2009

Learning nonsparse kernels by self-organizing maps for structured data

Fabio Aiolli
University of Padova

Giovanni Da San Martino
University of Padova

Markus Hagenbuchner
University of Wollongong, markus@uow.edu.au

Alessandro Sperduti
University of Padova

Learning nonsparse kernels by self-organizing maps for structured data

Abstract

The development of neural network (NN) models able to encode structured input, and the more recent definition of kernels for structures, makes it possible to directly apply machine learning approaches to generic structured data. However, the effectiveness of a kernel can depend on its sparsity with respect to a specific data set. In fact, the accuracy of a kernel method typically reduces as the kernel sparsity increases. The sparsity problem is particularly common in structured domains involving discrete variables which may take on many different values. In this paper, we explore this issue on two well-known kernels for trees, and propose to face it by recurring to self-organizing maps (SOMs) for structures. Specifically, we show that a suitable combination of the two approaches, obtained by defining a new class of kernels based on the activation map of a SOM for structures, can be effective in avoiding the sparsity problem and results in a system that can be significantly more accurate for categorization tasks on structured data. The effectiveness of the proposed approach is demonstrated experimentally on two relatively large corpora of XML formatted data and a data set of user sessions extracted from Website logs.

Disciplines

Physical Sciences and Mathematics

Publication Details

Aiolli, F., Martino, G. Da San., Hagenbuchner, M. & Sperduti, A. (2009). Learning nonsparse kernels by self-organizing maps for structured data. IEEE Transactions on Neural Networks, 20 (12), 1938-1949.

Learning Nonsparse Kernels by Self-Organizing Maps for Structured Data

Fabio Aiolli, Giovanni Da San Martino, Markus Hagenbuchner, *Member, IEEE*, and Alessandro Sperduti, *Senior Member, IEEE*

***Abstract*—The development of neural network (NN) models able to encode structured input, and the more recent definition of kernels for structures, makes it possible to directly apply machine learning approaches to generic structured data. However, the effectiveness of a kernel can depend on its sparsity with respect to a specific data set. In fact, the accuracy of a kernel method typically reduces as the kernel sparsity increases. The sparsity problem is particularly common in structured domains involving discrete variables which may take on many different values. In this paper, we explore this issue on two well-known kernels for trees, and propose to face it by recurring to self-organizing maps (SOMs) for structures. Specifically, we show that a suitable combination of the two approaches, obtained by defining a new class of kernels based on the activation map of a SOM for structures, can be effective in avoiding the sparsity problem and results in a system that can be significantly more accurate for categorization tasks on structured data. The effectiveness of the proposed approach is demonstrated experimentally on two relatively large corpora of XML formatted data and a data set of user sessions extracted from website logs.**

***Index Terms*—Kernel methods, self-organizing maps (SOMs), structured data, tree kernels.**

I. Introduction

THE self-organizing map (SOM) is a well-known unsupervised machine learning approach which has been successfully applied to data mining tasks where the clustering of high-dimensional data is required [1]. Recently, its scope has been extended to the treatment of structured data [2]–[6]. In fact, it is recognized that data from some learning domains are more appropriately represented by structures rather than by vectors. For example, in chemistry, a molecule is appropriately represented by an undirected graph where the vertices and the arcs represent the atoms and the bonds, respectively. Numerical labels can be assigned to vertices or arcs, providing descriptions of properties of the associated element. For example, a data label attached to a vertex could contain information about the atomic weight, degree of ionization, and the type of atom, whereas a label attached to an arc could indicate the valence of the bond. Another example of a domain where a structured representation of the data is a natural choice is the world wide web, which can be represented, for instance, by a single graph. In this case, vertices represent web documents which are interconnected by directed arcs representing the hyperlinks pointing from one web document to another. As an additional example, each web document can itself be represented as a tree through a structural decomposition of the document (e.g., by following its HTML structure).

Generally speaking, structured representations provide very versatile means to represent information. In fact, there are several applications where a structured data representation is essential for a given task. For example, search engines for the world wide web require knowledge on the structure of the web in order to function effectively [7], or in molecular chemistry, the representation of molecules should reflect the structural composition of atomic elements, as it is related to molecule's properties.

Traditional methods in machine learning deal with vectorial representations. Thus, some processing/encoding is required to map the structured information into a vector. The processing is clearly task specific and needs to be suitably designed for any new task. Moreover, in order to keep all meaningful structural information, an exponential number of structural features is typically needed, thus leading to serious computational issues. On the other hand, dropping many of these structural features may lead to a loss of potentially significant information for a given task.

Recent developments in machine learning have produced methods capable of processing structured information directly, such as kernel methods [8] and artificial neural networks (NNs) [2], [9]. Kernel methods work by *implicitly* mapping the structures into a (possibly infinite-dimensional) vector space through a positive–semidefinite similarity function, i.e., the kernel function. By defining a suitable kernel function for a given domain, any kernel method can be applied to any type of data. For example, the definition of kernels for structured domains (see [10] for an overview, and [11] for applications to computational biology) allows a better, and computationally feasible, exploitation of structural information. However, one problem with standard kernels for structures, such as the well-known subtree (ST) kernel [12] and subset tree (SST) kernel [13], already recognized in natural language processing applications, is that, in the case of large structures and many symbols, the feature space implicitly defined by these kernels is very sparse [14]. It is clear that any kernel machine cannot work effectively when used with these kernels.

Manuscript received May 09, 2008; revised August 08, 2009; accepted August 29, 2009. First published October 20, 2009; current version published December 04, 2009. The work of M. Hagenbuchner and A. Sperduti was supported by the Australian Research Council under Grant DP0774168.

F. Aiolli, G. Da San Martino, and A. Sperduti are with the Department of Pure and Applied Mathematics, University of Padova, Padova 35121, Italy (e-mail: aiolli@math.unipd.it; dasan@math.unipd.it; sperduti@math.unipd.it).

M. Hagenbuchner is with the School of Computer Science and Software Engineering, University of Wollongong, Wollongong, N.S.W. 2522, Australia (e-mail: markus@uow.edu.au).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNN.2009.2033473

On the other hand, the SOM-SD [2], an unsupervised NN for structures, is able to compress information by performing a projection of labeled trees and their STs onto a q–dimensional discrete lattice, preserving as much as possible the "topology" of the data in the original space. In other words, SOM-SDs map tree structured information in such a way that similar trees are mapped onto nearby areas in the lattice to form clusters [2]. This allows other procedures to use the projection instead of the associated tree as a basis for computation. For example, inexact graph matching is known to be computational expensive when executed in the domain of graphs. The same task is, instead, much simplified when applied to mapped structures. While SOM-SD can be used for classification tasks by recurring to a 1-NN rule [2], the resulting classifier can have less generalization capabilities than that of a kernel method.

In this paper, we propose a family of kernels defined on the top of a SOM-SD. This allows us to exploit the SOM-SD compression and "topology" preserving capabilities to define novel kernels with reduced sparsity. Specifically, since the similarities in the input space are preserved in the mapping space, it can be stated that the activation of the SOM-SD for a given tree is a representation of the tree in a (possibly compressed) fixed-dimensional space. The proposed family of kernels is then defined on the projection rather than on the original structures. On data where standard tree kernels are sparse, the possibility to perform an inexact tree matching through SOM-SD helps to define less sparse tree kernels. The experimental results obtained on a classification task involving a relatively large corpus of XML formatted data provide evidence that, when sparsity on the data is present, the use of the proposed kernels is able to improve the overall categorization performance over each individual method, i.e., either support vector machine (SVM) [8] using tree kernels or SOM-SDs equipped with a 1-NN classification rule. This demonstrates that neither tree kernels nor SOM-SDs are always able to retain all the relevant information for classification. The approach proposed in this paper can thus be considered as a first step in the direction of defining approaches able to fully exploit the structural information needed to solve learning tasks defined on structured domains. Further experiments on the same data set are devoted to demonstrate that the proposed method is quite robust with respect to the choice of SOM-SD (hyper)parameters. Moreover, empirical results, obtained by experiments involving data sets with different degrees of sparsity for one of the most popular tree kernels, seem to support the claim that the proposed method allows to construct nonsparse kernels and that, in general, sparser kernels tend to have worse performances. Finally, experimental results obtained on a data set involving a different domain, i.e., trees representing user sessions extracted from website logs, confirm the usefulness of the proposed approach.

This paper is organized as follows. Kernels for structured domains, a technique for assessing the quality of a kernel (kernel alignment), and the SOM-SD are introduced inSections II-A–II-C, respectively. A kernel based on SOM-SD is proposed in Section III and experimental findings are presented and discussed in Section IV. A general discussion on the proposed kernel can be found in Section V and conclusions are drawn in Section VI.

This paper is a significantly extended version of a previously published conference paper [15].

II. Background

This section introduces the main concepts which are used throughout the paper. Specifically, two kernels for trees (ST and SST) are described. Then, we describe the basic concepts underpinning one of the methods we used to evaluate the goodness of a general kernel in supervised settings, namely, kernel alignment. Finally, the SOM for structured data is described.

