Self-organising Map Techniques for Graph Data Applications to Clustering of XML Documents

A.C. Tsoi[1], M. Hagenbuchner[2], and
A. Sperduti[3]

[1] Monash e-Research Centre, Monash University
Victoria 3800, Australia
ahchung.tsoi@adm.monash.edu.au
[2] Faculty of Informatics, University of Wollongong,
Wollongong NSW 2522, Australia
markus@uow.edu.au
[3] Dipartimento di Matematica Pura ed Applicata, University of Padova,
Via G.B. Belzoni, 7, 35131 Padova, Italy
sperduti@math.unipd.it

Abstract. In this paper, neural network techniques based on Kohonen's self-organising map method which can be trained in an unsupervised fashion and applicable to the processing of graph structured inputs are described. Then it is shown how such techniques can be applied to the problems of clustering of XML documents.

1 Introduction

Neural networks have been one of the main techniques used widely in data mining. There are a number of popular neural network architectures, e.g. multilayer perceptrons [5], self organising maps [3], support vector machines [6]. However, most of these techniques have been applied to problems in which the inputs are vectors. In other words, the inputs to these neural network architectures are expressed in the form of vectors, often in fixed dimensions. In case the inputs are not suitably expressed in the form of vectors, they are made to conform to the fixed dimension vectorial format. For example, it is known that an image may be more conveniently expressed in the form of a graph, for instance, the image of a house can be expressed as a tree, with the source node (level 0) being the house, windows, walls, and doors expressed as leaves (level 1), and details of windows, walls and doors being expressed as leaves (level 2) of those leaves located in level 1, etc. These nodes are described by attributes (features, which may express colour, texture, dimensions) and their relationships with one another are described by links. Such inputs can be made to conform to a vectorial format if we "flatten" the structure and instead represent the information in each node in the form of a vector, and obtain the aggregate vector by concatenating the vectors together. Such techniques have been prevalent in the application of neural network architectures to these problems.

Recently, there have been some attempt in preserving the graph structured data as long as we can before applying the neural network technique. For example, in support vector machines, there have been some work in expressing the graph structured data in the form of string kernels [4], or spectrum kernels [4], and then use the "kernel trick" [6] in using the support vector machine machinery to process the data. Alternatively, another way to process the data is to preserve the graph structured data format, and modify the neural network techniques to process graph structured data. In this paper, we will not consider support vector machine further, and we will concentrate only on ways to modify a classic neural network technique, self- organising maps, so that it can accept graph structured inputs.

The structure of this paper is as follows: in Section 2, we will describe ways in which the classic self organising map idea can be extended to consider graph structured data, in what we called a self organising map for structured data (SOM-SD) technique and contextual self-organising map for structured data (CSOM-SD) technique. In Section 3, we will describe applications of these techniques to clustering XML documents. Some conclusions will be drawn in Section 4.

2 Self-organizing Map for Structured Data

The self-organising map concept [3] was developed to help identify clusters in multidimensional, say, p-dimensional datasets. The SOM does this by effectively packing the p-dimensional dataset onto a q-dimensional display plane, where we assume for simplicity $q = 2$ throughout this paper. The SOM consists of discretising the display space into $N \times N$ grid points, each grid point is associated with a p-dimensional vector, referred to in this paper, as an artificial neuron, or simply a neuron [1]. Each neuron has an associated p-dimensional vector, called a codebook vector. This codebook vector $\boldsymbol{m}$ has the same dimension as the i-the input vector $\boldsymbol{x}_i$. The neurons on the map are bound together by a topology, which is often either hexagonal or rectangular. The contents of these vectors are updated with each presentation of samples from the p-dimensional original data set. The contents of these vectors encode the relationships (distances) among the p-dimensional data. The result is that data items that were "similar" or "close" to each other in the original multidimensional data space are then mapped onto nearby areas of the 2-dimensional display space. Thus SOM is a topology-preserving map as there is a topological structure imposed on the nodes in the network. A topological map is simply a mapping that preserves neighborhood relations.

