Learning Discriminative Tree Edit Similarities for Linear Classification — Application to Melody Recognition

Aurélien Bellet[a], José F. Bernabeu[b,*], Amaury Habrard[c], Marc Sebban[c]

[a] *Magnet Team, INRIA Lille Nord Europe, 59650 Villeneuve dAscq, France*
[b] *Dept. Lenguajes y Sistemas Informaticos, Universidad de Alicante, Spain*
[c] *Univ Lyon, UJM-Saint-Etienne, CNRS, Institut d'Optique Graduate School, Laboratoire Hubert Curien UMR 5516, F-42023, Saint-Etienne, France*

Abstract

Similarity functions are a fundamental component of many learning algorithms. When dealing with string or tree-structured data, measures based on the edit distance are widely used, and there exist a few methods for learning them from data. In this context, we recently proposed GESL (Bellet et al., 2012), an approach to *string edit similarity* learning based on loss minimization which offers theoretical guarantees as to the generalization ability and discriminative power of the learned similarities. In this paper, we argue that GESL, which has been originally dedicated to deal with strings, can be extended to *trees* and lead to powerful and competitive similarities. We illustrate this claim on a music recognition task, namely melody classification, where each piece is represented as a tree modeling its structure as well as rhythm and pitch information. The results show that GESL outperforms standard as well as probabilistically-learned edit distances, and that it is able to describe consistently the underlying melodic similarity model.

Keywords: edit distance, convex optimization, tree-structured data, melody recognition

*Corresponding author
Email addresses: aurelien.bellet@inria.fr (Aurélien Bellet), jfbernabeu@dlsi.ua.es (José F. Bernabeu), amaury.habrard@univ-st-etienne.fr (Amaury Habrard), marc.sebban@univ-st-etienne.fr (Marc Sebban)

1. Introduction

Many applications require to deal with hierarchical information represented as trees such as Web/XML data processing in web information retrieval, syntactic analysis in natural language processing, structured databases, structured representations in images or symbolic representations in music information retrieval. In these areas, pattern recognition algorithms are used to perform classification or clustering tasks. Many of them rely on a notion of distance or similarity such as the k-Nearest-Neighbors, k-means or kernel-based classifiers. In the context of tree-structured data, tree edit distance (TED) (Bille, 2005) is a widely-used measure that extends the concept of string edit distance. TED is defined as the least costly set of basic operations to transform one tree into another. Typically, these edit operations include substitution, insertion or deletion of tree nodes.

TED-based approaches generally make use of an *a priori* fixed edit cost matrix. In most cases, the absence of explicit background knowledge about the problem at hand leads to a matrix that takes the form of a uniform distribution over the edit operations. However, it is worth noting that using a distance tailored to a specific task is key to improve the success of the recognition algorithm.

Following this idea, some research effort has been devoted to learning the edit costs from training data. Most of them use an EM-like algorithm: Neuhaus and Bunke (2004) learn a (more general) graph edit distance. Zigoris et al. (2006) train a parameterized tree alignment model for extracting fields from HTML search results. Boyer et al. (2007) optimize in the form of a stochastic transducer a stochastic version of the tree edit distance presented in Zhang and Shasha (1989). Bernard et al. (2008) extend the latter work to the Selkow algorithm (Selkow, 1977), while Dalvi et al. (2009) propose a solution to improve the two previous approaches using a more complex conditional transducer. Mehdad (2009) proposes a rather different method based on the Particle Swarm Optimization. In a recent paper, continuing the line of work initiated in Boyer et al. (2007), Emms (2012) proposes an EM cost adaptation algorithm for the Viterbi variant.

From this short survey, we see that relatively few methods have been proposed in the literature to deal with tree edit distance learning. Moreover, their most striking limitation is essentially due to algorithmic constraints coming from the underlying complexity of evaluating the tree edit distance

combined with the use of the EM algorithm that requires to iteratively recompute the updated distances. In this probabilistic context, tree edit distance learning quickly becomes intractable and the need of a new class of learning methods has emerged.

In a previous paper (Bellet et al., 2012), we proposed a new framework for learning **string** edit similarities addressing the main drawbacks described above. We presented a new algorithm, called *GESL* for *Good Edit Similarity Learning*, tailored to binary classification and originally designed for string-structured data. In GESL, we suggest to optimize the costs of the edit operations involved in the classical string edit distance (Levenshtein, 1966) to fulfill a goodness criterion according to the theory of (ϵ, γ, τ)-good similarity functions presented in Balcan et al. (2008). This method has been shown to be well suited to string edit similarity learning (Bellet et al., 2012), providing a procedure that (i) is very efficient (edit scripts need to be computed only once), (ii) has strong theoretical guarantees in the form of a generalization bound (suitable for problems with large alphabet), and (iii) can be used to learn low-error classifiers for the task at hand.

In this paper, our claim is that GESL deserves to be extended to tree-structured data to overcome the limitations of state of the art tree edit distance learning methods. In this context, our contribution is three-fold: first, we show that GESL can be easily adapted to trees without increasing the algorithmic complexity of the learning process and that we can derive generalization guarantees of the learned tree edit similarity. Second, we provide experimental evidences that GESL is well suited to deal with music recognition tasks. Indeed, musical pieces encoded in symbolic format (e.g. MIDI or MusicXML) can be efficiently represented by structured data such as trees (Rizo, 2010) (see Fig.3). We show on the PASCAL database that GESL outperforms state of the art tree edit distance algorithms. Our last contribution aims at showing that the models learned by GESL can provide a posteriori valuable knowledge from a music information retrieval point of view. Indeed, the learned edit similarity directly works on the structure of the data and thus on the melody structure itself. Therefore, it can give some insights about, e.g., the most common changes that make the tune still recognizable.

