A New Tree Kernel Based on SOM-SD

Fabio Aiolli, Giovanni Da San Martino, and Alessandro Sperduti

Dept. of Pure and Applied Mathematics, Via Trieste 63, 35131 Padova - Italy

Abstract. Many different paradigms have been studied in the past to treat tree structured data, including kernel and neural based approaches. However, both types of methods have their own drawbacks. Kernels typically can only cope with discrete labels and tend to be sparse. On the other side, SOM-SD, an extension of the SOM for structured data, is unsupervised and Markovian, i.e. the representation of a subtree does not consider where the subtree appears in a tree. In this paper, we present a hybrid approach which tries to overcome these problems. In particular, we propose a new kernel based on SOM-SD which adds information about the relative position of subtrees (the route) to the activation of the nodes in such a way to discriminate even those subtrees originally encoded by the same prototypes. Experiments have been performed against two well known benchmark datasets with promising results.

Key words: Tree Kernels, Kernel Methods, SOM, Supervised Learning

1 Introduction

Recently, there has been a great interest in the study of techniques able to learn in structured domains with no need to represent data in vectorial form. For example, kernels for structured domains (see [7] for an overview), allow for a direct exploitation of the structural information obtaining very good results in practice.

However, kernel for structures have the well known disadvantage that, in the case of large structures and many symbols, the feature space implicitly defined by these kernels is very sparse [10]. In fact, these kernels are usually defined in terms of the number of matching subparts and, whenever many different types of these parts can be found in data, these matches tend to be barely observed. As a result, kernel based learning methods like Support Vector Machines (SVM) [5] using these standard kernels cannot be trained effectively. They will tend to generate several support structures thus leading to a final model which is similar to the nearest neighbor rule. It is then clear that any kernel machine cannot work well when used together with these kernels.

A completely different approach for the treatment of structured data has been presented in [8], a neural network based method, called SOM-SD. This is an unsupervised learning algorithm which extends the SOM to structured domains. As SOMs, SOM-SD organizes data (structures in this case) onto a topological discrete lattice. Moreover, the neural approach is well suited to cope with structures having real valued labels, while in the case of standard kernels for trees labels are assumed to be discrete.

The ability of the SOM-SD to represent the data onto a lattice preserving as much as possible their topology in the original space provides a viable technique for defining

similarity functions based on matching of non identical structures in the original space. This idea has been exploited by the Activation Mask Kernels family of kernels (see [3, 1]), defined on top of a SOM-SD with the aim of exploiting both its compression and "topology" preserving capabilities. Experimental results with these kernels provided evidence that, when sparsity on the data was present, they were able to improve the overall categorization performance over each method taken individually, i.e. either SVM using tree kernels or SOM-SDs equipped with a 1-NN classification rule. This also demonstrates that, neither tree kernels nor SOM-SDs are always able to retain all the relevant information for classification.

One issue with the SOM-SD type of algorithms, and SOM-SD activation based kernels consequently, is that they are computed by almost neglecting the contextual information of substructures. More specifically, the path linking the root of a tree to the root of the subtree we want to represent is not actually considered when training takes place. This can be a problem when this type of information is relevant for a given task.

In this paper, we propose a new method which tries to fill this gap. The proposed method can be though of as a hybrid method which combines the SOM-SD neural approach, and the Activation Mask kernels, in conjunction with a standard kernel for trees devised in [2]. Specifically, we propose to add additional (contextual) information to the activations of the nodes in such a way to discriminate between subtrees encoded by the same prototypes. Then we propose to add a type of information which is orthogonal to the one processed by the SOM-SD, i.e. information about the antecedents of the root of a given subtree (here referred to as a *route*).

Experiments performed with this new kernel against two well known competition datasets have shown a systematic improvement with respect to baseline approaches.

2 Background

In the following sections, the SOM-SD, an extension presented in [1, 3] of the Self Organizing Maps for structured data, and the Activation Mask Kernel presented in [2], are sketched. Please, see the referred paper for details on these algorithms.

