A Supervised Self-Organizing Map for Structured Data*

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

[1] Faculty of Informatics, University of Wollongong, Australia
[2] Dip. di Informatica, Università di Pisa, Corso Italia, 40, 56125 Pisa, Italy

Abstract. A self organising map (SOM) for processing of structured data, using an unsupervised learning approach, called SOM-SD, has recently been proposed. Here, we suggest a new version of SOM, using the supervised learning approach. We compare the supervised version and the unsupervised version of SOM-SD on a benchmark problem involving visual patterns. As may be expected, the supervised version is able to solve the classification problem using very compact networks.

1 Introduction

The processing of structured data is usually confined to the domain of symbolic systems [2]. Recently, however, supervised neural networks for adaptive processing of structured data have been proposed [7,3]. These neural networks, called Recursive Neural Networks [7], are capable of learning a mapping from a domain of ordered directed acyclic graphs (DAGs), with labels attached to each node, to the set of real numbers. This unique ability has succesfully been exploited in Chemistry (e.g., QSAR studies), as well as in other application domains (for example [1,4]). Recursive neural networks are based on the idea of adaptive encoding of data structures. For example, in the special case of data structures being the set of labelled trees, the processing of the set of training samples begins from the leaf nodes, and progressively follows the graph structure until it reaches a root node. Each node could be represented by a multilayer feedforward neural network [7].

More recently, using a related approach, Self-Organizing Maps (SOMs) [6] have been extended to allow the mapping of structured objects into a topological map using the unsupervised learning approach[1]. It has been shown that this model, i.e., Self-Organizing Maps for Structured Data (SOM-SD), is able to arrange a set of labeled structures according both to structural similarities and similarities among the labels attached to nodes.

In this paper, we show how SOM-SD can also be trained using a supervised approach. This is a new direction in supervised learning to approach adaptive processing of structured data. So far, we are only aware of supervised learning methods using the Recursive Neural Network models. However, in this paper,

* Partially supported by MURST grant 9903244848 and MM09308497.
[1] *Self-Organizing Maps for Structured Data* manuscript in preparation.

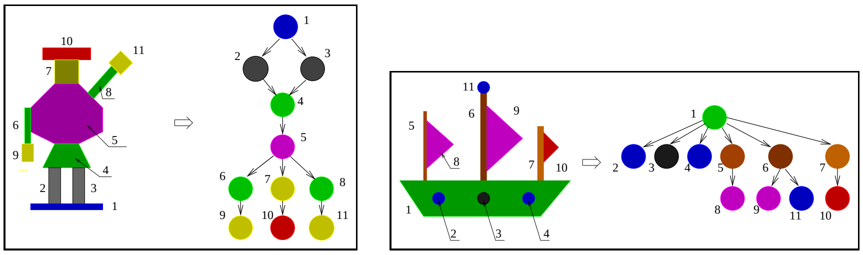


Fig. 1. Examples of DOAGs representing visual patterns. Labels are not shown.

we will show that it is possible to train the SOM-SD model using a supervised learning approach. We shall call this method sSOM-SD as opposed to the unsupervised version which we refer to as uSOM-SD.

There are several motivations for considering sSOM-SD instead of Recursive Neural Networks. First, the kind of supervision we are going to adopt is directly derived from the unsupervised algorithm, and, for this reason, it works with training sets where some of the examples may have unknown targets. In other words, sometimes the training examples may contain a mixture of target values, and missing target values. Moreover, sSOM-SD are able to map the training data structures to a low dimensional space and thus it is easier to understand how the network implements the solution to the computational task. Finally, with sSOM-SDs it is much easier to deal with many target classes, and it is easier to deal with the addition of new target classes.

2 Self-Organizing Maps for Structured Data

We shall use lowercase bold letters to denote vectors, uppercase bold letters to denote matrices, and calligraphic letters for representing graphs. A data structure $\mathcal{Y}$ is a directed acyclic graph (DAG) whose vertices are labelled by vectors of real-valued numbers which either represent numerical or categorical variables. Subscript notation will be used when referencing the labels attached to vertices in a data structure. Hence $\boldsymbol{y}_v$ denotes the vector labelling vertex $v \in \mathrm{vert}(\mathcal{Y})$, where $\mathrm{vert}(\cdot)$ denotes the set of vertices of the DAG $\mathcal{Y}$.

