Journal of Computational Science 16 (2016) 59–64

Contents lists available at ScienceDirect

Journal of Computational Science

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

Multilevel neuronal architecture to resolve classification problems with large training sets: Parallelization of the training process

Francisco Javier Martínez López[a], José Antonio Torres Arriaza[a], Sergio Martínez Puertas[b,*], María Mercedes Peralta López[a]

[a] Computer Department, University of Almería, Carretera de Sacramento S/N, La Cañada de San Urbano, 04120 Almería, Spain
[b] Math Department, University of Almería, Carretera de Sacramento S/N, La Cañada de San Urbano, 04120 Almería, Spain

ARTICLE INFO

Article history:
Received 27 September 2015
Received in revised form 18 January 2016
Accepted 3 April 2016
Available online 30 April 2016

Keywords:
Artificial neural networks
Radial basis function networks
Multilevel neural networks
CUDA
Parallelization

ABSTRACT

The value of radial base function networks (RBF) has been fully demonstrated and their application in a wide number of scientific fields is undisputed. A fundamental aspect of this tool focusses on the training process, which determines both the efficiency (success or "hit rat" in the subsequent classification) and the overall performance (runtime), since the RBF training phase is the most expensive phase in terms of time.

There is abundant literature on studies to improve these aspects, in which all the training techniques proposed are classified either as *iterative techniques*, with very short execution times for the training process, or as the traditional *exact techniques*, which excel in their high rates of accuracy in the classification.

In our field of study we require the smallest error possible in the classification process, and for this reason, our research opts for exact techniques, while we also work to improve the high latencies in the training process.

In a previous study, we proposed a pseudo-exact technique with which we improved the training process by an average of 99.1638177% using RBF-SOM architecture. In the present study we exploit one characteristic of this architecture, namely the possibility of parallelization of the training process.

Accordingly, our article proposes a RBF-SOM structure which, thanks to CUDA, parallelizes the training process. This we will denote as CUDA-RBF-SOM architecture.

1. State of the art: radial base function (RBF) networks

Radial basis function (RBF) networks are very useful to resolve problems where knowledge is scarce, fitting non-adjustable functions using statistical procedures and, in some cases, of conflicting knowledge. RBFs were introduced in the literature by Broomhead and Lowe [3] but it was Poggio and Girosi [30] who later offered the technique that allows a RBF the possibility of generalizing the solution of a problem.

According to [7], these "mathematical models developed to emulate the human brain" possess great computational power, which derives from their structure of distributed parallel computing. This structure enables the resolution of problems that would demand large amounts of time using "classic" computers. As a result, they have other properties that make them particularly attractive for solving many practical problems [16,44,17], including:

- They are non-linear distributed systems.
- They are fault tolerant systems.
- *Adaptability*: They have the capacity to modify the parameters on which their operation depends according to the changes that are produced in their work environment.
- They establish nonlinear relationships between data, since they do not have to meet the conditions of linearity, Gaussianity or stationarity [31].
- They admit the possibility of implementation in VLSI [26,25,27].

According to [10], thanks to these characteristics, they can be applied to a variety of fields of knowledge. The most notable of these are:

- *Classification*: With applications in medicine, for example, in the diagnosis of cardiopathology [18,9]; in pharmacy, for

* Corresponding author.
E-mail addresses: fml199@ual.es (F.J. Martínez López), jtorres@ual.es (J.A. Torres Arriaza), spuertas@ual.es (S. Martínez Puertas), mperalta@ual.es (M.M. Peralta López).

http://dx.doi.org/10.1016/j.jocs.2016.04.002
1877-7503/© 2016 Elsevier B.V. All rights reserved.

example, in the prediction of intoxication risk from digoxin [35]; signal processing, for example, in the equalization of communications channels, image pattern recognition and voice recognition [13,8,40,32,4,41,5,6]; and in economics, for example, in the concession of credit or determination of risk of bank failure.
- *Modeling*: With application in medicine, for example, in the prevention of cardiac degenerative diseases [1]; in pharmacy, for example, in the prediction of the concentration of gentamycin [11]; in signal processing, for example, in active noise elimination [36]; in economics, for example, in the prediction of electricity costs [12,20]; in environmental studies, for example, in the prediction of global temperature changes [28] or in automatic recognition of ocean structures [37]; and in robot behavior [19].

1.1. *Training algorithm for RBFs*

Mathematically, the training process of an RBF is defined by:

In a set of m elements that belongs to a training set $\vec{E}_{m\times n} = \{e_1, e_2, \ldots, e_m\}$ (with n characteristics per individual), and where $\vec{W} = \{w_1, w_2, \ldots, w_m\}$ are the weights of the neurons that comprise the network, the expression on which training of the RBF is based is:

$$rbf(\vec{x}) = \sum_{i=1}^{m} w_i \cdot \varphi_i(\|\vec{e}_i - \vec{x}\|), \tag{1}$$

where φ_i is the radial base function (Gaussian).

