ESANN'2001 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 25-27 April 2001, D-Facto public., ISBN 2-930307-01-3, pp. 359-364

Numerical implementation of continuous Hopfield networks for optimization

Miguel Atencia[2], Gonzalo Joya[1] and Francisco Sandoval[1]

[1]Departamento de Tecnología Electrónica
E.T.S.I.Telecomunicación
[2]Departamento de Lenguajes y Ciencias de la Computación
E.T.S.I.Informática
e-mail: matencia@lcc.uma.es
Universidad de Málaga, Campus de Teatinos, 29071 Málaga

Abstract. A novel approach is presented to implement continuous Hopfield neural networks, which are modelled by a system of ordinary differential equations (ODEs). The simulation of a continuous network in a digital computer implies the discretization of the ODE, which is usually carried out by simply substituting the derivative by the difference, without any further theoretical justification. Instead, the numerical solution of the ODE is proposed. Among the existing numerical methods for ODEs, we have selected the modified trapezoidal rule. The Hamiltonian Cycle Problem is used as an illustrative example to compare the novel method to the standard implementation. Simulation results show that this "numerical neural technique" obtains valid solutions of the problem and it is more efficient than other simulation algorithms. This technique opens a promising way to optimization neural networks that could be competitive with nonlinear programming methods.

1. Introduction

Since the seminal paper by Hopfield and Tank [7] was published, optimization has been a fundamental application of recurrent artificial neural networks (ANNs). However, neural techniques have not achieved enough performance to become competitive to nonlinear programming methods. These methods have the advantage of being problem-oriented, so any parameter may be adjusted to obtain maximal performance for a particular problem. On the contrary, computational neural networks are intended to be a general methodology. Thus, selection of neural models and adjustment of parameters must be performed for each particular problem. Moreover, no systematic and theoretically founded method has been found for this adjustment, which is usually carried out by

Thanks are due to Dr. Francisco Villatoro for suggesting the modified trapezoidal rule. This work has been partially supported by the Spanish Comisión Interministerial de Ciencia y Tecnología (CICYT), Project No. TIC98-0562

ESANN'2001 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 25-27 April 2001, D-Facto public., ISBN 2-930307-01-3, pp. 359-364

trial-and-error. Despite these drawbacks, optimization ANNs have achieved interesting results and further study could lead them to become as efficient as nonlinear programming methods.

After the target function is identified with the Lyapunov function of the network, the actual implementation of an optimization ANN comprises two additional stages. On one hand, the dynamics of the ANN must be selected either discrete [5] or continuous [6]. We restrict ourselves to continuous dynamics, as in a previous work [8] it was the only to exhibit adequate optimization capabilities. On the other hand, the selected dynamics must be implemented. Only an analogical device is capable of accurately representing a continuous model, but hardware implementation is costly and lacks the advantageous flexibility of the neural model. Thus, the implementation of ANNs is usually carried out through the software simulation of the model on a computer.

In Section 2. the continuous dynamics II is implemented, by means of the usual approximation of the derivative by the finite difference. In Section 3. the ordinary differential equation (ODE) that represents the continuous dynamics is solved by the modified trapezoidal rule. The usage of a numerical method was already suggested in [4] as an alternative to hardware implementation, but there exists no report about the effectiveness of this approach, to the best of our knowledge. As an illustrative example, the Hamiltonian Cycle Problem (HCP) is solved by means of both the standard and the numerical approach. Simulations suggest the enhanced performance of the novel method. Finally, in Section 4. some conclusions and directions of future research are exposed.

2. Continuous Dynamics

Hopfield ANNs are dynamical systems that have been proved to be asymptotically stable, as they have Lyapunov functions [5, 6, 1]. Convergence of the network implies seeking a minimum of a target function, if this is put into correspondence to the Lyapunov function. The evolution of a continuous ANN [6] is governed by the following system of nonlinear differential equations:

$$\frac{du_i}{dt} = -u_i + net_i \; ; \qquad s_i(t) = \mathrm{g}\left(\frac{u_i(t)}{\beta}\right) \; ; \qquad \mathrm{g}(x) = \frac{1}{1+e^{-x}} \tag{1}$$

where β is a parameter that controls the slope of the function g, and net_i is the linear term due to the input from the network, which, in the general higher order case, has the form:

$$net_i = \sum_{j=1}^{q} \sum_{\substack{(i_1,i_2\ldots i_j)\in C_j^n \\ i\neq i_1,i_2\ldots i_j}} w_{i\,i_1,i_2\ldots i_j}\, s_{i_1}\, s_{i_2}\, \ldots\, s_{i_j} - I_i \tag{2}$$

When the network is implemented on a digital computer, the differential is simply replaced by the finite difference:

$$\frac{\Delta u_i}{\Delta t} = -u_i + net_i \tag{3}$$

ESANN'2001 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 25-27 April 2001, D-Facto public., ISBN 2-930307-01-3, pp. 359-364

so that, when $\Delta t = 1$, the input potential to neuron i at simulation step $k+1$ is defined with respect to potential at step k as follows:

$$u_i(k+1) = net_i(k) \tag{4}$$

An alternative model [1] is defined by the equation $du_i/dt = net_i$, so its discretization is given by $\Delta u_i/\Delta t = net_i$. The implementation of this model, called continuous dynamics I [8], was shown to critically depend on the selection of Δt. In the sequel, we restrict ourselves to the dynamical equation (1).

