Growing Self Organizing Map with an Imposed Binary Search Tree for Discovering Temporal Input Patterns

Sumith Matharage[1], Upuli Gunasinghe[2] and Damminda Alahakoon[3]
[1,3]Cognitive and Connectionist Systems Lab, Faculty of IT, Monash University and [2]Department of Computer Science and Engineering, University of Moratuwa
[1]sumith.matharage@gmail.com, [2]upulipg@uom.lk and [3]Damminda.Alahakoon@infotech.monash.edu.au

Abstract – **In this paper the Binary Search Tree Imposed Growing Self Organizing Map (BSTGSOM) is presented as an extended version of the Growing Self Organizing Map (GSOM), which has proven advantages in knowledge discovery applications. A Binary Search Tree imposed on the GSOM is mainly used to investigate the dynamic perspectives of the GSOM based on the inputs and these generated temporal patterns are stored to further analyze the behavior of the GSOM based on the input sequence. Also, the performance advantages are discussed and compared with that of the original GSOM.**

I. INTRODUCTION

The family of Self Organizing Maps (SOMs) was pioneered and built by Kohonen and it has been extended in numerous disciplines throughout the time. The Growing Self Organising Map [1] is a significant milestone in the SOM extensions and has addressed its main limitations, the pre-selection of the architecture and excessive training time requirement, of the original SOM. In addition, a method of controlling the growth of the map was introduced in the GSOM using a spread factor and this has resulted in data analysts' having more control over the map to carry out their analysis.

GSOM has been used in many different domains such as Bioinformatics and Text Mining with proven benefits, because of its capacity for large-scale data handling, speed of processing and hierarchical clustering. In addition, many different extensions of GSOM [2, 3, 4] have been developed enhancing the original algorithm. But, none of them have addressed the temporal organization of GSOM based on the current input, focusing only on the overall arrangement of the GSOM.

The Binary Search Tree Imposed Growing Self Organizing Map provides a set of temporal tree based organizations, representing an individual input, in addition to the overall map. A single tree arrangement is a snapshot of the GSOM based on its current input. It will represent the relationships of the already created neurons in the network for the input to the network at the time that the snapshot is taken. The snapshot is a temporal arrangement specially focused on a particular input. So the temporal tree arrangements for each and every input are generated while training.

If this temporal behavior is analyzed, it represents the relationships of the already learnt knowledge to the new knowledge. It is well focused for the present input and arranged in a way that, the similar objects are closer to the new input and the distinct ones are further away from that. The GSOM only provides a global view of the map for all inputs with generalized relationships so that it caters to all inputs and presents them in a global manner. But the BSTGSOM provides temporal arrangements focused on the individual inputs, which are not available in any of the GSOM based algorithms, and would significantly contribute to knowledge discovery applications.

In addition, the BSTGSOM provides performance advantages over the original GSOM and its derivatives through reducing the number of calculations required for calculating the winning node. The imposed tree building can be used to find out the winner during the training and it will reduce the time to locate the winner. The performance analysis is discussed after the algorithm is presented.

The BSTGSOM algorithm, performance analysis, experiments and results and open problems are described in the next sections.

II. BSTGSOM ALGORITHM

The GSOM algorithm is modified to impose a Binary Search Tree. The method of finding the winner, neighborhood selection and weight adaptations are different from that of the GSOM and are discussed below.

The BSTGSOM also has the same three phases as in the GSOM, namely the initializing phase, the training phase and the smoothing phase. The algorithm is presented briefly here due to space limitation. All the terms and symbols used in this

978-1-4244-4837-1/09/$25.00 ©2009 IEEE

paper have the same meanings as in the GSOM[1] if not mentioned explicitly.

A. Initializing Phase

Four node initializing is used, which is similar to that of the GSOM.

Initializing the growing threshold and highest error value is also similar to the GSOM. The growing threshold is calculated as given in equation (1).

$$GT = -D * \ln(SF) \tag{1}$$

where GT is the Growing Threshold, D is number of input dimensions and SF is the Spread Factor.

B. Training Phase

Inputs are presented to the network one after the other and the training is conducted for each input for a predefined number of iterations. The training algorithm differs slightly for the first iteration from the other iterations and this is explained in detail below.

1. In the first iteration

The Euclidean difference between the first neuron and the input is calculated. A binary search tree is initialized with the first node as the root. The value of the node is the difference between the input and the neuron as given in equation (2)

$$v = \sqrt{\sum_{j=1}^{Dim}(x_j - w_j)^2} \tag{2}$$

where v, x, w are values of the node, input vector and weight vector respectively. Dim is the number of dimensions of the input.