A. Kernel for Trees

Kernel algorithms, such as SVM [8], require, in order to be applied to structured information, the definition of a kernel function, i.e., a similarity function, between any two structures. In the following, two of the most popular tree kernels, which will be used as (strong) baseline kernels in this paper, are described. In particular, a kernel for trees can be defined by considering SSTs as proposed in [13] or by considering matching STs as proposed in [12]. The structures considered in this paper are rooted positional trees with a known maximum outdegree. The SST kernel is based on counting matching STs between two input trees. Given an input tree T, let $h_s(T)$ be the number of times a ST s occurs in T (here s ranges over all possible STs of a data set).

The SST kernel can be efficiently calculated by a recursive procedure defined as

$$K(T_1,T_2) = \sum_{t_1\in N_{T_1}}\sum_{t_2\in N_{T_2}}\sum_{s=1}^{m} h_s(t_1)h_s(t_2) = \sum_{t_1\in N_{T_1}}\sum_{t_2\in N_{T_2}} C(t_1,t_2)$$

where N_T is the set of proper STs[1] of the tree T, $C(t_1,t_2) = \sum_{s=1}^{m} h_s(t_1)h_s(t_2)$ encompasses all possible SST matchings, and can be recursively computed according to the following rules.

1) If the productions[2] at t_1 and t_2 are different, then $C(t_1,t_2) = 0$.
2) If the productions at t_1 and t_2 are identical, and t_1 and t_2 have leaf children only (i.e., they are preterminal symbols), then $C(t_1,t_2) = \lambda$.
3) If the productions at t_1 and t_2 are identical, and t_1 and t_2 are not preterminals, then $C(t_1,t_2) = \lambda\prod_{j=1}^{nc(t_1)}(1 + C(ch_j[t_1],ch_j[t_2]))$, where $nc(t)$ is the number of children of a node t and $ch_j[t]$ is the jth child of a node t. Finally, $\lambda \in (0,1)$ is a weighting parameter whose purpose is to reduce the influence of larger STs [13].

On the other side, the ST kernel counts the number of shared proper STs. This value can be obtained by a simple modification of rule 3) of the SST. Specifically, the definition of C becomes $C(t_1,t_2) = \lambda\prod_{j=1}^{nc(t_1)} C(ch_j[t_1],ch_j[t_2])$. Of course, the ST kernel is less expressive than SST since its feature space is smaller. On the other hand, the smaller feature space reduces

[1]A proper ST rooted at a node n is defined as the ST composed by n and all of its descendants.

[2]A production is defined as the label of a node plus the labels associated to its children.

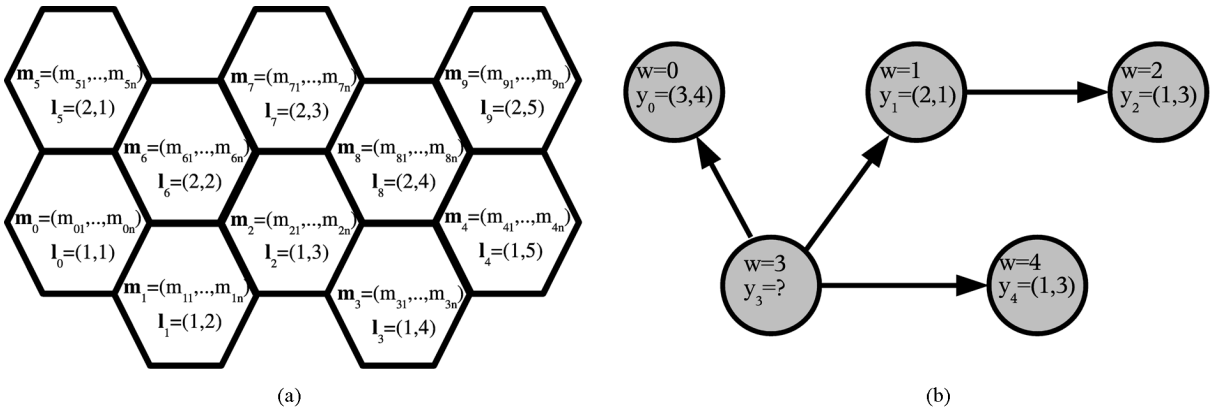


Fig. 1. (a) The 2-D map of size 2×5, and (b) a tree. Each hexagon refers to a neuron. ID, codebook vector $\mathbf{m}$, and coordinate value $\mathbf{l}$ for each neuron are shown. For each vertex of the tree, the vertex number w and coordinate of best matching codebook y are shown.

its computational complexity. In fact, the computational complexity in time of an ST kernel evaluation, exploiting a suitable indexing data structure (see, e.g., [12]), is $O(N_T \log N_T)$, while SST has a worst case computational complexity of $O(N_T^2)$, where $N_T = \max\{n_{T_1}, n_{T_2}\}$ and n_{T_1} and n_{T_2} are the number of nodes of trees T_1 and T_2, respectively.

B. Kernel Alignment

A way to assess the quality of a kernel function in a supervised setting is by using the kernel alignment [16]. Let $S = \{x_1, x_2, \ldots, x_n\}$ be the set of instances comprising a training set and k_1 and k_2 two kernel functions defined on S. The Gram matrix K related to a kernel k with respect to a set S is defined as $K(i,j) = k(x_i, x_j)$. The empirical alignment between k_1 and k_2 is defined as the Frobenius inner product between the corresponding normalized Gram matrices

$$A(k_1, k_2, S) = \frac{\langle K_1, K_2 \rangle_F}{\sqrt{\langle K_1, K_1 \rangle_F \langle K_2, K_2 \rangle_F}}$$

where $\langle K_1, K_2 \rangle_F = \sum_{i,j=1}^{n} K_1(i,j) K_2(i,j)$.

Values of $A()$ range from -1 to 1. The higher the value of $A(k_1, k_2, S)$, the higher the similarity between k_1 and k_2 with respect to S.

The value A can be used to measure how appropriate a kernel k is for a given two-class classification task by aligning k with a matrix Y defined as $Y(i,j) = y_i y_j$, where $y_i = \{-1, +1\}$ is the class associated to an instance x_i. In the case of a multiclass classification task, Y can be defined as $Y(i,j) = 1$ if $y_i = y_j$ and $Y(i,j) = 0$ if $y_i \neq y_j$. It is easy to see that the matrix Y defined according to the two definitions given above is always a positive definite one.

C. The SOM for Data Structures

The SOM-SD extends the SOM approach [1] by allowing to process structured input in the form of trees, where each vertex of a tree can have a label (e.g., a real valued vector). The SOM-SD can be understood as the recursive application of a standard SOM to individual nodes in a tree where the input is properly coded to take into consideration the structural information. As for the standard SOM, the SOM-SD consists of a number of neurons which are organized in a q-dimensional grid (usually $q = 2$). A codebook vector $\mathbf{m}$ is associated with each neuron. The network input for SOM-SD is a vector $\mathbf{x}_v$ representing the information of a vertex v of the input tree and it is built through the concatenation of the data label $\mathbf{v}$ attached to v and the coordinates obtained by the mapping of its child vertices $\mathbf{y}_{ch[v]}$ on the same map, so that $\mathbf{x}_v = [\mathbf{v}, \mathbf{y}_{ch[v]}]$. The vectors are made constant in size by assuming a maximum outdegree, say o, of any vertex in the data set. For vertices with less than o children, padding with a default coordinate, typically the "impossible" coordinate $(-1, -1)$, is applied. As a result, the input dimension is $n = p + 2o$, where p is the dimension of the data label and the constant 2 refers to the number of dimensions of the map which is the most commonly used. The codebook vectors $\mathbf{m} \equiv [\mathbf{m}^{\text{label}}, \mathbf{m}^{ch}]$ are of the same dimension.

Fig. 1 gives an exemplification of a SOM-SD computation. The figure shows a 2-D map of neurons organized in a 2×5 grid. The neurons honor a hexagonal neighborhood relationship with each other. Each neuron is uniquely identified by its coordinate vector $\mathbf{l}$, and has associated a k-dimensional codebook vector $\mathbf{m}$. Also shown in Fig. 1 is a tree with five vertices. The vertices are uniquely determined by an identifier w. For simplicity, no data label is assigned to vertices or arcs. The figure gives a snapshot of the situation during the training of a SOM-SD where some of the vertices have already been mapped. For example, the vertex $w = 1$ is assumed to have been mapped to the neuron at coordinate $(2, 1)$ whereas the mapping of vertex $w = 3$ may have yet to be determined. When processing vertex $w = 3$ the input vector $\mathbf{x}_3$ is formed through the concatenation of the mappings of the offspring of vertex $w = 3$. Hence, $\mathbf{x}_3 = [3, 4, 2, 1, 1, 3]$. The best matching codebook vector is then determined by using some similarity measure such as the Euclidean distance. The mapping $\mathbf{y}_3$ of vertex $w = 3$ is determined by taking the coordinate vector of the winning neuron. This example is indicative of the underlying training procedure which is similar to the standard SOM training algorithm, and can be described in detail as follows.

Consider a q-dimensional lattice of *neurons* representing the display space. Every neuron of the map is associated with an n-dimensional codebook vector $\mathbf{m}_i = (m_{i1}, \ldots, m_{in})^T$, where T transposes the vector. The neurons have a neighborhood relationship (where the hexagonal relationship is the most

common). The SOM-SD is trained by updating the elements of $\mathbf{m}_i$ through a three-step training procedure where vertices in the trees are processed in inverse topological order, from the leaf nodes to the root node as follows.