The rest of this paper is organized as follows. Section 2 introduces the background on trees and edit distance. Section 3 presents the adaptation of GESL to trees in the context of two variants of TED proposed in Selkow

(1977) and Zhang and Shasha (1989). Our experimental study is described in Section 5. We conclude in Section 6.

2. Background

We begin this part by making a little recap on the definition of a tree.

Definition 1. *Let $t = v(t_1, \ldots, t_n)$ be a tree where v is a node and $t_1, \ldots, t_n$ is an ordered sequence of (sub)trees. v is called the root of t. $t_1, \ldots, t_n$ are called the children of v. A node with no child ($n = 0$) is a leaf.*

The size of the tree t is denoted by $|t|$ and corresponds to the number of nodes constituting the tree. In the following, we assume each node to be labeled by a symbol σ coming from a set of labels Σ. We denote by T_Σ the set of trees that can be built from Σ.

In this paper, we stand in a classical supervised learning setting for binary classification. We assume we are given some labeled examples $z = (t, \ell)$ drawn from an unknown distribution P over $T_\Sigma \times \{-1, 1\}$, where $\{-1, 1\}$ are the binary labels of the examples. $K : T_\Sigma \times T_\Sigma \to [-1, 1]$ defines a pairwise similarity function over T_Σ, K is symmetric if for all $t, t' \in T_\Sigma$, $K(t, t') = K(t', t)$. A binary classifier h is a function $h : T_\Sigma \to \{-1, 1\}$.

The binary classifiers we consider rely on a similarity function that is based on the notion of edit distance over trees. The tree edit distance (TED) (Bille, 2005) can be seen as an extension or a generalization of the string edit distance (also known as the Levenshtein distance) and is based on three elementary edit operations: substitution, insertion or deletion of nodes. It is defined as follows.

Definition 2. *Let Σ be a set of labels and $\$$ be the empty symbol, T_Σ being the set of all possible trees (including the empty tree $\$$). The tree edit distance (TED) $e_T(t, t')$ between two trees t and t' is the minimum number of edit operations to change t into t'. The set corresponding to the minimum number of operations allowing one to change t into t' is called the optimal script.*

Like in the case of strings, TED can be computed using dynamic programming: when considering two trees of sizes m and n, where $m \leq n$, the best known algorithms for this problem, due to Zhang and Shasha (1989) and Klein (1998), have an $O(n^3 \log n)$ time complexity and an $O(mn)$ space complexity. In these approaches, when a tree node is deleted, all its children

are connected to its father. A less costly variant of these algorithms has been proposed by Selkow (1977), where deleting a node leads to the removal of the entire (sub)tree rooted at that node (see Figure 1 for an illustration). The insertion of a (sub)tree is also allowed and requires the iterative insertion of its nodes. Such a distance is relevant to specific applications. For instance, it would make no sense to delete a `<UL>` tag (i.e., a node) of an unordered list in a HTML document without removing the `<LI>` items (i.e., the subtree). In this case, the tree edit distance can be computed by dynamic programming in quadratic time.

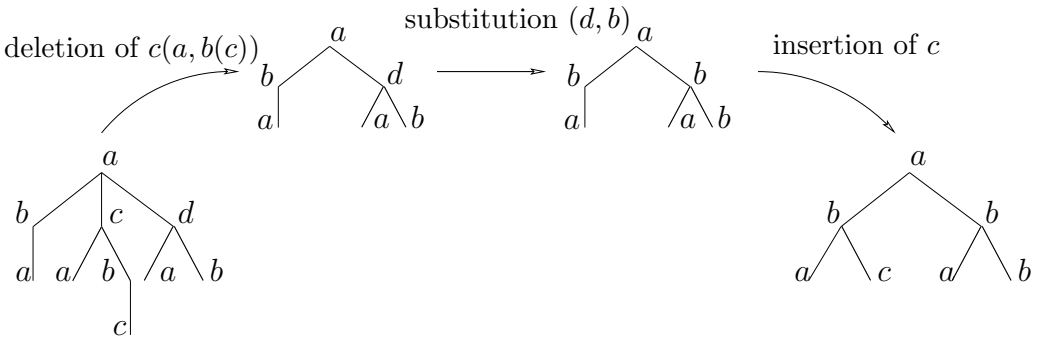


Figure 1: An optimal edit script according to Selkow tree edit distance algorithm Selkow (1977). The script begins by the deletion of the subtree $c(a, b(c))$, then it considers the substitution of two labeled nodes (d, b) and terminates with the insertion of the one node subtree c. Following this script between the two trees, the non-zero entries of the corresponding matrix # of Section 3.1 are: $\#_{(a,\$)}$, $\#_{(b,\$)}$, $\#_{(c,\$)}$, $\#_{(\$,c)}$, and $\#_{(d,b)}$; all these entries receive the value 1 since they are exactly used once in the script.

In the next section, we present the framework for learning good tree edit similarities.

