Optimization of Combining of Self Organizing Maps and Growing Neural Gas

Lukáš Vojáček[1(✉)], Pavla Dráždilová[2], and Jiří Dvorský[2]

[1] IT4Innovations, VŠB - Technical University of Ostrava,
17. listopadu 15/2172, 708 33 Ostrava, Czech Republic
lukas.vojacek@vsb.cz

[2] Department of Computer Science, VŠB - Technical University of Ostrava,
17. listopadu 15/2172, 708 33 Ostrava, Czech Republic
{pavla.drazdilova,jiri.dvorsky}@vsb.cz

Abstract. The paper deals with the issue of high dimensional data clustering. One possible way to cluster this kind of data is based on Artificial Neural Networks (ANN) such as Growing Neural Gas (GNG) or Self Organizing Maps (SOM). Parallel modification, Growing Neural Gas with pre-processing by Self Organizing Maps, and its implementation on the HPC cluster is presented in the paper. Some experimental results are also presented. We focus on effective preprocessing for GNG. The clustering is realized on the output layer of SOM and the data for GNG are distributed into parallel processes.

Keywords: Self organizing maps · Growing neural gas · High-dimensional dataset · High performance computing

1 Introduction

Recently, several topics and issues have emerged such as availability of big and/or high-dimensional data, petascale HPC systems aiming towards exascale supercomputers and need of processing this kind of data. Large high-dimensional data collections are commonly available in areas like medicine, biology, information retrieval, web analyze, social network analyze, image processing, financial transaction analysis and many others. To process such kind of data unsupervised learning algorithms, such as *Self Organizing Maps* (SOM) or *Growing Neural Gas* (GNG), are usually used. Various aspects of parallel implementation of these algorithms on HPC were studied e.g. [12,13].

The one of still preserving issues is efficient utilization of the computation resources. When speaking on SOM or GNG learning algorithms there are two challenges. The first one is fast computation of similarity in high-dimensional space and the second one is ideally uniform distribution of computation load among individual CPU cores.

Parallel implementation usually allocates one CPU core to group of neurons, evaluate similarity of these neurons with given input vector, find local best matching neuron and then using some form of communication to find a global

K. Saeed and W. Homenda (Eds.): CISIM 2016, LNCS 9842, pp. 277–286, 2016.
DOI: 10.1007/978-3-319-45378-1_25

best matching neuron in the whole neural network. The CPU cores are allocated regularly, using some pattern [12] regardless of input vectors distribution on neurons i.e. CPU cores causing a bottleneck in the parallel learning algorithm. To reduce the bottleneck input vectors preprocessing is done using small SOM and clustering algorithm. This allows us to improve distribution of neurons over CPU cores and subsequently speed-up the learning algorithm itself.

The paper is organized as follows. The Sect. 2 briefly describes used neural networks: SOM and GNG. Input vectors preprocessing algorithm is provided in Sect. 3. Experimental results are given in Sect. 4.

2 Artificial Neural Networks

In this section we will describe two types of neural networks, the first is Self Organizing Maps and the second is Growing Neural Gas and then we present a combination of SOM and GNG.

2.1 Self Organizing Maps

Self Organizing Maps (SOMs), also known as Kohonen maps, were proposed by Kohonen in 1982 [4]. SOM is a kind of artificial neural network that is trained by unsupervised learning. Using SOM, the input space of training samples can be represented in a lower-dimensional (often two-dimensional) space [5], called a *map*. Such a model is efficient in structure visualization due to its feature of topological preservation using a neighbourhood function.

SOM consists of two layers of neurons: an *input layer* that receives and transmits the input information, and an *output layer*, the map that represents the output characteristics. The output layer is commonly organized as a two-dimensional rectangular grid of nodes, where each node corresponds to one neuron. Both layers are feed-forward connected; each neuron in the input layer is connected to each neuron in the output layer. A real number, or weight, is assigned to each of these connections.

2.2 Growing Neural Gas

The representation of Growing Neural Gas is an undirected graph which need not be connected. Generally, there are no restrictions to the topology. The graph is generated and continuously updated by competitive Hebbian Learning [7,9]. According to the pre-set conditions, new neurons are automatically added and connections between neurons are subject to time and can be removed. GNG can be used for vector quantization by finding the code-vectors in clusters [3], image compression, disease diagnosis.