Step 1) One vertex from any tree in the training set is chosen ensuring that all of its offspring have already been processed. The input vector $\mathbf{x}$ is formed and its similarity to the codebook vectors is computed. When using the Euclidean distance measure, the winning neuron is obtained through

$$r = \arg\min_i \|(\mathbf{x} - \mathbf{m}_i)^T \mathbf{\Lambda}\|$$

where $\mathbf{\Lambda}$ is an $(n \times n)$-dimensional diagonal matrix; its diagonal elements $\lambda_{11} \cdots \lambda_{pp}$ are set to $\mu \in [0 \ldots 1]$ and all remaining diagonal elements are set to $1 - \mu$. The constant μ allows to balance the contribution of the data label component and the coordinate vector component to the Euclidean distance measure.

Step 2) $\mathbf{m}_r$ itself, as well as its topological neighbors, is moved closer to the input vector in the input space. The magnitude of the attraction is governed by the learning rate α and by a neighborhood function $f(\Delta_{ir})$, where Δ_{ir} is the topological distance between $\mathbf{m}_r$ and $\mathbf{m}_i$. The updating algorithm is given by

$$\Delta\mathbf{m}_i = \alpha(t) f(\Delta_{ir})(\mathbf{m}_i - \mathbf{x})$$

where α decreases to 0 with time t, and $f(.)$ is a neighborhood function. The most commonly used neighborhood function is the Gaussian function $f(\Delta_{ir}) = \exp\left(-\|\mathbf{l}_i - \mathbf{l}_r\|^2 / 2\sigma(t)^2\right)$, where the spread σ is called the neighborhood radius which decreases with time t, and $\mathbf{l}_r$ and $\mathbf{l}_i$ are the coordinates of the winning neuron and the ith neuron in the lattice, respectively.

Step 3) The coordinates of the winning neuron are passed on to the parent vertex which in turn updates its vector $\mathbf{y}$ accordingly.

Steps 1)–3) together constitute a single training step and they are repeated for every vertex in the input data set, and for a given number of times. The number of training steps must be fixed prior to the start of the training process because the rate of convergence in the neighborhood function and the learning rate are calculated accordingly.

The SOM-SD requires the processing of data in a strict causal manner (i.e., from the leaf nodes toward the root). It is not possible to process nodes in any other order since otherwise in Step 1) it is possible that not all the states of all neighbors are available. An extension circumventing this problem has been proposed in [3] and [4], by introducing a contextual SOM-SD (CSOM-SD). The CSOM-SD builds on the SOM-SD by adding a new step to the training procedure which takes both ancestors and descendants of a node into account. This is similar in nature to SOM-SD, and hence, the SOM-SD can be reduced to the CSOM-SD accordingly.

A number of parameters need to be set before starting the training of a SOM-SD. These parameters (network dimension, learning rate, number of training iterations) are problem dependent and are also required for the standard SOM. The weight value μ introduced with the SOM-SD is an additional parameter which can be computed while executing the training through a statistical analysis of the size and magnitude of the data labels which typically remain constant during training, and the coordinate vectors which can change during training. In other words, μ can be used to weight the input vector components so as to balance their influence on the distance measure in Step 1). In practice, however, it is often found that a smaller value for μ can help to improve the quality of the mappings. This is due to the recursive nature of the training algorithm and to the fact that a stronger focus on structural information helps to ensure that structural information is passed on more accurately to all causally related vertices when processing a tree.

III. Activation Mask Kernel

In this section, we show how novel tree kernels can be defined on the basis of a SOM-SD. The basic idea is to represent each vertex of a tree by its activation map with respect to a SOM-SD and then define a kernel which computes the dot product in this space. Specifically, with no loss in generality, we assume the neurons to be enumerated according to their position on the map, e.g., the one obtained by a bottom-up left-to-right visit of the map. According to this enumeration, each neuron is associated with a unique index $m \in \{1, \ldots, c\}$, where $c = ab$, and a and b are the horizontal and vertical dimensions of the map, respectively.

Let $\mathrm{ne}_\epsilon[m]$ denote the set of indices of neurons (from the SOM-SD) in the ϵ-neighborhood of the neuron with index m, i.e., $\{m' | \Delta_{m'm} \leq \epsilon\}$, where Δ is the topological distance defined on the 2-D map. An interesting measure of similarity between two STs which takes into account the topology induced by the SOM-SD can be defined as the cardinality of the intersection of the ϵ-neighbors of the neurons mostly activated by these STs. Let $m^*(t_1)$ and $m^*(t_2)$ be the indices of the winning neurons for the root vertices of STs t_1 and t_2, respectively, and

$$I_\epsilon(t_1, t_2) = \mathrm{ne}_\epsilon[m^*(t_1)] \cap \mathrm{ne}_\epsilon[m^*(t_2)] \tag{1}$$

be the set of indices of neurons shared by the two ϵ-neighbors, then a similarity measure between trees T_1 and T_2 can be defined by the function

$$K_\epsilon^{(I)}(T_1, T_2) = \sum_{t_1 \in T_1} \sum_{t_2 \in T_2} |I_\epsilon(t_1, t_2)|. \tag{2}$$

Alternative functions which emphasize the alignment between the activation profiles of two STs can be considered instead of the strict intersection. For example, it is possible to weight differently matching regions depending on their distance from the activated neurons

$$K_\epsilon(T_1, T_2) = \sum_{\substack{t_1 \in T_1, \\ t_2 \in T_2, \\ m \in I_\epsilon(t_1, t_2)}} Q_\epsilon(m, m^*(t_1)) Q_\epsilon(m, m^*(t_2))$$

where $Q_\epsilon(m, m')$ is a decreasing function of the distance $\Delta_{mm'}$ between map neurons with indices m and m', and $Q_\epsilon(m, m') = 0$ when the neurons are not in the ϵ-neighborhood of each other, i.e., $\Delta_{mm'} > \epsilon$. As an example, $Q_\epsilon(m, m')$ can be defined as

$$Q_\epsilon(m, m') = \begin{cases} \epsilon - \eta\Delta_{mm'}, & \text{if } \Delta_{mm'} \leq \epsilon \\ 0, & \text{otherwise} \end{cases} \quad (3)$$

where $0 \leq \eta \leq 1$ is a parameter determining how much the distance influences the neighborhood activation. The similarity function $K_\epsilon(T_1, T_2)$ is a kernel for any choice of $Q_\epsilon(m, m')$. A way to demonstrate it is to show that there exists a function ϕ such that for every T_1, T_2, we have $\phi(T_1) \cdot \phi(T_2) = K_\epsilon(T_1, T_2)$, i.e., K_ϵ can be expressed as a dot product in the feature space induced by ϕ.

Specifically, let us define a feature space of the same dimension as the map produced by the SOM-SD, i.e., c, thus obtaining $\phi(T) \in \mathbb{R}^c$. Given a tree T, we can define the mask $M \in \mathbb{R}^c$ where every element of M is associated to a neuron of the map. Let M be initially set to the null vector. The feature vector is then constructed by computing the best-matching neuron $m^*(t)$ for each ST $t \in T$ when presented to the SOM-SD. Then, the entries of M associated to neighbors within radius ϵ of $m^*(t)$ are updated according to $M_m = M_m + Q_\epsilon(m, m^*(t))$; finally, the feature vector $\phi(T)$ will be defined as $\phi(T) = [M_1, \ldots, M_c]$. At this point, it is easy to check that for a given tree T, $M_m(T) = \sum_{t \in T} Q_\epsilon(m, m^*(t))$ where t runs over all possible STs of T, and we can check that the kernel is obtained by performing the dot product in feature space, i.e.,

$$\begin{aligned} &M(T_1) \cdot M(T_2) \\ &= \sum_m M_m(T_1) M_m(T_2) \\ &= \sum_m \sum_{t_1 \in T_1} Q_\epsilon(m, m^*(t_1)) \cdot \sum_{t_2 \in T_2} Q_\epsilon(m, m^*(t_2)) \\ &= \sum_{t_1 \in T_1, t_2 \in T_2} \sum_m \left(Q_\epsilon(m, m^*(t_1)) \cdot Q_\epsilon(m, m^*(t_2)) \right) \\ &= \sum_{t_1, t_2} \sum_{m \in I_\epsilon(t_1, t_2)} \left(Q_\epsilon(m, m^*(t_1)) \cdot Q_\epsilon(m, m^*(t_2)) \right) \\ &= K_\epsilon(T_1, T_2) \end{aligned}$$

where the third derivation is justified by the fact that $Q_\epsilon(m, m^*(t)) = 0$ whenever m is not in the ϵ-neighborhood of $m^*(t)$.

Since this kernel is built on activation masks of a SOM-SD, we will refer to this approach as the activation mask kernel (AM-kernel).

Fig. 2 gives an example of construction of the feature space representation of three trees according to the AM-kernel. Fig. 2(a) reproduces three simple trees selected from the INEX 2005 data set (see Section IV) and Fig. 2(b) presents their activation masks referring to a 5×4 map. The height of each element of the map corresponds to the value of the activation. Note that the activations of a map depend on the parameter ϵ and that similar trees tend to have similar activation maps.