In general, the SOM is a model which is trained on a set of examples in an unsupervised fashion as follows:

For every input vector $\boldsymbol{x}_i$ in a training set, obtain the best matching codebook by computing

[1] This is called a neuron for historical reasons.

$$c = \arg\min_j \|\boldsymbol{x}_i - \mathbf{m}_j\| \tag{1}$$

where $\|\cdot\|$ denotes the Euclidean norm.

After the best matching unit $\mathbf{m}_c$ is found, the codebook vectors are updated. $\mathbf{m}_c$ itself as well as its topological neighbours are moved closer to the input vector in the input space i.e. the input vector attracts them. The magnitude of the attraction is governed by a learning rate α and by a neighborhood function $f(\Delta_{jc})$, where Δ_{jc} is the topological distance between $\mathbf{m}_c$ and $\mathbf{m}_j$. As the learning proceeds and new input vectors are given to the map, the learning rate gradually decreases to zero according to a specified learning rate function type. Along with the learning rate, the neighborhood radius decreases as well. The codebooks on the map are updated as follows:

$$\Delta\mathbf{m}_j = \alpha(t)f(\Delta_{jc})(\mathbf{m}_j - \mathbf{x}_i) \tag{2}$$

where α is a learning coefficient, and $f(.)$ is a neighborhood function which controls the amount by which the weights of the neighbouring neurons are updated. The neighborhood function can take the form of a Gaussian function $f(\Delta_{jc}) = \exp\left(-\frac{\|\mathbf{l}_j - \mathbf{l}_c\|^2}{2\sigma^2}\right)$ where σ is the spread, and $\mathbf{l}_c$ and $\mathbf{l}_j$ is the location of the winning neuron and the location of the j-the neuron respectively. Other neighborhood functions are possible. Equations (1) and (2) are computed for every input vector in the training set, and for a set number of iterations.

The SOM for Data Structures (SOM-SD) extends the SOM in its ability to encode directed tree structured graphs [1]. This is accomplished by processing individual nodes of a graph one at a time rather than by processing a graph as a whole. The network response to a given node v is a mapping of v on the display space. This mapping is called the *state* of v and contains the coordinates of the winning neuron. An input vector representation is created for every node in a graph G by concatenating a numeric data label $\boldsymbol{l}_v$ which may be attached to a node v with the *state* of each of the node's immediate offsprings such that $\boldsymbol{x}_v = [\boldsymbol{l}_v\ \boldsymbol{y}_{\mathrm{ch}[v]}]$, where $\mathrm{ch}[v]$ denotes the children of node v, and $\boldsymbol{y}_{\mathrm{ch}[v]}$ denotes the states or mappings of the children of v. The dimension of $\boldsymbol{x}$ is made constant in size by assuming a maximum dimension for $\boldsymbol{l}$ together with a maximum out-degree of a node. For nodes with less dimensions than the assumed, padding with a suitable value is applied. Since the initialization of $\boldsymbol{x}$ depends on the availability of all the children states, this dictates the processing of nodes in an inverse topological order (i.e. from the leaf nodes towards the root nodes), and hence, this causes information to flow in a strictly causal manner (from the leaf nodes to the root nodes).

A SOM-SD is trained in a similar fashion to the standard SOM with the difference that the vector elements $\boldsymbol{l}$ and $\boldsymbol{y}_{\mathrm{ch}}$ need to be weighted so as to control the influence of these components to a similarity measure. Equation (1) is altered to become:

$$c = \arg\min_j(\|(\boldsymbol{x}_v - \boldsymbol{m}_j)\mathbf{\Lambda}\|) \tag{3}$$

where $\boldsymbol{x}_v$ is the input vector for vertex v, $\boldsymbol{m}_i$ the i-the 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 $\lambda_{p+1,p+1} \cdots \lambda_{m,m}$ set to μ_2. The constants μ_1 and μ_2 control the influence of $\boldsymbol{l}_v$ and $\boldsymbol{y}_{\mathrm{ch}[v]}$ to the Euclidean distance in (3).

The rest of the training algorithm remains the same as that of the standard SOM. The effect of this extension is that the SOM-SD will map a given set of graphs, and all sub-graphs onto the same map. The SOM-SD includes the standard SOM and the SOM for data sequences as special cases.