If we express this as a matrix we have [15]:

$$\begin{pmatrix} \varphi_1(\vec{e}_1) & \cdots & \varphi_1(\vec{e}_m) \\ \vdots & \ddots & \vdots \\ \varphi_m(\vec{e}_1) & \cdots & \varphi_m(\vec{e}_m) \end{pmatrix} \times \begin{pmatrix} w_1 \\ \vdots \\ w_m \end{pmatrix} = \begin{pmatrix} y_1 \\ \vdots \\ y_m \end{pmatrix} \tag{2}$$

Thus, to obtain the unknown $\vec{w}$ (which is what the training stage consists of), the expression would be:

$$\begin{pmatrix} w_1 \\ \vdots \\ w_m \end{pmatrix} = \begin{pmatrix} \varphi_1(\vec{e}_1) & \cdots & \varphi_1(\vec{e}_m) \\ \vdots & \ddots & \vdots \\ \varphi_m(\vec{e}_1) & \cdots & \varphi_m(\vec{e}_m) \end{pmatrix}^{-1} \times \begin{pmatrix} y_1 \\ \vdots \\ y_m \end{pmatrix} \tag{3}$$

1.2. *Main drawback: complexity of the training process*

The main problem of RBFs stems from the costly training process in terms of time.

This training cost is mainly due to the computational requirement of its two principal operations [2,42]:

- Inverse of a $m \times m$ matrix: order of complexity $O(n^3)$.
- Multiplication of two matrices: order of complexity $O(n^3)$.

Other procedures exist that train the RBF using *iterative techniques* [29], which calculate the centroids of the kernel functions and create a RBF network with fewer RBF functions than the training set.

Nevertheless, these methods entail a significant simplification when the training sets are very large and they generate RBF networks that, in some cases, do not fit the problem with the required precision [38]. Furthermore, in [38] we compared the training time of the architecture proposed in [24] with the times of the classic training procedures, including iterative techniques and we conclude that the proposed architecture in [24] provides better execution times.

For all this reason, in our current piece of research, we have opted for exact techniques.

2. First improvement: subdivision of the training set using multilevel RBF-SOM neural architecture.

Recently, there are several techniques that employ neural architecture based on SOMs and radial basis function networks [22,23,39,43]. In [24], we proposed an architecture defined by a duo $\langle\{RBF_i\}, \{C_i\}\rangle$, where $\{RBF_i\}$ is a set of RBFs, each of which is applied on the set of the inputs space, and $\{C_i\}$ are the reference vectors of a SOM [21] (serially operating multipliers). These vectors C_i serve as activators of the RBFs that must be used for a given entry (Fig. 1).

2.1. *Training algorithm of the RBF-SOM architecture.*

1. Make an initial random partition of the training set into N groups:

$$\begin{pmatrix} e_{1,1} & \cdots & e_{1,t} \\ \vdots & \ddots & \vdots \\ e_{w,1} & \cdots & e_{w,t} \end{pmatrix} \longrightarrow \left\{ \begin{array}{c} \begin{pmatrix} e_{1,1} & \cdots & e_{1,t} \\ \vdots & \ddots & \vdots \\ e_{x,1} & \cdots & e_{x,t} \end{pmatrix} \\ \begin{pmatrix} e_{x+1,1} & \cdots & e_{x+1,t} \\ \vdots & \ddots & \vdots \\ e_{y,1} & \cdots & e_{y,t} \end{pmatrix} \\ \vdots \\ \begin{pmatrix} e_{h+1,1} & \cdots & e_{h+1,t} \\ \vdots & \ddots & \vdots \\ e_{w,1} & \cdots & e_{w,t} \end{pmatrix} \end{array} \right. \tag{4}$$

The reason for this (using a first partition at random and not based on any statistical procedure) is due to our efforts to reduce the computational requirements.
2. Apply the SOM to each of the training subsets.
(In our case, we used a hexagonal topology of p rows and k columns.)
3. From step 2, we are interested in the $p*k$ centroids of each of the SOMs.

$$\begin{array}{ccc} \begin{pmatrix} e_{1,1} & \cdots & e_{1,t} \\ \vdots & \ddots & \vdots \\ e_{x,1} & \cdots & e_{x,t} \end{pmatrix} & \longrightarrow & p*k \ \text{ centroids} \\ \begin{pmatrix} e_{x+1,1} & \cdots & e_{x+1,t} \\ \vdots & \ddots & \vdots \\ e_{y,1} & \cdots & e_{y,t} \end{pmatrix} & \longrightarrow & p*k \ \text{ centroids} \\ \vdots & \vdots & \vdots \\ \begin{pmatrix} e_{h+1,1} & \cdots & e_{h+1,t} \\ \vdots & \ddots & \vdots \\ e_{w,t} & \cdots & e_{w,t} \end{pmatrix} & \longrightarrow & p*k \ \text{ centroids} \end{array} \tag{5}$$

After this step, we will have $N*p*k$ centroids: $\vec{C} = \{c_1, c_2, \ldots, c_{N*p*k}\}$