ESANN 2011 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 27-29 April 2011, i6doc.com publ., ISBN 978-2-87419-044-5. Available from http://www.i6doc.com/en/livre/?GCOI=28001100817300.

Sparsity Issues in Self-Organizing-Maps for Structures

Markus Hagenbuchner[1], Giovanni Da San Martino[2], Ah Chung Tsoi[3] and Alessandro Sperduti[2] *

1- School of Computer Science and Software Engineering
University of Wollongong - Australia

2- Department of Pure and Applied Mathematics
University of Padova - Italy

3- Faculty of Information Technology
Macau University of Science and Technology - Macau SAR, China

Abstract. Recent developments with Self-Organizing Maps (SOMs) produced methods capable of clustering graph structured data onto a fixed dimensional display space. These methods have been applied successfully to a number of benchmark problems and produced state–of–the–art results. This paper discusses a limitation of the most powerful version of these SOMs, known as probability measure graph SOMs (PMGraphSOMs), viz., the sparsity induced by processing a large number of small graphs, which prevents a successful application of PMGraphSOM to such problems. An approach using the idea of compactifying the generated state space to address this sparsity problem is proposed. An application to an established benchmark problem, viz., the Mutag dataset in toxicology will show that the proposed method is effective when dealing with a large number of small graphs. Hence, this work fills a gap between the processing of a number of small graphs, and the processing of densely connected graphs using PMGraphSOMs.

1 Introduction

Self-Organizing Maps (SOMs) are one of the best known machine learning methods [1]. The SOMs have been very successful in tasks requiring the clustering or projection of high–dimensional data onto low dimensional display space. Some of the reasons of its success may be attributed to the fact that the learning algorithm is scalable (growing linearly with the size of a learning problem), it does not require ground truth information during the training phase, and the result is a topology preserving mapping which allows a direct visualization of the high dimensional data [1].

In its original form, the SOM was developed as an unsupervised clustering method for vectorial data [1]. Over time, the SOM has gradually been enhanced to allow the clustering of data sequences [1], ordered tree structured data [2], and directed or undirected graphs [3]. More recent work enabled the SOM to cluster generic graphs [4, 5, 6]. The *Probability Measure Graph-SOM* (PMGraphSOM) is arguably the most powerful SOM since it is a scalable approach capable of dealing with datasets as simple as a set of vectors and as complex as a set of one or more graphs which may contain directed or undirected links (or a mixture of both), cyclic structures, nodes that may be labeled by a numeric value, nodes featuring an arbitrary number of in- or out-degrees, etc [5]. In fact, the PMGraphSOM has been applied to a number of benchmark problems and has achieved state–of–the–art performances [4, 5, 7, 8].

*The work presented in this paper has been supported by the Australian Research Council in form of a Discovery Project grant (DP0774168) and a Linkage International Awards Grant (LX0882106).

However, a shortcoming of the PMGraphSOM was detected when it was used to cluster sparse graphs. Here we consider a graph is "sparse" if the ratio between the number of nodes and the number of links is small. As it will be discussed in section 3, the PMGraphSOM internally represents the neighbourhood of a node i by a vector of the same size of the PMGraphSOM map. The number of non-zero elements of such a vector is proportional to the number of neighbours of i. When the size of the PMGraph-SOM map is much larger than the out-degree of a node, which is verified in almost any practical application, the vector representing the node i tends to be sparse. This, in turn, increases the probability for any pair of input nodes to be considered orthogonal by the PMGraphSOM, thus preventing an effective learning process. In fact, the more sparse the input is, the more does the PMGraphSOM reduce itself to just memorizing the input data (rather than clustering them together). This paper will analyse the sparsity problem of the PMGraphSOM, and propose an approach to overcome such situations.

This paper is organized as follows: The PMGraphSOM is described in Section 2. The enhancement allowing a deployment of PMGraphSOM to accept sparse graph inputs is proposed in Section 3. Experiments and a comparison with results obtained by other researchers is offered in Section 4, and some conclusions are drawn in Section 5.