It should be noted that, unlike ST and SST kernels, the AM kernels do not require the introduction of the λ parameter since the number of involved features only scales linearly with the depth of the trees.

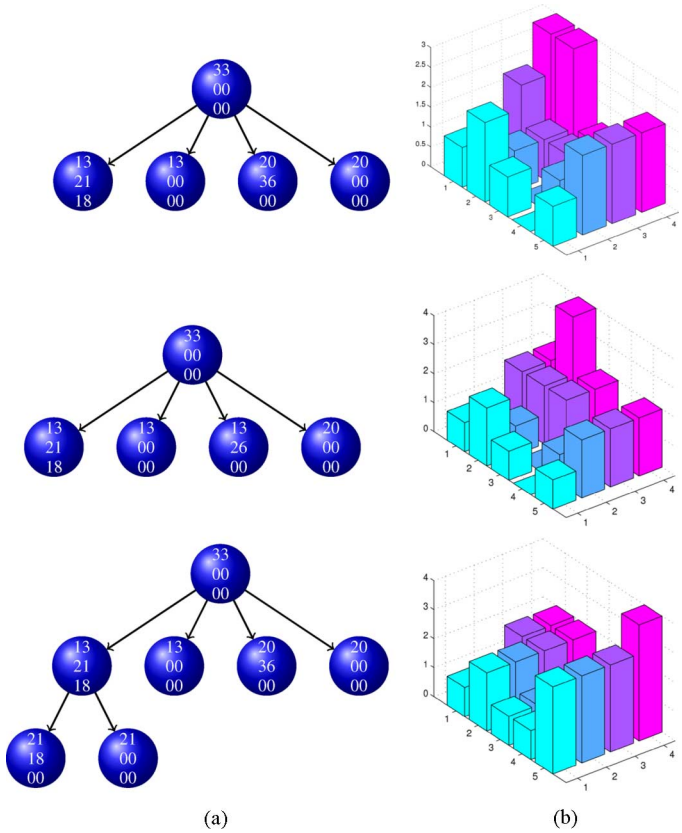


Fig. 2. Example of representation in feature space of three trees according to the AM-kernel for $\epsilon = 1$. (a) Three simple trees and (b) their activation masks referring to a 5×4 map. The height of each element of the map corresponds to the value of the activation.

Note that the proposed approach requires the initial training of a SOM-SD, which however is performed only once, thus not affecting the overall computational complexity of kernel evaluations.

IV. Experiments and Results

Experiments have been performed to evaluate the performances of the different methods, namely, SOM-SD, ST and SST kernels for trees, and the new AM kernel. In particular, we used a relatively large set of XML formatted documents which have been used for the 2005 INEX Competition[3] [17]. Specifically we have used the corpus (m-db-s-0), which consists of 9640 documents containing XML tags only, i.e., no further textual information available. All documents have one out of 11 target values. For the kernel methods, we used 3377 documents as training examples, while 1447 documents constitute the validation set. All remaining documents form the test set. Note that SOM-SD models are trained in an unsupervised fashion which does not benefit from having a validation data set.[4] Hence, the SOM-SD has been trained on the 3377 + 1447

[3]Data can be downloaded from http://xmlmining.lip6.fr

[4]It is a known problem with SOMs that the training parameters need to be set in a trial-and-error approach due to the unsupervised nature of the SOM training algorithm. Several SOMs are typically trained on the same data set using varying training parameters. The best performing SOM (i.e., best with respect to the clustering performance) is then normally chosen.

documents while using target labels solely for evaluating the performance of a trained network.

A tree structure is extracted for each of the documents in the data set by following the general XML structure within the documents. This was a simple process since there were no further attributes associated with the XML tags. This resulted in a data set consisting of 684 191 vertices (STs) with maximum outdegree 6418. Some preprocessing step was performed on the data set in order to reduce its dimensionality. Note that the applied preprocessing step is not strictly necessary, and has simply been done to help reduce the turn around time for the experiments. The preprocessing step removes redundancies in the data set without destroying relevant structural information, and hence, does by no means aim at transforming a graph structure into a simpler (i.e., vectorial) form. Moreover, this preprocessing replicates the procedures taken in [18] to produce the state-of-the-art performance on this data set.

A first step collapses redundant (repeated) sequences of tags within the same level of a structure. For example, the structure:

```
<BB>
  <a></a>
  <b></b>
  <a></a>
  <b></b>
  <a></a>
  <b></b>
</BB>
```

is consolidated to

```
<BB>
  <a></a>
  <b></b>
</B>.
```

A further dimensional reduction has been achieved by collapsing simple substructures which have the property of a data sequence into a single vertex. For example, the sequential structure `<A><b><c></c></b></A>` can be collapsed to `<A><b&c></b&c></A>` and even further to `<A&b&c>`. Note that this later step does not remove any information from the data set. It merely reduces the number of trivial nodes in a data set. The preprocessing step reduced the maximum outdegree to 32, and the total number of vertices to 124 359. This greatly reduces the size of the data set and, in practice, helped to reduce the execution times to hours.[5]

The SOM-SD was shown to produce the best known clustering performance on these data by participating and winning the international competition on the clustering of XML structured documents [18].

SVM-based multiclass classification of the data set was obtained by using the one-against-all methodology. First 11 binary classifiers, each devoted to recognize a single class, were trained. Then, the prediction for the 11-class classification task is given by the class whose associated classifier gets the highest confidence. Best hyperparameters for the different methods were selected comparing classification accuracies on the validation set. The obtained hyperparameters setting was then used to train a multiclass classifier on the union of the training and validation sets. The resulting classifier was then evaluated on the test set.

[5]Training a SOM-SD of size 125 required 48 h on a 2-GHz single core central processing unit (CPU). Without the removal of redundancies in the data set, it would have required weeks to train a SOM of the same size. After training, the application to the test set only required 7 min. Training an SVM with the SST kernel directly on the original data was estimated not feasible.

TABLE I
RESULTS AND STATISTICS OF THE TREE KERNELS

	classification error	Sparsity Index	Alignment
ST Kernel	11.27%	0.5471	0.567966
SST Kernel	11.21%	0.5471	0.462764

As a baseline, the SVM with ST and SST kernels [19] was applied to the same data set. Training times were very dependent on the used values for the hyperparameters, and ranged from few minutes to days of computation.[6] The obtained results, together with the values of the sparsity index and alignment for each kernel, all computed on the test set, are shown in Table I. In this table, the sparsity index is computed as the proportion of example pairs in the instance set S whose kernel value is 0

$$\frac{|\{i, j \in S | K(i,j) = 0\}|}{|S|^2}. \tag{4}$$

The alignment is computed as described in Section II-B. The best accuracy on test set has been obtained by the SST kernel with an error rate of 11.21%. This result was obtained by setting λ (see Section II-A) to 1.1 and setting the C hyperparameter of SVM to 10. Table I shows that both ST and SST kernels are very sparse.

The maps we used for this study were created by the SOM-SD software.[7]

Training a SOM-SD involves the setting of many parameters. Due to SOM-SD training times (e.g., about 12 h for a single large map (110×80) on an AMD Athlon(tm) 64 X2 Dual Core Processor 3800+), and the number of parameters involved, a comprehensive sampling of the parameter space was not feasible. Thus, we decided to run preliminary experiments involving the validation set to sort out the most relevant parameters with respect to the definition of the proposed kernels. The selected parameters were the map size, the number of training iterations, and the value of μ. For these parameters, the following values were used:

- map size: 110×80, 77×56, 55×40;
- number of training iterations: $32, 64, 128$;
- $\mu : 0.05, 0.25, 0.45, 0.65, 0.85$.

For what concerns the other SOM-SD (hyper)parameters, the following values were chosen: $\alpha = 1$, neighborhood radius $= 18$, type of α decrease $=$ sigmoidal, map topology $=$ hexagonal. By combining the above parameters, 45 different maps were built with the aim of spanning as much as possible the space of SOM-SD parameters and therefore getting insights on the dependency of the final results on the maps. After the training phase, each map was evaluated on the test set using a k-NN procedure with $k = 1$. Table II reports the classification performance of each map. Note that the resulting classification error ranges from significantly above the baseline (35.169%) to a very much lower values of the classification error (8.647%). This means that the results are indeed very sensitive to the parameters' choice.

[6]Similar computation times were observed when using the AM kernels.