3. Good Edit Similarity Learning for tree-structured data

We now present our procedure for learning good tree edit similarity. This procedure follows the principle of the algorithm GESL introduced by Bellet et al. (2012) which is based on three main steps. In this section, we detail each step of this process. First, we define a simple edit similarity based on edit scripts. Second, we optimize this similarity according to the framework of (ϵ, γ, τ)-good similarity of Balcan et al. (2008). This step leads to

good similarities allowing one to build, during a last step, powerful linear separators.

3.1. Tree Edit Script Based Similarity

Let C be a positive cost matrix of size $(|\Sigma| + 1) \times (|\Sigma| + 1)$ defining the possible edit operations over nodes of trees. $C_{i,j}$ gives the cost of the operation changing the symbol c_i into c_j, c_i and $c_j \in \Sigma \cup \{\$\}$. Let $\#(t, t')$ be a $(|\Sigma| + 1) \times (|\Sigma| + 1)$ matrix whose elements $\#_{i,j}(t, t')$ correspond to the number of times each edit operation consisting in changing c_i into c_j is used to turn t into t' in the optimal script obtained from any tree edit distance variant (Zhang and Shasha, 1989; Selkow, 1977). In other words, we consider the script corresponding to the minimum set of tree edit operations allowing one to change one tree into another one with respect to the tree edit distance considered (see Figure 1 for an illustration).

From these two matrices, the following edit function is then considered:

$$e_C(t, t') = \sum_{0 \leq i,j \leq \Sigma} C_{i,j} * \#_{i,j}(t, t').$$

An important point here is that the computation of e_C does not depend on the optimal script with respect to C. In other words, e_C is evaluated by considering the operations used in the optimal script, weighted by the custom costs C. Therefore, since the edit script defined by $\#(t, t')$ is fixed, $e_C(t, t')$ is nothing more than a linear function of the edit costs and can be optimized directly. Bellet et al. (2012) define then the edit similarity function as:

$$K_C(t, t') = 2e^{-e_C(t,t')} - 1.$$

The use of the exponential allows K_C to introduce non-linear information in the model and to belong to $[-1, 1]$ as required by Balcan et al.'s framework presented in the next section.

3.2. Learning Good Similarity Functions

In this section, we present our algorithm for learning good edit similarities. We first define the notion of good similarity. It relies on a relaxed version of Balcan et al.'s framework (Balcan and Blum, 2006; Balcan et al., 2008).

Definition 3 (Relaxed good tree edit similarity)**.** *A similarity function* $K : T_\Sigma \times T_\Sigma \to [-1, 1]$ *is an* (ϵ, γ, τ)*-good similarity function for a learning problem* P *if there exists a (random) indicator function* $R(t)$ *defining a (probabilistic) set of "reasonable trees" such that the following conditions hold:*

1. *A* $1 - \epsilon$ *probability mass of examples* (t, ℓ) *satisfy*

$$\mathbf{E}_{(t,\ell)}\left[\mathbf{E}_{(t',\ell')}\left[[1 - \ell\ell' K_C(t, t')/\gamma]_+ \,|R(t')\right]\right] \leq \epsilon'. \tag{1}$$

   where $[1 - c]_+ = \max(0, 1 - c)$ *corresponds to the hinge loss.*
2. $\Pr_{t'}[R(t')] \geq \tau$.

The first condition is essentially requiring that *on average a* $1 - \epsilon$ *proportion of trees* t *are* 2γ *more similar to random reasonable trees of the same class than to random reasonable trees of the opposite class*[1] and the second condition that *at least a* τ *proportion of the trees are reasonable*. Note that the reasonable instances are either provided by the application at hand (in this case, they can be viewed as prototypes) or are automatically selected from a set of so-called landmarks by solving a simple linear problem which is essentially a 1-norm SVM problem (see Balcan et al. (2008) for details - in this case, they can be viewed as support vectors).

Definition 3 is very interesting in two respects. First, it does not impose strong constraints on the form of the similarity functions considered. Second, these conditions are sufficient to learn a good linear space, i.e., to induce a linear separator $\boldsymbol{\alpha}$ in the space of the similarities to the reasonable trees: $h(\cdot) = \sum_{(t',\ell')\sim P|R(t')} \ell' K(\cdot, t')$, as illustrated in Figure 2 (see Balcan et al. (2008) for a formal proof).

[1]The original definition of Balcan et al. (2006; 2008) requires the property 1 to be true only on average over the reasonable examples. However, the use of the exponential in our similarity implies a non-convex formulation of the optimization problem considered for learning the costs. As a consequence, we propose to relax the original formulation by considering the surrogate version presented in Definition 3.

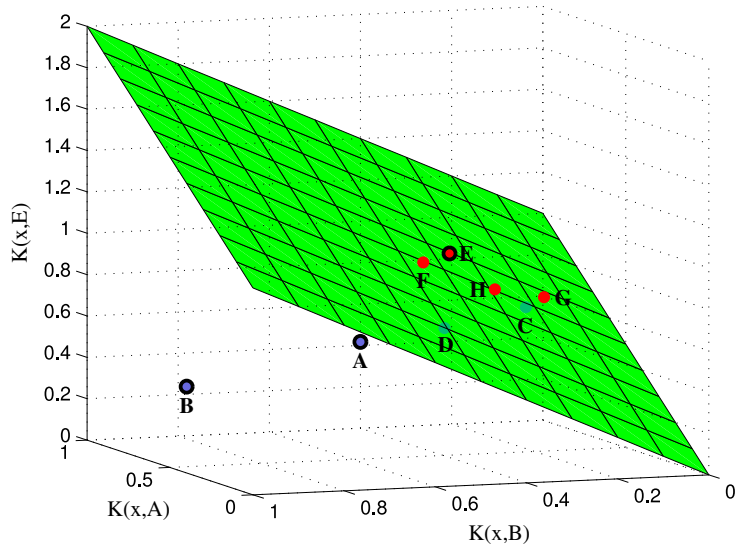


Figure 2: Linear separator $\boldsymbol{\alpha}$ learned from a training set of trees $\{(A,+1),(B,+1),(C,+1),(D,+1),(E,-1),(F,-1),(G,-1),(H,-1)\}$ thanks to an (ϵ,γ,τ)-good similarity function. A, B and E are reasonable trees. The good linear separator is learned in the 3D-space of the similarities to that reasonable points with respect to K.