2.1 Self-Organizing Maps for Structured Data

The SOM-SD extends the Self Organizing Map (SOM) approach [9] by allowing to process structured input. In this paper we are interested in structures in the form of (positional) trees, where each node v of a tree T can have a label (e.g., a real valued vector) $\mathbf{v}$ attached to it. Moreover, we assume that each child of a node v is associated to a specific position out of a maximum number o of available positions (the maximum out-degree of the trees we are interested in.) The i-th child of v will be denoted by $ch_i[v]$.

The SOM-SD can be understood as the recursive application of a standard SOM to individual nodes in a tree T 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. Given an input vector $\mathbf{x}$, let $\mathbf{y}_\mathbf{x}$ denote the coordinate vector of the winning neuron.

The network input for SOM-SD is a vector $\mathbf{x}_v$ representing the information of a node $v \in T$, 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 nodes on the same map, so that $\mathbf{x}_v = [\mathbf{v}, \mathbf{y}_{\mathbf{x}_{ch_1[v]}}, \ldots, \mathbf{y}_{\mathbf{x}_{ch_o[v]}}]$. In this representation, the "impossible" coordinate $(-1, -1)$, is used to represent a missing child. 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 $\mathbf{m} \equiv [\mathbf{m}^{label}, \mathbf{m}^{ch}]$ are of the same dimension.

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 nodes when processing a tree.

The SOM-SD then is able to map structures in input onto a discrete low dimensional lattice with the aim to preserve the topology of the input data. Structures which are similar tend to be mapped onto closer neurons. This is a key property for the intuition behind this paper. Figure 1 gives an examples of how trees are mapped into a SOM-SD. One can note similar (sub)trees are mapped close each other.

2.2 The Activation Mask Kernel

The unsupervised SOM-SD model can also be used to define a kernel for trees [3, 1]. The idea is to define a feature space having one dimension associated to each neuron of the map. Then a vectorial representation for a tree can be obtained by considering which ones of these neurons are activated for the nodes of the tree. Once the above representation has been computed for any pair of trees, a kernel can be promptly defined as the dot product of these representations.

More formally, given a SOM-SD map, let $\mathrm{ne}_\epsilon[\mathbf{y}(i)]$ denote the set of neurons (coordinates) in the ϵ-neighborhood of neuron i, i.e. $\{\mathbf{y}(j) | \Delta_{\mathbf{y}(i)\mathbf{y}(j)} \leq \epsilon\}$, where $\mathbf{y}(i) = (x_i, y_i)$ is the coordinate vector associated to neuron i, and Δ is the topological distance defined on the 2-dimensional map. Given two trees T_1 and T_2, we define the set of neurons (coordinates) shared by the two ϵ-neighbors related to nodes $v_1 \in T_1$ and $v_2 \in T_2$ as

$$I_\epsilon(v_1, v_2) = \mathrm{ne}_\epsilon[\mathbf{y}_{\mathbf{x}_{v_1}}] \cap \mathrm{ne}_\epsilon[\mathbf{y}_{\mathbf{x}_{v_2}}]. \quad (1)$$

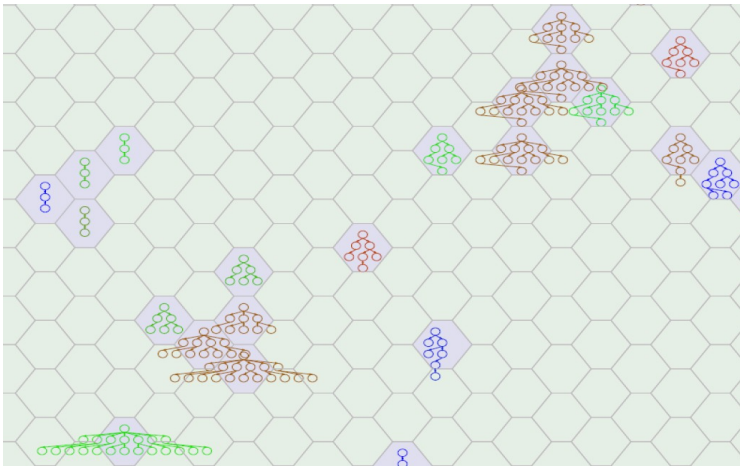

Fig. 1. Example of a SOM-SD mapping of a set of trees and their subtrees.