Then the differences between the input and the neurons will be calculated and the neurons will be added to the tree such that it preserves the binary search tree properties based on the input differences, moreover based on the value of the tree node.

At the end of the first iteration, the tree which includes all the neurons of the map will be generated and the wining neuron is the left most child of the tree based on the properties of the binary search tree.

The weights of the winner and its neighborhood are updated as given in equation (3). Detailed neighborhood selection is described later.

$$w_j(k+1) = \begin{cases} w_j(k), j \notin \mathrm{N}_{k+1} \\ w_j(k) + LR(k) * \left(x_k - w_j(k)\right), j \in N_{k+1} \end{cases} \tag{3}$$

$w_j(k)$and $w_j(k+1)$ are weights vectors of the node j before and after the adaptation, and $N_{k+1}$is the neighborhood of the winning neuron at (k+1)th iteration, LR(k), k$\in N_{k+1}$ is a sequence of positive parameters converging to zero as $k \to \alpha$.

The error value of the winner is updated similar to that of the GSOM as given in equation (4).

$$E_i(t+1) = E_i(t) + \sqrt{\sum_{j=1}^{Dim}(x_j - w_j)^2} \tag{4}$$

$E_i(t)$ and $E_i(t+1)$are errors of the neuron i before and after the error updating. *x, w* are input and weight vectors and *Dim* is the Dimension.

Finally the highest accumulated error value in the network is updated. If the highest error value exceeds the growing threshold, new nodes are generated or the error is distributed within the immediate neighborhood of the highest error neuron based on whether it is a boundary node in the network.

2. In the second iteration and onwards

The difference to the input is only calculated for the neurons that are within the neighborhood of the winner node of the previous iteration and those nodes will be rearranged in the tree, based on the newly calculated differences. The arrangements will be done such that it preserves the binary search tree properties making the left most children become the winner.

The weights of the winner and its neighbors are updated. The error value of the winner and the highest accumulated error are also updated. If the highest error value exceeds the growing threshold new nodes will be generated or the error will be distributed among the neighbors of the highest error node.

C. Smoothing Phase.

The tree generation mechanism is very much similar to that of the Training Phase. The only difference is that there will be no new neurons added during the smoothing phase.

The training and smoothing phases should be repeated for each input in the input set.

D. Neighborhood Selection

The neighborhood is selected based on the tree arrangement of the neurons. Some of the tree based SOM variations [5, 6] have used similar neighborhood selection mechanisms. Distance between the two adjacent neurons will be used as the basic unit when calculating the distance. In detail, the distance between the two neurons is defined as the number of connections that contains in the shortest path that connects the two neurons.

The neighborhood is specified using the neighborhood radius, which is defined as the maximum path length between the two nodes. So, all the nodes that lie in a distance less than the radius are considered to be in the neighborhood. In another words, the set of nodes that lie in a distance less than or equal to the radius is considered as the neighborhood of a given node. The following diagram illustrates the new neighborhood concept.

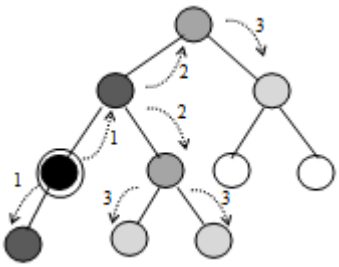

Fig. 1. Neighborhood of 3 of the selected (black) node. Neighborhood nodes with neighborhood of 1, neighborhood of 2 and neighborhood of 3 have highlighted with different gray level colors.

E. Weight Updating of the Newly Created Neurons

New nodes are created in the same manner as in the GSOM. The only change is, the weights of new nodes will be updated so that it will be closer to the winning neuron in the imposed tree not in the physical locations in the map. The weight initializing mechanism can be described as follows.

Case 1 - If only one node is added, it will be initialized so that it will become the left child of the highest error neuron. If the highest error neuron has a left child, weight of the new node is initialized as the middle value of the highest error neuron and its left child.

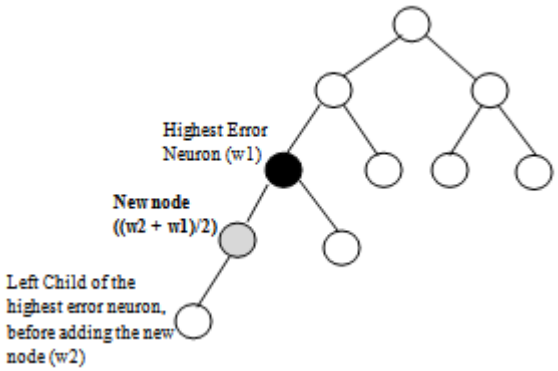

Fig. 2. New node weight initialization when there is no left child for the highest error neuron

If highest error neuron is a leaf node, weight of the new node = w1- Abs (w2-w1), w1 is the weight of the winner and w2 is weight of its parent.