The algorithm GESL has for objective to optimize the goodness of the similarity K_C so as to satisfy Condition 1 over all the pairs (z_i, z_j) of a training set of N_T labeled examples $T = \{z_i = (t_i, \ell_i)\}_{i=1}^{N_T}$. The optimization problem GESL for learning the costs C, which takes the form of a sparse convex program, is expressed as follows:

$$
\begin{array}{lll}
(GESL) & \min\limits_{C,B_1,B_2} & \frac{1}{N_T^2} \sum\limits_{1\leq i,j\leq N_T} V(C,z_i,z_j) \quad + \quad \beta\|C\|_{\mathcal{F}}^2 \\
& s.t. & V(C,z_i,z_j) = \left\{ \begin{array}{l} [B1 - e_C(t_i,t_j)]_+ \text{ if } \ell_i \neq \ell_j \\ [e_C(t_i,t_j) - B2]_+ \text{ if } \ell_i = \ell_j \end{array} \right. \\
& & B_1 \geq -\log(\tfrac{1}{2}), \quad 0 \leq B_2 \leq -\log(\tfrac{1}{2}), \quad B_1 - B_2 = \eta_\gamma \\
& & C_{i,j} \geq 0, \quad 0 \leq i,j \leq A,
\end{array}
$$

where $\beta \geq 0$ is a regularization parameter on edit costs, $\|\cdot\|_{\mathcal{F}}$ denotes the Frobenius norm and $\eta_\gamma \geq 0$ a parameter corresponding to the desired "margin". The relationship between the margin γ and η_γ is given by $\gamma = \frac{e^{\eta_\gamma}-1}{e^{\eta_\gamma}+1}$. The loss V essentially penalizes the violations of the goodness defined by

Equation 1.

Then, we optimize the costs of these edit operations by solving this optimization problem. This allows us, once again, to avoid using a costly iterative procedure: we only have to compute the tree edit script between two trees once, which dramatically reduces the algorithmic complexity of the learning algorithm. Moreover, the procedure of GESL allows one to learn good tree edit similarities with respect to Definition 3 implying the ability to learn good linear separators for tree structured data which is presented in the next section.

3.3. Classifier Learning: Automatic Selection of the Reasonable Trees

As mentioned previously, reasonable points play an important role because the optimal linear classifier, named $\boldsymbol{\alpha}$, is built in the space of the similarities to that points. In areas where some ground truth is available, these reasonable points can be given by an expert. In this case, provided that we have access to a similarity that satisfies the goodness property of Definition 3, the optimal classifier is directly obtained by using the similarities to that points. In complex domains such as in music recognition, setting the set of reasonable points is a tricky task. To overcome this problem, a strategy, as suggested by Balcan et al. (2008), consists in using d (unlabeled) examples as landmarks, d_l labeled examples and in solving the following simple linear problem which is essentially a 1-norm SVM problem:

$$\min_{\boldsymbol{\alpha}} \sum_{i=1}^{d_l} \left[1 - \sum_{j=1}^{d} \alpha_j \ell_i K(t_i, t'_j) \right]_+ + \lambda \|\boldsymbol{\alpha}\|_1, \tag{2}$$

where $\|\cdot\|_1$ corresponds to the L_1-norm.
An important feature of (2) is the L_1-regularization on $\boldsymbol{\alpha}$, which induces sparsity. Therefore, it allows automatic selection of reasonable points controlling the sparsity of the solution: the larger λ, the sparser $\boldsymbol{\alpha}$. Moreover, by solving (2), we directly get a good (with generalization guarantees) linear separator in the space of the similarities to the reasonable points. In the experimental section, we will show that these points provide valuable information.

We present in the next section the formalism used for tree representation of melodies and we use the properties of GESL to derive theoretical

guarantees in this context.

4. Tree-structured representation of melodies and theoretical guarantees

We use a tree-based symbolic representation of melodies as suggested by previous works for melody classification (Habrard et al., 2008; Rizo et al., 2009). The representation uses rhythm for defining the tree structure and pitch information for node labeling. To represent the note pitches in a monophonic melody M, we use symbols σ from a pitch representation alphabet Σ_p. In this paper, the interval from the tonic of the song modulo 12 is used as a pitch descriptor and the symbol '$-$' represents rests ($\Sigma_p = \{p \in \mathcal{N} \mid 0 \leq p \leq 11\} \cup \{-\}$). In the tree representation, each melody bar is represented by a tree, $t \in T_{\Sigma_p}$. Bars are coded by separated trees and then they are linked to a common root (see Fig. 3 for an illustration). The level of a node in the tree determines the duration it represents. During the tree construction, nodes are created top-down when needed and guided by the meter, to reach the appropriate leaf level corresponding to the associated note duration. The corresponding leaf node is labelled with the pitch representation symbol, $\sigma \in \Sigma_p$ (for the details see Rizo et al., 2009; Rizo, 2010). Once the tree has been built, a bottom-up propagation of the pitch labels is performed to label all the internal nodes, using melodic analysis rules (Illescas et al., 2007).