Then, the Activation Mask Kernel is defined by taking:

$$K_\epsilon(T_1, T_2) = \sum_{\substack{v_1 \in T_1, \\ v_2 \in T_2, \\ \mathbf{y} \in I_\epsilon(v_1, v_2)}} Q_\epsilon(\mathbf{y}, \mathbf{y}_{\mathbf{x}_{v_1}}) Q_\epsilon(\mathbf{y}, \mathbf{y}_{\mathbf{x}_{v_2}}), \tag{2}$$

where $Q_\epsilon(\mathbf{y}, \mathbf{y}')$ is inversely proportional to the distance $\Delta_{\mathbf{y}\mathbf{y}'}$ between map neurons with coordinates $\mathbf{y}$ and $\mathbf{y}'$ and $Q_\epsilon(\mathbf{y}, \mathbf{y}') = 0$ when the neurons are not in the ϵ-neighborhood of each other, i.e. when $\Delta_{\mathbf{y}\mathbf{y}'} > \epsilon$. In [1], $Q_\epsilon(\mathbf{y}, \mathbf{y}')$ is defined as

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

where $0 \leq \eta \leq 1$ is a parameter determining how much the distance influences the neighborhood activation.

Thus, the representation of a tree T into the feature space induced by the map is defined as the vector $\phi(T)$ with i-th component $\phi_i(T) = \sum_{v \in T} Q_\epsilon(\mathbf{y}(i), \mathbf{y}_{\mathbf{x}_v})$.

Figure 2 gives an example of construction of the feature space representation of 3 trees according to the AM-kernel ($\epsilon = 2$, $\eta = 1$). On the lower part of the image three simple trees selected from the INEX 2005 dataset (see section 4.1) and on the right part 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 tree at the left side is more similar to the tree at the center than to the tree at the right side, and this is reflected in the activation masks.

In [3, 1] it has been shown that the similarity function $K_\epsilon(T_1, T_2)$ is a kernel for any choice of $Q_\epsilon(\mathbf{y}, \mathbf{y}')$ and that the complexity of its evaluation is $O(a \cdot b \cdot (|T_1| + |T_2|))$, where $a \cdot b$ is the size of the map. The overall computational complexity is not affected by the required initial training of the SOM-SD as it is performed only once.

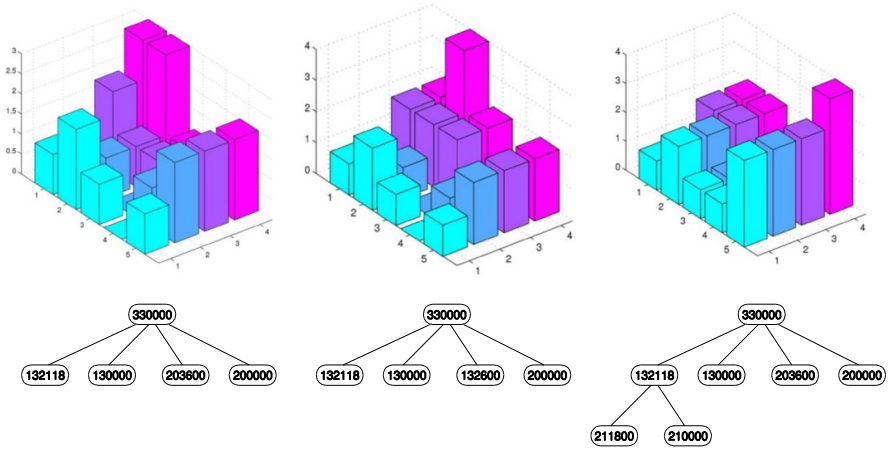

Fig. 2. Example of feature space construction for the AM-Kernel ($\epsilon = 2$, $\eta = 1$). The figure presents three trees (bottom) and the corresponding feature mapping onto a 5×4 map. Note that the feature maps of the first two trees are quite similar while they both differ from the third one, as expected.

3 Adding Route Information to the Activation Mask Kernel

In this section, we introduce the main contribution of the paper, i.e. the extension of the feature space of the Activation Mask Kernel with features that are based on route information. Intuitively, a route in a tree explicitly keeps information about the position of the nodes with respect to adjacent nodes.