GNG works by modifying the graph, where the operations are the addition and removal of neurons and edges between neurons.

To understand the functioning of GNG, it is necessary to define the learning algorithm. The algorithm published in [12] is based on the original algorithm [2,3], but it is modified for better continuity in the SOM algorithm.

Remark. The notation used in the paper is briefly listed in Table 1.

Table 1. Notation used in the paper

Symbol	Description
M	Number of input vectors
n	Dimension of input vectors, number of input neurons, dimension of weight vectors in GNG output layer neurons
N	Current number of neurons in GNG output layer
N_{max}	Maximum allowed number of neurons in GNG output layer
n_i	i-th input neuron, $i = 1, 2, \ldots, n$
N_i	i-th output neuron, $i = 1, 2, \ldots, N$
N_F	the fattest neuron
T	Number of epochs
l_{c_1}	Learning factor of BMU_1
l_{nc_1}	Learning factor of BMU_1 neighbours
e_i	Local error of output neuron N_i, $i = 1, 2, \ldots, N$
C_i	ith cluster witch contains similar input vectors
m_i	Number of input vectors in cluster C_i
$\boldsymbol{Z}_i$	Centroid of cluster C_i
α	Error e_i reduction factor
β	Neuron error reduction factor
γ	Interval of input patterns to add a new neuron
a_{max}	Maximum edge age

3 Combination of SOM and GNG

In our previous paper [12], we focused on a combination of SOM and GNG, where the basic idea was to pre-process the input data by SOM, as a result of which there are clusters of similar data. Subsequently, we created the same number of GNG network as clusters, and assigned each cluster to one GNG. Each GNG creates its own neural map and after the learning process is finished, the results are merged. The entire description above can be summarized as follows: Help speeding up computation parallelization is shown in Fig. 1 where the top layer of parallelization (SOM) is described in a previous paper [13]. In this chapter we will describe an improved method focusing on the creating clusters of input data and optimization of used resources.

To improve the efficiency of parallelization is needed to clusters of input data for GNG network contained approximately the same number of input data. Based on the fact that when the GNG assigned more input vectors, the calculation takes longer. In the past to create clusters we used to spanning tree algorithm [12] which, however, does not reflect the number of input vectors

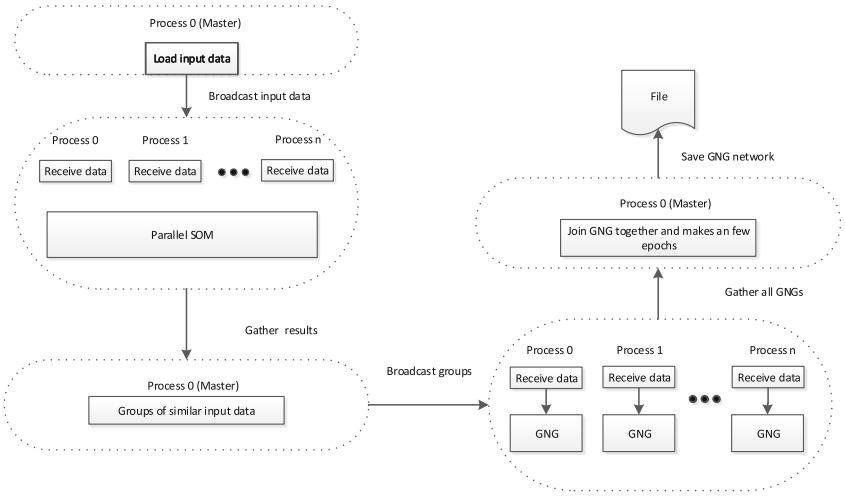


Fig. 1. Parallel algorithm

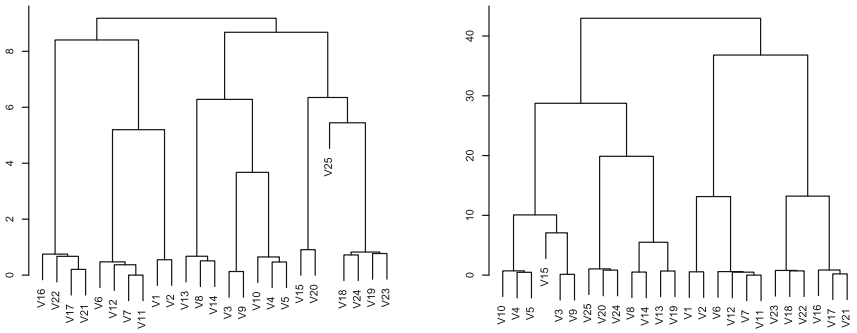


Fig. 2. Dendrograms of hierarchical clustering (Average-linkage and Ward's method) of output layer of SOM 5×5

