ESANN'2005 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 27-29 April 2005, d-side publi., ISBN 2-930307-05-6.

Contextual Processing of Graphs using Self-Organizing Maps

Markus Hagenbuchner[1], Alessandro Sperduti[2] and Ah-Chung Tsoi[3] *

1- University of Wollongong, NSW - Australia

2- Università di Padova - Italy

3- Australian Research Council, Canberra - Australia

Abstract. This paper introduces a novel approach to Self-Organizing Maps which is capable of processing graphs such that the context of vertices and sub-graphs are considered in the mapping process. The result is that any vertex in a graph is mapped onto an n-dimensional map depending on its label and the graph structure as a whole. Experimental results demonstrate that the proposed approach achieves the desired outcomes.

1 Introduction

Self Organizing Maps (SOMs) are Neural Network Models typically trained in an unsupervised fashion. SOMs are commonly employed to tasks that require dimension reduction or the clustering of data vectors [1]. Recent developments extended the SOM's capabilities by allowing the direct processing of more general types of data such as sequences (e.g., TKM [2], RecSOM [3]), and directed graphs (SOM-SD [4]). In particular, a SOM-SD processes directed acyclic graphs, with numerical labels attached to vertices, in a bottom-up fashion, starting from the vertices with no offsprings up to vertices with no parents. A SOM-SD network processes a vertex at a time, starting from the vertices with no offsprings and moving to internal vertices according to some inverse topological order of the vertices of each graph. Let $\boldsymbol{y}_v = f(\boldsymbol{x}_v)$ be the response of the network at vertex v. Then, the network input $\boldsymbol{x}_v$ for v is defined as a vector obtained by the concatenation of the data label $\boldsymbol{l}_v \in \mathbb{R}^p$ and coordinates of the mapping of children vertices $\boldsymbol{y}_{ch[v]}$ (i.e., the coordinates of the winner neuron for each child) such that $\boldsymbol{x}_v = [\boldsymbol{l}_v, \boldsymbol{y}_{ch[v]}]$. These input vectors can be made constant in size if the maximum outdegree of any vertex in the dataset is known. Assuming that the maximum outdegree is o, then for vertices with less than o children the missing children are encoded by a default code, using, for example, the "impossible" winning coordinate $(-1, -1)$ for a 2-D map. Training a SOM-SD network is similar to a standard SOM with two differences: (a) the need to consistently update the $\boldsymbol{x}_v$ vectors at each iteration on the presentation of the training set, and (b) the introduction of weight values in the routine which computes the best matching codebook:

$$r = \arg\min_i \|(\mathbf{x}_v - \mathbf{m}_i)\mathbf{\Lambda}\| \tag{1}$$

where $\mathbf{x}_v$ is the input vector for vertex v, $\mathbf{m}_i$ the i-th codebook, and $\mathbf{\Lambda}$ is a $m \times m$ dimensional diagonal matrix with its diagonal elements $\lambda_{1,1} \cdots \lambda_{p,p}$ set to μ_1, and

*This work was partially supported by MIUR grant n. 20033091149_005 and an ARC Discovery grant.

$\lambda_{p+1,p+1} \cdots \lambda_{m,m}$ set to μ_2. The constant μ_1 influences the contribution of the data label component to the Euclidean distance (1), and μ_2 controls the influence of the children's coordinates to the same Euclidean distance.