Definition 1 (Route). *Let T be a (positional) tree, $v_1, v_2 \in T$ any two nodes in the tree, with v_2 being a descendant of v_1. Then the route from v_1 to v_2 in T, denoted by $\pi(v_1, v_2)$, is the sequence of indexes of edges connecting the consecutive nodes in the path connecting nodes v_1 and v_2.*

Figure 3 gives an example of a tree and a route computed between nodes **a** and **e**. The nodes connected by red edges represent the path connecting nodes **a** and **e**. The route connecting nodes **a** and **e** is represented by the sequence $(2, 3)$, since node **b** is the second child of **a** and node **e** is the third child of **b**. It must be pointed out that a route is not a path, since a path can be understood as a route where we retain the information about the label attached to each node belonging to the path.

The concept of route is useful for those tree domains where it may be important to "recognize" that a specific subtree, or family of subtrees, occurs into a specific "location" within a tree, e.g. at the end of a specific route, with no consideration of the labels attached to nodes crossed by the route. It must be noticed that, because of the causal style of processing, SOM-SD cannot discriminate among different occurrences of the same subtree within the same or different trees. In fact, when computing the winning neuron for a node in a tree, no information about the node's ancestors is used (see Fig. 4). It is true that the Activation Mask Kernel can actually exploit information

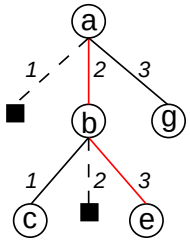

Fig. 3. An example of a route connecting nodes labeled with **a** and **e**. The positional nature of the tree is shown in the figure by representing missing edges by dashed lines and missing nodes by black squares. Edges crossed by the route are colored in red. The route is formed by the sequence $2, 3$ since node **b** is the second child of **a** and node **e** is the third child of **b**.

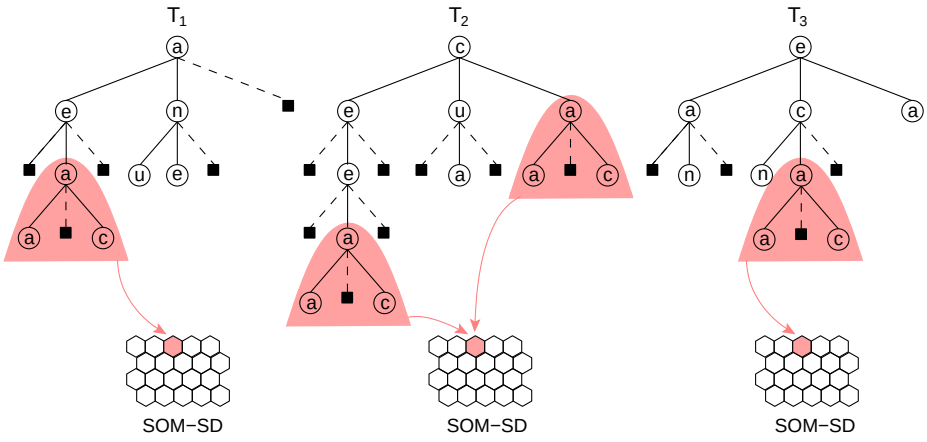

Fig. 4. Since SOM-SD is a causal model, the different occurrences of subtree **a(a,_,c)** in the trees $T_1,\ T_2,\ T_3$, get the same winner.

about the node's ancestors, since the activation map used by the kernel is obtained by collecting the information about winners for all the nodes in the tree. However, this information is heavily dependent on the label attached to each node. This can be easily understood by recalling that the winner neuron is determined by combining the information about labels and structure. Thus, the feature space exploited by the Activation Mask Kernel does not possess single features based on route information.