[7]http://www.artificial-neural.net/software.html

TABLE II
CLASSIFICATION ERROR OF THE SOM-SD MAPS. LOWEST ERROR IS IN BOLD

map	map size	training iterations	μ	test error (%)
1	110 × 80	128	0.05	12.264
2	110 × 80	128	0.25	14.259
3	110 × 80	128	0.45	11.370
4	110 × 80	128	0.65	10.455
5	110 × 80	128	0.85	**8.647**
6	110 × 80	32	0.05	12.617
7	110 × 80	32	0.25	16.587
8	110 × 80	32	0.45	10.912
9	110 × 80	32	0.65	15.423
10	110 × 80	32	0.85	11.661
11	110 × 80	64	0.05	13.282
12	110 × 80	64	0.25	11.723
13	110 × 80	64	0.45	14.238
14	110 × 80	64	0.65	11.245
15	110 × 80	64	0.85	8.855
16	55 × 40	128	0.05	21.638
17	55 × 40	128	0.25	29.079
18	55 × 40	128	0.45	28.081
19	55 × 40	128	0.65	21.326
20	55 × 40	128	0.85	22.511
21	55 × 40	32	0.05	35.169
22	55 × 40	32	0.25	32.488
23	55 × 40	32	0.45	27.770
24	55 × 40	32	0.65	22.137
25	55 × 40	32	0.85	25.629
26	55 × 40	64	0.05	31.844
27	55 × 40	64	0.25	27.541
28	55 × 40	64	0.45	27.749
29	55 × 40	64	0.65	20.121
30	55 × 40	64	0.85	19.144
31	77 × 56	128	0.05	21.451
32	77 × 56	128	0.25	24.215
33	77 × 56	128	0.45	23.488
34	77 × 56	128	0.65	16.296
35	77 × 56	128	0.85	9.956
36	77 × 56	32	0.05	16.234
37	77 × 56	32	0.25	22.282
38	77 × 56	32	0.45	19.310
39	77 × 56	32	0.65	18.624
40	77 × 56	32	0.85	17.585
41	77 × 56	64	0.05	17.169
42	77 × 56	64	0.25	22.864
43	77 × 56	64	0.45	21.721
44	77 × 56	64	0.65	9.457
45	77 × 56	64	0.85	15.735

TABLE III
CLASSIFICATION ERROR OF THE AM KERNEL. THE ϵ THAT WOULD BE SELECTED IN VALIDATION FOR EACH MAP IS IN BOLD. THE MAP THAT WOULD BE SELECTED IN VALIDATION IS UNDERLINED. THE IMPROVEMENT WITH RESPECT TO SST FOR EXAMPLE IS COMPUTED AS $100 \times (\epsilon_{st} - \epsilon_{am})/\epsilon_{st}$

Map	Classification error with Activation Mask kernel						Improvement ratio (%) w.r.t	
	$\epsilon = 0$	$\epsilon = 1$	$\epsilon = 2$	$\epsilon = 3$	$\epsilon = 4$	$\epsilon = 5$	SST	SOM-SD
1	7.046	**6.215**	6.153	6.340	6.672	6.693	44.6	49.3
2	7.358	6.444	**6.381**	6.589	6.485	6.755	43.1	55.2
3	6.610	5.383	**5.238**	5.654	6.132	6.194	53.3	53.9
4	6.319	5.841	5.716	**6.256**	6.568	6.568	44.2	40.2
5	6.464	5.965	6.277	**6.693**	7.254	6.340	40.3	22.6
6	6.028	5.280	**5.196**	5.695	6.236	6.630	53.6	58.8
7	6.610	5.945	5.259	5.425	5.487	**5.508**	50.9	66.8
8	6.506	5.737	5.737	5.903	**5.737**	5.986	48.8	47.4
9	5.965	5.737	**5.924**	6.153	6.194	6.049	47.2	61.6
10	6.028	5.695	**5.945**	6.173	6.194	6.402	47.0	49.0
11	6.485	6.090	**6.028**	6.402	6.485	6.340	46.2	54.6
12	6.818	5.633	**5.342**	5.674	5.425	5.882	52.3	54.4
13	6.444	5.321	**5.404**	5.300	5.550	5.591	51.8	62.0
14	6.111	5.217	5.280	5.425	**5.758**	5.924	48.6	48.8
15	6.672	5.529	**5.882**	6.049	6.153	6.589	47.5	33.6
16	7.462	**6.984**	7.524	7.691	7.545	7.566	37.7	67.7
17	7.732	7.358	**6.838**	7.088	7.005	7.233	39.0	76.5
18	7.878	7.982	**8.273**	8.252	7.753	8.252	26.2	70.5
19	7.026	7.192	6.901	**7.524**	7.275	7.005	32.9	64.7
20	7.358	7.067	7.566	**7.649**	7.462	7.732	31.8	66.0
21	8.501	8.127	8.293	**8.293**	8.584	9.146	26.0	76.4
22	8.834	**9.083**	8.626	8.938	9.104	8.938	19.0	72.0
23	8.397	8.397	8.127	8.293	8.065	**8.190**	26.9	70.5
24	**8.605**	8.792	8.418	8.709	8.584	8.481	23.2	61.1
25	8.273	**8.481**	8.481	8.917	8.709	8.481	24.3	66.9
26	**8.896**	8.543	8.626	8.377	8.356	8.501	20.6	72.1
27	7.628	**7.296**	6.859	7.233	7.379	7.400	34.9	73.5
28	6.880	7.129	6.963	7.337	7.483	**7.441**	33.6	73.2
29	7.649	**7.129**	7.919	7.732	7.608	7.940	36.4	64.6
30	**8.460**	7.899	8.148	8.148	7.857	8.190	24.5	55.8
31	7.753	**7.483**	7.774	8.044	8.896	7.899	33.2	65.1
32	7.171	7.213	**7.483**	8.106	8.086	7.566	33.2	69.1
33	7.067	6.693	**6.547**	6.610	6.527	6.859	41.6	72.1
34	6.381	6.444	6.028	**6.527**	6.901	6.942	41.8	59.9
35	6.444	5.571	**5.716**	6.194	6.360	7.067	49.0	42.6
36	6.319	**5.716**	5.737	6.901	6.028	6.069	49.0	64.8
37	7.587	7.213	**6.818**	6.901	7.192	6.818	39.2	69.4
38	6.256	**6.256**	6.007	6.236	6.527	6.880	44.2	67.6
39	7.400	**6.631**	6.776	7.254	7.795	7.524	40.8	64.4
40	6.735	**6.360**	6.090	6.776	6.693	6.797	43.3	63.8
41	**6.340**	6.132	6.527	6.568	7.504	7.192	43.4	63.1
42	7.026	6.402	6.714	**7.441**	6.776	6.880	33.6	67.5
43	7.026	6.901	7.026	**7.171**	7.192	7.628	36.0	67.0
44	7.129	6.319	**6.194**	6.402	6.818	6.672	44.7	34.5
45	6.277	**6.111**	6.485	6.506	6.444	7.171	45.5	61.2
Mean	7.110	6.687	6.694	6.968	7.041	7.109	39.5	60.5
Std	0.814	1.030	1.044	0.996	0.956	0.893	9.423	11.980

Experiments proceeded by testing the AM-kernels defined in Section III. For each map and for different values of ϵ [see (1)] a kernel was defined. For each kernel, the C parameter of the SVM was selected on the validation set from the following values: $0.001, 0.01, 0.1, 1, 10, 100, 1000$. Finally, with the selected value, an SVM was trained on the union of the training and validation sets and then evaluated on the test set.

The classification error of each AM-kernel is reported in Table III. In the last two columns of the table, we have reported the error improvement (in percent) obtained by the best performing kernel on the validation set (in bold) when varying the ϵ value with respect to SOM-SD and SST performance, respectively. Specifically, let $\epsilon_{sd}, \epsilon_{st}, \epsilon_{am}$ be the SOM-SD, SST, and AM errors, respectively, then the improvement with respect to (w.r.t.) SOM-SD is computed as $100 \times (\epsilon_{sd} - \epsilon_{am})/\epsilon_{sd}$. Similarly, the improvement w.r.t. SST is computed as $100 \times (\epsilon_{st} - \epsilon_{am})/\epsilon_{st}$.

In these experiments, the use of the AM-kernel always improved the classification performance. In some cases, the error is reduced up to 76.5% with respect to SOM-SD and 53.6% with respect to the SVM with SST kernel. The cumulative low standard deviation (see bottom of table) obtained for the AM kernels suggests that the improvement is quite independent with respect to the chosen map. In order to further discuss the dependence of the AM-kernel accuracy from the related map, a graphical comparison among the classification error on the test set of the methodologies involved in the experiment has been made in Fig. 3. The error values of the AM-kernel are related to the ϵ value selected on validation. The plot suggests that the map accuracy influences the error of the AM-kernel. Nevertheless, starting from any map, the error obtained by the AM-kernel is significantly lower than the SST and SOM-SD ones.