2 The PMGraphSOM

The PMGraphSOM maintains many similarities with Kohonen's original SOM: it consists of an n–dimensional display space [1], the display space is formed by a set of codebooks which are arranged as a regular grid, and the training algorithm consists of a two–step procedure which processes vectorial inputs. The differences are: PMGraph-SOM processes nodes in a graph and the input vectors are formed dynamically in order to account for the dependencies of a node on other nodes. The PMGraphSOM allows the processing of graphs the nodes of which are labeled by a numeric vector. The input $\mathbf{x}_i$ associated with the i-th node in a graph is formed by concatenating any existing label $\mathbf{l}_i$ that may be attached to the i-th node with a *state* vector $\mathbf{M}_{ne[i]}$. $\mathbf{M}_{ne[i]}$ encodes the information about the neighbors directly connected to it [2]. Let us first consider the training algorithm of a PMGraphSOM to explain the formation of $\mathbf{M}_{ne[i]}$. Given a set of codebook vectors organized on a discrete two-dimensional grid of size $x_1 \times x_2$. There is one codebook vector for each grid point and the dimension of all codebook vectors is the same as the dimension of the input vectors $\mathbf{x}$. The codebook vectors are initialized with random values. The codebook vectors are updated according to the following two–step training procedure:

1. **Competitive step:** Randomly select a node i from any graph in the training dataset, form the input vector $\mathbf{x}_i = (\mathbf{l}_i, \mathbf{M}_{ne[i]})$, then find the best matching codebook value using:

$$r = \arg\min_j \left\{ \mu_1 \|\mathbf{x}_i - \mathbf{m}_j\|^2 + \mu_2 \sum_{\ell=1}^{x_1 \times x_2} \sum_{k \in ne[\ell]} \frac{1}{\sqrt{2\pi}\sigma(t)} \exp\left\{ -\frac{\|\mathbf{c}_\ell - \mathbf{c}_k\|^2}{2\sigma(t)^2} \right\} \right\} \tag{1}$$

[1] Without loss of generality, this paper will assume that $n = 2$.

[2] A node is said to be a *direct neighbor* if a link exists to a given node.

ESANN 2011 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 27-29 April 2011, i6doc.com publ., ISBN 978-2-87419-044-5. Available from http://www.i6doc.com/en/livre/?GCOI=28001100817300.

where $\mathbf{c}_\ell$ denotes the coordinates of the neuron in the node which is connected to the current node n. $\sigma(t)$ is a monotonically decreasing function in t. The constants μ_i, $i = 1, 2$ are designed to moderate the effect of the influence of the current node or those in the neighborhood of the current node n. The winning codebook $\mathbf{m}_r$ is said to be *activated* by the current input.

2. **Cooperative step:** All codebook vectors are updated by $\Delta\mathbf{m}_s = \alpha(t)f(\Delta_{sr})(\mathbf{m}_s - \mathbf{x}_i)$, where $\alpha(t)$ is a learning rate which decreases to zero with time t, $f(\Delta_{sr}$ is said to be a neighborhood function which is commonly defined as $f(\Delta_{sr}) = \exp(-\|\mathbf{c}_s - \mathbf{c}_r)\|^2/2\sigma(t)^2)$, $\mathbf{c}_s$ is the coordinate of the s-th codebook in the map, $\mathbf{c}_r$ is the coordinate of the winning codebook, σ is a parameter called the *neighborhood radius* which decreases to 1 with time t.