This tree representation of melodies defines a particular class of data for which we can derive a consistency theorem tailored to our method. For any bar of duration d_{bar}, the duration of a note encoded by a node v in a tree is defined according to its depth in the tree: $\text{nodedur}(v) = \frac{1}{2^{depth(v)}} \times d_{bar}$, where $\text{depth}(v)$ denotes the depth of the node v. In order to limit the depth of the tree, a minimum note duration is generally fixed (Rizo, 2010) and defined by a constant Min_d. Symmetrically, the depth of a node v encoding a note of duration d is given by: $\text{depth}(v) = \log_2(\frac{d_{bar}}{d})$. If Max_d is the maximum duration of a bar, we have for any bar of duration d_{bar} and any note of duration d encoded by a node v,

$$\text{depth}(v) = \log_2(\frac{d_{bar}}{d}) \leq \log_2 \left(\frac{\text{Max}_d}{\text{Min}_d} \right).$$

The depth of the tree represents the number of nodes on the path from

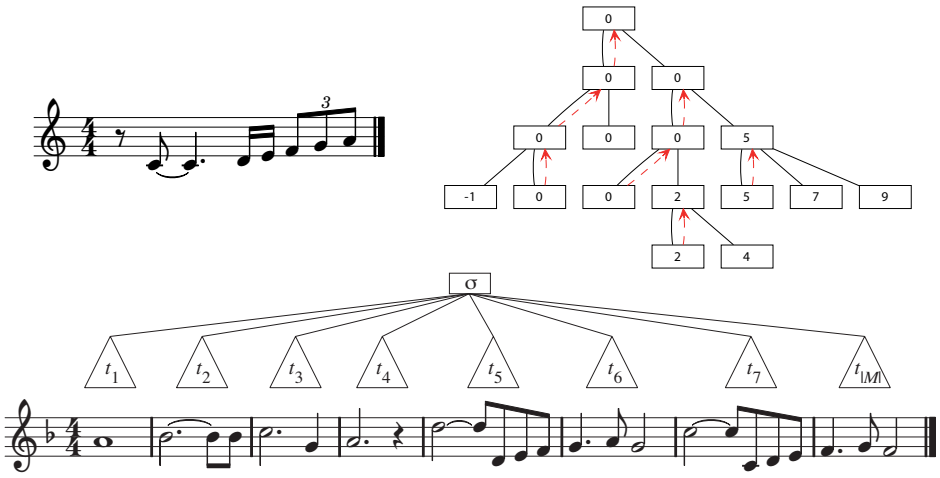


Figure 3: Above: tree representation of a one-bar melody with an example of how pitch labels are propagated. Below: tree representation of a melody (each bar is represented as a tree).

the root to the node and needed to encode the note in the tree; $\log_2\left(\frac{\text{Max}_d}{\text{Min}_d}\right)$ represents the maximum node depth and thus the maximum number of nodes needed to encode a note in a tree. Now, if N_{maxb} is the maximum number of bars in a melody, $\frac{\text{Max}_d}{\text{Min}_d}$ gives the maximum number of notes in a bar and then we can bound the maximum size of a tree encoding a melody by:

$$N_{maxb} \times \frac{\text{Max}_d}{\text{Min}_d} \times \log_2\left(\frac{\text{Max}_d}{\text{Min}_d}\right).$$

Then, we have the following generalization bound for the GESL tree edit version. Indeed, one can relate the true loss of the learned matrix C_T, $L(C_T) = \mathbb{E}_{(z,z')}[V(C_T, z, z')]$, with its empirical expected loss $L_T(C_T)$ (over the pairs of the set T) with the following theorem.

Theorem 1. *Let N_{maxb}, Max_d and Min_d be respectively the maximum number of bars, the maximum duration of bars and the minimum duration of a note in the melodies. Let T be a sample of N_T trees drawn i.i.d. from an unknown distribution P, C_T the matrix learned by GESL (using all the possible pairs from T), with probability $1-\delta$ we have the following bound for $L(C_T)$:*

$$L(C_T) \leq L_T(C_T) + 2\frac{\kappa}{N_T} + \left(2\kappa + (3B_\gamma\left(\frac{2N_{notes}}{\sqrt{\beta B_\gamma}} + 3\right)\right)\sqrt{\frac{\ln(2/\delta)}{2N_T}},$$

with $N_{notes} = N_{maxb} \times \frac{Max_d}{Min_d} \times \log_2\left(\frac{Max_d}{Min_d}\right)$, $\kappa = \frac{6N_{notes}^2}{\beta}$ *and* $B_\gamma = \max(\eta_\gamma, -\log(1/2))$.

Proof. The proof follows the same principle as in Bellet et al. (2012), and uses the fact that the maximum number of nodes in a tree is bounded by N_{notes}. □

The previous result highlights important aspects of our method. First, it has a convergence rate in $O(\sqrt{1/N_T})$ which a classical rate for concentration bounds. Another point is that the method is independent from the alphabet size - *i.e.* from the pitch representation - which indicates that the method should scale well with large alphabets. Lastly, by considering the framework of Section 3.2, it provides a consistency justification of our approach by showing that the goodness of the similarity is ensured, which implies the existence of a good linear classifier in the space defined by the reasonable trees.