According to these experiments, the method used for selecting the parameters is reliable. In fact, if for each map we select the best performance obtained on the test set and we subtract this value from the performance obtained by the value of ϵ selected on the validation set (in bold), the mean value obtained over the set of maps is 0.25 (with standard deviation 0.256). Moreover, selecting both the map and the ϵ in validation

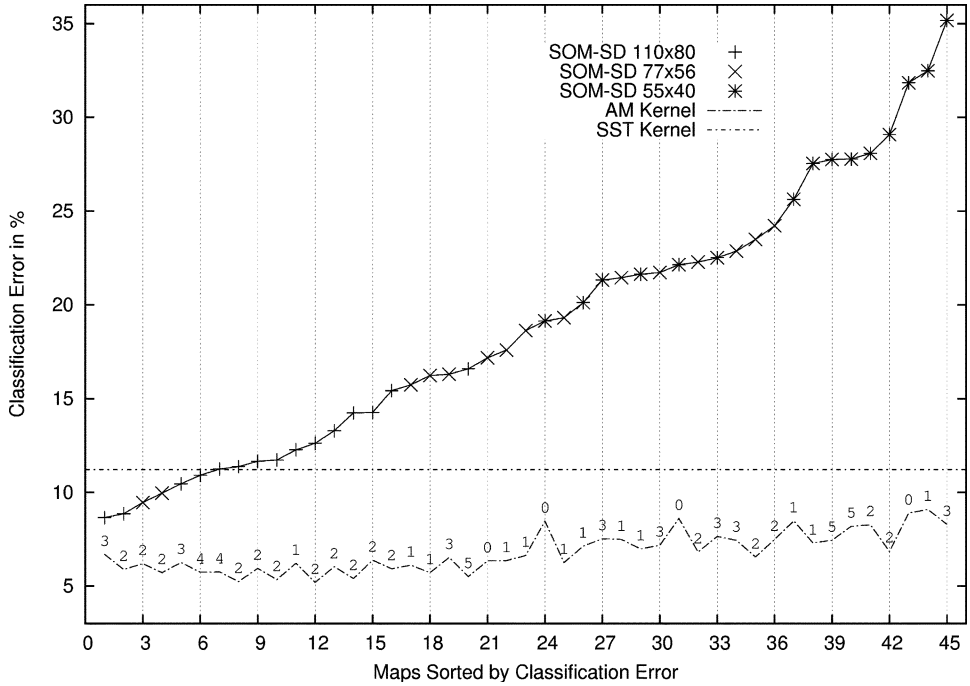


Fig. 3. Comparison between classification error of the different techniques on the INEX 2005 test data set. Maps on the x-axis are sorted by SOM-SD classification error. The error values of the AM-kernel are related to the ϵ value selected on validation (which is reported in correspondence of the map error value).

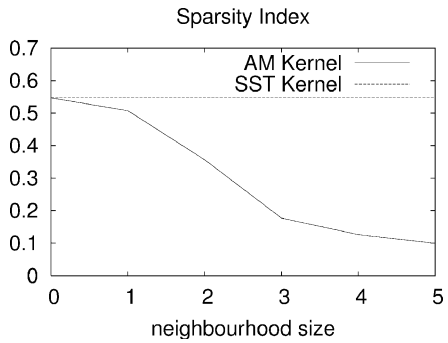


Fig. 4. Statistics on sparsity of AM and SST kernels. Plot refers to map 6.

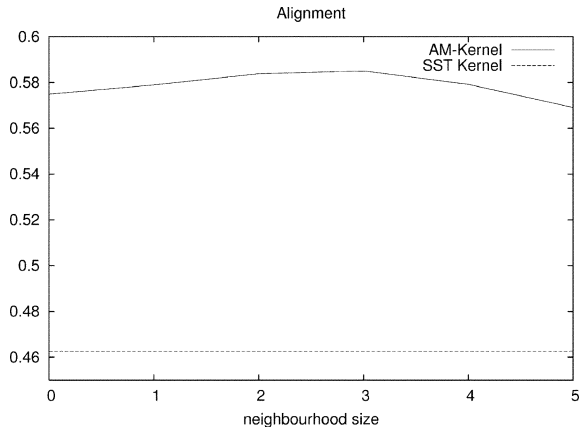


Fig. 5. Alignment of AM and SST kernels. The illustration corresponds to map 6.

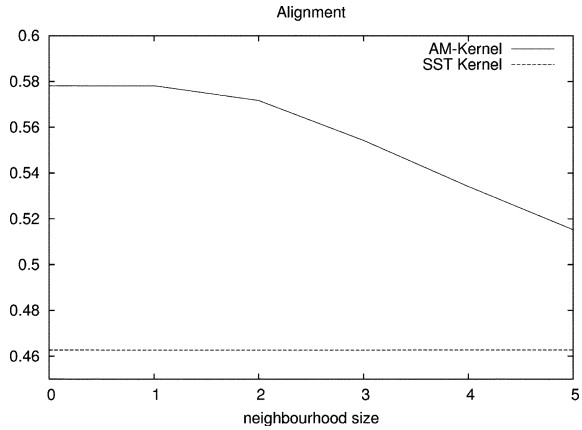


Fig. 6. Alignment of AM and SST kernels. Plot refers to map 19.

would have led us to obtain the best result of Table III (the underlined value).

In order to explain the obtained results, we collected statistics about sparsity and alignment on the test with respect to AM neighborhood size. For what concern sparsity, since the plots for each map show similar behavior, we have plotted the statistics collected for a representative map, i.e., map 6, in Fig. 4.

On the contrary, there are basically two different types of plots for alignment. The first one, exemplified in Fig. 5, is most common for 110×80 and 77×56 maps, while the second one, typical of 55×40 maps, is exemplified in Fig. 6.

The plots clearly show that the AM kernel is far less sparse than the SST one. However, as statistics for the highest values of ϵ point out, low sparsity does not guarantee high accuracy. High ϵ values may overrepresent a structure on the map and thus making it similar to structures which should be considered different for the current task. The different behavior of small maps alignment plots can be explained by the fact that, if the map is not large enough, different structures can be represented by neighboring prototypes. In such cases, any ϵ value larger than 0 may again represent similarly structures that should be different. In other words, the value of ϵ should not be too high with respect

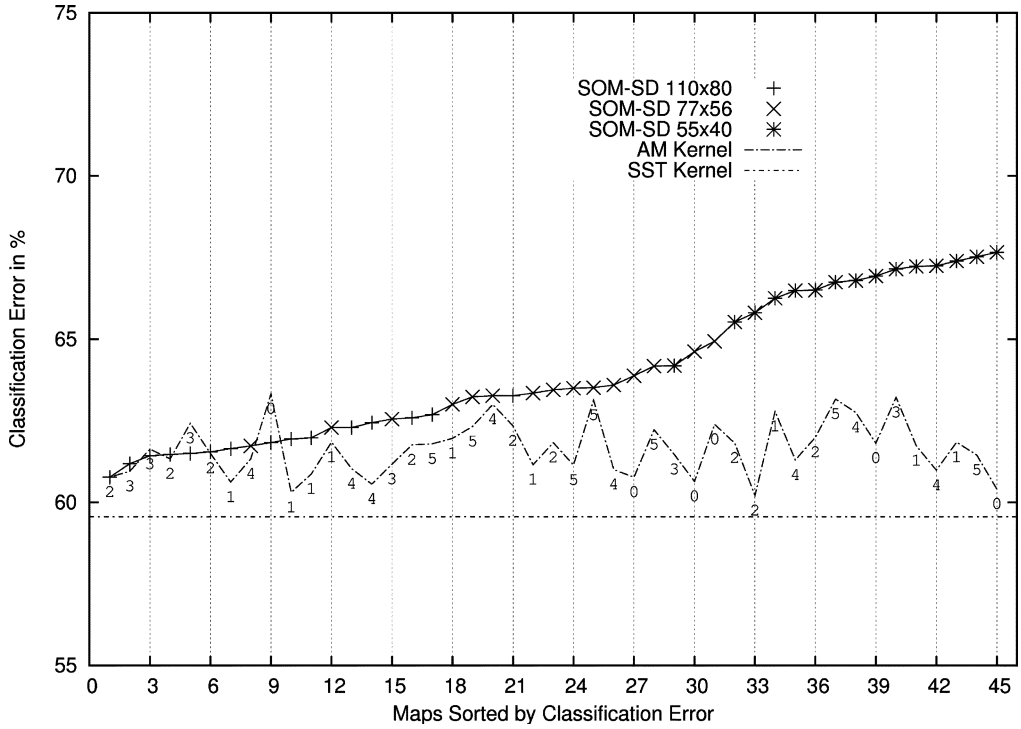


Fig. 7. Comparison between classification error of the different techniques on the INEX 2006 test data set. Maps on the x-axis are sorted by SOM-SD classification error. The error values of the AM-kernel are related to the ϵ value selected on validation (which is reported in correspondence of the map error value).

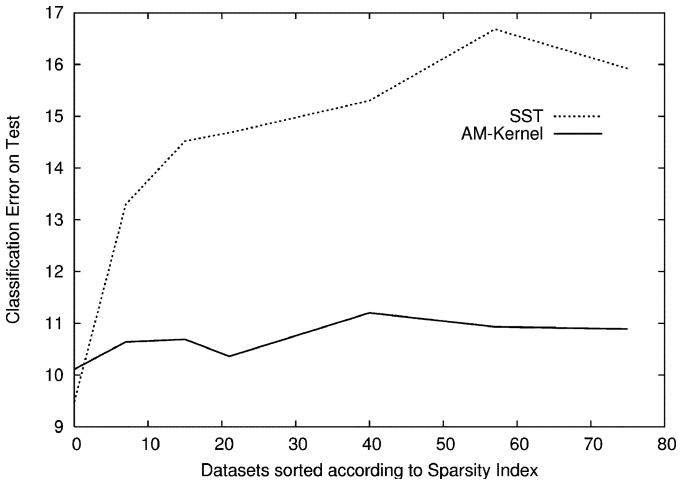


Fig. 8. Classification error of the SST and AM-kernel on various data sets with a different level of sparsity.

to the size of the map. In order to further sustain our claim that the AM kernel is especially useful for tasks in which traditional kernels are sparse, we ran the same set of experiments on a similar, but nonsparse data set involving XML documents which has been used for the 2006 INEX Competition [17]. In this case, the training, validation, and test sets consisted of 4237, 1816, and 6054 documents, respectively. Each document belongs to 1 out of 18 classes. By applying the same methodology as in the previous experiment, the following results were obtained. The sparsity of the SST kernel is 0.0025, its alignment is 0.316, and its classification error is 59.31%. In this case, the mean sparsity of the AM kernels, computed over 45 different maps, ranges from 0.0026 (with standard deviation 0.0000051) to 0.0003 (with standard deviation 0.0003034) when considering the same set of values for the ϵ parameter. The SOM-SD classification error ranges from 67.66% to 60.77% with a mean value of 63.98%. The test error of the AM kernel varies from 64.24% to 58.24% with a mean value of 61.579%.

