[1, 15]$, the number of neighbors for KNN in the range $[1, 15]$, the kernel bandwidth for SVM in the range $[0.1, 10]$, the sparsity parameter λ for the goodness classifier in the range $[10^{-}5, 10]$, and the regularization strength β for GESL and our proposed metric learning in the range $2{\cdot}K{\cdot}m{\cdot}[10^{-6}, 10^{-2}]$. We chose the number of prototypes for our proposed metric learning scheme, as well as the number of neighbos for GESL as the optimal number of prototypes K for MRGLVQ.

As implementations, we used custom implementations of KNN, MRGLVQ, the goodness classifier, and GESL, and the LIBSVM standard implementation for SVM (Chang & Lin, 2011). All experiments were performed on a consumer-grade laptop with an Intel Core i7-7700 HQ CPU. Our custom implementations as well as an example experimental script are available at `https://doi.org/10.4119/unibi/2919994`.

The results of our experiments are displayed in Table 5. In all data sets and for all classifiers, LVQ metric learning yields lower classification error compared to GESL. For the Strings data set we can also verify this result statistically with a one-sided Wilcoxon signed rank test ($p < 10^{-4}$). Furthermore, in all but the Leukemia data set, LVQ metric learning yields the overall best classification results, and is close to optimal for the Leukemia data set (0.2% difference). In all cases, LVQ metric learning could improve the accuracy for KNN, in five out of six cases for SVM (the exception being the Cystic data set), in four out of six cases for MRGLVQ (in Sorting and Leukemia it stayed equal), and in three out of six cases for the goodness classifier. For the Strings and Sentiment data sets we can also verify this

Table 1. The mean test classification error and runtimes for metric learning, averaged over the cross validation trials, as well as the standard deviation. The x-axis shows the metric learning schemes, the y-axis the different classifiers used for evaluation. The table is sub-divided for each data set. The lowest classification error for each data set is highlighted via bold print.

classifier	initial	GESL	LVQ
	Strings		
KNN	$21.0 \pm 10.2\%$	$23.0 \pm 10.8\%$	$\mathbf{0.0 \pm 0.0}\%$
MRGLVQ	$36.0 \pm 15.7\%$	$34.0 \pm 11.0\%$	$\mathbf{0.0 \pm 0.0}\%$
SVM	$9.0 \pm 11.2\%$	$10.0 \pm 8.6\%$	$\mathbf{0.0 \pm 0.0}\%$
goodness	$11.5 \pm 9.3\%$	$0.5 \pm 2.2\%$	$\mathbf{0.0 \pm 0.0}\%$
runtime [s]	0 ± 0	0.030 ± 0.002	$1.077 \pm 0.098\%$
	MiniPalindrome		
KNN	$12.5 \pm 11.2\%$	$12.5 \pm 7.9\%$	$10.4 \pm 9.4\%$
MRGLVQ	$2.1 \pm 5.1\%$	$4.2 \pm 6.5\%$	$\mathbf{0.0 \pm 0.0}\%$
SVM	$4.2 \pm 6.5\%$	$20.8 \pm 15.1\%$	$\mathbf{0.0 \pm 0.0}\%$
good	$6.2 \pm 6.8\%$	$14.6 \pm 5.1\%$	$8.3 \pm 10.2\%$
runtime [s]	0 ± 0	0.103 ± 0.014	2.785 ± 0.631
	Sorting		
KNN	$15.6 \pm 8.8\%$	$18.8 \pm 16.4\%$	$10.9 \pm 8.0\%$
MRGLVQ	$14.1 \pm 10.4\%$	$14.1 \pm 8.0\%$	$14.1 \pm 8.0\%$
SVM	$10.9 \pm 8.0\%$	$\mathbf{9.4 \pm 8.8}\%$	$\mathbf{9.4 \pm 8.8}\%$
good	$15.6 \pm 11.1\%$	$17.2 \pm 14.8\%$	$17.2 \pm 9.3\%$
runtime [s]	0 ± 0	0.352 ± 0.102	3.358 ± 0.748
	Cystic		
KNN	$31.2 \pm 6.6\%$	$32.5 \pm 10.1\%$	$28.1 \pm 8.5\%$
MRGLVQ	$34.4 \pm 6.8\%$	$33.1 \pm 9.8\%$	$30.0 \pm 10.1\%$
SVM	$28.1 \pm 9.0\%$	$33.1 \pm 8.9\%$	$29.4 \pm 12.5\%$
good	$28.1 \pm 8.5\%$	$26.2 \pm 14.4\%$	$\mathbf{24.4 \pm 13.3}\%$
runtime [s]	0 ± 0	0.353 ± 0.292	$0.864 \pm 0.767\%$
	Leukemia		
KNN	$7.5 \pm 2.6\%$	$8.2 \pm 4.6\%$	$7.3 \pm 4.3\%$
MRGLVQ	$9.5 \pm 4.0\%$	$10.9 \pm 4.7\%$	$9.5 \pm 3.0\%$
SVM	$7.0 \pm 4.1\%$	$8.8 \pm 2.9\%$	$6.8 \pm 4.7\%$
good	$\mathbf{6.1 \pm 4.3}\%$	$10.0 \pm 4.4\%$	$6.3 \pm 3.8\%$
runtime [s]	0 ± 0	2.208 ± 0.919	$6.550 \pm 2.706\%$
	Sentiment		
kNN	$40.2 \pm 2.8\%$	–	$38.2 \pm 3.3\%$
MRGLVQ	$44.0 \pm 2.6\%$	–	$41.3 \pm 5.7\%$
SVM	$34.3 \pm 3.0\%$	–	$\mathbf{33.3 \pm 3.6}\%$
goodness	$43.7 \pm 1.9\%$	–	$42.5 \pm 3.1\%$
runtime [s]	0 ± 0	–	69.385 ± 58.064