In this paper, we assume that instances in the learning domain are DOAGs (directed ordered acyclic graphs) or DPAGs (directed positional acyclic graphs). The *indegree* of node v is the number of incoming edges to v, whereas the *outdegree* of v is the number of outgoing edges from v.

We shall require the DAG (either DOAG or DPAG) to possess a supersource[2], i.e. a vertex $s \in \mathrm{vert}(\mathcal{D})$ such that every vertex in $\mathrm{vert}(\mathcal{D})$ can be reached by a directed path starting from s. Given a DAG $\mathcal{D}$ and $v \in \mathrm{vert}(\mathcal{D})$, we denote by $\mathrm{ch}[v]$ the set of children of v, and by $\mathrm{ch}_k[v]$ the k-th child of

[2] If no supersource is present, a new node connected with all the nodes of the graph with null *indegree* can be added.

v. We shall denote by $\#^{(c)}$ the class of DAGs with maximum outdegree c. The class of all data structures defined over the label universe domain $\boldsymbol{\mathcal{Y}}$ and skeleton in $\#^{(c)}$ will be denoted as $\boldsymbol{\mathcal{Y}}^{\#^{(c)}}$. The void DAG will be denoted by the special symbol ξ.

In Fig. 1, we show some examples of visual patterns represented as DOAGs. Each node of the graph represents a different colored component of the visual pattern, while edges are used to encode the concept of "connected with".

Here we are interested in a *feature map* $\widehat{\mathcal{M}} : \boldsymbol{\mathcal{Y}}^{\#^{(c)}} \rightarrow \boldsymbol{\mathcal{A}}$, where the input space is a structured domain with labels in $\boldsymbol{\mathcal{Y}}$. Specifically, we are interested in maps which can be realized by the following recursive definition:

$$\widehat{\mathcal{M}}(\mathcal{D}) = \begin{cases} nil_{\boldsymbol{\mathcal{A}}} & \text{if } \mathcal{D} = \xi \\ \mathring{\mathcal{M}}\left(\boldsymbol{y}_v, \widehat{\mathcal{M}}(\mathcal{D}^{(1)}), \ldots, \widehat{\mathcal{M}}(\mathcal{D}^{(c)})\right) & \text{otherwise} \end{cases}$$

where $nil_{\boldsymbol{\mathcal{A}}}$ is a special void coordinate into the discrete output space $\boldsymbol{\mathcal{A}}$, and $\mathring{\mathcal{M}}$: $\boldsymbol{\mathcal{Y}} \times \boldsymbol{\mathcal{A}}^c \rightarrow \boldsymbol{\mathcal{A}}$ is a SOM, defined on a generic node, which takes as input the label $\boldsymbol{y}$ of the node and the 'encoding' of the subgraphs $\mathcal{D}^{(1)}, \ldots, \mathcal{D}^{(c)}$ according to the $\widehat{\mathcal{M}}$ map. $\widehat{\mathcal{M}}(\mathcal{D})$ can be computed by starting to apply $\mathring{\mathcal{M}}$ to leaf nodes (nodes with null outdegree), and proceeding with the application of $\mathring{\mathcal{M}}$ bottom-up from the frontier to $s \in \text{vert}(\mathcal{D})$

Assuming that each label in $\boldsymbol{\mathcal{Y}}$ is encoded in $\boldsymbol{\mathcal{U}} \subset \mathbb{R}^m$, for each node v in $\text{vert}(\mathcal{D})$, we have a vector $\boldsymbol{u}_v$ of dimension m. Moreover, the display output space $\boldsymbol{\mathcal{A}}$, where $\boldsymbol{\mathcal{A}} \equiv [1..n_1] \times [1..n_2] \times \cdots \times [1..n_q]$, is realized through a q dimensional lattice of neurons[3]. Thus, the winning neuron is represented by the coordinates $(i_1, .., i_q)$.

