”Kernelized” Self-Organizing Maps for Structured Data

Fabio Aiolli[1], Giovanni Da San Martino[1], Alessandro Sperduti[1], and
Markus Hagenbuchner[2]

1- University of Padova - Dept of Pure and Applied Mathematics
via Trieste 63, Padova - Italy

2- University of Wollongong - Faculty of Informatics
Northfields Avenue, NSW 2522 - Australia

Abstract.
The suitability of the well known kernels for trees, and the lesser known Self-Organizing Map for Structures for categorization tasks on structured data is investigated in this paper. It is shown that a suitable combination of the two approaches, by defining new kernels on the activation map of a Self-Organizing Map for Structures, can result in a system that is significantly more accurate for categorization tasks on structured data. The effectiveness of the proposed approach is demonstrated experimentally on a relatively large corpus of XML formatted data.

1 Introduction

Graphs provide very versatile means to represent information. In fact, there are numerous applications where a graph structured representation of data forms an essential element for a given task. For example, the search engines for the World Wide Web require the knowledge of the structure of the Web in order to function effectively, or in molecular chemistry the structural composition of atomic elements must be respected since it is related to molecule’s properties.

Traditional methods in machine learning deal with vectorial information. Thus, preprocessing is required to encode graph structured information by a vector. However, a pre-processor is always task specific and needs to be suitably designed for any new task. The pre-processing of information can also result in the loss of some information which may be important for a given task.

Recent developments in Machine Learning produced methods capable of processing graph structured information directly. For example, the definition of kernels for structured domains (e.g. [1]) allows a better exploitation of structural information. Advances in artificial neural networks produced both supervised (e.g. [2]) and unsupervised (e.g. [3]) models that process graph structured information. This paper addresses the suitability of kernels for structures [1] and Self-Organising Maps for Structured Data [4] for classification tasks involving structures. It will be shown that each of these two approaches have some significant drawbacks but that the combination of these approaches can result in a system that is superior to both.

One problem of kernel for structures, such as the well known Subtree kernel (ST) [5] and Subset tree kernel (SST) [6], already recognized in applications to Natural Language Processing, is that in the case of large structures and many symbols, the feature space implicitly defined by these kernels is very sparse [7]. In fact, it will be shown that the classification performance of the Self-Organizing Maps for structured data (SOM-SD) can be significantly higher than the ones of ST and SST kernels in many real world applications. We believe that this is mainly due to the ability of SOM-SD to compress structural information, thus avoiding the sparsity problem suffered by the tree kernels. However, the compression property of SOM-SD together with the unsupervised nature of the model, may lead to the loss of relevant information for the categorization task.

This paper studies kernels defined on SOM-SD activations that, while benefit from the spatial reduction of the SOM-SD, try to recover "topological" information (i.e., structural similarity) not exploited by this technique.

The experiments performed show that such new kernels are able to improve the overall categorization performance, thus demonstrating, at least in principle, that neither tree kernels, nor SOM-SD, are able to retain all the relevant information. This means that more effort is needed to try to close this gap, which the approach proposed in this paper already contributed in part to fill in.

This paper is organized as follows: A brief description of kernels for structured domains and SOM-SD is given in Section 2 and Section 3 respectively. The kernel for SOM-SD is proposed in Section 4, and experimental findings are presented in Section 5. Conclusions are drawn in Section 6.

2 Kernel for Trees

Kernel algorithms, such as the Support Vector Machine (SVM), require the definition of a kernel function, i.e. a similarity function between any two elements of a domain. In the following two of the most popular tree kernels, which will be used as (strong) baseline kernels in this paper, are introduced. In particular, a kernel for trees can be defined by considering subset trees (SST) as proposed in [6] or by considering matching subtrees (ST) as proposed in [5].

The SST kernel is based on counting matching subtrees between two input trees. Given an input tree T, let $h_s(T)$ be the number of times a subtree s occurs in T (here s ranges over all possible subtrees of a dataset).