One drawback of this result is the dependency on a given maximum tree size N_{notes}. Fortunately, this constraint can be relaxed by assuming the trees to be generated from probabilistic models where the probability to generate trees with size larger than an integer k is bounded, like probabilistic finite tree automata. Indeed, it can be shown that with probability $1 - \delta$ for any sample of N_T trees, we have for any tree t belonging to this sample,

$$|t| < \frac{\log(N_T U/\delta)}{\log(1/\rho)},$$

with $U > 0$ and $0 < \rho < 1$ some constants (Denis et al., 2008). By combining this result with the previous theorem, one can obtain a bound independent from the maximum tree size, see (Bellet et al., 2012) for the technical details.

5. Experiments in melody recognition

In this section, we provide an experimental evaluation of GESL for tree edit distance learning. Our objective is two-fold: first, we show that this approach allows us to learn good similarities leading to very accurate linear classifiers. Second, we illustrate how the notion of reasonable points can be used in this framework to extract semantic information from the learned similarity.

5.1. Pascal database

To perform our experiments, we focus on a standard task in music information retrieval, namely *melody classification*. We evaluate GESL using the *Pascal* database[2] consisting of a set of 420 monophonic 8-12 bar incipits of 20 worldwide well known tunes of different musical genres. For each song, a canonical version was created using a score editor. Then it was synthesized and the audio files were given to three amateur and two professional musicians (a classical and a jazz player). Each musician listened to the songs (to identify the part of the tune to be played) and they played them on MIDI controllers (four keyboards, one guitar), real-time sequencing them 20 times with different embellishments and without avoiding performance errors. This way, for each of the 20 original scores, 21 different sequences were built. The task consists in identifying a target melody using a set of different variations played by musicians for the 20 different tunes corresponding to the target classes. In this context, we use the symbolic representation with trees, as presented in the previous section, using rhythm for defining the tree structure and pitch information for node labeling.

5.2. Experimental setup

A three-fold cross-validation scheme is carried out to perform the experiments, where 2/3 of the database is used for training and 1/3 for testing. We compare five edit similarities: (i) the *Selkow* tree edit distance (Selkow, 1977), which constitutes the baseline, (ii) a stochastic version of the Selkow distance, Selkow$_{sto}$ (Bernard et al., 2008) learned from an EM-like algorithm based on the software SEDiL (Boyer et al., 2008), (iii) the Zhang-Shasha tree edit distance (Zhang and Shasha, 1989), (iv) K_{Selkow}, learned by GESL using the Selkow edit scripts, and (v) $K_{Zhang-Shasha}$, learned by GESL using the Zhang-Shasha edit scripts. Note that it was not possible to evaluate the stochastic version of the Zhang-Shasha distance learned with SEDiL because the learning procedure was intractable, that confirms our claim mentioned in the introduction.

Let us remind that GESL is specifically dedicated to optimize similarities that are then efficiently used in the learning of a linear separator (by solving problem (2) of Section 3.3). Therefore, in a first series of experiments, the melodies are classified using this linear separator, learned from

[2] This name comes from the Pascal European Network of Excellence (Pattern Analysis, Statistical Modeling and Computational Learning).

the five similarity measures (Selkow, Selkow$_{sto}$, Zhang-Shasha, K_{Selkow} and $K_{Zhang-Shasha}$)[3]. Even though GESL has not been designed for nearest-neighbor-like algorithms, we perform a second series of experiments where the melodies are classified with the 1-nearest neighbor rule by directly making use of the five similarities.

To deal with the multi-class setting, we use a standard one versus one procedure: a binary linear classifier is learned for each pair of classes (C_j, C_k), that is, the binary linear classifier $h_{(j,k)}$ is learned considering the instances of the class C_j (resp. C_k) as positive (resp. negative) data. Therefore, we learn $\binom{n}{2}$ (binomial coefficient) binary classifiers, where n is the number of classes (in our problem 20 classes and 190 binary classifiers). Each classifier determines if a melody belongs to the class C_j or to the class C_k. Then, we apply a majority vote strategy to decide the final classification for each melody.

5.3. Results and edit cost analysis

Results are reported in Table 1. We can make the following remarks.

- Using a linear classifier, the similarities learned by GESL allow significant improvements of the classification accuracy. $K_{Zhang-shasha}$ achieves the best overall performance (95.0%) improving the results mentioned in Habrard et al. (2008) for the same dataset. These results confirm the interest to optimize with GESL a tree edit distance (according to the goodness criterion of Definition 3) which is then used to learn a simple linear separator in the space of the similarities to the reasonable points.

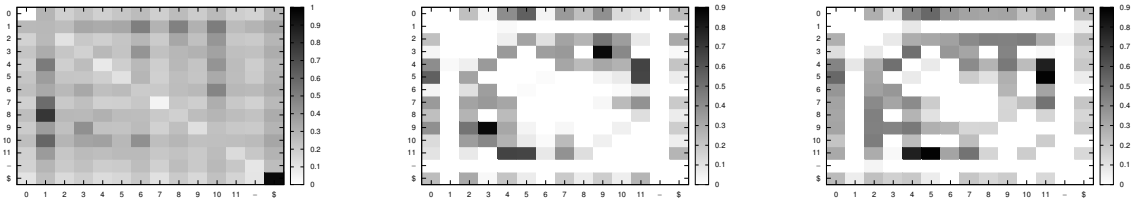
- In the second series of experiments, we can note that the tree edit distance $Zhang - Shasha$ allows us to reach the best result (however, smaller than 95.0%). Even though K_{Selkow} and $K_{Zhang-Shasha}$ have not been optimized for a nearest neighbor classifier, it is worth noting that the tree edit cost learning procedure of GESL remains competitive with the two others.