With the above assumptions, we have $\mathring{\mathcal{M}}$: $\mathbb{R}^m \times ([1..n_1] \times \cdots \times [1..n_q])^c \rightarrow [1..n_1] \times \cdots \times [1..n_q]$, and the $m + cq$ dimensional input vector $\boldsymbol{v}$ to $\mathring{\mathcal{M}}$, representing the information about a generic node v, is defined as

$$\boldsymbol{v} = \left[\boldsymbol{u}_v \ \mathcal{D}_{ch_1[v]} \ \mathcal{D}_{ch_2[v]} \ \cdots \ \mathcal{D}_{ch_c[v]}\right],$$

where $\mathcal{D}_{ch_i[v]}$ denotes the coordinates vector[4] of the winning neuron for the subgraph pointed by the i-th pointer of v. Each neuron with coordinates vector $\boldsymbol{c}_j$ in the q dimensional lattice has an associated vector weight $\boldsymbol{w}_{\boldsymbol{c}_j} \in \mathbb{R}^{m+cq}$. The weights associated with each neuron in the q dimensional lattice $\mathring{\mathcal{M}}$ can be trained using the standard SOM's learning process. The training algorithm for $\widehat{\mathcal{M}}$ is shown in Fig. 2.

The coordinates for the (sub)graphs are stored in $\mathcal{D}_v$, once for each processing of graph $\mathcal{D}$, and then used when needed[5] for the training of $\mathring{\mathcal{M}}$. The

[3] In this paper, experiments are performed by using $q = 2$.

[4] The null pointer $nil_{\boldsymbol{\mathcal{A}}}$ can be defined, for example, by the vector with all components at -1.

[5] Notice that the use of an inverted topological order guarantees that the updating of the coordinate vectors $\mathcal{D}_v$ is performed before the use of $\mathcal{D}_v$ for training.

Unsupervised Stochastic Training Algorithm for $\widehat{\mathcal{M}}$

input: Set of training DAGs $T = \{\mathcal{D}_i\}_{i=1,\ldots,N}$, c maximum outdegree of DAGs in T, map $\mathring{\mathcal{M}}$;
begin
randomly set the weights for $\mathring{\mathcal{M}}$;
repeat
  randomly select $\mathcal{D} \in T$ with uniform distribution;
  `List(`$\mathcal{D}$`)` $\leftarrow$ an inverted topological order for vert($\mathcal{D}$);
  for v $\leftarrow$`first(List(`$\mathcal{D}$`))` **to** `last(List(`$\mathcal{D}$`))`**do**
    train($\mathring{\mathcal{M}}$ $\left(\left[\boldsymbol{u}_v\ \mathcal{D}_{ch_1[v]}\ \mathcal{D}_{ch_2[v]}\ \cdots\ \mathcal{D}_{ch_c[v]}\right]\right)$);
    $\mathcal{D}_v \leftarrow \mathring{\mathcal{M}}\ (\left[\boldsymbol{u}_v\ \mathcal{D}_{ch_1[v]}\ \mathcal{D}_{ch_2[v]}\ \cdots\ \mathcal{D}_{ch_c[v]}\right])$;
end

Fig. 2. The unsupervised SOM-SD algorithm.

vector is an approximation of the true coordinate vector for the graph rooted in v, however, if the learning rate η is small, this approximation may be negligible. The call $train()$ refers to a single step of standard SOM training.