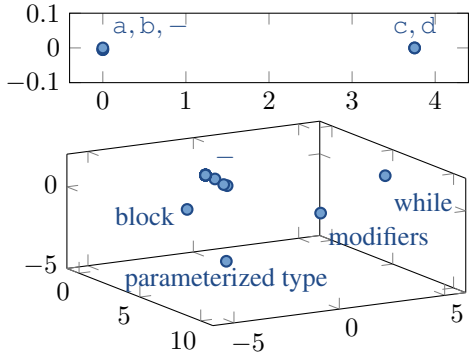

Figure 1. A PCA of the learned embeddings for the Strings (top) and the MiniPalindrome data set (bottom), covering 100% and 83.54% of the variance respectively.

result statistically with $p < 0.05$ for all classifiers.

Note that the focus of our work is to improve classification accuracy via metric learning, not to develop state-of-the-art classifiers as such. However, we note that our results for the Sorting data set outperform the best reported results by Paaßen et al. (2016) of 15%. For the Cystic data set we improve the AUC from $76.93 \pm 0.97\%$ mean and standard deviation across crossvalidation trials to $79.2 \pm 13.6\%$, and for the Leukemia data set from $93.8 \pm 3.3\%$ to $94.6 \pm 4.5\%$. Both values are competitive with the results obtained via recursive neural networks and a multitude of graph kernels by Gallicchio & Micheli (2013). For the Sentiment data set, we obtain a SVM classification error of 27.51% on the validation set, which is noticeably worse than the reported literature results of around 12.5% (Socher et al., 2013). However, we note that we used considerably less data to train our classifier (only 500 points for the validation).

Interestingly, GESL tended to decrease classification accuracy compared to the initial tree edit distance. Likely, GESL requires more neighbors for better results (Bellet et al., 2012). However, scaling up to a high number of neighbors lead to prohibitively high runtimes for our experiments such that we do not report these results here. These high runtimes can be explained by the fact that the number of slack variables in GESL increases with $\mathcal{O}(m \cdot K)$ where m is the number of data points and K is the number of neighbors. The scaling behavior is also visible in our experimental results. For data sets with little data points and neighbors, such as Strings, MiniPalindrome, and Sorting, GESL is 10 to 30 times faster compared to our proposed method. However, for Cystic and Leukemia, the runtime advantage shrinks to a factor of 2 to 3.