The SST kernel can be efficiently calculated by a recursive procedure defined as: $K(T_1, T_2) = \sum_{t_i \in T_1} \sum_{t_2 \in T_2} \sum_{s=1}^{m} h_s(t_1) h_s(t_2) = \sum_{t_1 \in T_1} \sum_{t_2 \in T_2} C(t_1, t_2)$, where $C(t_1, t_2) = \sum_{s=1}^{m} h_s(t_1) h_s(t_2)$ can be recursively computed according to the following rules: *i)* if the productions[1] at t_1 and t_2 are different then $C(t_1, t_2) = 0$; *ii)* if the productions at t_1 and t_2 are the same, and t_1 and t_2 have only leaf children (i.e. they are preterminals symbols) then $C(t_1, t_2) = \lambda$; *iii)* if the productions at t_1 and t_2 are the same, and t_1 and t_2 are not pre-terminals, then $C(t_1, t_2) = \lambda \prod_{j=1}^{nc(t_1)} (1+C(ch_j[t_1], ch_j[t_2]))$, where $nc(t_1)$ is the number of children of t_1 and $ch_j[t]$ is the j-th child of node t, and $\lambda > 0$ is a weighting parameter whose purpose is to reduce the influence of larger subtrees [6].

The ST kernel counts the number of shared full subtrees and can be obtained by substituting rule *iii)* for the SST with $C(t_1, t_2) = \lambda \prod_{j=1}^{nc(t_1)} C(ch_j[t_1], ch_j[t_2])$.

3 The SOM for data structures

The SOM-SD extends the Self Organizing Map (SOM) approach [8] by allowing to process structured input in the form of directed acyclic graphs, where each vertex of

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

the graph can have a label (e.g., a real valued vector). The SOM-SD can be understood as the recursive application of a standard SOM where the input is properly coded to take into consideration the structural information. The network input for SOM-SD is a vector $\mathbf{x_v}$ representing the information of a vertex v of the input graph and it is built through the concatenation of the data label $\mathbf{v}$ which may be attached to v and the coordinates of the mapping, on the same network, of child nodes $\mathbf{y}_{ch[v]}$ of v so that $\mathbf{x}_v = [\mathbf{v}, \mathbf{y}_{ch[v]}]$. The vectors are made constant in size by assuming a maximum outdegree o of any vertex in the dataset. For vertices with less than o children, padding with a default coordinate, typically the “impossible” coordinate (-1,-1), is applied. As a result, the $\mathbf{x}$ are $k = p + 2o$ dimensional vectors, where p is the dimension of the data label and the constant 2 refers to the number of dimensions of the map. The codebook vectors $\mathbf{m} \equiv [\mathbf{m}^{label}, \mathbf{m}^{ch}]$ are of the same dimension.

In order to account for the coding of both data label and structural information in the same input item, the winning neuron is selected by using a modified Euclidean distance, i.e. $r = \arg\min_i \|\mu_1(\mathbf{v}-\mathbf{m}_i^{label})+\mu_2(\mathbf{y}_{ch[v]}-\mathbf{m}_i^{ch})\|$ where the constant μ_1 influences the contribution of the data label component to the Euclidean distance, and μ_2 controls the influence of the children’s coordinates to the Euclidean distance. In order to obtain unique value pairs for μ_1 and μ_2 it is commonly assumed $\mu_2 = 1 - \mu_1$, $\mu_1 \in [0, 1]$. The difficulty of this approach is the estimation of good values for the parameters μ_i. Using large values for μ_2 increases the mapping precision according to structural information but neglects the importance of the information provided by the data labels. Setting large values for μ_1 increases the focus on the data label but reduces the SOM-SD’s ability to represent structural information, and reduces the accuracy with which information is passed on to parent nodes. The SOM-SD is trained similarly to the traditional SOM with the exception of a modified distance measure (as shown above), and the fact that the network input $\mathbf{x}_v$ needs to be updated according to possible changes in $\mathbf{y}_{ch[v]}$ during training.