Case 2 - If there are 2 nodes, nodes will be initialized as left and right Children. Weight of the new node is initialized in a similar manner to the case 1, based on whether the highest error neuron has left or right child nodes.

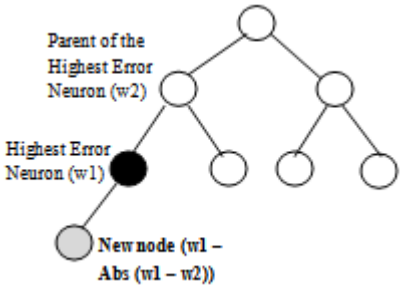

Fig. 3. New node weight initialization when highest error neuron has a left child

Case 3 - If there are 3 nodes, nodes will be initialized as left child, right child and its parent. All the nodes are initialized similar to case 1 and case 2. The parent's weight value is always equal to the average value of the highest error neuron's weight and its parent's weight.

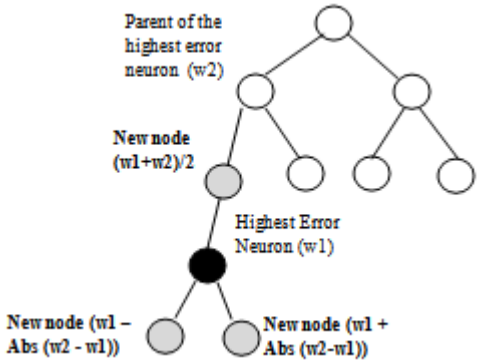

Fig. 4. New node weight initialization for case 3, when the highest error neuron is a leaf node

III. PERFORMANCE ANALYSIS

When considering the complexity of the algorithms, the major performance drawback in GSOM is the process of finding the winning neuron. It involves computing the difference between the input and neurons for every input in each iteration. Simply the total number of difference calculations is equal to the number of iterations multiplied by the number of inputs. As the GSOM grows, the numbers of nodes in the map increases. Also, when the input set is large, to get finer clusters it needs to have a higher number of iterations, making it computationally expensive.

But in the BSTGSOM, the new way of calculating the winning node reduces the number of calculations needed to find the winning neuron. Since calculations are based on the binary search tree arrangement it involves comparing only the

neurons in the winner's neighborhood in the following iterations, giving a significant performance improvement.

Experiments were done based on both the algorithms and they have shown that BSTGSOM is much faster than the GSOM.

IV. EXPERIMENTAL RESULTS

Experiments were done on a basic zoo data set. Since this is a new algorithm, this small data set was used to demonstrate the functionality of the algorithm. The zoo data set is very useful to demonstrate a clustering algorithm. The results from the BSTGSOM for the zoo data set under different spread factors are illustrated in figure 5 and figure 6.

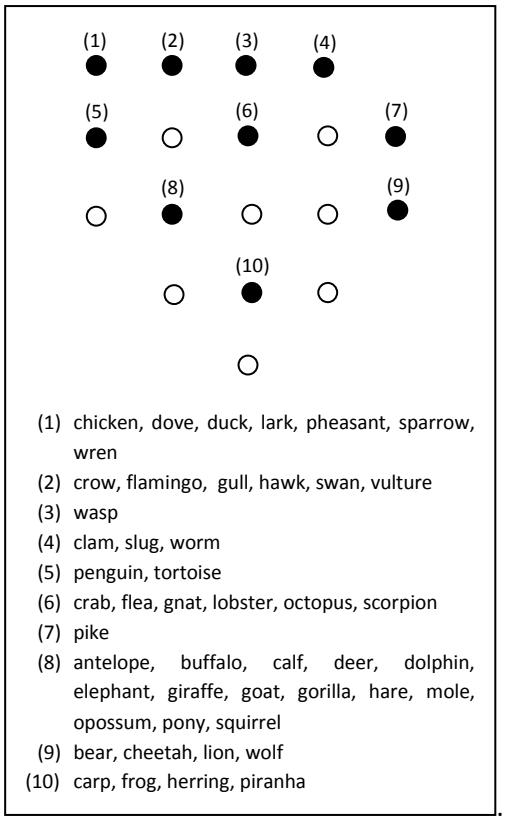

Fig. 5. BSTGSOM for the zoo data set with SF = 0.8

Figure 7 demonstrates the functionality of the BST model using the zoo data. It can be seen that with each new input the existing map readjusts itself, to match the new input, while maintaining the inter cluster relationships.

