Neurocomputing 73 (2010) 1794–1800

Contents lists available at ScienceDirect

Neurocomputing

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

Fast hopfield neural networks using subspace projections

Daniel Calabuig *, Sonia Gimenez, Jose E. Roman, Jose F. Monserrat

Universidad Politécnica de Valencia, Camino de Vera s/n, 46022 Valencia, Spain

ARTICLE INFO

Available online 12 March 2010

Keywords:
Hopfield Neural Networks
Linear Constraints
Projection.

ABSTRACT

Hopfield Neural Networks are well-suited to the fast solution of complex optimization problems. Their application to real problems usually requires the satisfaction of a set of linear constraints that can be incorporated with an additional violation term. Another option proposed in the literature lies in confining the search space onto the subspace of constraints in such a way that the neuron outputs always satisfy the imposed restrictions. This paper proposes a computationally efficient subspace projection method that also includes variable updating step mechanisms. Some numerical experiments are used to verify the good performance and fast convergence of the new method.

1. Introduction

A Hopfield Neural Network (HNN) is a specific kind of recurrent neural network designed for the minimization of an energy function that contains several terms [9]. From the Hopfield neuron model, any problem that can be written in terms of a second order Lyapunov function can be solved with a quasi-optimal solution using HNNs. These neural networks have gained much relevance in the last decade as a good tool to solve complex optimization problems, mainly thanks to their fast response time. Certainly, one of the main advantages of neural techniques is the high computational speed obtained through their hardware implementations, which is even more valuable when considering their usage for industrial applications. Actually, the use of HNNs has been recently suggested for several time-constrained practical problems due to this characteristic—see for example [2,4,5].

However, HNNs have also acquired many detractors because of the poor quality of the obtained solutions, mostly when the energy function is non-convex [8]. This condition provokes some additional problems related to the convergence to non-desired local minima. These problems could be solved by means of a fine tuning of the energy function parameters that properly penalizes the non-desired states [4]. In general, a mathematical analysis of the borderline cases must be performed in order to derive the proper weighting values. However, this problem becomes even more complex in practical optimization problems, since reality imposes a set of strict constraints that must be taken into account. Consequently, in real problems a set of linear constraints is incorporated in the energy function, adding some additional constraint violation terms [9]. Despite being a common practice, some authors have demonstrated that the inclusion of the violation terms results in more likely invalid solutions [13] and even in a change in the network behaviour [3]. The underlying problem that causes these convergence problems lies in the fact that the cost terms run contrary to the constraint terms and, as a result, the networks converge to local minima far from the absolute minimum.

In order to tackle this problem, some authors proposed to confine the HNN to the feasible constraint subspace, hence ensuring the final solution validity [6,11]. Chu [6] proposed to project the energy gradient, which indicates the direction of movement, modifying the neuron inputs in such a way that the neuron outputs are always in the constraint plane. Although this seminal work was the first one bringing forward the usage of Projection-based HNNs (P-HNNs), it assumed a continuous-time neural network and no reference was made to a more realistic implementation which is inherently discrete in time. The main consequence is that discrete-time implementations continuously separate from the feasible subspace due to large computational errors when neurons are near the extremes. Moreover, the projection matrix was explicitly calculated from the constraints matrix, without considering issues of practical relevance such as computational efficiency and numerical robustness. On the other hand, Smith et al. [11] defined an iterative mechanism based on the integration of the projection of the neuron outputs – instead of projecting the energy gradient – together with an annealing technique for escaping local minima by permitting, in a controlled way, increments of the energy function. Comparing this mechanism with [6], both use the same calculation method for the projection matrix. However, Smith et al. incorporated more effective means to guarantee stability and convergence to feasible solutions, but at the expense of extremely increasing the computational burden.

* Corresponding author. Tel.: +34 96 3879582.
E-mail addresses: dacaso@iteam.upv.es (D. Calabuig), sogico@teleco.upv.es (S. Gimenez), jroman@itaca.upv.es (J.E. Roman), jomondel@iteam.upv.es (J.F. Monserrat).

0925-2312/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.neucom.2009.12.031

This paper presents a new, computationally efficient subspace projection method that includes the variable updating step mechanism proposed by Talaván and Yáñez [12]. Once the direction of movement is confined in the subspace of constraints, neuron states are modified so that the energy is reduced as much as possible. The proposed method performs projections by means of the orthogonalization with respect to an appropriate basis of vectors. This basis is dynamically augmented in order to guarantee that the modified neuron state vector does not exit the space of allowed solutions. This numerical procedure for the projection is computationally efficient.

Concerning the implementation of this Fast HNN (F-HNN), it is worth highlighting that, although some authors state that the HNN response could be attained in a few microseconds [2], this order of magnitude corresponds with the analogue implementation of continuous HNN. Nevertheless, so far the performance of HNN has been determined assuming a discrete-time implementation in computers and without implementing the analogue circuit. This fact is due to the enormous size of the hardware network when considering a high number of neurons and the difficulty of accurately implementing the resistor values, which can change network behaviour. In fact, the few hardware implementations found in the literature have been developed on digital devices, for example [1], in order to reduce the size of the network and solve the precision problems in the resistors' values. However, these implementations lose the benefits of the parallel interworking, since they make use of a central processing unit to update the neuron outputs sequentially in each iteration. An implementation on a computer can exploit the increasing potential of parallelism offered by current processor architectures. The proposed F-HNN method retains this inherent capability of parallelization so that fast response times can be expected. Even in a sequential implementation, the method presents advantages. Some numerical experiments are used to verify the performance and fast convergence of the proposed method as compared with other P-HNN techniques.