These two steps are repeated for a given number of training iterations. Note that when processing directed graphs, the input vector is formed by setting $\mathbf{x}_i = (\mathbf{l}_i, \mathbf{M}_{pa[i]}, \mathbf{M}_{ch[i]})$, where $pa[i]$ refers to the parent nodes of node i, and $ch[i]$ the child nodes of the same node respectively. The algorithm produces an activation for each presented node. This activation is the network's response to the given input. Once an activation is computed for each node in a training dataset, this information is used to initialize the state vector $\mathbf{M}$. In other words, $\mathbf{M}$ provides information about the mappings of a node's neighbors. $\mathbf{M}$ forms a part of an input vector, and the PMGraphSOM encodes a node's label as well as the context within which it occurs in a graph. Applied iteratively, and to all nodes, the context of a node is eventually passed to all other (connected) nodes in a graph. Note that this procedure allows the PMGraphSOM to encode any number of nodes which may or may not be connected by links. This means that the PMGraphSOM considers a set of graphs as a single graph consisting of disconnected sub–graphs. To account for the Euclidean similarity measure used, the activations of a node's neighbors are encoded by the state vector as follows: $M_i = 1/(\sigma(t)\sqrt{2\pi}) \cdot \exp(-\|\mathbf{c}_s - \mathbf{c}_r)\|^2/2\sigma(t)^2)$, where M_i is the i-th element of the state vector. This means that the dimension of M is $x_1 \times x_2$ (the number of codebooks on the map). The advantage of this approach is that the dimension of the input vectors becomes independent of the degree of a node. However, this can have a profound effect when processing data consisting of many small graphs.

3 Self-Organizing Maps for Sparse Graphs

In general, the more complex the learning problem is, the larger the map needs to be, and the dimension of the state vector grows with the size of the map. This is fine for learning problems featuring graphs which are richly connected. A problem arises when attempting to encode a large set of small graphs. In such cases, although the dimensionality of $\mathbf{M}$ is large, since each node has only few neighbors, $\mathbf{M}$ represents the encoding of only a few nodes. Moreover, during training, the neighborhood radius $\sigma(t)$ decreases to 1, and hence the elements in $\mathbf{M}$ become increasingly sparse, until towards the end of the training process there are only $n_{[ne]}$ non-zero elements in $\mathbf{M}$, where $n_{[ne]}$ is the number of neighbors of a node. This means that $\mathbf{M}$ becomes very sparse. The problem with this is that the Euclidean distance measure used can no longer differentiate between the mappings of neighbors of two different nodes. The same problem existed in GraphSOM, a predecessor of PMGraphSOM which was shown to have the afore–mentioned deficiency [9]. A possible approach for the elimination of the sparse graph problem lies in the explicit listing of coordinate values for all of a node's neighbors. The approach is similar to the one taken by self–organizing map for structured data (SOM-SD) [2]. However, the SOM-SD can only process directed ordered graphs. This is because the SOM-SD lists the coordinate values in order of the occurrence of the offsprings of a node (the first coordinate values correspond to the first offspring of a node, the second coordinate values to the second offspring, and so on). Thus, the order of these coordinate values is very important since it could limit the capability of a PMGraphSOM to process unordered and undirected graphs. Since the aim is to maintain similarities between (sub–)graphs rather than to maintain any order that may exist in a graph, we propose to *sort* the coordinate values prior to further processing. Thus, the state vector $\mathbf{M}$ in Eq(1) becomes a sorted list of coordinate values. This approach is valid for the following reasons: (a) the dimension of $\mathbf{M}$ gives the maximum number of neighbors of any node (the degree of a graph). Since we are dealing with sparse graphs, the degree is either known or can be estimated. Padding with a set of illegal coordinate values, such as (-1,-1), can be used for nodes with less than the maximum number of neighbors. (b) a PMGraphSOM is known to map similar sub–structures to nearby regions on the map, and hence, the sorting of coordinate values ensures that the mapping of similar sub-structures (rooted by the neighbors of a node) is listed in order of similarity. (c) the dimension of the new state vector $\mathbf{M}$ is small since we deal with sparse graphs. When dealing with densely connected graphs we can fall back on using the original PMGraphSOM approach [5] instead.

The method presented in this section requires the sorting of the state vector. Since the size of the state vector depends on the connectivity of a graph, and since good sorting algorithms have a linear-logarithmic complexity, the proposed approach does not scale very well with the level of connectivity of a graph. However, in practice, this is not an issue since we assume to deal with sparse graphs (which contain only few links). In the following, we will refer to a PMGraphSOM with the proposed compact representation of state information as *compact PMGraphSOM*.