V. CONCLUSION AND OPEN PROBLEMS

The new algorithm presented in this paper extends one of the key advantages of the popular SOM algorithm. One of the main reasons for the popularity of the SOM is that it not only clusters a data set but provides a mapping of the data which is

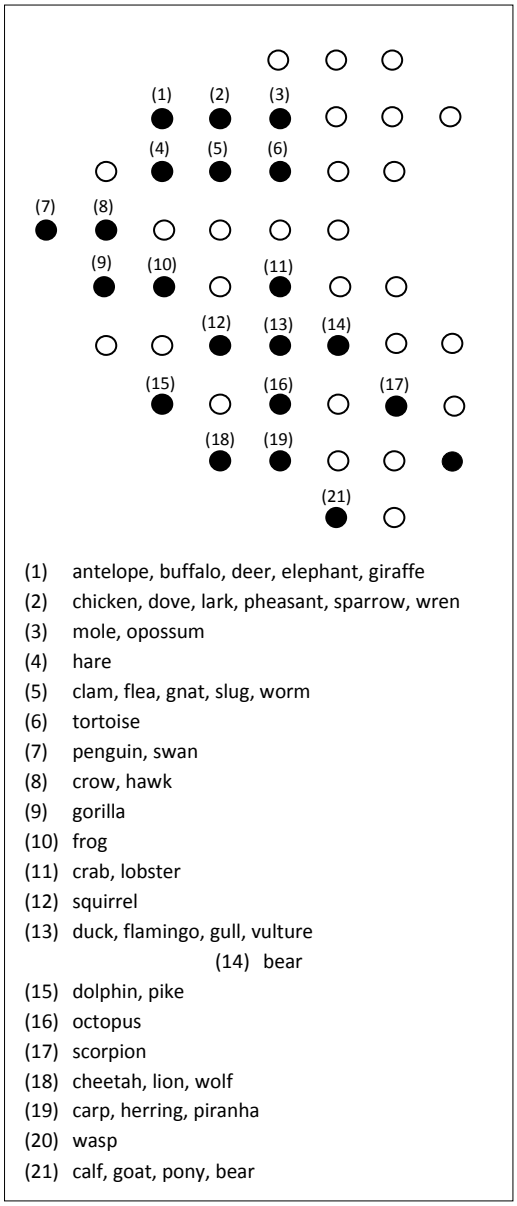

Fig.. 6. BSTGSOM for the zoo data set with SF = 0.1

a two dimensional representation of the neighborhood relationships among the data. As such the clusters with similar or 'close' features will be near to each other and vice versa. This characteristic has made the SOM a popular data visualization technique. The proposed algorithm is based on the structure adapting (dynamic structure) SOM called the GSOM and has further extended the GSOM with a dynamic, adaptive cluster visualization ability. Once the map is trained, it does not become a static structure (as with the traditional SOM), but provides the user with a new structure for each new input, which shows the positioning of the existing clusters in comparison to the new input. The important characteristic in the model is that all the clusters are repositioned according to their relationship to the new input,

while maintaining their inter cluster relationships. This is a model which has very high potential in many application areas, especially where changing input patterns could be analyzed based on their effect on existing clusters. We hope to evaluate the potential value of this technique with dynamic text streams, for identifying the matching and 'close by' clusters to new text input.