This way of processing the data corresponds to assuming a causal dependence of the output of a vertex v only on the outputs already computed for the children of v. This property, especially in the case of discrete labels, turns out to be very useful in reducing the computational burden, since, if a substructure is shared by several different structures, the output for such substructure needs to be computed only once. Unfortunately, from a computational point of view, this also means that not all directed graphs can be discriminated by this way of processing the structures. In fact, a graph with vertices v_1, v_2, v_3, v_4 and labels $\boldsymbol{l}_{v_1} = A$, $\boldsymbol{l}_{v_2} = B$, $\boldsymbol{l}_{v_3} = C$, $\boldsymbol{l}_{v_4} = D$, and arcs $v_1 \rightarrow v_2$, $v_1 \rightarrow v_3$, $v_2 \rightarrow v_4$, $v_3 \rightarrow v_4$, cannot be discriminated from a graph with vertices $v_1, ..., v_5$, labels $\boldsymbol{l}_{v_1} = A$, $\boldsymbol{l}_{v_2} = B$, $\boldsymbol{l}_{v_3} = C$, $\boldsymbol{l}_{v_4} = D$, $\boldsymbol{l}_{v_5} = D$, and arcs $v_1 \rightarrow v_2$, $v_1 \rightarrow v_3$, $v_2 \rightarrow v_4$, $v_3 \rightarrow v_5$, since vertex v_4 and vertex v_5, because of the causality assumption, cannot generate a different output. A possible solution to this problem, in the context of supervised learning for structured domains, has been proposed by using the Contextual Recursive Cascade-Correlation model [5], where information about the parents of a vertex v coded in frozen units is exploited by the candidate units to generate an output that depends on the "context" of v.

In this paper, we explore the possibility to introduce this ability into unsupervised neural network models and in particular into an extended SOM-SD, that we will refer to as Contextual SOM-SD.

2 The Contextual Processing of Graphs

In order to achieve contextual processing of information by a SOM-SD when processing a vertex v of a graph, it is necessary to have available the network response to the parent vertices when processing a child vertex, i.e. the states $\boldsymbol{y}_{pa[v]}$. This information is not available since the network processes data in a bottom-up fashion.

The solution presented in this section exploits the fact that $\boldsymbol{y}_{pa[v]}$ becomes available once the SOM-SD has been fully trained. Then, a second map is trained using the $\boldsymbol{y}_{pa[v]}$ as additional (static) information. The approach recursively trains a new map and initializes the $\boldsymbol{y}_{pa[v]}$ using the relevant information available from the previously trained map. If we denote the output of a standard SOM-SD with $\boldsymbol{y}_v^{(0)}$ (i.e., the network at level 0), then the input vector for the second map (i.e., the network at level 1) will be defined as $\boldsymbol{x}_v^{(1)} = [\boldsymbol{l}_v, \boldsymbol{y}_{ch[v]}^{(1)}, \boldsymbol{y}_{pa[v]}^{(0)}]$. Applied recursively, this scheme will include not only the information about the parents of a vertex, but also the information about its ancestors. Thus, a SOM-SD at level $i > 0$ will have as input the vector $\boldsymbol{x}_v^{(i)} = [\boldsymbol{l}_v, \boldsymbol{y}_{ch[v]}^{(i)}, \boldsymbol{y}_{pa[v]}^{(i-1)}]$, where we assume that training for network at level $i-1$ has been completed.

The minimum number of levels required to guarantee that the full context of every vertex is covered, depends on the longest path among two vertices of any graph in the dataset. If the longest path has length d, then the training of $d+1$ networks will ensure that the full context of any vertex in the graphs is considered.

A generalization of this approach would be to consider for network at level i, the use of information coming from all the previously trained networks. Thus, the input

vector for level $i > 0$ can be defined as

$$\boldsymbol{x}_v^{(i)} = [\boldsymbol{l}_v, \boldsymbol{y}_{ch[v]}^{(0)}, \boldsymbol{y}_{ch[v]}^{(1)}, \ldots, \boldsymbol{y}_{ch[v]}^{(i)}, \boldsymbol{y}_{pa[v]}^{(0)}, \boldsymbol{y}_{pa[v]}^{(1)}, \ldots, \boldsymbol{y}_{pa[v]}^{(i-1)}].$$

This approach, however, has the drawback that the size of the input vector increases with the number of levels, and so do the number of parameters. Another possible generalization would be to explicitly represent information about descendants and ancestors of a vertex v (i.e., $ch[ch[v]]$, $ch[ch[ch[v]]]$, ..., and $pa[pa[v]]$, $pa[pa[pa[v]]]$, ...) into the input vector.

On the basis of this description, the training procedure for a Contextual SOM-SD can be described as follows:

Step 1 Train a SOM-SD network as described in Section 1.