4 Activation Mask Kernel

Here we show how novel tree kernels can be defined on the basis of a SOM-SD map. The basic idea is to represent each node of a tree into the activation map (feature space) of a SOM-SD and then define a kernel which computes the dot product in this space.
Let $\mathrm{ne}_\epsilon[m]$ denote the set of neurons (SOM-SD map nodes) in the ϵ-neighbourhood of the neuron m, i.e. $\{m' | \Delta_{m'm} \leq \epsilon\}$, where Δ is the topological distance defined on the two dimensional map. An interesting measure of similarity between two subtrees which takes into account the topology induced by the SOM-SD can be defined as the cardinality of the intersection of the ϵ-neighbours of the neurons mostly activated by those subtrees. Let $m^*(t_1)$ and $m^*(t_2)$ be the coordinates of the winning neurons for the root nodes of subtrees t_1 and t_2, respectively, and

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

be the set of nodes shared by the two ϵ-neighbours, 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 subtrees can be considered instead of the strict intersection. For example, it is possible to weight differently matching regions depending on the distance from the activated neurons:

$$K_\epsilon(T_1, T_2) = \sum_{t_1 \in T_1, t_2 \in T_2} \sum_{m \in I_\epsilon(t_1, t_2)} Q_\epsilon(m, m^*(t_1)) Q_\epsilon(m, m^*(t_2)) \tag{3}$$

where $Q_\epsilon(m, m')$ is inversely proportional to the distance $\Delta_{mm'}$ between map nodes m and m' and $Q_\epsilon(m, m') = 0$ when the nodes are not in the ϵ-neighbourhood of each other, i.e. $\Delta_{mm'} > \epsilon$. As an example, $Q_\epsilon(m, m')$ can be defined as

$$Q_\epsilon(m, m') = \begin{cases} \frac{\epsilon - \eta \Delta_{mm'}}{\epsilon} & \text{if } \Delta_{mm'} \leq \epsilon \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

where $0 \leq \eta \leq 1$ is a parameter determining how much the distance influences the neighbourhood activation. Note that (2) can be obtained from (4) setting $\eta = 0$.
The similarity function $K_\epsilon(T_1, T_2)$ is a kernel for any choice of $Q_\epsilon(m, m')$. A way to demonstrate this is to show that there exists a function ϕ such that for every T_1, T_2 $\phi(T_1) \cdot \phi(T_2) = K_\epsilon(T_1, T_2)$, i.e. K_ϵ can be expressed as a dot product in a feature space induced by ϕ. Let us define a feature space which has the same dimension as the map produced by the SOM-SD. Let $n \times o$ be the size of the rectangular grid of the map, then $\phi(T) \in \Re^{n \times o}$. Now, given a tree T, we define the mask $M \in \Re^{n \times o}$ where every element of M is associated to a node of the map. Let M be initially set to the null vector. The feature vector is then constructed by computing the best-matching map node $m^*(t)$ for each subtree $t \in T$ when presented to the SOM-SD. Then, the entries of M associated to neighbours 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 $\phi(T) = [M_1, \ldots, M_{n \times o}]$. At this point is 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 subtrees of T, and we can check that the kernel is obtained by performing the dot product in feature space, i.e.

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

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

5 Experiments and Results

Performances of the different methods are evaluated by using a relatively large set of XML formatted documents which were made available as part of the INEX Initiative[2]. Specifically the corpus (`m-db-s-0`) consists of $9,640$ documents containing XML tags only, i.e. no further textual information available. All documents have one out of 11 possible target value. 3377 dataset documents are marked as being the training set, 1447 as being the validation set and all remaining documents form the test set. A tree structure is extracted for each of the documents in the dataset by following the general XML structure within the documents. The dataset is composed of $684,191$ nodes (subtrees) with maximum out-degree $6,418$. A pre-processing step was performed on the dataset in order to reduce its dimensionality. First, repeated sequences of tags within the same level of a structure were collapsed. A further dimension reduction has been achieved by collapsing simple sub-structures which have the property of a data sequence into a single node. The pre-processing step reduced the maximum out-degree to 32, and the total number of nodes to $124,359$. SVM-based multiclass classification of the dataset was obtained by using the one-against-all method. First 11 binary classifiers, each devoted to recognize a single class, were trained. Then the 11-class classification was chosen to be the prediction of the 1-class classifier with highest confidence. Best parameters were selected comparing classification performance on the validation set. The same parameter 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. In order to have a baseline, the SVM with ST and SST kernels was applied to the dataset. Best ST performance has an error rate of 11.27%. It was obtained setting λ (see Section 2) to 1.1 and setting the usual c parameter of the SVM to 10. Best SST performance is 11.21% ($c = 10$ and $\lambda = 0.1$).
The maps used for this study were created by the SOMS-SD Simulator[3]. 9 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 from the maps. However, due to SOM-SD training times and the number of parameters involved, a comprehensive investigation was not feasible. Map 9 is the exact reproduction of the best SOM-SD map for this dataset [9].