in clusters. To obtain clusters of neurons we now use two different clustering algorithms. The first algorithm is agglomerative hierarchical clustering methods of calculation for determining the distance between clusters. Practically useful methods are *Ward's method*, *Centroid-linkage* and *Average-linkage* (AVL), see Fig. 2. The second clustering algorithm is the algorithm PAM [8], which like the above hierarchical algorithms, operates with a matrix of distances between the neurons in the output layer. The thus formed clusters are input to the following algorithm, which subdivides the neurons of the output layer SOM into clusters containing the closest possible number of the input data. The clusters are created on the basis of Algorithm 1.

We proposed an optimization for the calculation of GNG networks, which aims to optimize the maximum utilization of the allocated resources, but on

Algorithm 1. Calculate the distribution of neurons to efficiently parallelize

Input : The set of clusters $\mathbf{C}' = \{C'_1, \ldots, C'_m\}$ which contain neurons from output layers of SOM ($k \times k$ size) where training vectors are mapped to specific neurons. The clustering was done using algorithm with the best Silhouette index (Figs. 3 and 4)

Output: The new set of clusters $\mathbf{C} = \{C_1, \ldots, C_n\}$ where $m < n$

1. For $k = \{5, 6, 7, 8\}$ find in each SOM ($k \times k$) fattest neuron N_F which satisfies

$$F = \arg \max_{i=1,\ldots,N} |\mathsf{N}_i|$$

   where $|\mathsf{N}_i|$ denotes number of input vectors mapped to neuron N_i.
2. Find a $K \in \{5, 6, 7, 8\}$ for which value $\mathsf{N}_F^k - \frac{1}{k^2}M$ is minimal. So that SOM ($K \times K$) has the smallest dispersion of $|\mathsf{N}_i|$. Furthermore, we will work with SOM ($K \times K$). For sake of simplicity the index K will be omitted from N_F^K in following text.
3. Divide the cluster C'_i with the fattest neuron N_F to two clusters C_{N_F} and C_i where cluster C_{N_F} contain only one neuron N_F and cluster $C_i = C'_i \backslash \{\mathsf{N}_F\}$.
4. For others clusters C'_j verify: if number of input data $|C'_j|$ in cluster C'_j $|C'_j| > |\mathsf{N}_F|$, then split C'_j into two clusters $C_{j,1}$ and $C_{j,2}$ where cluster $C_{j,1}$ contain fattest neuron from C'_j and cluster $C_{j,2} = C'_j \backslash C_{j,1}$.
5. Return new set of cluster.

condition that the computing time must be similar. The principle of optimization is based on the idea that individual computing resources will count more GNG networks than only one – as it has until now. In Algorithm 2, the overall functionality is described. Here it is necessary to mention two facts regarding point 4. Firstly, it is a variation of a known problem *Subset sum* [6], which is an NP complete problem [1]. But at the beginning it was defined that the size of the output map of SOM is small and therefore the maximum possible number of clusters is also small, and therefore negligible. And secondly the maximum limit is reduced by 10 % on the grounds that to work with each GNG network time for I/O operations must be added.

Algorithm 2. Algorithm to optimize the utilization of computing resources

1. From the set of clusters $\mathbf{C} = \{C_1, \ldots, C_n\}$; take cluster C_{N_F} with the greatest number of input vectors.
2. Remove C_{N_F} from $\mathbf{C}$.
3. Assign cluster C_{N_F} to unused computing resource.
4. Select the set of clusters from set $\mathbf{C}$ wherein the sum of their input vectors is approaching number of input vectors in $0.9|C_{\mathsf{N}_F}|$
5. Remove used clusters from the set $\mathbf{C}$.
6. Assign selected clusters to the unused computing resource.
7. If $\mathbf{C}$ is not empty return to step 4.

