ESANN'2008 proceedings, European Symposium on
Artificial Neural Networks - Advances in Computational Intelligence and Learning
Bruges (Belgium), 23-25 April 2008, d-side publi., ISBN 2-930307-08-0.

Design of Oscillatory Recurrent Neural Network Controllers with Gradient based Algorithms

Guillaume Jouffroy
AI laboratory, University Paris 8, gj@ai.univ-paris8.fr

Abstract. In this paper we address the question of the search of parameters value for recurrent neural networks to have an oscillatory behavior. A generalized partial teacher forcing is formalized when target signals are not all available. The drawbacks of these algorithms are covered, and a modified version is proposed toward a better general hybrid partial teacher forcing algorithm. The scope of shaping the oscillator is addressed.

1 Introduction

It is a difficult problem to obtain a *stochastic* learning rule to find parameters such that a Recurrent Neural Network (RNN) behaves as an oscillator, as the bifurcation phenomenon and the nonlinearities of the models arise. This may be seen by the relatively limited literature on the subject. Indeed beside analytic methods, for a lot of RNN oscillatory controllers, a priori parameters estimation or genetics algorithms (for e.g.[1]) are used, and in most of the works which tackle the question of a learning algorithm, algorithms can only be applied to a restricted neural architecture [2]. An interesting approach, which however does not directly concern the oscillatory parameters search, can be found in [3]. This work deals with the interesting idea of "shaping" the oscillatory signal we mentioned above, using reinforcement learning in the feedback.

The gradient descent based technique applied to the learning of continuous signals is a good candidate to build up an oscillatory RNN controller. Indeed there are several variations to optimize the search, whether it is in computation complexity [4] or in search speed with modified direction search principles [5].

This paper is structured as follows. In the second section we present how a RNN can be trained to produce oscillations, using a gradient descent algorithm with the so called "teacher forcing" principle. Then, in the third section we show how the need for the teacher signals can be reduced to a general formulation. In section four we discuss the drawbacks of this method together with new ideas about how convergence may be better obtained, and we give an example toward a generalized *hybrid* model. We discuss how the portrait of the resulting oscillator can be modified. In the last section we give concluding remarks.

2 Gradient descent with teacher forcing

For all the neural networks let us consider a simple fully connected RNN of the form

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{W}), \tag{1}$$

where $\mathbf{x} \in \mathbb{R}^n$ is the state vector of the network, and $\mathbf{W} \in \mathbb{R}^{n \times n}$ is the matrix of the weight connexions between neurons (w_{ij}being the connexion from neuron i to neuron j). Fully connected means every neuron is connected to all the others, and also self-connected ($w_{ii} \neq 0$). Each neuron model of this network is simply

$$\mathbf{f}(\mathbf{x}, \mathbf{W}) = -(1/\tau)(\mathbf{x} + \mathbf{W}s(\mathbf{x})) + \mathbf{I}, \tag{2}$$

with $s(x)$ a squashing function such as $tanh(x)$, $\tau \in \mathbb{R}^n$ a time constant vector and $\mathbf{I} \in \mathbb{R}^n$ an external input which we set for now as 0. It has been shown by [6] that the external inputs can be used to control the frequency of the oscillator.

The purpose of the learning algorithm is to match after transient, the state $\mathbf{x}$ of the network with a target "teacher" signal vector $\mathbf{x}^*$, by means of adapting the weight matrix $\mathbf{W}$. In the usual case the learning is achieved when an error criterion E, $E \in \mathbb{R}^n$ is less or equal than a minimum $\epsilon \in \mathbb{R}$, $\epsilon \simeq 0$

$$E = \frac{1}{2}(\mathbf{x} - \mathbf{x}^*) \circ (\mathbf{x} - \mathbf{x}^*) < \epsilon, \tag{3}$$

the operator $\circ$ being the Hadamard product.