Step 2 Build a new set of input vectors $\mathbf{x}$ through the concatenation of the data label $\boldsymbol{l}_v$, the coordinates of the mapping of child vertices $\boldsymbol{y}_{ch[v]}$, and the coordinates of the mapping of parent vertices $\boldsymbol{y}_{pa[v]}$ in the previously trained map. The vectors can be made constant in size if the maximum outdegree and the maximum indegree of any vertex in the dataset are known. Assuming that the maximum outdegree is o and the maximum indegree is q, then for vertices with less than o children and less than q parents, padding with a default value is applied. As a result, the $\mathbf{x}$ is an $k = p + 2o + 2q$ dimensional vector. The codebook vectors $\mathbf{m}$ for the new SOM-SD are of the same dimension.

Step 3 Train a new SOM-SD network as in Section (1) with the only difference that $\mathbf{\Lambda}$ in Equation 1 is now a $k \times k$ dimensional diagonal matrix with diagonal elements $\lambda_{1,1} \cdots \lambda_{p,p}$ set to μ_1, $\lambda_{p+1,p+1} \cdots \lambda_{m,m}$ set to μ_2, and all remaining diagonal elements are set to μ_3. The constant μ_3 influences the contribution of the parent coordinates to the Euclidean distance, while μ_1 and μ_2 control the influence of the data label component and the children's coordinates to the Euclidean distance.

Iterate through **Step 2** and **Step 3** at least d times, where d is the maximum length of the longest path between any two nodes in the graph.

Note that this approach, given a sufficiently large map, ensures that identical substructures that are part of different graph structures are mapped to different locations. It can be assumed that a map properly trained will map vertices to the same or nearby location only if the underlying graph structure is similar. Note also that the complexity of the training algorithm increases linearly with the size of the training set, the size of the network, and d. Hence, the approach provides a mechanism which is capable of finding similar or matching graphs in linear time.

The problem is that the approach is limited to data where the maximum indegree, outdegree, and d are known *a priori*. As a result, it is not possible to directly process cyclic or undirected graphs. In addition, the algorithm requires an appropriate choice for the weight value μ_3 which may need to be found through trial and error.

3 Experiments

We use an artificial set of graphs from [6]. The datasets in [6] provide a benchmark on which most of the recursive neural network architectures were tested. The dataset

used contains 1250 trees with a total of 9420 vertices or substructures. The maximum out-degree is six, the maximum indegree is one, and $d = 2$. The vertices of the tree are labeled by a two dimensional numerical value. Some of the trees are identical in structure and differ only in the label attached to the vertices. Hence, to capture the context of any sub-structure, it suffices to train a Contextual SOM-SD with 3 levels [1].

A large range of networks were trained. Here we illustrate some results when training a SOM-SD of size 24×16, which is as small as we could go without compromising too much on the mapping precision. Larger networks generally produced improved mappings but were not used here to ease the illustration task.

The network featured a hexagonal neighborhood and was trained for 200 iterations with an initial neighborhood radius of 29 and an initial learning rate 1.0. The resulting network is shown in Fig. 1. In Fig. 1, the hexagons refer to the codebook location, the fill color shows the level of activity of the codebook, where white implies that the associated codebook was not involved in the mapping of any sub-structure, darker colors indicate that more sub-structures were mapped at that location. The best matching sub-structure is superimposed on every neuron which was activated at least once. For example, the upper left map shows the SOM-SD at level 0. Root vertices were mapped at the upper left corner of this map, intermediate vertices were largely mapped near the right hand end of the map, whereas leaf vertices were mapped at the lower left.

Since at level 0 the SOM-SD is trained in a strictly causal manner where context information is available only from the children of a vertex, it can be assumed that many of the sub-structures which are identical in structure are mapped onto the same or nearby neurons even if the associated completed tree is of a very different architecture. To verify this claim: that through the contextual processing of patterns it becomes possible to diversify identical sub-structures according to the context in which they occurred inside the tree, we examined an arbitrarily chosen neuron at location 18,1 as an example (the neuron 18,1 is highlighted in the SOM-SD level 0 in Fig. 1) and found that 16 different trees were involved in the mapping of the same sub-structure at this location. We also found that the trees involved were of very different shape, where some featured just 6 vertices in total while others featured up to 11 vertices in different configurations.