In Fig. 7, we also report a comparison between classification error of the different techniques on the INEX 2006 test data, using the same visualization method explained for Fig. 3. It can be observed that, while in almost all cases the AM kernel improves on the corresponding SOM-SD map, the SST kernel returns a better performance, thus suggesting that when the SST kernel is not sparse, it can be quite effective.

In order to make an empirical analysis of the relationship between sparsity and classification error, we ran a number of experiments on a set of artificial data sets with different values of sparsity. We considered the two-class problem of discriminating the examples of the INEX 2006 data set belonging to class 8 from the examples belonging to any other class. From this data set, we created seven data sets with the following values of the sparsity index with respect to SST kernel: 0.0025, 0.07, 0.15, 0.21, 0.40, 0.57, 0.75. The data sets were obtained by concatenating to each label a uniformly generated random number. In this way, identical labels have a chance to be transformed into different labels, thus adding sparsity. The number of digits composing the random numbers is constant and thus no different labels can become equal. By varying the range of the random numbers the desired level of sparsity can be obtained. The test has been performed on the following map: size $= 110 \times 80$, training iterations $= 64$, $\mu_1 = 0.45$. Fig. 8 compares the classification error on the test set of the SST and AM-kernel on each data set. The best parameter of the SST, i.e., λ, is selected on validation among the following values $\{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 1.1\}$. The

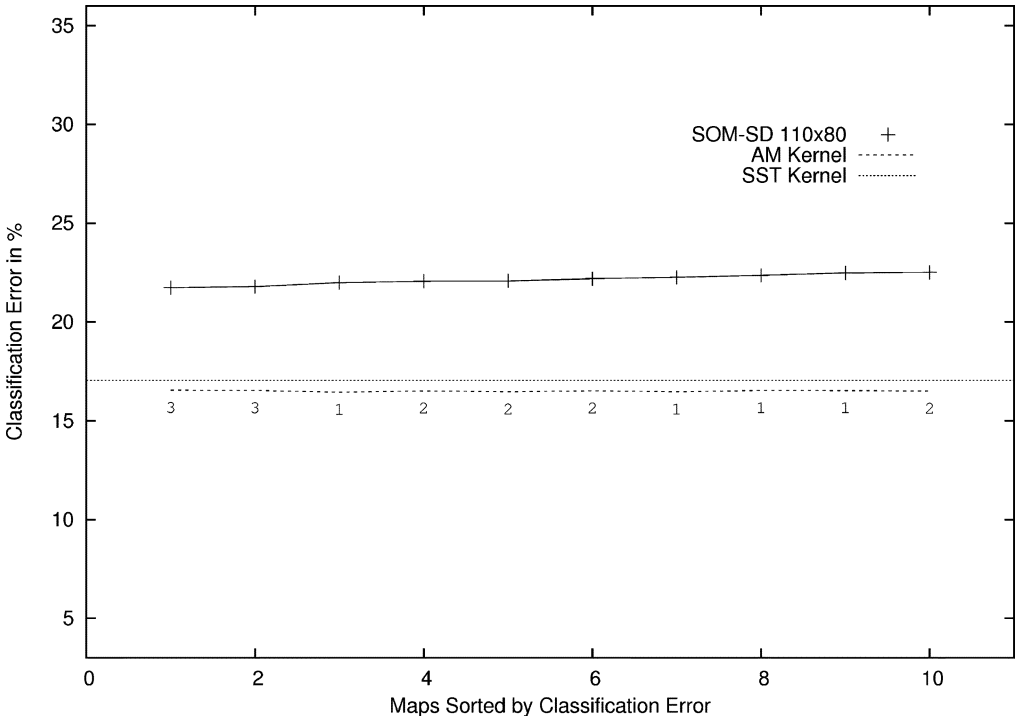


Fig. 9. Comparison between classification error (using threefold cross validation) of the different techniques on the LOGML test data set. Maps on the x-axis are sorted by SOM-SD classification error. The error values of the AM-kernel are related to the ϵ value selected on validation (which is reported in correspondence of the map error value).

best ϵ value of the AM-kernel is selected in validation among the usual values (from 0 to 5). The experiments suggest that the AM-kernel is quite robust to the increase of sparsity.

It has not been formally demonstrated that the low-dimensional representation obtained by a SOM-SD is always the one best representing the topology of the input space. While the demonstration is beyond the scope of this paper, we empirically investigated the usefulness of the SOM-SD learning algorithm by running the same set of experiments on the INEX 2005 data set starting from random, i.e., nontrained, maps as in [18]. Since the number of training iterations was fixed to 0, 15 maps were created. Classification error on the test set of the AM-kernel (results on the validation set are very similar) ranges from 90.36% to 28.21% with a mean value of 51.55% and standard deviation 17.68. Results are most evidently correlated with the parameter μ_1: higher values of μ_1 give lowest classification error. This is not surprising since being the map random the structural information contained in the neurons is useless or misleading. Thus the best results are obtained by giving more importance to label information. The results of the last experiment clearly show the usefulness of the SOM-SD learning algorithm.

In order to assess whether the obtained results were dependent on the specific data sets involving XML documents, we decided to perform additional experiments involving a different type of data. We selected the LOGML data set, which is typically used for data mining research. It consists of user sessions of the website of the Computer Science Department, Rensselaer Polytechnic Institute, Troy, NY,[8] collected over a period of three weeks. Each user session consists of a graph and contains the websites a user visited on the computer science domain. These graphs were transformed to trees by only enabling forward edges starting from the root node. The goal of the classification task is to discriminate between users who come from the "edu" domain and users from another domain, based upon the users browsing behavior. Three data sets are available. They comprise 8074, 7409, and 7628 examples, respectively. The maximum outdegree of the trees is 137. The data sets are unbalanced: for each of them about 76% examples belong to the positive class. The data sets are very sparse with respect to the SST tree kernel: the mean of the three sparsity index values is 0.9595.

Because of the availability of the three data sets, it was natural to compute the classification error of the SST and the AM-Kernel by performing a threefold cross validation considering, in each round, one of the data sets as the test set. Ten maps were trained combining the following parameter values:

- map size: 110×80;
- number of training iterations: $32, 64$;
- $\mu : 0.05, 0.25, 0.45, 0.65, 0.85$.

In Fig. 9, a comparison between classification error of the different techniques on the LOGML test set is plotted, again following the same style of presentation as in Fig. 3. It can be noted that the results obtained for this data set show less variance since they are obtained by a threefold cross-validation approach. Also for this data set, the AM kernel was able to get a significant improvement over SOM-SD, and over SST, although the improvement in this case is smaller.

V. Discussion

Our approach is based on the assumption that the projection of the data on a lower dimensional space should respect the

[8]http://www.cs.rpi.edu

topology of the data in the input space. Under this assumption, the proposed AM-kernel is less sparse than standard tree kernels. The use of the SOM-SD as a dimensionality reduction technique proved to be effective in practice. However, the SOM-SD has a heuristic nature, thus the proposed approach may benefit from the use of more principled dimensionality reduction techniques. In some situations, the characteristics of the SOM-SD may pose challenges to its applicability. As stated in Section II-C, codebook vectors are a representation of the labels in vectorial form. When node labels belong to a symbolic domain, in order to avoid to impose a metric on them, any pair of different labels should be represented by orthogonal vectors. If the label domain is large, the size of the neurons, and thus the time required for training the SOM-SD, greatly increases. Note that kernels for trees can treat efficiently any symbolic domain. On the other hand, note that SOM-SD handles naturally the case in which the labels are represented by numerical vectors. Standard kernels for trees, on the contrary, cannot deal effectively with data from continuous domains. While the formulation of the tree kernels could be easily modified to handle soft matching, the resultant complexity would be quadratic. The AM-kernel requires $O(n \cdot c \cdot n_T)$ to map the nodes in the reduced space and $O(c)$ to compute the kernel. There are also computational issues related to the size of the representation of the children nodes, i.e., $2o$. The parameter o should be fixed *a priori* in accordance to the maximum outdegree of a tree in the data set. However, when there are only a few trees in the data set with very high outdegree, they can be trimmed in order to speed up the learning phase. The computational challenges of the SOM-SD due to the size of the representation of the neurons could be handled by recurring to the kernel SOM approach [20], which uses only an implicit representation of the neurons.