Our proposal is to enrich the feature space of the Activation Mask Kernel by introducing explicit information about routes. Let v be a node belonging to a tree T, and let π_v^T be the route associated to it. Then, we define $\mathrm{ne}_\epsilon^\pi[\mathbf{y}_{\mathbf{x}_v}] = \{\pi_u^T | u \in T,\ \Delta_{\mathbf{y}_{\mathbf{x}_v},\mathbf{y}_{\mathbf{x}_u}} \leq \epsilon\}$. Notice that for $\epsilon = 1$, $\mathrm{ne}_\epsilon^\pi[\mathbf{y}_{\mathbf{x}_v}]$ just contains routes of nodes of T that share the same winning neuron with coordinate vector $\mathbf{y}_{\mathbf{x}_v}$, i.e. if $\pi_v^T \in \mathrm{ne}_0^\pi[\mathbf{y}_{\mathbf{x}_v}]$ and $\pi_u^T \in \mathrm{ne}_0^\pi[\mathbf{y}_{\mathbf{x}_v}]$, with $u \neq v$, then $\mathbf{y}_{\mathbf{x}_v} = \mathbf{y}_{\mathbf{x}_u}$. Given two trees T_1 and T_2, we define the set of routes

shared via SOM-SD by the two π-ϵ-neighbors related to nodes $v_1 \in T_1$ and $v_2 \in T_2$ as

$$I_\epsilon^\pi(v_1, v_2) = \mathrm{ne}_\epsilon^\pi[\mathbf{y}_{\mathbf{x}_{v_1}}] \cap \mathrm{ne}_\epsilon^\pi[\mathbf{y}_{\mathbf{x}_{v_2}}]. \tag{4}$$

Then, we define the kernel contribution of the routes as:

$$K_\epsilon^\pi(T_1, T_2) = \sum_{\substack{v_1 \in T_1, v_2 \in T_2 \\ \mathbf{y}_{\mathbf{x}_{v_1}} = \mathbf{y}_{\mathbf{x}_{v_2}} \\ \pi' \in I_\epsilon^\pi(v_1, v_2)}} Q_\epsilon(\mathbf{y}_{\mathbf{x}_{v_1}}, \mathbf{y}_{\mathbf{x}_{\pi'|T_1}}) Q_\epsilon(\mathbf{y}_{\mathbf{x}_{v_2}}, \mathbf{y}_{\mathbf{x}_{\pi'|T_2}}), \tag{5}$$

where $\pi'|T_i$ refers to the node reached by following route π' starting from the root of T_i. Note that eq. 5 can be computed efficiently by explicitly representing each feature since the number of distinct routes for a tree is at most $|T|$.

The final kernel is defined as $\hat{K}_\epsilon(T_1, T_2) = K_\epsilon(T_1, T_2) + K_\epsilon^\pi(T_1, T_2)$.

In Fig. 5 we give an example of how routes are exploited.

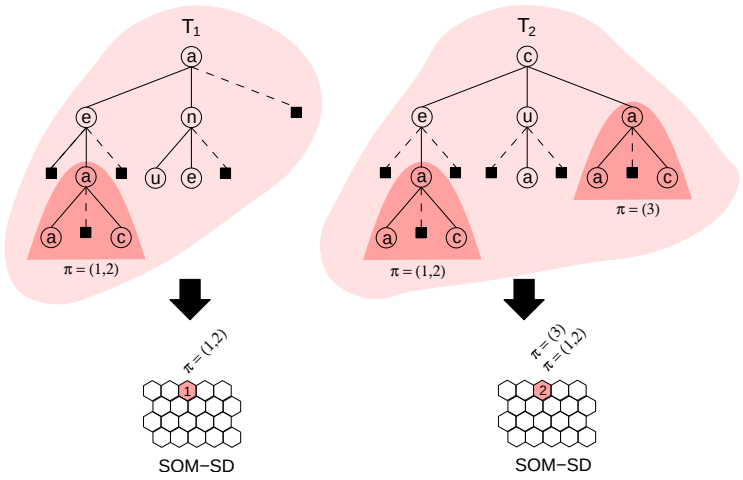

Fig. 5. In the proposed representation, the activation mask for each tree is enriched by associating to each neuron the routes to (sub)trees for which the neuron is a winner. In the figure, two examples are shown, where only the contribution of subtree **a(a,_,c)** is reported.

4 Experiments and Results

In order to test the effectiveness of the proposed approach, two experiments on multi-class classification problems were performed. The datasets considered are derived from the INEX 2005 and INEX 2006 competitions [6], respectively.

