Neurocomputing 104 (2013) 72–82

Contents lists available at SciVerse ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Growing neural gas efficiently

Daniel Fišer*, Jan Faigl, Miroslav Kulich

Department of Cybernetics, Faculty of Electrical Engineering, Czech Technical University in Prague, Technická 2, 6 Prague 166 27, Czech Republic

ARTICLE INFO

Article history:
Received 28 September 2011
Received in revised form
30 July 2012
Accepted 7 October 2012
Communicated by B. Hammer
Available online 14 November 2012

Keywords:
Self-organizing map
Growing neural gas
Nearest neighbor search
3-D surface reconstruction

ABSTRACT

This paper presents optimization techniques that substantially speed up the Growing Neural Gas (GNG) algorithm. The GNG is an example of the Self-Organizing Map algorithm that is a subject of an intensive research interest in recent years as it is used in various practical applications. However, a poor time performance on large scale problems requiring neural networks with a high amount of nodes can be a limiting factor for further applications (e.g., cluster analysis, classification, 3-D reconstruction) or a wider usage. We propose two optimization techniques that are aimed exclusively on an efficient implementation of the GNG algorithm internal structure rather than on a modification of the original algorithm. The proposed optimizations preserve all properties of the GNG algorithm and enable to use it on large scale problems with reduced computational requirements in several orders of magnitude.

1. Introduction

The Growing Neural Gas (GNG) [1] is one of the Self-Organizing Map (SOM) [2] algorithms for unsupervised learning. Unsupervised learning (or sometimes called learning without teacher) is a learning method that works solely with an input data and has no information about the desired output. Input data are consecutively presented to SOM in a form of input signals and SOM changes its topological structure to fit to the input data using its own specific mechanism – self-adaptation. The GNG and other growing neural networks (such as Growing Cell Structures [3], Growing Grid [4], etc.) use furthermore a growing mechanism for a gradual adaptation and a self adjusting of its own size. The growing neural network starts in some minimal state (e.g., with some minimal number of neurons in the network), which is adapted to input data. Then, it grows (increases its size) and adapts again. This cycle is repeated until a desired resolution of the neural network is reached.

SOM algorithms are used in various applications such as vector quantization [5–7], cluster analysis [1,3,8–10], classification [11,6], and 3-D reconstruction [12–15], but a poor time performance, especially on large scale problems, can be a limiting factor for further applications or a wider usage. Although several approaches addressing the computational requirements have been proposed [16,17,7], we have found out that the GNG algorithm can be implemented in a more efficient way than a direct implementation of the original description. Moreover, based on our application of the GNG algorithm in a real problem of 3D reconstruction[1] [18], we identified the most time consuming operations of the GNG algorithm and proposed optimization techniques significantly reducing the real required computational time. Hence, the goal of this paper is to show how to overcome the issue of the poor time performance and how to implement the GNG algorithm with optimizations providing a significant speedup.

The paper is organized as follows. First, a detailed description of the GNG algorithm is presented in the next section to identify and understand its most time-consuming parts. In Section 3, an overview of the related work is presented. Sections 4 and 5 are dedicated to description of the speedup techniques proposed. The real benefit of the techniques is evaluated in Section 6, the discussion of the experimental results is presented in Section 7, and the concluding remarks are presented in Section 8.

2. The growing neural gas

A GNG network structure is a graph consisting of a set of nodes and a set of edges connecting the nodes. Each node has associated a weight vector corresponding to the node's position in the input space and an error variable intended for identification of the parts of the network least adapted to input signals. Each edge is unambiguously identified by a pair of nodes. The schema of the GNG is depicted in Algorithm 1 with supporting functions in Algorithm 2 and notation used through this paper can be seen in Table 1.

* Corresponding author.
E-mail addresses: danfis@danfis.cz (D. Fišer), faiglj@fel.cvut.cz (J. Faigl), kulich@labe.felk.cvut.cz (M. Kulich).

[1] Videos with visualization of the 3-D reconstructions are available at http://www.youtube.com/watch?v=yoPcZpCPfyI, http://www.youtube.com/watch?v=oXx3oJ8omOQ, http://www.youtube.com/watch?v=j_t8LkAXS9Q.

0925-2312/$ - see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.neucom.2012.10.004

The GNG works as follows. After initialization, which places two randomly generated nodes into a network, two main phases are alternating until a selected stopping criterion is met. The first phase ("self-organizing") is adaptation, which is performed in λ steps. In each step, random input signal is generated and the neural network adapts itself to it: a connection between two nodes nearest to the input signal is strengthened (or created if it does not exist), then the nearest node and all its topological neighbors (nodes connected directly to the node by an edge) move towards the input signal and the nearest node's error is increased. This helps to identify areas where nodes are not sufficiently adapted to input signals. After that, the aging mechanism of edges is triggered – those edges that were not strengthened for a long time (the age of the edge is higher than A_{max}) are removed from the network. In the last step of the adaptation, an error of each node is decreased. Using this mechanism the neural network "forgets" old errors and thus it can focus on the most recent ones.