4 Experiments

We will describe different datasets and we will provide experiments with datasets in this section.

4.1 Experimental Datasets and Hardware

Two datasets were used in the experiments. The first dataset was commonly used in Information Retrieval – *Medlars*. The second one was the test data for the elementary benchmark for clustering algorithms [11].

Weblogs Dataset. To test the learning algorithm effectiveness on high dimensional datasets, a Weblogs dataset was used. The Weblogs dataset contained web logs from an Apache server. The dataset contained records of two month's requested activities (HTTP requests) to the NASA Kennedy Space Center WWW server in Florida[1]. The standard data preprocessing methods were applied to the obtained dataset. The records from search engines and spiders were removed, and only the web site browsing was left (without download of pictures and icons, stylesheets, scripts etc.). The final dataset (input vector space) was of a dimension 90,060 and consisted of 54,961 input vectors. For a detailed description, see our previous work [10], where a web sites community behaviour was analyzed.

Medlars Dataset. The Medlars dataset consisted of 1,033 English abstracts from medical science[2]. The 8,567 distinct terms were extracted from the Medlars dataset. Each term represents a potential dimension in the input vector space. The term's level of significance (weight) in a particular document represents a value of the component of the input vector. Finally, the input vector space has a dimension of 8,707, and 1,033 input vectors were extracted from the dataset.

Experimental Hardware. The experiments were performed on a Linux HPC cluster, named Anselm, with 209 computing nodes, where each node had 16 processors with 64 GB of memory. Processors in the nodes were Intel Sandy Bridge E5-2665. Compute network is InfiniBand QDR, fully non-blocking, fat-tree. Detailed information about hardware is possible to find on the web site of Anselm HPC cluster[3].

4.2 First Part of the Experiment

The first part of the experiments was oriented towards the comparison of quality of clustering by the agglomerative hierarchical clustering and PAM Clustering. The used dataset was *Weblogs*. All the experiments were carried out for

[1] The collection can be downloaded from http://ita.ee.lbl.gov/html/contrib/NASA-HTTP.html.

[2] The collection can be downloaded from ftp://ftp.cs.cornell.edu/pub/smart. The total size of the dataset is approximately 1.03 MB.

[3] https://support.it4i.cz/docs/anselm-cluster-documentation/hardware-overview.

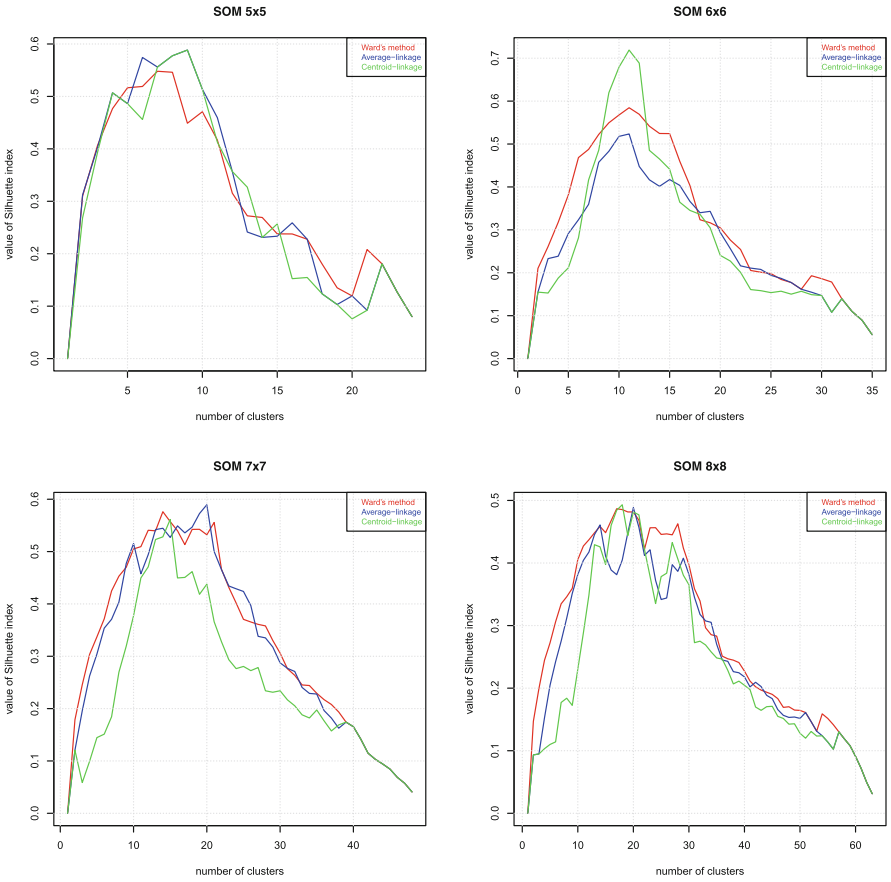
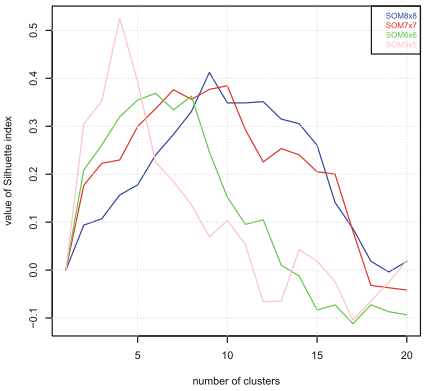