The 16 trees were then selected and individually mapped at the SOM-SD at level 2. We also mapped all sub-structures of these trees. Two examples are given in Fig. 1. The lower two maps show the mapping of all sub-structures of two of the trees. The highlighted neuron shows the mapping of the sub-structure which was mapped at location 18,1 in the SOM-SD level 0. It can be observed, that despite of the fact that both trees (the zoomed structures) feature similar sub-structures, all of them are mapped onto a different location. The observation made here is of a general nature in that such observations were made for many other trees and sub-structures. It can be concluded that as a result each sub-structure is a representation of the associated tree structure as a whole, and hence, it is demonstrated that the proposed approach to the contextual processing of information is effective.

These visual observations were supported by quantitative performance measures computed using $e = 1/N \sum_{i=1,n_i \neq 0}^{N} m_i/n_i$ and $E = 1/N \sum_{i=1,n_i \neq 0}^{N} M_i/n_i$, where n_i is the number of sub-structures mapped at location i, m_i is the greatest number of

[1] Software and datasets used for the experiments are available from www.artificial-neural.net.

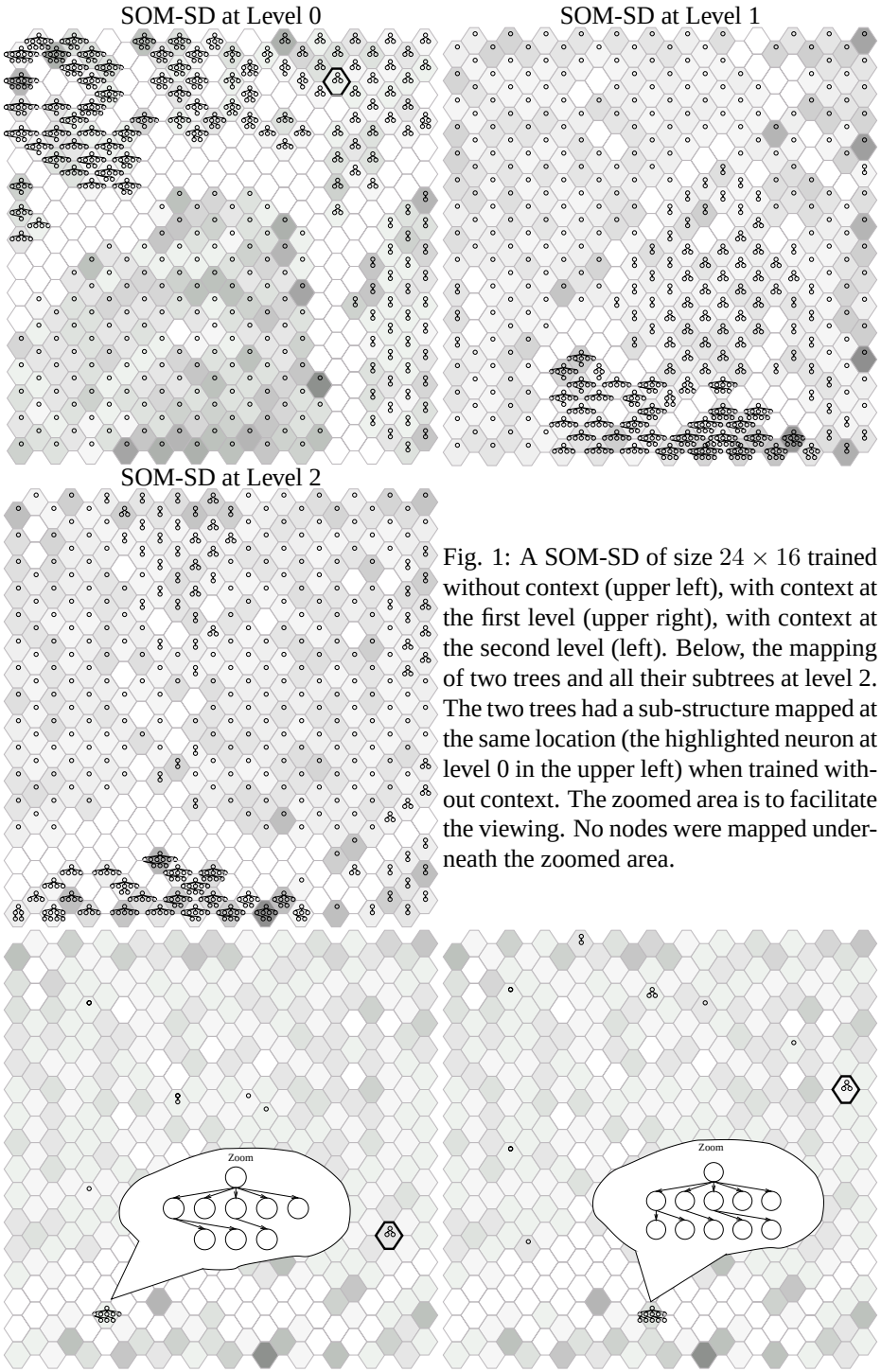

Fig. 1: A SOM-SD of size 24×16 trained without context (upper left), with context at the first level (upper right), with context at the second level (left). Below, the mapping of two trees and all their subtrees at level 2. The two trees had a sub-structure mapped at the same location (the highlighted neuron at level 0 in the upper left) when trained without context. The zoomed area is to facilitate the viewing. No nodes were mapped underneath the zoomed area.

Table 1: The mapping performance. Higher values indicate better performance.

Level	e	E
0	0.923	0.278
1	0.947	0.488
2	0.904	0.448