In order to analyze the understandability of the inferred models, let us compare the edit cost matrices learned on the one hand with GESL for both

[3]Note that the distances are used as a similarities and normalized to stand in the interval $[-1, 1]$.

Table 1: Success rates (%) and standard deviation obtained from the five edit similarities in 1NN and linear classifications on the *Pascal* corpus.

Approach	Success rate Linear Classifier	Success rate 1-NN
Selkow	90.2 ± 1.2	90.5 ± 0.9
Selkow$_{sto}$	88.6 ± 0.9	91.5 ± 0.4
Zhang-Shasha	91.9 ± 0.9	93.6 ± 0.5
K_{Selkow}	93.1 ± 0.5	90.5 ± 0.2
$K_{Zhang-Shasha}$	95.0 ± 0.7	90.2 ± 1.4



(a) SEDiL (Selkow). (b) GESL (Selkow). (c) GESL (Zhang and Shasha).

Figure 4: Learned edit costs by GESL and SEDiL algorithms.

Selkow and Zhang & Shahsha tree edit distances, and on the other hand with SEDiL (only for Selkow, because Zhang and Shasha's is intractable). In order to represent graphically these matrices, they have been averaged and normalized over the three folds. Our aim here is to see if the learned edit costs are able to model the changes and mistakes that can be found in the corpus. Mistakes correspond in general to note elisions, grace notes insertions, the substitution of the intended note by another at a distance of a semitone or a tone, and note duration and onset changes.

The matrix learned by SEDiL is shown in Fig. 4(a), while matrices learned by GESL are presented in Fig. 4(b) and (c). We can see that in the edit cost matrix learned with SEDiL from an EM-like algorithm, small costs (represented by bright cells) are mainly located in the diagonal of the matrix. This means that substitutions of a pitch by itself are favored. However, this reduces the ability of the model to adapt to small variations caused by the musicians. In this case, the model will prefer to delete and insert a pitch

(the last row and last column contain indeed brighter cells too) to fit the canonical version. We claim that this does not model well the reality. Unlike matrix (a), the edit cost matrices (b) and (c) learned with GESL allow some distortions. We can see that the smallest costs are found not only in the substitution of a pitch by itself (the diagonal) but also in substitution of a pitch by a close pitch of about one or two semitones. This corresponds to the substitution of the intended note by another at a distance of a semitone or a tone caused by mistakes of the musicians when they played the songs.

Moreover, in both plots issued from GESL, we can identify two bright row and column that correspond to the operation of replacing a symbol σ by the rest symbol '$-$' ($(\sigma \to -)$ or $(- \to \sigma)$). It is worth noting that this perfectly models the insertions or deletions of rests by the musicians when they want to produce some effect to the melody feeling.

5.4. Reasonable points analysis

Finally, we provide a brief analysis of the reasonable points automatically selected by solving problem (2) of Section 3.3. Intuitively, these representative points should be some discriminative prototypes the classifier is based on. To make the analysis easier, we consider a restricted task where the goal is to predict if a melody belongs to the classical music genre or if it is a children's song. These two styles correspond to two classes allowing us to turn the task into a binary classification problem. The examples of the classical genre correspond to songs belonging to the classes '`Toccata and fugue`', '`Avemaria`', '`Ode to joy`', '`Bolero`' and '`Lohengrin, wedding march`' of the *Pascal* corpus. The children's class is formed by the songs coming from '`Alouette`', '`Oh! Susanna`', '`Happy birthday`', '`Twinkle twinkle little star`' and '`Jingle bells`' classes. From these data, we build a learning and a test sample such that there are 7 instances for each *Pascal* class in the test and 14 in the training set. Therefore, each class has 35 examples in test and 70 for training. Table 2 shows the number of reasonable points for each class. We can see that (beyond a high accuracy) in the classical class there are 4 reasonable points that belong to '`Toccata and fugue`' and 5 to '`Lohengrin, wedding march`'. These two songs seem thus to be good representatives for that class. In the same way, the '`Oh! Susanna`' and '`Happy birthday`' songs provide many (9 out of 15) discriminative prototypes for the children class. All in all, these four songs provide about 62% (18 out of 29) of the reasonable points while they represent only 40% of the training songs. Said differently, the

Table 2: Number of reasonable points used to learn a linear classifier between classical and children's music.

Song	# reasonable points
Toccata and fugue	4
Avemaria	1
Ode to joy	2
Bolero	2
Lohengrin, wedding march	5
Alouette	2
Oh! Susanna	4
Happy birthday	5
Twinkle twinkle little star	2
Jingle bells	2
Success (%)	94.29

predicted label of a new song will be mainly defined by its similarity to those 4 songs.

6. Conclusions

In this paper, we investigated a new framework for learning tree edit distances thanks to a convex optimization problem based on the framework of GESL, originally developed for strings. We have shown that this framework is adequately tailored to tree edit distance allowing us to have strong theoretical justification while having an efficient procedure to deal with complex distance, such as the Zhang-Shasha one, which is a clear advantage in comparison of EM-based methods that quickly become intractable.