With contextual SOM for graphs (CSOM-SD), the network input is formed by additionally concatenating the state of parent nodes and children nodes to an input vector such that $\boldsymbol{x}_v = [\boldsymbol{l}\ \boldsymbol{y}_{\mathrm{ch}[v]}\ \boldsymbol{y}_{\mathrm{pa}[v]}]$, where $\boldsymbol{y}_{\mathrm{pa}[v]}$ are the states of the parent nodes and $\boldsymbol{y}_{\mathrm{ch}[v]}$ are the states of the children nodes. The problem with this definition is that a circular functional dependency is introduced between the connected vertices v and $\mathrm{pa}[v]$, and so, neither the state for node v nor the state of its parents $\mathrm{pa}[v]$ can be computed. One possibility to compute these states could be to find a joint stable fix point to the equations involving all the vertices of a structure. This could be performed by initializing all the states with random values and then updating these initial states using the above mentioned equations, till a fixed point is reached. Unfortunately, there is no guarantee that such a fixed point would be reached. Moreover, even if sufficient conditions can be given over the initial weights of the map to guarantee stability, i.e. the existence of the fixed point, there is no guarantee that training will remain valid on such sufficient conditions over the weights.

A (partial) solution to this dilemma has been proposed in [2]. The approach is based on an K-step approximation of the dynamics described above: Let

$$\boldsymbol{y}^t = h(\boldsymbol{x}_v^{t-1}), t = 1, \ldots, K \tag{4}$$

where $h(\cdot)$ computes the state of node v by mapping the input $\boldsymbol{x}_v^{t-1}$, and $\boldsymbol{x}_v^{t-1} = [\boldsymbol{l}_v\ \boldsymbol{y}_{\mathrm{ch}[v]}^{t-1}\ \boldsymbol{y}_{\mathrm{pa}[v]}^{i-1}]$. The algorithm is initialized by setting $\boldsymbol{y}_{\mathrm{ch}[v]}^0 = \boldsymbol{y}_{\mathrm{pa}[v]}^0 = k$, where $k = [-1, -1]$, an impossible winning coordinate. In other words, the approach iteratively re-computes the states of every node in a graph K-times. Then, the network input can be formed by setting $\boldsymbol{x}_v = [\boldsymbol{l}\ \boldsymbol{y}_{\mathrm{ch}[v]}^K\ \boldsymbol{y}_{\mathrm{pa}[v]}^K]$. A suitable value for K could be, for instance, the maximum length of any path between any two nodes in the graph. Although such a choice does not guarantee the full processing of contextual information due to possible latency in the transfer of contextual information from one vertex of the structure to its neighbors vertices, this value for K seems to be a good tradeoff between contextual processing and computational cost.

Training is performed similar to the training of SOM-SD with the difference that $\boldsymbol{\Lambda}$ is now a $n \times n$ matrix, $n = \dim(\boldsymbol{x})$ with $\lambda_{m+1,m+1} \cdots \lambda_{n,n}$ set to the constant μ_3. All other elements in $\boldsymbol{\Lambda}$ are the same as defined before.

3 Experiments

The corpus (`m-db-s-0`) considered consists of 9,640 XML formatted documents which were made available as part of the INEX Initiative (INitiative for the Evaluation of XML Retrieval). Each of the XML formatted documents describes an individual movie (e.g. the movie title, list of actors, list of reviewers, etc.). It was built using the IMDB database. Each document is labelled by one thematic category which represents the genre of the movie in the original collection and one structure category. There are 11 thematic categories and 11 possible structure categories which correspond to transformations of the original data structure. Note that the target labels are used solely for testing purposes, and hence, are ignored during the training process.

A tree structure was extracted for each of the documents in the dataset by following the general XML structure within the documents. The resulting dataset featured $9,640$ tree structured graphs, one for each XML document in the dataset. The maximum depth of any graph is 3, the maximum outdegree is $6,418$, and the total number of nodes in the dataset is $684,191$. Hence, the dataset consists of shallow tree structures which can be very wide. A three-dimensional data label is attached to every node in the dataset indicating the XML-tag it represents (more on this below). There were a total of 197 different tags in the dataset.