After the training phase each map was evaluated on the test set with a k-nn procedure with $k = 1$. Table 1 reports the classification performance of each map. Notice that the worst classification error, 14.06%, is significantly above the baseline (11.21%). Experiments proceeded by testing the activation mask kernels (AM) defined in Section 4.

Map	1	2	3	4	5	6	7	8	9
Error rate(%)	11.69	14.01	14.06	11.46	12.23	7.30	9.28	6.14	6.10

Table 1: Classification error of the SOM-SD maps.

First, a new dataset in which each sample was transformed into its activation mask, was created. This process was repeated for each map and for different values of ϵ (see

[2] *http://xmlmining.lip6.fr*

[3] *http://www.uow.edu.au/˜markus/apods/software.html*

eq. 1). Table 2 summarizes the results obtained. In these experiments the use of the Activation Mask kernel always improved the classification performance. Moreover the low standard deviation of the AM kernels suggests that the improvement is quite independent with respect to the SOM-SD used. A reasonable technique for selecting the neighbourhood size for the AM is to select the ϵ best performing (together with all other parameters) on the validation set. In our experiments this technique would have led to a mean error rate of 5.82% with standard deviation of 0.77, which would place amongst the best with respect to the approaches listed in Table 2.

	ST	SST	SOM-SD	AM	AM	AM	AM	AM
				$\epsilon = 1$	$\epsilon = 2$	$\epsilon = 3$	$\epsilon = 4$	$\epsilon = 5$
Mean error(%)	11.27	11.21	10.25	5.84	5.69	5.91	5.92	5.99
Std. Deviation	-	-	3.16	0.78	0.67	0.88	0.87	0.75

Table 2: Mean classification error and standard deviation over all maps for the Subtree kernel, Subset Tree kernel, SOM-SD and Activation Mask kernel with different neighbourhood sizes.

6 Conclusion

When facing datasets with large structures and many symbols, the feature space implicitly defined by Subset and SubTree kernels are sparse, thus leading to poor classification performance. The SOM-SD is a method for dimensionality reduction which is "topology" preserving. In this paper a new class of kernels defined on SOM-SD activations is studied. Experiments have shown that the new kernels improve both the SOM-SD and tree kernels classification performances.

References

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

[2] P. Frasconi, M. Gori, and A. Sperduti. A general framework for adaptive processing in data structures. *IEEE Transactions on Neural Networks,* Vol 9(5):768–785, 1998.

[3] 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.

[4] M. Hagenbuchner, A. Sperduti, and A.C. Tsoi. Contextual self-organizing maps for structured domains. In *Workshop on Relational Machine Learning*, pages pp. 46–55, 2005.

[5] S.V.N. Vishwanathan and A. J. Smola. Fast kernels on strings and trees. In *Proceedings of Neural Information Processing Systems 2002*, 2002.

[6] M. Collins and N. Duffy. New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron. In *ACL02,* 2002.

[7] 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.

[8] T. Kohonen. *Self-Organizing Maps*. Springer, Berlin, Heidelberg, 1995.

[9] F. Trentini, M. Hagenbuchner, A. Sperduti, F. Scarselli, and A. Tsoi. A self-organising map approach for clustering of xml documents. In *Proceedings of the WCCI*, Vancouver, Canada, July 2006. IEEE Press.