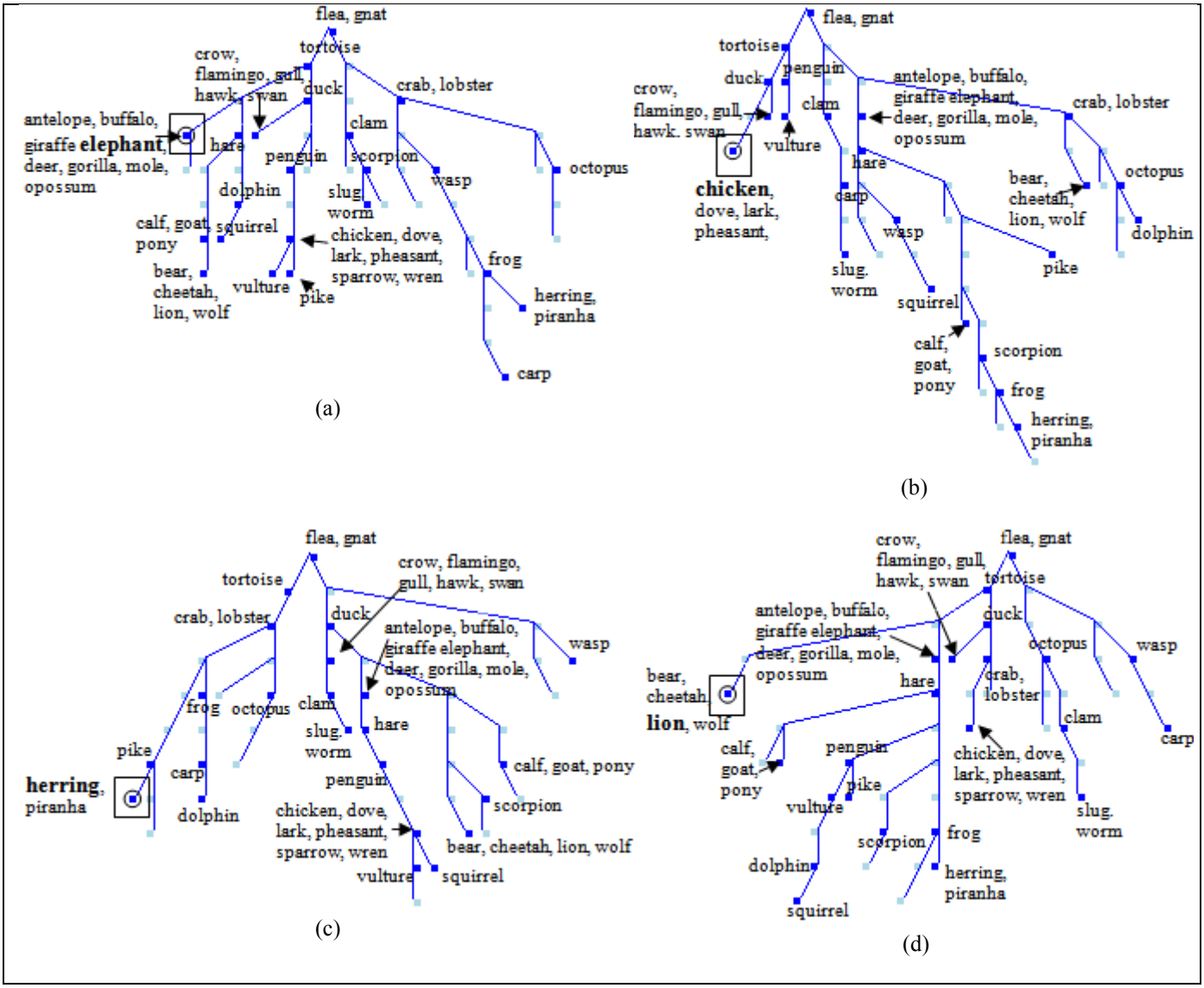

Fig. 7. Readjustments of the map and inter cluster relationship for the different inputs. (a) readjusted map for the input elephant (b) readjusted map for the input chicken (c) readjusted map for the input herring (d) readjusted map for the input lion

REFERENCES

[1] Alahakoon L.D., Halgamuge S.K. and Srinivasan B., Dynamic Self Organising Maps with controlled growth for Knowledge Discovery, IEEE Transactions on Neural Networks, 11;3, 2000.

[2] Wang H., Azuaje F and Black N., An Integrative Framework for improving Biomedical Pattern Discovery and Visualization, IEEE Transactions on Information Technology in Bio Medicine, 8(1): 16-27, 2004.

[3] Wang H., Azuaje F and Black N., Improving BioModicular Pattern Discovery and visualization with Hybrid Self-Adaptive Networks, IEEE Transactions on Nano BioScience, 1(4) : 146-166, 2002

[4] Andreas Nurnberger, Clustering of Document Collections using a Growing Self-Organising Map, In: Proc. Of BISC International workshop on Fuzzy Logic and the Internet (FLINT 2001), pp. 136-141, ERL, College of Engineering, UC Berkeley, CA, 2001.

[5] Pakkanen J., The Evolving Tree, a new kind of self-organizing neural network. In: proceedings of the Workshop on Self-Organizing Maps 2003, Kitakyushu, Japan, pp. 311-316, 2003.

[6] Astudillo, C.A., Oommen, B.J.: A Novel Self Organizing Map Which Utilizes Imposed Tree-Based Topologies. In 6th international conference on Computer Recognition Systems, 2009.