While for the SOM-SD and CSOM-SD there is no specific need to pre-process this set of graphs, we decided to apply a pre-processing step in order to reduce the dimensionality of the dataset. This allows for a reasonable turn around time for the experiments. Dimension reduction was achieved by consolidating XML tags as follows: Repeated sequences of tags within the same level of a structure are consolidated. For example, the structure:

```
<BB>
    <a></a>
    <b></b>                                   <BB>
    <a></a>                                       <a></a>
    <b></b>        is consolidated to             <b></b>
    <a></a>                                   </BB>
    <b></b>
</BB>
```

A justification for taking this step is inspired by operations in regular expressions. For example, the expression $(ab)^n$ can be simulated by repeatedly presenting ab n-times. Hence, it suffices to process the consolidated structure n times. There were many trees which exhibited such repeated sequences of tags. The consequence of this pre-processing step is that the maximum outdegree is reduced to just 32.

A further dimension reduction is achieved by collapsing sequences into a single node. For example, the sequential structure `<A><b><c></c></b></A>` can be collapsed to `<A><b&c></b&c></A>`, and further to `<A&b&c>`. Since the deepest graph is of depth 3, this implies that the longest sequence that can be collapsed is of length 3. This pre-processing step reduces the total number of nodes in the dataset to $247,140$.

A unique identifier (ID) is associated with each of the 197 XML tags. In order to account for nodes which represent collapsed sequences, we attach a three dimensional data label to each node. The first element of the data label gives the ID of the XML tag it represents, the second element of the data label is the

Label	Frequency
1	598
10	386
11	448
2	486
3	701
4	172
5	435
6	231
7	261
8	769
9	333
Total	4820

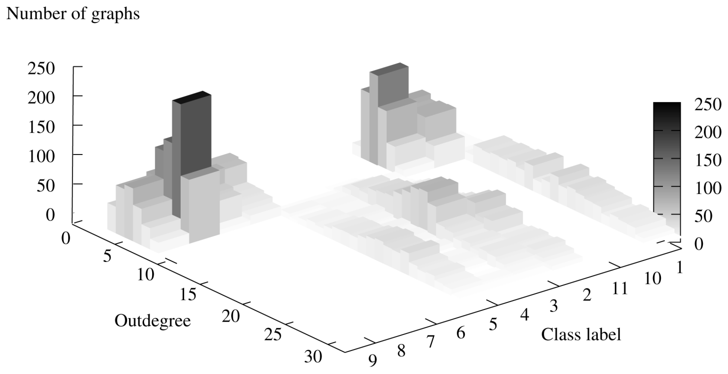

Fig. 1. Properties of the training set: The table (left) shows the number of graphs in each of the 11 classes. The plot (right) shows the distribution of outdegrees in the dataset. Shown are the number of graphs (z-axis) which have a given outdegree (y-axis) and belong to a given class (x-axis).

ID number of the first tag of a collapsed sequence of nodes, and consequently, the third element is the ID of the tag of the leaf node of a collapsed sequence. For nodes which do not represent a collapsed structure, the second and third element in the data label will be set to zero.

The resulting dataset consists of 4,820 graphs containing a total of 124,360 nodes (training set), and 4,811 graphs containing a total of 122,780 nodes (test set). The training set was analysed for its statistical properties; and the results are shown in Figure 1. It is observed that the training set is unbalanced. For example, the table on the left of Figure 1 shows that there are only 172 samples of the pattern instance denoted by "4" but over 700 instances of patterns from the instance denoted by "3". Also, the 3-D plot in Figure 1 shows that the distribution of outdegrees can vary greatly. For example, there is only one instance in the pattern class denoted by "8" which has an outdegree of 10 while there are over 270 instances for the same pattern class with outdegree 5. There are also a number of pattern classes which are similar in features such as class "10" and class "11" which are of similar size and are of similar structure.