2. Mathematical foundations of the fast hopfield neural network

The objective of this section is to describe the main mathematical principles on which the F-HNN proposed in this paper is based. With this aim, the classical HNN is firstly explained. Next, there is a description of the two main concepts that, jointly used, enable the fast convergence of the new neural network: the variable updating step and projection of the energy gradient. In Section 2.5, another possibility that consists in executing variable reduction methods before the optimum search is discussed, justifying why this option was dismissed. The last part of this section deals with some efficient computational methods to reduce the number of operations and numerical errors. Section 3 summarizes the set of steps to be taken in each update of the F-HNN state.

2.1. Fundamentals of hopfield neural networks

HNNs can be completely defined with an energy function of the form [9]:

$$E(\mathbf{v}(t)) = -\frac{1}{2}\mathbf{v}(t)'\mathbf{T}\mathbf{v}(t) - \mathbf{i}'\mathbf{v}(t), \tag{1}$$

where $\mathbf{v}(t)$ is the $N\times 1$ vector of the neuron outputs at time t with elements $V_i(t) \in [0,1]$, $\mathbf{T}$ is an $N\times N$ symmetric matrix, $\boldsymbol{i}$ is an $N\times 1$ vector, with elements T_{ij} and I_i, respectively, of constant parameters that define the neural network, and N is the number of neurons. These parameters should be selected so that the energy minima inside the unit hypercube, i.e., $V_i(t)\in[0,1]$, are the desired solutions of the optimization problem.

Let $\mathbf{d}(t)$ be defined as the $N\times 1$ vector of the updating direction at time t. Then, each neuron is updated following:

$$V_i(t+1) = V_i(t) + \Delta_i(t), \tag{2}$$

$$\Delta_i(t) = \begin{cases} 0, & d_i(t) > 0, \quad V_i(t) = 1, \\ 0, & d_i(t) < 0, \quad V_i(t) = 0, \\ \beta(t)d_i(t), & \text{otherwise}, \end{cases} \tag{3}$$

where $\beta(t) > 0$ is the updating step at time t, and $d_i(t)$ is the i-th element of $\mathbf{d}(t)$. The updating step and direction must be selected to make the network converge towards a local minimum of the energy. For that reason, the updating direction is typically minus the energy gradient in the bibliography, since it points to the maximum decrement of the energy. However, any direction with a negative directional derivative decreases the energy too and hence is also a good candidate. The updating step is typically constant but, if so, has to be sufficiently small to prevent oscillations. Therefore, a lot of iterations are needed until the convergence point is reached.

2.2. Variable updating step

The Variable Updating Step (VUS) technique was proposed by Talaván and Yáñez [12] to increase the convergence speed of the neural network. The idea is to move in the direction $\mathbf{d}(t)$ until the energy minimum – in that direction – is reached. The energy over direction $\mathbf{d}(t)$ is:

$$E(\mathbf{v}(t)+\alpha\mathbf{d}(t)) = -\frac{1}{2}(\mathbf{v}(t)+\alpha\mathbf{d}(t))'\mathbf{T}(\mathbf{v}(t)+\alpha\mathbf{d}(t)) - \mathbf{i}'(\mathbf{v}(t)+\alpha\mathbf{d}(t)), \tag{4}$$

$$E(\mathbf{v}(t)+\alpha\mathbf{d}(t)) = E(\mathbf{v}(t)) - S_1\alpha + S_2\alpha^2, \tag{5}$$

$$S_1 = \mathbf{d}(t)'(\mathbf{T}\mathbf{v}(t)+\mathbf{i}), \tag{6}$$

$$S_2 = -\mathbf{d}(t)'\mathbf{T}\mathbf{d}(t). \tag{7}$$

Parameter α is any possible updating step. The energy of Eq. (5) is a quadratic function with respect to α. Thus, it has a critical point that can be either a maximum or a minimum. This critical point is [12]:

$$\left.\frac{dE(\mathbf{v}(t)+\alpha\mathbf{d}(t))}{d\alpha}\right|_{\alpha=\alpha_0} = 0 \Rightarrow \alpha_0 = \frac{S_1}{S_2}. \tag{8}$$

It is worth noting that S_1 is minus the directional derivative of the energy over the direction $\mathbf{d}(t)$. Therefore, let us assume that $S_1 \geq 0$ and hence that direction $\mathbf{d}(t)$ points towards points with less energy, i.e.:

$$\exists\varepsilon > 0: \quad \forall\alpha, 0<\alpha<\varepsilon \rightarrow E(\mathbf{v}(t)+\alpha\mathbf{d}(t)) \leq E(\mathbf{v}(t)). \tag{9}$$

Moreover, S_2 is the second derivative of the energy of Eq. (5) with respect to α. Therefore, if $S_2 > 0$, then the critical point α_0 is a minimum of Eq. (5). Conversely, if $S_2 < 0$, then α_0 is a maximum. In the first case, the VUS should be α_0 at most, since greater steps will further not reduce the energy and could even increase it. In the second case, the VUS can be as high as possible, since the energy will continuously decrease in the direction $\mathbf{d}(t)$.

Additionally to the previous paragraph, the VUS must satisfy some other constraints. More specifically, the updated neuron outputs must remain in the unit hypercube, i.e., $\mathbf{v}(t)+\beta(t)\mathbf{d}(t)\in[0,1]^N$. For this aim it is necessary to take into account the distances between the current neuron outputs and the extremes 0 and 1. Let us define $\mathbf{l}(t)$ as the $N\times 1$ vector of limits