Each component x_i^* of the teacher vector $\mathbf{x}^*$ is of the form $x_i^* = \sin(t + \phi_i)$. It is known that as noted for example in [2], there are restrictions between the phase latencies ϕ_i of the teacher signals. For the 2 neurons case, when a phase latency of 0 or π between teachers is approached, the error E correction time goes to infinity, i.e. convergence is not possible. One can easily see that for a two neurons network, a phase latency of π would plot a straight line in the phase plane which shows non-uniqueness of solutions. With one more dimension, we obtain a circle-like trajectory which is solvable by the three neurons network. For any phase latency between output neurons, one should use a combination of ϕ_i which lets the system find a solution. For a RNN of $n = 2$ neurons, we will choose $\phi_1 = 0$ and $\phi_2 = \pi/2$, $\eta = 0.1$, which gives the fastest convergence speed, in at most 300 time steps.

During the learning process the output of each neuron is replaced by the target signal, which means the neurons are thus independent from each other [7]. An other option is to only provide the target signals for the self connections, i.e. the connections of weight w_{ii}. In this case more computation is required as neurons have to be trained all together. The choice rely only upon technical considerations. Forcing the network with the oscillatory targets prevents the instability of the gradient [8].

The weight w_{pq} is adjusted in the following way

$$\dot{w}_{pq} = -\eta \sum_{i=1}^{n} \frac{\partial E}{\partial x_i} z_{pq}^i, \tag{4}$$

with $\eta \in \mathbb{R}$ is the learning rate, which we set as $\eta = 0.1$. $\mathbf{z} \in \mathbb{R}^{n \times n^2}$ is the sensitivity of the state of the system with respect to the weight matrix $\mathbf{W}$, whose elements for the teacher forcing gives

$$\dot{z}_{pq}^i = \frac{\partial f_i(\mathbf{x}(\mathbf{W}), W)}{\partial w_{pq}} = -\frac{1}{\tau} z_{pq}^i + s(h_j(t)) \tag{5}$$

The above equation should be solved from the equation (1) being forced, i.e. the target signal x_i^* in place of the input and/or output of the neuron i. There is a redundancy in solving this same differential equation for every weight. Such a redundancy can be reduced using the Green's function method proposed in [4].

The generated signals from the resulting network are not sinusoidal, but they are bent because of the non-linear squashing function s. To obtain a better signal shape, one should use a squashing function which is as linear as possible in $x \in [-1;1]$ but still continuous. Note that the quality of the signal's shape is sensitive to the integration step. We can also replace this function by a purely linear one. However the weights should be constant, otherwise the purely conjugate eigenvalues can easily escape from the imaginary axis, which means a loss of the limit cycle.

3 Partial teacher forcing

In general in robotics we can not know the teacher signal for every neuron of the network, but we can only provide the feedback signal of the output neurons.

The sensitivity equation (5) can be written for the general case in a matrix form

$$\dot{\mathbf{z}} = \mathbf{J}_{\mathbf{f_x}}(M)\mathbf{z} + \mathbf{J}_{\mathbf{f_w}}(M), \tag{6}$$

where $\mathbf{J}_{\mathbf{f_x}}(M)$ and $\mathbf{J}_{\mathbf{f_w}}(M)$ are the jacobian matrices of the function $\mathbf{f}$ respectively to $\mathbf{x}$ and $\mathbf{w}$ at the point M. In the teacher forcing case (5) we have

$$\dot{\mathbf{z}} = -\mathbf{I}\mathbf{z} + \mathbf{J}_{\mathbf{f_w}}(M), \tag{7}$$

$\mathbf{I}$ being the identity matrix. For convenience $\mathbf{z}$ is the matrix of elements z_{pq}^i, i being the column index representing the ith neuron, and the indices pq for the line vectors, $i,p,q \in [1;n]$ of w_{pq}, the weight matrix element of $\mathbf{W}$ ($z_{pq}^i = \partial x_i/\partial w_{pq}$).