Many results reported on the continuous model (1), actually implement the discrete equation (4), although sometimes this fact is not explicitly stated. The dynamical properties of this discrete-time continuous-state model differ from those of the continuous model. In particular, its stability is guaranteed only if neurons are updated asynchronously [2, 9, 13]. This fact is an important drawback, as it discourages any implementation by parallel algorithms on multiprocessor computers, which could lead to performance enhancements.

When applying a continuous-state model to an optimization problem that requires integer solutions, which is the most interesting case [12], some mechanism must be incorporated to force the states to discrete values. Usually, this is accomplished by gradually incrementing the slope of the continuous sigmoid function g, until, in the limit, it becomes a discrete step function. In our simulations, whenever the state approaches a fixed point inside the hypercube of states, the parameter β is decreased. As an example, the discretization (4) of the continuous model (1) is applied to the solution of the HCP and the results are presented in the next Section.

3. Numerical Implementation

In this Section, the implementation of the continuous model is accomplished by solving the equation (1) through a numerical method. An immediate question arises as to which method should be selected. Numerical analysts have developed a wide range of methods for different applications. The intuitive substitution of the differential by the finite difference leads to the well-known Euler method, which would be implemented by the vector equation $\Delta\vec{u}/\Delta t = -\vec{u} + \vec{net}$. When comparing this equation to those in the previous Section, it appears that this is the same dynamical equation as (3), but now the neurons are updated synchronously. As was mentioned before, this model is not guaranteed to be stable, so it is ruled out. An interesting alternative, which reduces the human time to program the simulation of the ANN, is the usage of a canned method, implemented by a commercial numerical package. We have tried the MATLAB implementation of the classical Runge-Kutta method, i.e. the function `ode45` [10]. Both this method and the adaptive step method suggested in [4] result in excessive computational cost.

Finally, we have chosen the trapezoidal rule, which approximates the solution of a -possibly multidimensional- differential equation $ds/dt = \mathrm{f}(s,t)$ by the iterative procedure $s_{n+1} = s_n + (h/2)\left(\mathrm{f}(s_n,t_n) + \mathrm{f}(s_{n+1},t_{n+1})\right)$, where h

ESANN'2001 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 25-27 April 2001, D-Facto public., ISBN 2-930307-01-3, pp. 359-364

is the step size. This nonlinear equation should be solved for s_{n+1}, because the trapezoidal rule is an implicit method. Alternatively, we estimate s_{n+1} by the Euler method, and get an explicit method that is called the modified trapezoidal method [3]:

$$\tilde{s}_{n+1} = s_n + h\ \mathrm{f}(s_n, t_n)\,; \qquad s_{n+1} = s_n + \frac{h}{2}\ (\mathrm{f}(s_n, t_n) + \mathrm{f}(\tilde{s}_{n+1}, t_{n+1})) \tag{5}$$

which is used to implement a continuous Hopfield ANN by calculating the function f from equation (1), yielding the following formula:

$$\mathrm{f}(s) = \frac{1}{\beta}\, s\,(1-s)\,(-\beta\ \mathrm{g}^{-1}(s) + net(s)) \tag{6}$$

where the functional relation $\mathrm{g}' = \mathrm{g}(1-\mathrm{g})$ has been used and the input potential u has been replaced by its value $\beta\,\mathrm{g}^{-1}(s)$ as a function of the state s. Examining equations (5) and (6), it is clear that the most complex operation of the method is the evaluation of the nonlinear function g^{-1}, and only two evaluations are needed for each step. Therefore, the computational complexity of the modified trapezoidal rule is roughly twice that of the Euler method of the same step size. However, in practice, the trapezoidal rule is expected to outperform the Euler method, because a larger step size may be selected without compromising stability.

When programming the numerical method, we realized that there exist important differences between implementing a Hopfield ANN and solving an ODE that represents a physical system. First of all, in the ANN we do not know a priori the time interval that the solution must span, so the implementation process must iterate until a stable state is reached. This fact must be taken into account when programming the method. Secondly, the final state of the ANN represents the solution of the considered problem, so only the final vector, not the whole trajectory, is of interest. Thus, the program has no need to keep intermediate states and a great amount of memory is saved. As a consequence of the importance of the final, stable state, rather than the intermediate states, the A-stability of the method is the main issue. This consideration strongly supports the trapezoidal rule due to its better stability properties [3], when compared to the Euler method. On the contrary, accurate representation of the trajectory is unimportant, so the step size may be incremented to reduce the computational cost, as long as stability is preserved. Finally, the vector equation (5) involves synchronous updating of neurons while the usual implementation given by equation (4) is asynchronous. Matrix synchronous operations can be efficiently implemented on parallel computers, thus leading to enhanced performance.