VI. Conclusion

In practical applications involving structured data, the use of a kernel method may not give an optimal performance because of the sparsity of the adopted kernel. This is particularly true for structured data involving discrete variables. In this paper, we have shown an example of this issue related to the SST and ST kernels applied to XML documents represented as trees. We have suggested that such sparsity can be reduced by first learning a similarity function on the trees and then by exploiting it for defining nonsparse kernels. Specifically, we have suggested to learn such similarity function by means of a SOM-SD, which is an unsupervised, dimensionality reduction, and "topology" preserving algorithm for structured data. Then, a family of kernels for trees is defined on the top of the SOM-SD map. The aim of this approach is to learn, in an unsupervised fashion, a kernel which is neither sparse nor uninformative. Experimental results on a relatively large corpus of XML documents, for which both SST and ST kernels exhibit the sparsity problem, have shown that the new kernels are able to improve on the performance of both SOM-SD and the standard tree kernels. This improvement is quite independent from the map used to define the kernel, thus showing that the proposed approach is quite robust. Experimental results obtained on a similar data set, for which, however, SST and ST kernels do not exhibit the sparsity problem, show that there is not a significant improvement in performances. Thus, it seems reasonable to state that the proposed approach is particularly suited when standard tree kernels are sparse. This statement has been experimentally verified by artificially modifying the previous data set in order to progressively increase its sparsity. Finally, experimental results obtained on an additional structured data set involving a different domain confirmed the performance improvements when using AM kernels.

References

[1] T. Kohonen, *Self-Organizing Maps*. Berlin, Germany: Springer-Verlag, 1995.

[2] M. Hagenbuchner, A. Sperduti, and A. Tsoi, "A self-organizing map for adaptive processing of structured data," *IEEE Trans. Neural Netw.*, vol. 14, no. 3, pp. 491–505, May 2003.

[3] M. Hagenbuchner, A. Sperduti, and A. C. Tsoi, "Contextual processing of graphs using self-organizing maps," in *Proc. Eur. Symp. Artif. Neural Netw.*, 2005, pp. 399–404.

[4] M. Hagenbuchner, A. Sperduti, and A. Tsoi, "Contextual self-organizing maps for structured domains," in *Proc. Workshop Relat. Mach. Learn.*, 2005, pp. 46–55.

[5] M. K. M. Rahman, W. P. Yang, T. W. S. Chow, and S. Wu, "A flexible multi-layer self-organizing map for generic processing of tree-structured data," *Pattern Recognit.*, vol. 40, no. 5, pp. 1406–1424, 2007.

[6] M. Hagenbuchner, A. Sperduti, and A. C. Tsoi, "Self-organizing maps for cyclic and unbounded graphs," in *Proc. Eur. Symp. Artif. Neural Netw.*, 2008, pp. 203–208.

[7] S. Brin and L. Page, "The anatomy of a large-scale hypertextual Web search engine," *Comput. Netw. ISDN Syst.*, vol. 30, no. 1–7, pp. 107–117, 1998.

[8] C. Cortes and V. Vapnik, "Support-vector networks," *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, 1995.

[9] P. Frasconi, M. Gori, and A. Sperduti, "A general framework for adaptive processing in data structures," *IEEE Trans. Neural Netw.*, vol. 9, no. 5, pp. 768–785, Sep. 1998.

[10] T. Gartner, "A survay of kernels for structured data," *ACM SIGKDD Explorat. Newslett.*, vol. 5, no. 1, pp. 49–58, 2003.

[11] B. Schölkopf, K. Tsuda, and J. P. Vert, *Kernel Methods in Computational Biology*. Cambridge, MA: MIT Press, 2004.

[12] S. Vishwanathan and A. J. Smola, "Fast kernels on strings and trees," in *Proc. Neural Inf. Proces. Syst.*, 2002, pp. 569–576.

[13] M. Collins and N. Duffy, "New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron," in *Proc. Assoc. Comput. Linguistics*, 2002, pp. 263–270.

[14] J. Suzuki and H. Isozaki, "Sequence and tree kernels with statistical feature mining," in *Advances in Neural Information Processing Systems*, Y. Weiss, B. Schölkopf, and J. Platt, Eds. Cambridge, MA: MIT Press, 2006, vol. 18, pp. 1321–1328.

[15] F. Aiolli, G. Da San Martino, A. Sperduti, and M. Hagenbuchner, "Kernelized self organizing maps for structured data," in *Proc. Eur. Symp. Artif. Neural Netw.*, Apr. 24–27, 2007, pp. 19–24.

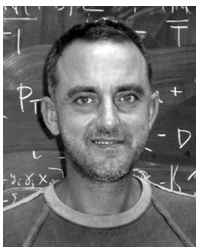
[16] N. Cristianini, J. Shawe-Taylor, and A. Elisseeff, "On kernel-target alignment," [Online]. Available: citeseer.comp.nus.edu.sg/480615.html

[17] L. Denoyer and P. Gallinari, "Report on the xml mining track at inex 2005 and inex 2006: categorization and clustering of xml documents," *SIGIR Forum*, vol. 41, no. 1, pp. 79–90, 2007.

[18] F. Trentini, M. Hagenbuchner, A. Sperduti, F. Scarselli, and A. Tsoi, "A self-organising map approach for clustering of xml documents," in *Proc. Int. Joint Conf. Neural Netw.*, Vancouver, BC, Canada, Jul. 2006, pp. 1805–1812.

[19] A. Moschitti, "A study on convolution kernel for shallow semantic parsing," in *Proc. Assoc. Comput. Linguistics*, Barcelona, Spain, 2004, pp. 335–342.

[20] D. MacDonald and C. Fyfe, "The kernel self-organising map," in *Proc. 4th Int. Conf. Knowl.-Based Intell. Eng. Syst. Allied Technol.*, 2000, vol. 1, pp. 317–320.



Fabio Aiolli received the Laurea degree in computer science, the Laurea degree in computer science technologies, and the Ph.D. degree in computer science from University of Pisa, Pisa, Italy, in 1999, 2003, and 2004, respectively.

He has been Project Manager for the European project IST-BIZON, in charge of the "Centro META, Consorzio Pisa Ricerche," between 2002 and 2004. He was Postdoctoral Researcher at the Computer Science Department, University of Pisa, between 2003 and 2004. Between 2004 and 2005, he was paid Visiting Scholar at the University of Illinois at Urbana-Champaign, Urbana. From 2005 to 2006, he held a Postdoctoral position at the Dipartimento di Matematica Pura ed Applicata, University of Padova, Padova, Italy, where since 2006, he has been an Assistant Professor (Ricercatore). His research activity is mainly in the area of machine learning, pattern recognition, and kernel methods, including study of methodologies for the solution of complex prediction problems, with structured input/output, both for the supervised and nonsupervised cases.

Giovanni Da San Martino received the B.S. and M.S. degrees in computer science from University of Pisa, Pisa, Italy, in 2003 and 2005, respectively, and the Ph.D. degree in computer science from University of Bologna, Bologna, Italy, in 2009.

In 2009, he spent a period at the University of Bristol, U.K., supported by a fellowship from the Department of Pure and Applied Mathematics of the University of Padova, Padova, Italy. Currently, he has a Postdoctoral Research Fellowship from the University of Padova. His research interests include pattern recognition, kernel methods, image processing, neural networks, and swarm intelligence.

Markus Hagenbuchner (M'03) received the Ph.D. degree in computer science from University of Wollongong, Wollongong, N.S.W., Australia.

Currently, he is a Senior Lecturer at the School of Computer Science and Software Engineering, University of Wollongong. He joined the machine learning research area in 1992, started to focus his research activities on neural networks for the graph structured domain in 1998, and pioneered the development of self-organizing maps for structured data. His contribution to the development of a self-organizing map for graphs led to winning the international competition on document mining on several occasions. He is the team leader of the machine learning group at the University of Wollongong. His current research interest is on the development of supervised and unsupervised machine learning methods for the processing of complex data structures in data mining applications.

Dr. Hagenbuchner has been the Co-Chair for the 2008 Artificial Intelligence Conference, and is a program committee member for the 2010 International Workshop on Artificial Neural Networks in Pattern Recognition (ANNPR). He has been a reviewer of international standing for the Australian Research Council since 2004, and has been an invited guest speaker at various international venues.

Alessandro Sperduti (M'98–SM'08) received the Laurea and Ph.D. degrees in computer science from University of Pisa, Pisa, Italy, in 1988 and 1993, respectively.

In 1993, he spent a period at the International Computer Science Institute, Berkeley, CA, supported by a postdoctoral fellowship. In 1994, he moved back to the Computer Science Department, University of Pisa, where he was Assistant Professor, and Associate Professor. Currently, he is Full Professor at the Department of Pure and Applied Mathematics, University of Padova, Padova, Italy. His research interests include pattern recognition, image processing, neural networks, kernel methods, and hybrid systems. In the field of hybrid systems, his work has focused on the integration of symbolic and connectionist systems. He contributed to the organization of several workshops on this subject. He is the author of more than 120 refereed papers mainly in the areas of neural networks, pattern recognition, and kernel methods.

Dr. Sperduti served in the program committee of several conferences on neural networks, machine learning, artificial intelligence, and information retrieval. He gave several tutorials within international schools and conferences, such as 1997, 1999, and 2001 International Joint Conference on Artificial Intelligence. He is in the editorial board of the journals *Neural Information Processing–Letters and Reviews*, *AI Communications*, *Neural Processing Letters*, and the IEEE TRANSACTIONS ON NEURAL NETWORKS. He is chair of the IEEE Data Mining Technical Committee and a member of the Executive Board of the European Neural Networks Society.