3 The Supervised SOM-SD

Given a training set $T = \{(\mathcal{D}_i, t_i)\}_{i=1,..,N}$, the SOM-SD model can be used in a supervised mode as follows. Let us consider the data structure $\mathcal{D}_i$ with desired value t_i, the input vector for each node of this data structure is extended by adding a new field containing the information about the target:

$$\boldsymbol{v} = \left[\boldsymbol{u}_v\ \mathcal{D}_{ch_1[v]}\ \mathcal{D}_{ch_2[v]}\ \cdots\ \mathcal{D}_{ch_c[v]}\ \boldsymbol{t}_i\right],$$

where $\boldsymbol{t}_i \in I\!R^r$ is a real valued vector representing the target t_i. In this way, the information associated to data structures in the same target class $\tilde{t}$, will have identical last field $\tilde{\boldsymbol{t}}$. Thus, using the unsupervised algorithm on these new representations, the model will partition the neurons in subsets, each one associated with a different target class. We can trivially state that the sSOM-SD computes the a posteriory probability of a data structure given the target class and the training set, i.e., $p(\mathcal{D}_i|t_i, T)$.

After unsupervised training, the neurons in the map are labelled by the most frequent target associated to the data structures for which the neuron is the winner, i.e., given a neuron $n \in \mathring{\mathcal{M}}$, and the set $Data(n,t) = \{\mathcal{D}|(\mathcal{D}, t) \in T$ and n wins the competition on $supersource(\mathcal{D})\}$, let $WinnerLabel(n) = \arg\max_{t \in C} |Data(n,t)|$. The label associated to each neuron is then used to classify a new data structure $\mathcal{D}$. Specifically, the nodes of $\mathcal{D}$ are presented to $\mathring{\mathcal{M}}$, having extended the input vector with the null vector (i.e., the target

Training Algorithm for $\widehat{\mathcal{M}}$

input: $T = \{(\mathcal{D}_i, t_i)\}_{i=1,\dots,N}$, c (maximum outdegree of DAGs in T), $\mathring{\mathcal{M}}$;
begin
call the unsupervised algorithm on the extended data;
/* Labelling of Nodes */
foreach $n \in \mathring{\mathcal{M}}$;
$label(n) \leftarrow WinnerLabel(n)$;
end

Classification Algorithm for $\widehat{\mathcal{M}}$

input: $\mathcal{D}$;
begin
`L` $\leftarrow$ an inverted topological ordered list for vert($\mathcal{D}$);
for $v \leftarrow$`first(L)` **to** `last(L)`**do**
$\mathcal{D}_v \leftarrow \mathring{\mathcal{M}}\left(\left[\boldsymbol{u}_v\ \mathcal{D}_{ch_1[v]}\ \mathcal{D}_{ch_2[v]}\ \cdots \cdots\ \mathcal{D}_{ch_c[v]}\ \mathbf{0}\right]\right)$;
$w \leftarrow winner(supersource(\mathcal{D}))$;
return $label(w)$;
end

Fig. 3. The training and classification algorithms for the supervised mode.

value is unknown), and the label associated with the winning neuron for the supersource of $\mathcal{D}$ is returned. The training and classification algorithms for the supervised mode are described in Fig. 3.

4 Experiments

In this section we compare the performance of the uSOM-SD vs. the sSOM-SD model on a classification task with 12 classes. The basic idea is to "measure" the difficulty of the problem via the uSOM-SD model (without using the target information) and then to compare the results with the sSOM-SD model (using the target information).

The data set was taken from the *Policemen Benchmark* [5] which we extended by merging patterns from three different domains (houses, ships, policemen) to increase the complexity of the learning problem, and the number of possible instances (classes). In essence, the *Policemen Benchmark* [5] is a set of labeled DOAGs extracted from images produced by means of a context free attributed plex grammar. Structures are extracted by employing a simple scan line algorithm where images are scanned from bottom to top, and from left to right. The first colored object found constitutes the root node. Objects directly connected form the offsprings. The algorithm proceeds recursively until all objects in an image are considered. Each node created receives a two dimensional label stating the $\{x, y\}$ coordinate of the center of gravity of the represented object.