Fig. 3. Quality of hierarchical clustering of output layer of SOM k × k

20 epochs; the random initial values of neuron weights in the first epoch were always set to the same values. The tests were performed for SOM with rectangular shape 5 × 5 neurons, 6 × 6 neurons, 7 × 7 neurons and 8 × 8 neurons. The metrics used for determining the quality of clustering is *Average Silhouette Index* (ASI). The achieved quality of clustering for agglomerative hierarchical clustering is presented in Fig. 3 and for PAM Clustering is presented in Fig. 4.

4.3 Second Part of the Experiment

The second part of the experiments was oriented towards comparing the time efficiency of algorithms PAM and AVL. The GNG parameters are as follows $\gamma = 100$, $e_w = 0.05$, $e_n = 0.006$, $\alpha = 0.5$, $\beta = 0.0005$, $a_{max} = 160$, $N_{max} = 1221$, $T = 200$. The used dataset was *Weblogs*. Dimensions of SOM are 5×5, 6×6, 7×7 and 8×8. Number of cores are 32 for each group and the groups are computed sequentially (Tables 2 and 3).



SOM dimensions	# of clusters	ASI
5×5	4	0.53
6×6	6	0.37
7×7	10	0.38
8×8	9	0.41

Fig. 4. Quality of PAM clustering of output layer of SOM k × k

Table 2. Computing time with respect to number of cores, standard GNG algorithm, dataset Medlars

SOM		Numbers of vectors		Time [mm:ss]			
Dimension	Time [s]	Max	Second max	4* PAM	10 PAM	9* AVL	4 AVL
5×5	13	6852	1636	14:19	14:37	14:44	14:22
				6* PAM	10 PAM	11* AVL	6 AVL
6×6	18	5918	1900	13:50	14:12	14:11	14:02
				10* PAM	6 PAM	20* AVL	6 AVL
7×7	27	5607	1795	13:57	13:47	14:26	13:48
				9* PAM	6 PAM	20* AVL	6 AVL
8×8	35	5192	1437	14:03	13:51	14:41	13:37

Table 3. The quality of algorithms (the lower is better)

4* PAM	10 PAM	9* AVL	4 AVL
1.24	1.23	1.24	1.2
6* PAM	10 PAM	11* AVL	6 AVL
1.22	1.24	1.22	1.26
10* PAM	6 PAM	20* AVL	6 AVL
1.14	1.38	1.4	1.18
9* PAM	6 PAM	20* AVL	6 AVL
1.24	1.04	1.3	1.18

4.4 Third Part of the Experiment

The third part of experiments was oriented towards speedup and resource optimization. The GNG parameters are as follows $\gamma = 200$, $e_w = 0.05$, $e_n = 0.006$, $\alpha = 0.5$, $\beta = 0.0005$, $a_{max} = 160$, $N_{max} = 1000$, $T = 200$. Medlars dataset was used. The tests were performed for SOM with a rectangular shape of 5×5 neurons. The achieved speedup is presented in Tables 4 and 5. Both samples have similar speedup but fundamentally differ in efficiency; in the fastest case without optimization the efficiency is 0.02, but with optimization it is 0.23.