In ablation studies, we studied the difference between GESL and our proposed method in more detail. We observed that considering the average over all co-optimal edit scripts, and considering LVQ prototypes instead of ad-hoc nearest neighbors, improved GESL on the MiniPalindrome data set, worsened it for the Strings data set, and showed no remarkable difference for the Sorting, Cystic, and Leukemia data set. We also compared LVQ metric learning without the embedding approach and with the embedding approach. Interestingly, the pseudo-edit distance performed worse when considering embeddings, while the actual edit distance performed better when considering embeddings. In general, GESL variants performed better for the pseudo-edit distance than for the actual edit distance, and LVQ variants performed better for the actual edit distance compared to the pseudo edit distance. We report the full results in the supplementary material (Paaßen, 2018b).

Beyond classification accuracy, our metric learning approach permits to inspect the resulting embedding. Figure 1 displays a PCA of the embeddings learned for the Strings and MiniPalindrome data set respectively. As we can see, the embedding for the Strings data set captures the objective of the task perfectly: Both `a` as well as `b` symbols are irrelevant for class discrimination and are thus embedded at the origin, while `c` and `d` are embedded far from the origin, but both at the same location. For MiniPalindrome, we also observe that most syntactic concepts are embedded at zero, indicating that a combination of the four remaining concepts is sufficient for class discrimination; namely the block concept, which captures the nesting structure of the program, the while concept, which is specific to one of the classes, the modifiers concept, which can serve to count the number of variables and methods in the program, and the parametrized type concept, which distinguishes programs with advanced data structured from programs with primitive data structures.

6. Conclusion

In this contribution we have proposed a novel approach for edit distance learning on trees with three distinct characteristics: First, our objective is the generalized learning vector quantization cost function, which pulls data points closer to the closest prototype for their own class and pushes them away from the closest prototype for a different class; second, we consider not only a single optimal edit script between trees but a summary of *all* co-optimal edit scripts; finally, we do not learn a cost function for the edits directly, but instead a vectorial embedding of the label alphabet, which can be interpreted subsequently, and guarantees metric properties. In our experiments we have shown that our proposed metric learning scheme improves upon the state-of-the-art of good edit similarity learning for trees on a diverse tree data sets including Java program syntax trees, tree-based molecule representations from a biomedical task, and syntax trees in natural language processing.

Limitations of our work include that an improvement of the loss function for the pseudo edit distance does still not strictly imply an improvement of the loss for the actual edit distance, and that improvements in classification accuracy are small for some classifiers and some data sets. Future research should investigate the relation between pseudo edit distance and edit distance, as well as the relation between the number of prototypes and metric learning performance in more detail. It may also be worthwhile to study different cost functions, in particular probabilistic ones, which may be compatible with probabilistic models of the edit distance. Still, we regard our existing contribution as a meaningful step towards edit distances on trees which are both discriminative as well as interpretable and can thus enhance accuracy and understanding on classification tasks of structured data.

Acknowledgements

Funding by the DFG under grant number HA 2719/6-2 and the CITEC center of excellence (EXC 277) is gratefully acknowledged. Thanks to David Nebel and our anonymous reviewers for helpful comments and suggestions.

References

Aiolli, F. and Donini, M. EasyMKL: A scalable multiple kernel learning algorithm. *Neurocomputing*, 169:215–224, 2015. doi:10.1016/j.neucom.2014.11.078.

Aiolli, F., Martino, G. D. S., and Sperduti, A. An efficient topological distance-based tree kernel. *IEEE Transactions on Neural Networks and Learning Systems*, 26(5): 1115–1120, 2015. doi:10.1109/TNNLS.2014.2329331.

Bacciu, D., Micheli, A., and Sperduti, A. Generative kernels for tree-structured data. *IEEE Transactions on Neural Networks and Learning Systems*, 2018. doi:10.1109/TNNLS.2017.2785292. in press.

Balcan, M.-F., Blum, A., and Srebro, N. A theory of learning with similarity functions. *Machine Learning*, 72(1):89–112, Aug 2008. doi:10.1007/s10994-008-5059-5.

Bellet, A., Habrard, A., and Sebban, M. Good edit similarity learning by loss minimization. *Machine Learning*, 89(1): 5–35, Oct 2012. doi:10.1007/s10994-012-5293-8.

Bellet, A., Habrard, A., and Sebban, M. A survey on metric learning for feature vectors and structured data. *arXiv e-prints*, 2014. URL http://arxiv.org/abs/1306.6709.