4.1 Data Description and Experimental Setting

The INEX 2005 Competition dataset is formed by XML documents describing movies from the IMDB site[1]. The dataset employed in the experiments is obtained from the (m-db-s-0) corpus of INEX 2005 after a preprocessing phase. It consists of 9631 documents containing XML tags only. The data are divided into 11 classes. The preprocessing is described in detail in [11]. The choice of such preprocessing is motivated by the need to obtain input structures of manageable size and to compare to the techniques which won the competition. The mean size of the input structures has been reduced from 684191 vertices with maximum out-degree 6418 to 124359 vertices with maximum outdegree 32. The data are divided into training, validation and test sets containing 3397, 1423 and 4811 documents, respectively. The dataset is unbalanced and sparse with respect to two of the most popular kernels for trees: the Subtree (ST) and the Subset tree kernels (SST) [4]. Their sparsity index, computed as the proportion of example pairs in the dataset whose kernel value is 0, is 0.54 (see [1] for details).

The INEX 2006 dataset is derived from the IEEE corpus and it's composed of 12107 scientific articles from IEEE journals in XML format. It includes XML formatted documents, each from one of 18 different journals. Each different journal corresponds to one class. The data have been preprocessed with the same methodology used for INEX 2005. The data is again split into training, validation and test sets containing 4251, 1802 and 6054 documents, respectively. The dataset is unbalanced and non sparse with respect to ST and SST kernels. In fact, their sparsity index is 0.002489.

Experiments proceeded as follows. First, for each dataset, five maps were trained with the SOM-SD software[2]. The maps were chosen among the 45 described in [1] by sorting them in ascending order according to the classification error, computed with a 1-NN procedure, and then selecting one map every 11. This choice allows us to investigate the dependency of the error of our kernel from the map. Only 5 maps were selected because of the need to reduce the duration of the experiments. The maps selected for INEX 2005 and INEX 2006 are listed in table 1 and table 2, respectively. The parameters not listed in the two tables are kept fixed: $\alpha = 1$, neighbourhood radius=18, type of α decrease=sigmoidal, map topology=hexagonal. Given a map, the set of features of the activation mask kernel with information about the routes, were computed as described in sections 2.2 and 3. Experiments with the SVM[3] and the proposed kernel, AMπ, were performed by selecting, on the validation set, the ϵ of the AM kernel and the c of the SVM among these values: $1 \leq \epsilon \leq 6$, $c \in \{0.001, 0.01, 0.1, 1, 10, 100, 1000\}$. The parameter η of the AM kernel is set to 1. Finally, the performance of the best parameter setting, for each map, was checked on the test set. The classification of a tree according to the SOM-SD is the one of the neuron representing the root of the tree. The class of a neuron is the most frequent class of the trees of the training set represented by such neuron.

[1] http://www.imdb.com

[2] *http://www.uow.edu.au/∼markus/apods/software.html*

[3] http://disi.unitn.it/moschitti/Tree-Kernel.htm

Table 1. Characteristics of the SOM-SD maps trained on the INEX 2005 dataset and classification error for the SOM-SD, AM and AMπ kernels.

Map #	Size	Learning Iterations	μ	SOM-SD error %	AM error %		AMπ error %	
				test	valid	test	valid	test
1	110 x 80	128	0.85	8.65	6.33	6.41	3.45	3.48
2	110 x 80	32	0.05	12.62	5.77	5.60	3.59	3.16
3	77 x 56	32	0.65	18.62	8.22	7.18	3.52	3.14
4	55 x 40	128	0.85	22.51	7.38	7.24	3.45	3.23
5	55 x 40	32	0.25	32.49	9.84	9.36	3.45	3.39

Table 2. Characteristics of the SOM-SD maps trained on the INEX 2006 dataset and classification error for the SOM-SD, AM and AMπ kernels.