As an example, the method given by equations (5) and (6) is applied to the solution of the HCP [11], for a 5 node graph. In Table 1 the results are shown and compared to the usual asynchronous implementation given by equation (4), showing a greater performance of the numerical method. In both methods, the usual strategy to approach a discrete state was needed: the

ESANN'2001 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 25-27 April 2001, D-Facto public., ISBN 2-930307-01-3, pp. 359-364

parameter β is decreased whenever the evolution has reached a stable state. No special attention has been paid to the optimization of the simulation program. Performance increment is expected when the implementation is refined.

	Average time (seconds)	Feasible solution rate
Asynchronous simulation	23.41	100 %
Modified trapezoidal rule	10.97	100 %

Table 1: Solution of the 5 node HCP with the usual asynchronous simulation and the novel numerical implementation. The experiment was repeated 100 times to obtain average values.

In the presented simulations, the step size was defined as $h = 0.1$. Previous trials showed that a larger size could produce instability phenomena at the end of the simulation, near a discrete state. This fact can be intuitively justified in the following way: for a linear equation $y' = \lambda y$, the region of absolute stability of a numerical method depends on the product $h\lambda$. The linearization of our nonlinear equation is roughly proportional to $1/\beta$, which plays the role of λ. At the end of the simulation, β has been repeatedly decreased so as to reach a discrete state. Thus, if h is too high, the method may be out of the region of absolute stability. With an adequate modification of β the step size could be increased obtaining enhanced performance. The selection of the parameters of the method is crucial, and could lead to important refinements. This subject deserves more attention and is left for further research.

4. Conclusions and future directions

A novel approach is presented for the implementation of continuous Hopfield neural networks. The most important drawback of current simulation algorithms of continuous ANNs is the unaffordable computational cost. In this paper, the differential equation that represents the dynamics of the network is integrated by a numerical method. After several considerations, we have chosen the modified trapezoidal rule, which presents better stability properties than the simpler Euler method. The method has been applied to the solution of the Hamiltonian Cycle Problem, and simulations show that this technique presents higher performance, when compared to current algorithms.

When comparing the effectiveness of optimization algorithms, it should be emphasized that nonlinear programming is a decades old discipline, and it is still under a tremendous research effort. It is unfair requiring that novel techniques, such as neural networks, are immediately competitive to classical methods. The results in this paper must be analyzed under this view. They are still far from being competitive to nonlinear programming, but the way is open to several refinements. First of all, the presented algorithm allows for synchronous updating of neurons, and a parallel implementation on a multiprocessor computer is straightforward. Rigourous results from numerical analysis

ESANN'2001 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 25-27 April 2001, D-Facto public., ISBN 2-930307-01-3, pp. 359-364

may lead to an adequate adjustment of parameters such as the step size and the slope $1/\beta$ of the transfer function. Also, the selection of the numerical method is an open question. Those methods that preserve the fixed points of the original differential equation, even if a large step size is selected, are an interesting choice. All these techniques could be combined so that neural networks become efficient optimization algorithms. If they are considered as inferior, it should not be done before they have tried their best.

References

[1] S. Abe. Theories on the Hopfield neural networks. *IJCNN*, I, 1989.

[2] J. Bruck. On the convergence properties of the Hopfield model. *Proc. IEEE*, 78, 1990.

[3] C. Gear. *Numerical Initial Value Problems in Ordinary Differential Equations*. Prentice-Hall, 1971.

[4] J. Hertz, A. Krogh, and R. Palmer. *Introduction to the theory of neural computation*. Addison-Wesley, 1991.

[5] J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proc. Natl. Acad. Sci. USA*, 79:2554–2558, 1982.

[6] J. Hopfield. Neurons with graded response have collective computational properties like those of two-state neurons. *Proc. Natl. Acad. Sci. USA*, 81:3088–3092, 1984.

[7] J. Hopfield and D. Tank. 'Neural' computation of decisions in optimization problems. *Biol. Cybern.*, 52:141–152, 1985.

[8] G. Joya, M. A. Atencia, and F. Sandoval. Hopfield neural networks for optimization: Study of the different dynamics. *Neurocomputing (accepted)*, 2000.

[9] P. Koiran. Dynamics of discrete time, continuous state Hopfield neural networks. *Neural Computation*, 6:459–468, 1994.

[10] *MATLAB Function Reference*. The MathWorks Inc., 2000.

[11] S. Mehta and L. Fulop. An analog neural network to solve the hamiltonian cycle problem. *Neural Networks*, 6:839–881, 1993.

[12] M. Vidyasagar. Minimum-seeking properties of analog neural networks with multilinear objective functions. *IEEE Trans. On Automatic Control*, 40(8):1359–1375, 1995.

[13] L. Wang. On the dynamics of discrete-time, continuous-state Hopfield neural networks. *IEEE Trans. On Circuits and Systems-II*, 45(6):747–749, 1998.