Bellet, A., Bernabeu, J. F., Habrard, A., and Sebban, M. Learning discriminative tree edit similarities for linear classification - application to melody recognition. *Neurocomputing*, 214:155 – 161, 2016. doi:10.1016/j.neucom.2016.06.006.

Biehl, M., Hammer, B., Schleif, F.-M., Schneider, P., and Villmann, T. Stationarity of matrix relevance LVQ. In *Proceedings of the 2015 International Joint Confgerence on Neural Networks (IJCNN 2015)*, pp. 1–8, 2015. doi:10.1109/IJCNN.2015.7280441.

Boyer, L., Habrard, A., and Sebban, M. Learning metrics between tree structured data: Application to image recognition. In Kok, J. N., Koronacki, J., Mantaras, R. L. d., Matwin, S., Mladenič, D., and Skowron, A. (eds.), *Proceedings of the 18th European Conference on Machine Learning (ECML 2007)*, pp. 54–66. Springer, 2007. doi:10.1007/978-3-540-74958-5_9.

Chang, C.-C. and Lin, C.-J. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2(3):27:1–27:27, 2011. doi:10.1145/1961189.1961199. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

Cho, K., van Merrienboer, B., Gülçehre, Ç., Bougares, F., Schwenk, H., and Bengio, Y. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In Moschitti, A., Pang, B., and Daelemans, W. (eds.), *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*, pp. 1724–1734, 2014. URL https://www.aclweb.org/anthology/D14-1179.

Cover, T. M. and Hart, P. E. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13 (1):21–27, 1967. doi:10.1109/TIT.1967.1053964.

Da San Martino, G. and Sperduti, A. Mining structured data. *Computational Intelligence Magazine*, 5(1):42–49, Feb 2010. doi:10.1109/MCI.2009.935308.

Gallicchio, C. and Micheli, A. Tree echo state networks. *Neurocomputing*, 101:319–337, 2013. doi:10.1016/j.neucom.2012.08.017.

Gisbrecht, A., Schulz, A., and Hammer, B. Parametric nonlinear dimensionality reduction using kernel t-SNE. *Neurocomputing*, 147:71–82, 2015. doi:10.1016/j.neucom.2013.11.045.

Hammer, B., Micheli, A., and Sperduti, A. *Adaptive Contextual Processing of Structured Data by Recursive Neural Networks: A Survey of Computational Properties*, pp. 67–94. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007. doi:10.1007/978-3-540-73954-8_4.

Hashimoto, K., Goto, S., Kawano, S., Aoki-Kinoshita, K. F., Ueda, N., Hamajima, M., Kawasaki, T., and Kanehisa, M. KEGG as a glycome informatics resource. *Glycobiology*, 16(5):63R–70R, 2006. doi:10.1093/glycob/cwj010.

Kohonen, T. *Learning Vector Quantization*, pp. 175–189. Springer Berlin Heidelberg, 1995. doi:10.1007/978-3-642-97610-0_6.

Kulis, B. Metric learning: A survey. *Foundations and Trends in Machine Learning*, 5(4):287–364, 2013. doi:10.1561/2200000019.

Liu, D. C. and Nocedal, J. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45(1):503–528, 1989. doi:10.1007/BF01589116.

Mokbel, B., Paaßen, B., Schleif, F.-M., and Hammer, B. Metric learning for sequences in relational LVQ. *Neurocomputing*, 169:306–322, 2015. doi:10.1016/j.neucom.2014.11.082.

Nebel, D., Hammer, B., Frohberg, K., and Villmann, T. Median variants of learning vector quantization for learning of dissimilarity data. *Neurocomputing*, 169:295–305, 2015. doi:10.1016/j.neucom.2014.12.096.

Paaßen, B., Mokbel, B., and Hammer, B. Adaptive structure metrics for automated feedback provision in intelligent tutoring systems. *Neurocomputing*, 192:3–13, 2016. doi:10.1016/j.neucom.2015.12.108.

Paaßen, B., Hammer, B., Price, T. W., Barnes, T., Gross, S., and Pinkwart, N. The Continuous Hint Factory - providing hints in vast and sparsely populated edit distance spaces. *Journal of Educational Datamining*, 2018. URL http://arxiv.org/abs/1708.06564. accepted.