If we can only partially teacher force the network, i.e. we provide a target signal(s) x_i^* only for some neuron(s) i, the sensitivity equation (5) in the general case can be written as

$$\tau \dot{z}_{pq}^i + z_{pq}^i = \sum_{j=1}^{n} a_{ji} w_{ji} \frac{\partial s(\overline{x}_j)}{\partial \overline{x}_j} z_{pq}^j \quad + \quad \varsigma_{iq} \left[b_{pq}^i s(\overline{x}_p) + c_{pq}^i s(\overline{x}_p^*) \right], \tag{8}$$

with $1 \leq i,p,q \leq n$. $a_j = b_j = 0$ and $c_j = 1$, when j is a neuron output, $a_j = b_j = 1$ and $c_j = 0$ otherwise. $\varsigma_{iq} = 1$ if $i = q$, 0 otherwise, or in the context of the matrix form (6), with the present neuron model, the components of $\mathbf{J}_{\mathbf{f_x}} = A$ read

$$A_{ij} = -a_{ij} + b_{ij} w_{ji} \frac{\partial s(x_j)}{\partial x_j}, \tag{9}$$

with $a_{ij} = 1$ when $i = j$, 0 otherwise, $b_{ij} = 1$ when i is not a forced neuron, 0 otherwise. $\mathbf{J}_{\mathbf{f_w}} = \mathbf{B}$ has the same form as $\mathbf{z}$ and its elements are such that $B_{pq}^i = 0$ when $i \neq q$, $B_{pq}^i = s(x_p^*)$ if p is a forced neuron, $B_{pq}^i = s(x_p)$ otherwise.

We can also notice that even if the error term (3) decreases exponentially, the weights take an infinite time to become stable, though the values obtained give a reasonable oscillatory behavior. This problem has some stability drawbacks for the resulting system. Typically, the non-forced neuron(s) change(s) amplitude throughout the learning process whereas the forced neuron(s) reach(es) its(their) target fast. The weights convergence can also be infinite when the phase of teacher signals are close even if there is a sufficient number of neurons to solve the problem with the teacher forcing algorithm.

In a perspective of relearning a different oscillatory behavior, or in case of injury, we should notice that with this algorithm, for some solutions, not all neurons are used. This means algorithms such as pruning can not be used. The non-forced neurons have also to be initialized to a non-zero state so that they can be used (otherwise the learning equations give zero all the time), or with random input weights.

4 Toward an hybrid partial teacher forcing

The partial teacher forcing has two drawbacks we can address here: weight convergence and no use of all neurons.

If we add a constraint to the non-forced neurons such that they do not compromise the convergence of the forced neurons to their target signal, and reach a reasonable and stable amplitude, both problems can be dealt with. The idea is to split the error function, in two different error functions, one for the neurons which have to match a target, and one for the non-forced neurons. So the error would be

$$E = \begin{cases} \frac{1}{2}(x_i - x_i^*)^2 & \text{if } i \text{ is a forced neuron} \\ E_h & \text{otherwise} \end{cases} \tag{10}$$

The error of the non-forced neurons E_h needs to include characteristics of the forced neurons, i.e. this error represents a criterion of "correlation" in a wide sense. There are a lot of possibilities concerning this criterion (relative distance, angle difference...). As an example we choose for a network of $n = 2$, the neuron 2 being forced

$$E_h = \frac{1}{2}(\dot{\mathbf{x}}^T \mathbf{x})^2 \tag{11}$$

One can easily see that all the components $\frac{\partial E}{\partial x_i}$ are already computed elsewhere in the learning process. Here again $\eta = 0.1$. Note that these error elements are calculated with the forced equations of the system (1). Fig. 1 shows the evolution of the weight matrix $\mathbf{W}$ during the learning process. The weight elements are all stable after transient and the system is robust to the perturbation of a weight. The convergence takes at most 2500 time steps, depending on the initial conditions. The non-forced neuron reaches the same stable amplitude as the forced one. The matrix weight obtained is $\mathbf{W} = [1.31\ 1.27; -1.19\ 1.13]$.
The existence of a limit cycle can be proved with the Dulac's criterion and the eigenvalues. The network will converge to the limit cycle with initial conditions

ESANN'2008 proceedings, European Symposium on
Artificial Neural Networks - Advances in Computational Intelligence and Learning
Bruges (Belgium), 23-25 April 2008, d-side publi., ISBN 2-930307-08-0.