sub-structures which are identical in structure and are mapped at location i. Similarly, M_i is the greatest number of identical complete trees which are associated with the sub-structure mapped at location i. N is the total number of neurons activated by at least one sub-structure during the mapping process. Hence, e is an indicator of the quality of the mapping of sub-structures, and E indicates the quality of the contextual mapping process. Values in e and E can be within $(0; 1]$, where 1 indicates a *perfect* mapping, and a value closer to 0 indicates a poor mapping. Results illustrated in Table 1 show that e_i remains largely unchanged for the three levels, while E nearly doubles from level 0 to level 1 but remains unchanged between level 1 and level 2. The latter is mostly due to the limited size of the SOM-SD used in the experiments which was too small to allow a better diversification in the mappings. Other experiments with larger maps not shown here produced further significant increases for E at the higher level.

4 Conclusions

It was demonstrated that the proposed approach is effective in diversifying the mapping of vertices and sub-structures according to the context in which they occur inside a tree. The approach is simple in that it is a straightforward extension to a SOM-SD, and as such, leaves the computational complexity unchanged at a linear rate.

Future work will address the computational power of the proposed approach in greater detail, and will consider an online version of the contextual SOM-SD where the parent state is updated during the training procedure, and, hence, a single layer SOM-SD network should suffice to encode contextual information.

References

[1] T. Kohonen. *Self-Organizing Maps*, volume 30 of *Springer Series in Information Sciences*. Springer, Berlin, Heidelberg, 1995.

[2] G. Chappell and J. Taylor. The temporal kohonen map. In *Neural Networks*, number 6, pages 441–445, 1993.

[3] T. Voegtlin. Recursive self-organizing maps. *Neural Networks*, 15:979–992, 2002.

[4] 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.

[5] A. Micheli, D. Sona, and A. Sperduti. Contextual processing of structured data by recursive cascade correlation. *IEEE Transactions on Neural Networks*, 15:1396–1410, 2004.

[6] M. Hagenbuchner and A.C. Tsoi. A benchmark for testing adaptive systems on structured data. In Michel Verleysen, editor, *7th European Symposium on Artificial Neural Networks*, ISBN 2-9600049-9-X, pages pp. 63–68, Bruges, Belgium, April 1999. D-Facto.