Paaßen, B. MiniPalindrome, 2016a. Bielefeld University, doi:10.4119/unibi/2900666.

Paaßen, B. Java Sorting Programs, 2016b. Bielefeld University, doi:10.4119/unibi/2900684.

Paaßen, B. Revisiting the tree edit distance and its backtracing: A tutorial. *ArXiv e-prints*, 2018a. URL https://arxiv.org/abs/1805.06869.

Paaßen, B. Tree edit distance learning via adaptive symbol embeddings: Supplementary materials and results. *ArXiv e-prints*, 2018b. URL https://arxiv.org/abs/1805.07123.

Pennington, J., Socher, R., and Manning, C. D. GloVe: Global vectors for word representation. In Moschitti, A., Pang, B., and Daelemans, W. (eds.), *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*, pp. 1532–1543, 2014. URL http://www.aclweb.org/anthology/D14-1162.

Sato, A. and Yamada, K. Generalized learning vector quantization. In Tesauro, G., Touretzky, D., and Leen, T. (eds.), *Proceedings of the 7th conference on Advances in Neural Information Processing Systems (NIPS 1995)*, pp. 423–429. MIT Press, 1995. URL https://papers.nips.cc/paper/1113-generalized-learning-vector-quantization.

Schleif, F.-M. and Tino, P. Indefinite proximity learning: A review. *Neural Computation*, 27(10):2039–2096, 2015. doi:10.1162/NECO_a_00770.

Schneider, P., Biehl, M., and Hammer, B. Adaptive relevance matrices in learning vector quantization. *Neural Computation*, 21(12):3532–3561, 2009. doi:10.1162/neco.2009.11-08-908.

Schneider, P., Bunte, K., Stiekema, H., Hammer, B., Villmann, T., and Biehl, M. Regularization in matrix relevance learning. *IEEE Transactions on Neural Networks*, 21(5):831–840, 2010. doi:10.1109/TNN.2010.2042729.

Shervashidze, N., Schweitzer, P., Leeuwen, E. J. v., Mehlhorn, K., and Borgwardt, K. M. Weisfeiler-Lehman graph kernels. *Journal of Machine Learning Research*, 12(Sep):2539–2561, 2011. URL http://www.jmlr.org/papers/v12/shervashidze11a.html.

Smith, T. F. and Waterman, M. S. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147(1):195–197, 1981. doi:10.1016/0022-2836(81)90087-5.

Socher, R., Perelygin, A., Wu, J. Y., Chuang, J., Manning, C. D., Ng, A. Y., and Potts, C. Recursive deep models for semantic compositionality over a sentiment treebank. In Baldwin, T. and Korhonen, A. (eds.), *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP 2013)*, pp. 1631–1642, 2013. URL http://www.aclweb.org/anthology/D/D13/D13-1170.pdf.

Sutskever, I., Vinyals, O., and Le, Q. V. Sequence to sequence learning with neural networks. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. D., and Weinberger, K. Q. (eds.), *Proceedings of the 27th Conference on Advances in Neural Information Processing Systems (NIPS 2014)*, pp. 3104–3112. Curran Associates, Inc., 2014. URL https://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks.

Yanardag, P. and Vishwanathan, S. Deep graph kernels. In *Proceedings of the 21th International Conference on Knowledge Discovery and Data Mining (KDD 2015)*, pp. 1365–1374, New York, NY, USA, 2015. ACM. doi:10.1145/2783258.2783417.

Zeng, Z., Tung, A. K. H., Wang, J., Feng, J., and Zhou, L. Comparing stars: On approximating graph edit distance. *Proceedings of the VLDB Endowment*, 2(1):25–36, 2009. doi:10.14778/1687627.1687631.

Zhang, K. and Shasha, D. Simple fast algorithms for the editing distance between trees and related problems. *SIAM Journal on Computing*, 18(6):1245–1262, 1989. doi:10.1137/0218082.

Zhang, K., Statman, R., and Shasha, D. On the editing distance between unordered labeled trees. *Information Processing Letters*, 42(3):133–139, 1992. doi:10.1016/0020-0190(92)90136-J.