As a result, the dataset provided graphs with the following properties: Policemen have a maximum outdegree of 3, Houses have 5, and Ships have 6; the minimum (maximum) depth for Policemen is 4 (5), for Houses is 2 (3), for Ships is 1 (2); the minimum (maximum) number of nodes for Policemen is 9 (11), Houses 4 (7), and Ships 3 (13). Hence, policemen patterns produce

Class 'a': policemen with rised left arm ×
Class 'b': policemen with lowered left arm ○
Class 'c': ships with 2 masts ✳
Class 'd': ships with all 3 masts ●
Class 'e': houses without windows ◇
Class 'f': houses with 1 window at L-L corner ▲
Class 'g': houses with 1 window at U-L corner ▼
Class 'h': houses with 1 window at U-R corner □
Class 'i': houses with 2 windows at L-L and U-L corners ▽
Class 'j': houses with 2 windows at L-L and U-R corners +
Class 'k': houses with 2 windows at U-L and U-R corners △
Class 'l': houses with all 3 windows ■

Fig. 4. Definition of the 12 classes and their symbols as used later in Fig. 5. L-L means "lower left", U-L "upper left", and U-R "upper right".

deep narrow graphs, ships and houses have a flat and wide data structure. Some graph structures produced by the ships and houses are identical in structure whereas there is no graph structure derived from policemen images that is contained in the domain houses or ships. Thus, some patterns can be distinguished only through features encoded in the labels. For example, when considering policemen, the location of the arm is not encoded in the graph structure, but in the data label, while the graph structure is not affected.

These three domains were combined to form one training set so that the maximum outdegree of all nodes in the training set is 6. As a result (assuming $q = 2$) every input vector $\boldsymbol{v}$ is of dimension $2 + 2 \times 6$. The training (test) set contained $3,750$ $(3,750)$ graphs with a total of $29,808$ $(29,984)$ nodes.

For the supervised task, 12 classes were defined as shown in Fig. 4. The number of examples for each class both in the training and test sets were as in the following table:

	'a'	'b'	'c'	'd'	'e'	'f'	'g'	'h'	'i'	'j'	'k'	'l'
Training Set	645	605	937	313	28	59	58	172	53	170	188	522
Test Set	670	580	944	306	21	59	64	195	71	173	194	473

Experiments were designed to demonstrate the typical behaviour of sSOM-SD networks in comparison with that of an uSOM-SD network. Networks were trained on the same data set where class memberships formed an additional input when using a supervised training regime – class information was omitten for the unsupervised method. However, to allow the evaluation of the network performance and a comparison with sSOM-SD, we labeled the neurons by a symbolic class label after training the network. All results described in this section were obtained by SOM-SD networks with hexagonal topology and the Gaussian neighborhood function. Networks ranging from a size of 102×118 down to a size 2×5 have been trained. Table 1 displays the performance of these networks. It shows that sSOM-SD network perform with 100% correct classification rate down to a very small network size with just 200 neurons whereas the uSOM-SD performs hardly at a level of 90% on a network with more than 60 times as many neurons.

In Fig. 5 we display where root nodes are mapped onto the network. It is observed that an uSOM-SD maps root nodes close to the upper right corner of a network. This is mainly due to the fact that the NULL identifier for missing offsprings for leaf nodes has been chosen to be -1 and hence, are values that are nearest to the origin of the map. Thus, leaf nodes tend to be mapped closer to the lower left end of a map. For sSOM-SD networks this is different. Here, the class information has a strong influence on where a root node is mapped. As it can been observed from the plot, some classes are mapped onto a single neuron, whereas others form relatively large clusters. This is due to the fact that some classes are represented through a larger number of different instances than others. For example, there are many different instances of ships (e.g. many ways of arranging the masts, sails, flags) than there are instances for houses which often look very similar. Concerning intermediate and leaf nodes (not shown in Fig. 5), they activate different regions of the maps.

We also observered how a sSOM-SD network represents data on a much smaller output scale. With smaller networks, the training algorithm gradually forces data to be mapped onto smaller clusters until almost all classes are mapped onto a single neuron.