4 Experiments and Comparisons

The proposed approach is applied to a challenging benchmark problem consisting of a set of chemical molecules, viz. the Mutag benchmark problem [10]. The Mutag problem consists of a classification task; currently the best known supervised approach performs in the region of 90% accuracy. We will use this dataset since it features molecule structures which can readily be represented as graphs, and since the number of connections is limited. Due to the complexity of the learning problem, when applying a Self-Organizing Map algorithm, this would normally require a larger display space. Hence, the Mutag dataset is ideally suited to illustrate the sparse graph problem of PMGraphSOM and the effectiveness of the proposed approach. The Mutag dataset consists of a total of $3,371$ nodes in 188 graphs. To reflect the well known properties of atoms, the maximum number of connections (the degree) for any one node in the graph is 4.

Several maps were trained by varying the μ weight values, learning rate, network size, and neighborhood radius. We kept the number of training iterations constant at 100, and used a hexagonal neighborhood relationship for all experiments. The observations made were that the compact PMGraphSOM significantly improved the diversi-

ESANN 2011 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 27-29 April 2011, i6doc.com publ., ISBN 978-2-87419-044-5. Available from http://www.i6doc.com/en/livre/?GCOI=28001100817300.

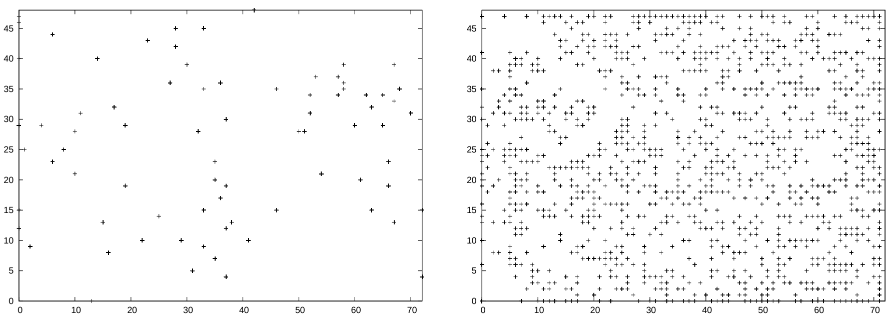

Fig. 1: Mapping of the nodes in the mutag dataset using PMGraphSOM (left), and by using the compact state representation (right).

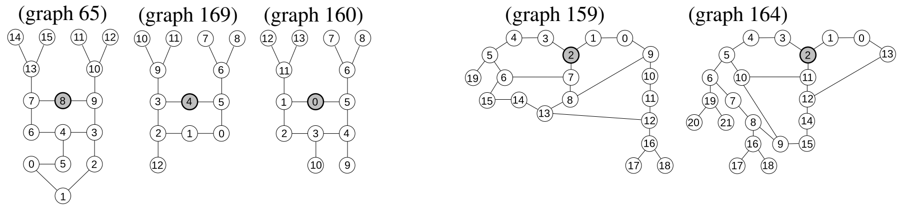

Fig. 2: Three graphs rooted by the gray node as they were found mapped at the lower left corner of the map (left), and two graphs (rooted by the gray node) which were mapped at a distant location in the upper right corner of the same map.

fication of the mappings, and hence, resulted in a considerably improved network utilization. Figure 1 provides a visualization of a typical observation for a network of size 76×48. We undertook two approaches in order to obtain a deeper insight into whether the improved diversification actually resulted in useful mappings. First, we used the target labels that were available with this dataset to compute the classification performance of these SOMs. For the maps shown in Figure 1, the classification performance is 75.38% for the PMGraphSOM, and 84.10% for the compact PMGraphSOM. This is a very respectable result given that the SOMs were trained unsupervised, and provides a good indication on the effectiveness of the proposed method. Secondly, we looked at nodes that were mapped nearby on the map generated by the compact PMGraphSOM. An example of the observation made is depicted in Figure 2. The three leftmost graphs shown in Figure 2 were mapped at coordinates (10,0), (11,0), and (12,0) respectively. These coordinates refer to codebooks located at the lower left corner of the map which were activated by at least one node. The gray colored node is the node that activated the codebook. It can be observed that the three nodes are located in a very similar context within the respective graph. In contrast the nodes and associated graph mapped at the coordinate (70,47), (71,47) near the upper right corner of the map are shown on the righthand side of Figure 2. Again, it can be observed that the nodes appear in a very similar context within the respective graphs. The figure also made it clear that the nodes mapped in different regions of the map occur in a very different context. This