There are 2,872 unique sub-structures in the training set. This is an important statistical figure since it gives an indication to how much more information is provided to a SOM-SD when compared to the flat vectors used for the SOM. And hence, the larger the number of unique sub-structures in the training set, the greater the potential diversification in the mapping of the data will be. Similarly, there are 96,107 unique nodes in different contextual configurations in the training set. This shows that the CSOM-SD is provided with a greater set of diverse features in the training set, and hence, may be capable of diversifying in the mapping of the data even further. Thus, this dataset provides a challenging learning problem on which various SOM models will be tested.

All SOMs illustrated in this section used a hexagonal topology, and a Gaussian neighborhood function. For the SOM-SD and the CSOM-SD, when generating the input vectors $\boldsymbol{x}_i$ for nodes with less than the maximum outdegree, padding was performed using the impossible coordinate $[-1, -1]$.

The standard SOM is trained on $4,820$ data *vectors*, each one represents an XML document. The i-the element in the data vector represents the frequency of the i-the XML tag within a document. Thus, the input vectors for the SOM are 197 dimensional containing all the information about the XML tags in a document but do not contain any information about the topological structure between the XML tags.

Thus, the SOM is trained on relatively few high-dimensional data vectors while the SOM-SD or the CSOM-SD is being trained on a large number of nodes which are represented by a relatively small size vectors. For the SOM we chose $64 \times 48 = 3,072$ as the size of the network. The total number of network parameters for the SOM is $3,072 \times 197 = 605,184$. Since the codebook dimensions for the SOM-SD is $3+32\times 2 = 67$, this implies that a SOM-SD needs to feature at least $9,033$ codebooks to allow a fair comparison. Accordingly, the CSOM-SD should feature at least $8,771$ neurons. However, since the SOM-SD (and to an even greater extent the CSOM-SD) is to encode a larger feature set which includes causal (contextual) information about the data, this implies that the SOM-SD (CSOM-SD) will potentially diversify the mapping of the data to a greater extent than a SOM would do. Hence, this would justify the choice of even larger networks for the SOM-SD and CSOM-SD respectively for the comparisons. However, we chose to use these network sizes as these suffice to illustrate the principal properties of the models.

It is evident that a simple quantization error is an insufficient indicator of the performance of a SOM-SD or a CSOM-SD since such an approach neglects to take into account the fact that structural information is being mapped. In fact, there are a number of criteria with which the performance of a SOM-SD or a CSOM-SD can be measured.

Retrieval capability (R): This reflects the accuracy of retrieved data from the various SOM models. This can be computed quite simply if for each XML document d_j a target class $y_j \in \{t_1, \ldots, t_q\}$ is given. Since each XML document is represented by a tree, in the following, we will focus our attention just on the root of the tree. Thus, with r_j we will refer to the input vector for SOM, SOM-SD or CSOM-SD representing the root of the XML document d_j. The R index is computed as follows: the mapping of every node in the dataset is computed; then for every neuron i the set $win(i)$ of root nodes for which it was a winner is computed. Let $win_t(i) = \{r_j | r_j \in win(i) \text{ and } y_j = t\}$, the value $R_i = \max_t \frac{|win_t(i)|}{|win(i)|}$ is computed for neurons with $|win(i)| > 0$ and the index R computed as $R = \frac{1}{W} \sum_{i,|win(i)|>0} R_i$, where $W = \sum_{i,|win(i)|>0} 1$ is the total number of neurons which were activated at least once by a root node.

Classification performance (C): This can be computed as follows:

$$C_j = \begin{cases} 1 \text{ if } y_j = t_r^*, & t_r^* = \arg\max_t |win_t(r)| \\ 0 \text{ else} \end{cases},$$

where r is the index of the best matching codebook for document d_j (typically measured at the root node). Then,

$$C = \frac{1}{N}\sum_{j=0}^{N} C_j,$$

where N is the number of documents (graphs) in the test set. Values of C and R can range within $(0:1]$ where values closer to 1 indicate a better performance.