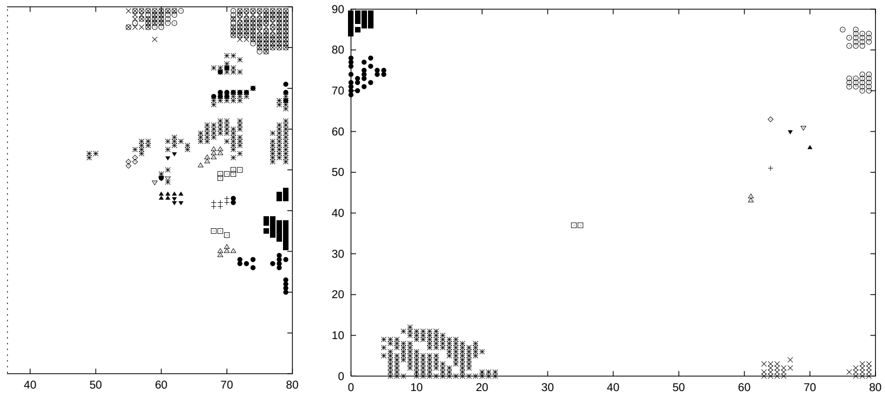
5 Discussion and Conclusions

In this paper, we demonstrated a new way of addressing supervised learning problem involving structured objects through an extention to the uSOM-SD algorithm. Specifically, we showed how class memberships of structured data can be incorporated in the learning process to yield improved network performances.

The methodology described in this paper has distinctive advantages over many other supervised learning methods. The algorithm allows to process data with incomplete or missing class labels without further modification. Moreover, it has been demonstrated that a sSOM-SD is able to efficiently represent structured information in a highly reduced manner by mapping it

Size (w x h = # nodes)	sSOM-SD (train-test)	uSOM-SD (train-test)
102 x 118 = 12036	100% - 100%	91.547% - 89.227%
80 x 90 = 7200	100% - 100%	88.347% - 87.36%
60 x 70 = 4200	100% - 100%	86.853% - 85.227%
40 x 50 = 2000	100% - 100%	85.305% - 84.107%
20 x 30 = 600	100% - 100%	77.547% - 74.773%
10 x 20 = 200	100% - 100%	63.067% - 62.933%
5 x 10 = 50	93.173% - 93.12%	54.933% - 54.72%
2 x 5 = 10	80.587% - 79.28%	47.387% - 46.693%

Table 1. Comparison between sSOM-SD and uSOM-SD on the supervised task.



unsupervised : *root nodes*   *supervised* : *root nodes*

Fig. 5. Unsupervised and supervised maps of dimension 80×90. Only root nodes are shown; symbols refer to the class assigned to the nodes. For the unsupervised map, only the portion of the map activated by root nodes is shown.

onto a relatively small, often two dimensional output space. It has become evident that through incorporating class information into the learning process, we are able to improve the network performance considerably over the unsupervised counterpart.

References

1. A.M. Bianucci, A. Micheli, A. Sperduti, and A. Starita. Analysis of the internal representations developed by neural networks for structures applied to qsar studies of benzodiazepines. *Journal of Chemical Information and Computer Sciences*, 41(1):202–218, 2001.
2. H. Bunke and A. Sanfeliu, editors. *Syntactic and Structural Pattern Recognition : Theory and Applications.* Series in computer science ; v. 7. Singapore ; New Jersey : World Scientific, c1990. ISBN-9971505525.
3. P. Frasconi, M. Gori, and A. Sperduti. A general framework for adaptive processing in data structures. In *IEEE Trans on Neural Networks*, volume Vol 9, pages 768–785, 1998.
4. C. Goller. *A Connectionist Approach for Learning Search-Control Heuristics for Automated Deduction Systems.* PhD thesis, Technical University Munich, Computer Science, 1997.
5. M. Hagenbuchner and A-C. Tsoi. The traffic policeman benchmark. In Michel Verleysen, editor, *European Symposium on Artificial Neural Networks*, ISBN 2-9600049-9-X, pages 63–68. D-Facto, April 1999.
6. T. Kohonen. *Self-Organisation and Associative Memory.* Springer, 3rd edition, 1990.
7. A. Sperduti and A. Starita. Supervised neural networks for classification of structures. *IEEE Trans Neural Networks*, Vol. 8(No. 3):714–735, 1997.