Map #	Size	Learning Iterations	μ	SOM-SD error %	AM error %		AMπ error %	
				test	valid	test	valid	test
1	110 x 80	128	0.05	60.77	59.22	61.07	57.72	60.16
2	110 x 80	32	0.85	61.98	58.77	59.93	57.22	59.67
3	77 x 56	128	0.85	63.45	59.22	61.37	58.50	59.49
4	55 x 40	64	0.05	66.25	59.94	61.75	60.44	61.73
5	55 x 40	32	0.45	67.66	59.55	61.77	57.50	59.26

4.2 Results and Discussion

Table 1 summarizes the results on the INEX 2005 dataset. Note that the results of the AM kernel differ from those in [1] because a newer version of the SVMlight software has been employed. The AMπ kernel show a clear improvement for all maps both on validation and test with respect to the SOM-SD alone and the AM kernel. The mean classification error of the AM kernel is 7.508 with standard deviation 1.610 on the validation set and 7.158 with standard deviation 1.4 on the test set. The mean classification error of the AMπ kernel on validation is 3.492 with standard deviation 0.062, and 3.28, with standard deviation 0.148, on the test set. The low values of the mean and standard deviations for the AMπ kernel suggests that the accuracy of the kernel does not seem to depend on the employed map.

The results on the INEX 2006 dataset are summarized in table 2. Except for one case on the validation set, the AMπ kernel always improves with respect to the AM kernel and always improves with respect to the SOM-SD alone. The mean classification error of the AM kernel is 59.34 with standard deviation 0.435 on the validation set and 61.178 with standard deviation 0.755 on the test set. The mean classification error of the AMπ kernel on validation is 58.276 with standard deviation 1.299, and 60.062, with standard deviation 0.989, on the test set. Although AMπ kernel shows more variability on the results, on average an improvement of 1.116 is obtained on the test set.

5 Conclusions

The Activation Mask Kernel is a tree kernel based on SOM-SD. Here we have proposed an extension of the Activation Mask Kernel which adds to the feature space information

about routes, i.e. how to reach a specific node in a tree by starting from its root. In our proposal, the contribution to the kernel of two trees given by the sharing of a specific subtree is reinforced if the root of the subtree can be reached by the same route in both trees. The SOM-SD is basically enriched with information about routes, and the new kernel computed by adding to the Activation Mask Kernel the contribution due to routes. This extension is supposed to be particularly effective when the tree domain is such that the relative location of a subtree, or family of subtrees, within the tree is important for the task. Experimental results obtained on XML datasets seem to confirm the usefulness of the proposed approach.

References

1. F. Aiolli, G. Da San Martino, M. Hagenbuchner, and A. Sperduti. Learning nonsparse kernels by self organizing maps for structured data. In *IEEE Transactions on Neural Networks*, volume 20, pages 1938–1949, December 2009.
2. F. Aiolli, G. Da San Martino, and A. Sperduti. Route kernels for trees. In *International Conference on Machine Learning*, page 3, 2009.
3. F. Aiolli, G. Da San Martino, A. Sperduti, and M. Hagenbuchner. "kernelized self organizing maps for structured data". In *ESANN 2007 Conference*, April 24-27 2007.
4. M. Collins and N. Duffy. New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron. In *ACL02*, 2002.
5. Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Mach. Learn.*, 20(3):273–297, 1995.
6. 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*, 41(1):79–90, 2007.
7. Thomas Gartner. A survey of kernels for structured data. *SIGKDD Explorations*, 5(1):49–58, 2003.
8. M. Hagenbuchner, A. Sperduti, and A.C. Tsoi. A self-organizing map for adaptive processing of structured data. *IEEE Transactions on Neural Networks*, 14(3):491–505, May 2003.
9. T. Kohonen. *Self-Organizing Maps*. Springer, Berlin, Heidelberg, 1995.
10. J. Suzuki and H. Isozaki. Sequence and tree kernels with statistical feature mining. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems 18*, pages 1321–1328. MIT Press, Cambridge, MA, 2006.
11. F. Trentini, M. Hagenbuchner, A. Sperduti, and F. Scarselli. A self-organising map approach for clustering of xml documents. In *Proceedings of the International Joint Conference on Neural Networks, IJCNN 2006, part of the IEEE World Congress on Computational Intelligence, WCCI 2006, Vancouver, BC, Canada, 16-21 July 2006*, pages 1805–1812. IEEE Press, 2006.