Clustering performance (P): A more sophisticated approach is needed to compute the ability of a SOM-SD or a CSOM-SD to suitably group data on the map. In this paper the following approach is proposed:

1. Compute the quantities R_i as defined above, and let $t_i^* = arg\max_t |win_t(i)|$.
2. For any activated neuron compute the quantity:

$$P_i = \frac{\sum_{j=1}^{|\mathcal{N}_i|} \frac{|win_{t_i^*}(j)|}{|win(j)|} + \frac{|win_t(i)|}{|win(i)|}}{|\mathcal{N}_i| + 1} = \frac{\sum_{j=1}^{|\mathcal{N}_i|} \frac{|win_{t_i^*}(j)|}{|win(j)|} + R_i}{|\mathcal{N}_i| + 1}$$

   where $\mathcal{N}_i = \{v | v \in \mathrm{ne}[i], win(v) \neq \emptyset\}$.
3. The overall neural network performance is then given by:

$$P = \frac{\sum_i P_i}{W}.$$

A performance value close to 1 indicates a perfect grouping, while a value closer to 0 indicates a poor clustering result. Thus, this measure indicates the level of disorder inside a SOM-SD or CSOM-SD.

Structural mapping precision (e and E): These indices measure how well structural (e) and contextual structural (E) information are encoded in the map. A suitable method for computing the structural mapping precision was suggested in [2]. In this case, just the skeleton of the trees is considered, i.e. the information attached to vertices is disregarded, and only the topology of the trees is considered. Notice that these measures do not consider the information about the class to which an XML document (i.e., a tree) belongs. For this reason, all the neurons of a map are now considered, since we are also interested in neurons which are winners for sub-structures. These two measures e and E are respectively computed as follows

$$e = \tfrac{1}{N}\textstyle\sum_{i=1,n_i\neq 0}^{N} \tfrac{m_i}{n_i} \qquad \text{and} \qquad E = \tfrac{1}{N}\textstyle\sum_{i=1,n_i\neq 0}^{N} \tfrac{M_i}{n_i}$$

where n_i is the number of sub-structures mapped at location i, m_i is the greatest number of sub-structures which are identical 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 of e and E close to 1 indicate a very good mapping (indeed a *perfect* mapping if the value is 1), and values closer to 0 indicate a poor mapping.

Compression ratio: This is the ratio between the total number of root nodes in the training/test set, and the number of neurons actually activated by root nodes in the training/test set. The higher the compression, the fewer the number of neurons are involved in the mapping process. Extremely high or extremely low compression ratios can indicate a poor performance. The compression ratio can vary between 0 and N, where N is the number of root nodes in the training/test set.

A number of SOMs, SOM-SDs, and CSOM-SDs were trained by varying the training parameters, and initial network conditions. We used the classification measure C as a general benchmark on which to optimize the performance of the various models. A total of 56 experiments were executed for each of the SOM models, and every experiment was repeated 10 times using a different random initialization of the map as a starting point. The experiments varied the following training parameters: number of training iterations i, initial neighborhood radius $r(0)$, initial learning rate $\alpha(0)$, and the weight values μ (in this order). The set of training parameters which maximised the classification performance of the three models is shown below.

	size	# iterations	$\alpha(0)$	$r(0)$	μ_1	μ_2	μ_3
SOM	64×48	32	1.0	4	1.0	–	–
SOM-SD	110×81	62	1.0	38	0.11	0.89	–
CSOM-SD	110×81	12	0.7	15	0.11	0.88	0.01 [2]

It is observed that the SOM-SD required more training iterations and a larger initial neighborhood radius to achieve optimum classification performance (on the training set). It was also observed that the classification performance of the CSOM-SD improved with smaller values for μ_3 reaching an optimum for $\mu_3 = 0.0$. However, setting μ_3 to zero would reduce the CSOM-SD to a SOM-SD, and hence, would be an unsuitable choice for the comparisons. In this case we have set μ_3 to a small value.