ESANN 2011 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 27-29 April 2011, i6doc.com publ., ISBN 978-2-87419-044-5. Available from http://www.i6doc.com/en/livre/?GCOI=28001100817300.

visually confirms that the compact PMGraphSOM creates useful topology preserving mapping. Moreover, we also observed that the training times were generally shorter for the compact PMGraphSOM. This is simply due to the fact that the input dimension for the PMGraphSOM is dominated by the 76×48-dimensional state vector, whereas the state vector of the compact PMGraphSOM was just 4×2-dimensional.

5 Conclusions

In this paper, we have presented one way to overcome an issue when we apply PM-GraphSOM to a large number of small graphs. This approach is applied to a benchmark problem of classification in a toxicology dataset: the Mutag dataset. It is found that the proposed approach is effective in dealing with sparse graphs by improving network utilization and the quality of the mappings. Future work includes a formal investigation of the sparsity problem. Such investigation would identify the conditions under which the proposed compact PMGraphSOM is superior to the PMGraphSOM. Further future work includes the extension of the compact PMGraphSOM to be trained using a supervised learning approach. By providing the neural network architecture with more informative feedback through a target value, hopefully the supervised version of the compact PMGraphSOM will achieve results which are comparable, if not better than other approaches to tackling this benchmark problem.

References

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

[2] 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.

[3] M. Hagenbuchner, A. Sperduti, and A.C. Tsoi. Contextual processing of graphs using self-organizing maps. In *European symposium on Artificial Neural Networks*, 27 - 29 April 2005.

[4] L. Di Noi, M. Hagenbuchner, F. Scarselli, and A. C. Tsoi. Web spam detection by probability mapping graphsoms and graph neural networks. In Konstantinos Diamantaras, Wlodek Duch, and Lazaros Iliadis, editors, *Artificial Neural Networks - ICANN 2010*, volume 6353 of *Lecture Notes in Computer Science*, pages 372–381. Springer Berlin / Heidelberg, 15-18 September 2010.

[5] S. Zhang, M. Hagenbuchner, A.C. Tsoi, and A. Sperduti. Self organizing maps for the clustering of large sets of labeled graphs. In N. Fuhr et al., editor, *LNCS 4862, Lecture Notes in Computer Science*, pages 207–221, Berlin, 2009. Springer-Verlag Berlin Heidelberg.

[6] S. Günter and H. Bunke. Self-organizing map for clustering in the graph domain. *Pattern Recognition Letters*, 23(4):405–417, 2002.

[7] M. Kc, M. Hagenbuchner, A.C. Tsoi, F. Scarselli, M. Gori, and S. Sperduti. Xml document mining using contextual self-organizing maps for structures. In *Lecture Notes in Computer Science*, volume 4518, pages 510–524. Springer-Verlag Berlin Heidelberg, 2007.

[8] M. Hagenbuchner, A. Sperduti, A.C. Tsoi, F. Trentini, F. Scarselli, and M. Gori. Clustering xml documents using self-organizing maps for structures. In N. Fuhr et al., editor, *LNCS 3977, Lecture Notes in Computer Science*, pages pp. 481–496. Springer-Verlag Berlin Heidelberg, 2006.

[9] M. Hagenbuchner, S. Zhang, A.C. Tsoi, and A. Sperduti. Projection of undirected and non-positional graphs using self organizing maps. In *European Symposium on Artificial Neural Networks - Advances in Computational Intelligence and Learning*, pages 559–564, Bruges, Belgium, 2009.

[10] A.K. Debnath, R.L. Lopez de Compadre, G. Debnath, A.J. Shusterman, and C. Hansch. Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. correlation with molecular orbital energies and hydrophobicity. *J. Med. Chem.*, 34:786–797, 1991.