In the second phase ("growing"), a new node is created and connected into the network. The node's error is used for an identification of the area where the adaptation was least successful – the node with the largest error and its neighbor with the largest error are found. A new node is created at the halfway between them. The errors of those nodes are decreased.

Algorithm 1. The original growing neural gas algorithm.

GNG()
1 initialize the set G by two nodes with random weight vectors
2 $c \leftarrow 0$
3 $s \leftarrow 0$
4 $\vec{\xi} \leftarrow$ random input signal
5 $s \leftarrow s+1$
6 $v,\mu \leftarrow$ TWO_NEAREST_NODES($\vec{\xi}$)
7 **foreach** n **in** N_v
8 $A_{n,v} \leftarrow A_{n,v}+1$
9 INC_ERROR (c, s, v, $\|\vec{w}_v - \vec{\xi}\|^2$)
10 $\vec{w}_v \leftarrow \vec{w}_v + \epsilon_b(\vec{\xi} - \vec{w}_v)$
11 $\vec{w}_n \leftarrow \vec{w}_n + \epsilon_n(\vec{\xi} - \vec{w}_n)$, $\forall n \in N_v$
12 create an edge between v and μ if it does not exist
13 $A_{v,\mu} \leftarrow 0$
14 **foreach** a,b **in** all edges in map
15 **if** $A_{n,v} > A_{max}$
16 delete edge connecting n and v and all nodes w/o edges
17 **if** $s = \lambda$
18 GNG_NEW_NODE(c)
19 $c \leftarrow c+1$
20 $s \leftarrow 0$
21 DEC_ALL_ERROR(β)
22 **if** stopping criterion is met
23 terminate algorithm
24 **else**
25 go to step 4.

Table 1
Notation.

G	The set of all nodes in the network
v, μ	Nodes
N_v	The set of all topological neighbors of node v
$\vec{w}_v$	The weight vector of a node v
E_v	The error of a node v
$A_{v,\mu}$	The age of the edge between the nodes v and μ
c	The cycle counter
s	The step counter

Algorithm 2. Functions for the original GNG.

INC_ERROR(c, s, v, ν)
1 $E_v \leftarrow E_v + \nu$

DEC_ERROR(c, v, α)
1 $E_v \leftarrow \alpha E_v$

SET_ERROR(c, v, ν)
1 $E_v \leftarrow \nu$

DEC_ALL_ERROR(β)
1 $E_n \leftarrow \beta E_n$, $\forall n \in G$

LARGEST_ERROR (c)
1 $q \leftarrow \arg\max_{n \in G} E_n$
2 $f \leftarrow \arg\max_{n \in N_q} E_n$
3 **return** q, f

Algorithm 3. The growing neural gas algorithm.

GNG()
1 initialize the set G by two nodes with random weight vectors
2 $c \leftarrow 0$
3 **while** stopping criterion is not met
4 **for** $s \leftarrow 0$ **to** $\lambda - 1$
5 $\vec{\xi} \leftarrow$ random input signal
6 GNG_ADAPT(c, s, $\vec{\xi}$)
7 GNG_NEW_NODE(c)
8 $c \leftarrow c+1$

GNG_ADAPT(c, s, $\vec{\xi}$)
1 $v,\mu \leftarrow$ TWO_NEAREST_NODES ($\vec{\xi}$)
2 INC_ERROR(c, s, v, $\|\vec{w}_v - \vec{\xi}\|^2$)
3 $\vec{w}_v \leftarrow \vec{w}_v + \epsilon_b(\vec{\xi} - \vec{w}_v)$
4 $\vec{w}_n \leftarrow \vec{w}_n + \epsilon_n(\vec{\xi} - \vec{w}_n)$, $\forall n \in N_v$
5 create an edge between v and μ if it does not exist
6 $A_{v,\mu} \leftarrow 0$
7 **foreach** n **in** N_v
8 $A_{n,v} \leftarrow A_{n,v}+1$
9 **if** $A_{n,v} > A_{max}$
10 delete edge connecting n and v and all nodes w/o edges
11 DEC_ALL_ERROR(β)

GNG_NEW_NODE(c)
1 $q,f \leftarrow$ LARGEST_ERROR(c)
2 $\vec{w}_r \leftarrow \frac{\vec{w}_q + \vec{w}_f}{2}$
3 delete an edge connecting q and f and create two new edges between r and q and between r and f
4 DEC_ERROR(c, q, α)
5 DEC_ERROR(c, f, α)
6 SET_ERROR(c, r, $(E_q + E_f)/2$)

2.1. An alternative formulation of the GNG algorithm

We had observed that the original description of the GNG made by Bernd Fritzke in [1] can be reformulated without changes of the algorithm behavior, i.e., the new algorithm works exactly in the same way as the original one. The reformulation can be considered as "cosmetic"; however, it allows a more straightforward application of the proposed optimizations of the time consuming operations. Thus, the alternative formulation helps in further explanation of the speedup techniques proposed, and