Table 1. Best results obtained during the experimentation with maps of size 64×48 (SOM), and for maps of size 110×81 (SOM-SD and CSOM-SD)

	train set						test set					
	C	R	P	e	E	Z	C	R	P	e	E	Z
SOM	90.5%	0.90	0.73	1.0	1.0	2.45	76.5%	0.92	0.73	1.0	1.0	2.45
SOM-SD	92.5%	0.92	0.78	0.77	0.50	5.13	87.3%	0.93	0.79	0.76	0.50	4.9
CSOM-SD	83.9%	0.87	0.73	0.91	0.30	8.53	78.6%	0.88	0.71	0.90	0.37	8.54

The performances of the three SOM models are illustrated in Table 1 with the above mentioned performance measures. From Table 1 it can be observed that a standard SOM is capable of classifying over 90% of patterns in the training

[2] Smallest non-zero value tried. Setting $\mu_3 = 0.0$ resulted in a better classification performance but would reduce the CSOM-SD to a SOM-SD.

set correctly despite of no information about the underlying causal or contextual configuration of XML tags is provided to the training algorithm. However, it was found that the SOM generalizes poorly. In comparison, the SOM-SD improved the classification rate by a noticeable amount, and was able to generalize over unseen data very well. As is observed from the compression ratio Z, the performance increase of the SOM-SD comes despite a doubling of the compression ratio. This is a clear evidence that causal information about the order of XML tags allows (a) to diversify the mapping of nodes to a considerably larger extend, and (b) the diversification in the mappings can result in an overall improvement of the classification or clustering performances. In contrast, the inclusion of contextual information did not help to improve on the classification performance as it is observed from the results obtained from the CSOM-SD. It is found that contextual information helped to diversify the mapping of nodes by almost double when compared to the SOM-SD. This is indicated by the larger compression ratio. Thus, it is evident that a correct classification of the graphs in the dataset is independent of the contextual information about the XML tags within the original documents. When paired with the greater data compression which is the result of a greater diversification in the mapping of nodes, this produced a relative overall reduction in classification performance for the CSOM-SD, and explains the observation that the performance optimum of the CSOM-SD is at $\mu_3 = 0$.

In addition, it is observed that a CSOM-SD performs worse on the performance measure E than a SOM-SD. This is a result which arose out of the fact that the experiments were to optimize the classification performance C. It was found that a CSOM-SD improves on C when using $\mu_3 \to 0$. However, setting $\mu_3 = 0$ would reduce the CSOM-SD to a SOM-SD and would have denied us from making a comparison between the models. Instead, we chose a small value for μ_3 so as to allow such comparisons, and still produce reasonable classification performances. Using a very small μ_3 reduces the impact of contextual information to the training algorithm. When paired with the increased compression ratio in the mapping of root nodes, this resulted in a relative decrease in the performance on E. Note that the standard SOM performed at $e = E = 1$. This is due to the fact that a SOM handles the simplest type of data structures (viz. single nodes). These render all structures in the dataset identical, resulting in the observed performance values.

A closer look at the mapping of (training) data is made in the standard SOM Figure 2(a). The hexagons in Figure 2(a) refer to the neurons on the map. The brightness of the grid intersection represents the number of training data which are assigned to the grid point due to their closeness in the original input space. Thus by examining the brightness in the grid, it is possible to gain an appreciation of the way the given training dataset can be grouped together, according to their closeness in the original input space. Every neuron is also filled in with a pattern indicating the class that most frequently activated the neuron. There are 11 different fill in patterns for the 11 possible classes. Neurons which are not filled in are not activated by any vector in the training set. It can be observed that a number of well distinct clusters have formed on the map,

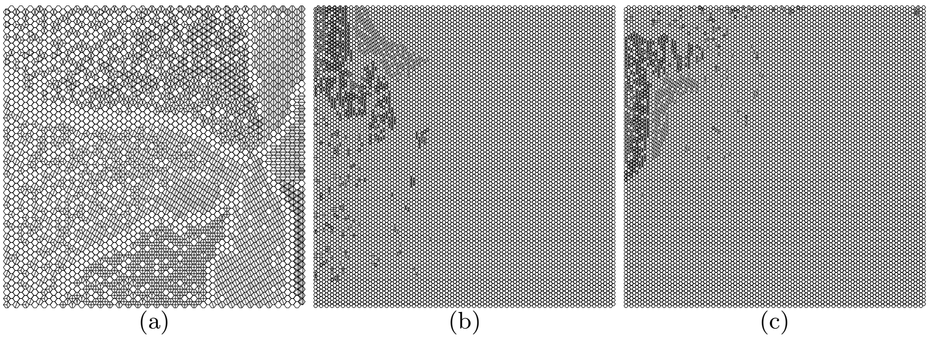

Fig. 2. The mapping of the training vectors on a standard SOM is shown in (a). The mapping of root nodes (training vectors) on a SOM-SD is shown in (b). The mapping of root nodes (training vectors) on a CSOM-SD is shown in (c).

most of which correspond nicely with the target label that is associated with the training data. Most clusters are separated from each other by an area of neurons which were not activated. This may indicate a good result since the presence of such border regions should allow for a good generalization performance; a statement which could not be confirmed when evaluating the test set.