We experimentally showed on a music recognition task that this framework is able to build very accurate classifiers improving state-of-the-art results for this problem. This experiment was done in a multi-class setting showing that GESL can also deal with this context. Moreover, we illustrated that the produced models can be used to provide a semantic analysis of the knowledge learned from the data.

A perspective of this work would be to study other definitions of tree edit similarities. Indeed, GESL is based on a linear combination of the edit script

operations plugged in an exponential but we could imagine other strategies tailored to the application at hand. Another interesting future work would be to adapt the similarity learning procedure directly to the multi-class setting instead of binary classification. Finally, the efficiency of this tree edit distance learning framework (both in terms of accuracy and running time) opens the door to an extensive use of tree edit distance in larger-scale applications such as natural language processing or XML data classification.

Acknowledgments

This work was supported by a grant from CPER Nord-Pas de Calais/FEDER DATA Advanced data science and technologies 2015-2020 and the Spanish Ministerio de Economía y Competitividad project TIMuL (No. TIN2013–48152–C2–1–R supported by UE FEDER funds).

References

Balcan, M.-F., Blum, A., 2006. On a Theory of Learning with Similarity Functions. In: Proceedings of the 23rd International Conference on Machine Learning (ICML). pp. 73–80.

Balcan, M.-F., Blum, A., Srebro, N., 2008. Improved Guarantees for Learning via Similarity Functions. In: Proceedings of the 21st Annual Conference on Learning Theory (COLT). pp. 287–298.

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

Bernard, M., Boyer, L., Habrard, A., Sebban, M., 2008. Learning probabilistic models of tree edit distance. Pattern Recognition 41 (8), 2611–2629.

Bille, P., 2005. A survey on tree edit distance and related problems. Theoretical Computer Science 337 (1-3), 217–239.

Boyer, L., Esposito, Y., Habrard, A., Oncina, J., Sebban, M., 2008. Sedil: Software for edit distance learning. In: Proceeding of the European Conference on Machine Learning and Lnowledge Discovery in Databases (ECML/PKDD), Part II. Vol. 5212 of LNCS. pp. 672–677, available from `http://labh-curien.univ-st-etienne.fr/SEDiL/`.

Boyer, L., Habrard, A., Sebban, M., 2007. Learning metrics between tree structured data: Application to image recognition. In: Proceedings of the European Conference on Machine Learning (ECML). Vol. 4701 of LNCS. pp. 54–66.

Dalvi, N., Bohannon, P., Sha, F., 2009. Robust web extraction: an approach based on a probabilistic tree-edit model. In: Proceedings of the ACM SIGMOD International Conference on Management of Data. pp. 335–348.

Denis, F., Gilbert, E., Habrard, A., Ouardi, F., Tommasi, M., 2008. Relevant representations for the inference of rational stochastic tree languages. In: Proceedings of the 9th International Colloquium on Grammatical Inference (ICGI). Vol. 5278 of LNCS. Springer, pp. 57–70.

Emms, M., 2012. On stochastic tree distances and their training via expectation-maximisation. In: Proceedings of the 1st International Conference on Pattern Recognition Applications and Methods - ICPRAM 2012 - Volume 1. pp. 144–153.

Habrard, A., Iñesta, J. M., Rizo, D., Sebban, M., 2008. Melody recognition with learned edit distances. In: Proceedings of the IAPR International Workshop Structural and Syntactic Pattern Recognition (SSPR). Vol. 5342 of LNCS. pp. 86–96.

Illescas, P. R., Rizo, D., Iñesta, J. M., 2007. Harmonic, melodic, and functional automatic analysis. In: Proceedings of the International Computer Music Conference (ICMC). Vol. I. pp. 165–168.

Klein, P., 1998. Computing the edit-distance between unrooted ordered trees. In: Proceedings of the 6th European Symposium on Algorithms (ESA). Vol. 1461 of LNCS. Springer, pp. 91–102.

Levenshtein, V. I., 1966. Binary codes capable of correcting deletions, insertions and reversals. Soviet Physics-Doklandy 6, 707–710.

Mehdad, Y., 2009. Automatic cost estimation for tree edit distance using particle swarm optimization. In: Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the AFNLP (ACL-IJCNLP). pp. 289–292.

Neuhaus, M., Bunke, H., 2004. A probabilistic approach to learning costs for graph edit distance. In: Proceedings of the 17th International Conference on Pattern Recognition (ICPR). IEEE, pp. 389–393.

Rizo, D., November 2010. Symbolic music comparison with tree data structures. Ph.D. thesis, Universidad de Alicante.

Rizo, D., Lemström, K., Iñesta, J. M., 2009. Tree representation in combined polyphonic music comparison. Proceeding of International Symposium on Computer Music Modeling and Retrieval, Genesis of Meaning in Sound and Music (CMMR) 5493, 177–195.

Selkow, S., 1977. The tree-to-tree editing problem. Information Processing Letters 6 (6), 184–186.

Zhang, K., Shasha, D., 1989. Simple fast algorithms for the editing distance between trees and related problems. SIAM Journal of Computing 18 (6), 1245–1262.

Zigoris, P., Eads, D., Zhang, Y., 2006. Unsupervised learning of tree alignment models for information extraction. In: ICDM Workshops. pp. 45–49.