Table 4. Combination SOM and GNG's

Cores	Time	Speedup	Efficiency
1	02:28:46	1.00	1.00
32	00:18:01	8.26	0.26
64	00:17:51	8.33	0.13
128	00:16:37	8.95	0.07
256	00:10:42	13.90	0.05
512	00:07:27	19.97	0.04
1024	00:06:37	22.48	0.02

Table 5. Combination SOM and GNG's and resource optimization

Cores	Time	Speedup	Efficiency
1	02:28:46	1.00	1.00
32	00:18:02	8.26	0.26
64	00:17:48	8.33	0.13
96	00:06:39	22.37	0.23
-	-	–	–
-	-	–	–
-	-	–	–

5 Conclusion

In this paper, we presented experiments with preprocessing of data in output layer of SOM. Different clustering algorithm was used and different quality of clusters were obtained. The global input data preprocessed on this SOM was used as an input for GNG neural network. This approach allows us almost uniformly distribute data on computation cores efficiently utilize them. The achieved speed-up is also very good. In future work we intend to focus on sparse date, and improved acceleration and use Xeon phi for better speed-up.

Acknowledgment. This work was supported by The Ministry of Education, Youth and Sports from the Large Infrastructures for Research, Experimental Development and Innovations project "IT4Innovations National Supercomputing Center LM2015070" and partially supported by Grant of SGS No. SP2016/68, VŠB – Technical University of Ostrava, Czech Republic.

References

1. Carlson, J.: The Millennium Prize Problems. American Mathematical Society, Providence (2006)
2. Fritzke, B.: A growing neural gas network learns topologies. In: Advances in Neural Information Processing Systems, vol. 7, pp. 625–632. MIT Press, Cambridge (1995)

3. Holmström, J.: Growing neural gas experiments with GNG, GNG with utility and supervised GNG. Master's thesis, Uppsala University, 30 August 2002
4. Kohonen, T.: Self-organization and Associative Memory. Springer Series in Information Sciences, vol. 8, 3rd edn. Springer, Heidelberg (1984)
5. Kohonen, T.: Self Organizing Maps, 3rd edn. Springer, Heidelberg (2001)
6. Li, J., Wan, D.: On the subset sum problem over finite fields. Finite Fields Appl. **14**(4), 911–929 (2008). http://www.sciencedirect.com/science/article/pii/S1071579708000208
7. Martinetz, T.: Competitive hebbian learning rule forms perfectly topology preserving maps. In: Gielen, S., Kappen, B. (eds.) ICANN 1993, pp. 427–434. Springer, London (1993). http://dx.doi.org/10.1007/978-1-4471-2063-6_104
8. Park, H.S., Jun, C.H.: A simple and fast algorithm for k-medoids clustering. Expert Syst. Appl. **36**(2, Part 2), 3336–3341 (2009). http://www.sciencedirect.com/science/article/pii/S095741740800081X
9. Prudent, Y., Ennaji, A.: An incremental growing neural gas learns topologies. In: Proceedings of 2005 IEEE International Joint Conference on Neural Networks, 2005, IJCNN 2005, vol. 2, pp. 1211–1216, July–4 August 2005
10. Slaninová, K., Martinovič, J., Novosád, T., Dráždilová, P., Vojáček, L., Snášel, V.: Web site community analysis based on suffix tree and clustering algorithm. In: Proceedings - 2011 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology - Workshops, WI-IAT Workshops 2011, pp. 110–113 (2011)
11. Ultsch, A.: Clustering with SOM: U*C. In: Proceedings of Workshop on Self-organizing Maps, Paris, France, pp. 75–82 (2005)
12. Vojáček, L., Dráždilová, P., Dvorský, J.: Combination of self organizing maps and growing neural gas. In: Saeed, K., Snášel, V. (eds.) CISIM 2014. LNCS, vol. 8838, pp. 100–111. Springer, Heidelberg (2014)
13. Vojáček, L., Martinovič, J., Dvorský, J., Slaninová, K., Vondrák, I.: Parallel hybrid SOM learning on high dimensional sparse data. In: Chaki, N., Cortesi, A. (eds.) CISIM 2011. CCIS, vol. 245, pp. 239–246. Springer, Heidelberg (2011)