In comparison, the mapping of root nodes in the training set on a trained SOM-SD is shown in Figure 2(b). Neurons which are not filled in are either not activated by a root node, or are activated by a node other than the root node. It can be observed in Figure 2(b) that large sections of the map are not activated by any root node. This is due to the fact that root nodes are a minority in the dataset. Only $4,824$ nodes out of the total $124,468$ nodes in the training set are root nodes. Hence, only a relatively small portion of the map is activated by root nodes. It is also observed that graphs belonging to different classes form clear clusters some of which are very small in size. This observation confirms the experimental findings which show that the SOM-SD will be able to generalize well.

Figure 2(c) gives the mapping of the root nodes as produced by the CSOM-SD. Again, it is found that the largest portion of the map is filled in by neurons which are either not activated or are activated by nodes other than the labelled root nodes. Clear clusters are formed which are somewhat smaller in size when compared to those formed in the SOM-SD case. This illustrates quite nicely that the CSOM-SD is compressing the "root" data considerably more strongly than the SOM-SD since contextual information is also encoded which requires additional room in the map. Nevertheless, the observation confirms that the CSOM-SD will also be able to generalize well even though some of the performance indices may be worse than when compared to a SOM-SD of the same size. This can be expected since the CSOM-SD compresses the "root" data more strongly.

4 Conclusions

The clustering of graphs and sub-graphs can be a hard problem. This paper demonstrated that the clustering task of general types of graphs can be performed in linear time by using a neural network approach based on Self-Organizing Maps. In addition, it was shown that SOM-SD based networks can produce good performances even if the map is considerably smaller than the size of the training set. Using larger maps will generally improve the performance further though this was not illustrated in this paper.

Specifically, it was shown that the given learning problem depends on the availability of causal information about the XML tags within the original document in order to produce a good grouping or classification of the data. The incorporation of contextual information did not help to improve on the results further.

The training set used in this paper featured a wide variety of tree structured graphs. We found that most graphs are relatively small in size, only few graphs were either very wide or featured many nodes. This creates imbalances in features represented in a training set which is known to negatively impact the performance of a neural network. Similarly it is true when considering the way we generated data labels for the nodes. An improvement of these aspects (i.e. balancing the features in the training set, using unique labels which are equiv-distant to each other) should help to improve the network performances. An investigation into the effects of these aspects is left as a future task.

Furthermore, it was shown that the (C)SOM-SD models map graph structures onto a finite regular grid in a topology preserving manner. This implies that similar structures are mapped onto nearby areas. As a consequence, these SOM models would be suitable for inexact graph matching tasks. Such applications are considered as a future task.

References

1. M. Hagenbuchner, A. Sperduti, and A. Tsoi. A self-organizing map for adaptive processing of structured data. *IEEE Transactions on Neural Networks*, 14(3):491–505, May 2003.
2. M. Hagenbuchner, A. Sperduti, and A. Tsoi. Contextual processing of graphs using self-organizing maps. In *European symposium on Artificial Neural Networks*, Poster track, Bruges, Belgium, 27 - 29 April 2005.
3. T. Kohonen. *Self-Organizing Maps*, volume 30 of *Springer Series in Information Sciences*. Springer, Berlin, Heidelberg, 1995.
4. C. Leslie, E. Eskin, and W. Noble. Spectrum kernel: A string kernel for svm protein classification. *Proceedings of the Pacific Symposium on Biocomputing*, pages 474–485, 2002.
5. D. MacKay. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 1st edition, 2003.
6. S. A. Schlkopf, B. *Learning with Kernels*. MIT Press, Cambridge, MA, 1st edition, 2002.