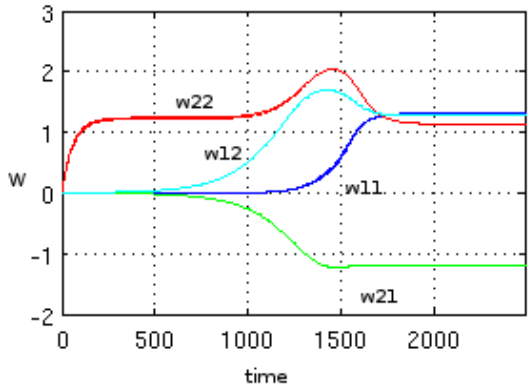

Figure 1: Modified partial teacher forcing: weights are stable after transient, and only at most 8 times slower than the teacher forcing with all target signals known.

on $\mathbf{x}$ being up to twice the amplitude of the target signals, without the need to learn to the system a suitable attractive vector field contrary to [2]. Eq. (10) makes the limit cycle to be a circle. This circle can easily be stretched in any direction to change the phase relationship between neurons, using in place of (10), $\mathbf{A}$-orthogonality instead of orthogonality, i.e. $E_h = \frac{1}{2}(\dot{\mathbf{x}}^T \mathbf{A}\mathbf{x})^2$.

The circle is stretched/contracted in the directions of the eigenvectors of $\mathbf{A}$, if the eigenvalues are greater/less than one. This 'shaping' of the limit cycle means to change the phase relationship between quiescent neurons. This is different to many works, where the phase relationship between *oscillatory neurons* are dealt with, but here we have no explicit phase information.

5 Shape of the oscillator

We already mentioned the fact that the signals generated after the learning process may be altered in their shape because of the pseudo-linear part of the squashing function s. This idea can be used to modify the shape of the oscillator signals in an on-line manner. For example if we let $s(x) = tanh(2/(1+e^{-\alpha x})+1$, we can change the portrait of the oscillator, increasing α, changing neurons signal from sinusoidal to triangular. However it should be noted that it has the drawback of getting an increase of the expected amplitude, because we change the point on s where x is maximum. When $\alpha < 2$, s become linear, and thus the limit cycle is no more guaranteed. α should also not be too big (e.g. 250), as s will tend to a step function which contains a discontinuity. The shape of the oscillator's portrait can also be modified if we change the time constants τ_i. If they are identical, a change will only affect the speed of the orbits. However a change on a particular time constant will affect the amplitude of its neuron i. The amplitude of the other neurons will all get a roughly equivalent loss of

amplitude, but the amplitude of the neuron i will be much more decreased.

6 Conclusion

Many works concerning oscillatory controllers in the robotics field deal with evolutionary process, and this leaves several dynamic processes in the unknown. We have presented the teacher forcing algorithm in the usual way together with a general matrix formulation, for the design of oscillatory recurrent neural network controllers. We have then given a generalized formulation of how to use a partial teacher forcing when a target signal is not available for all neurons and we have shown how to reduce its drawbacks. The error is split in two different types of functions: the error for the forced neuron to match its target, and the error for the non-forced neuron, which is a "correlation" factor with the forced neuron. Our results show that indeed, contrary to the partial teacher forcing, during the learning, stable weights are found in a finite time, and that the non-forced neuron reaches the same stable amplitude as the forced one. More results need to be done to see if this method may be applied to arbitrary functions. We also address the question of shaping a default oscillator to its need, instead of the design of a specific oscillator. The methods presented here demonstrate that there are simple ways to be able to design oscillatory controllers without loosing the understanding of the dynamics that arise, which are hard to see in the widely used evolutionary methods for such a task.

References

[1] Raffaele M. Ghigliazza and Philip Holmes. A minimal model of a central pattern generator and motoneurons for insects locomotion. *SIAM Journal on Applied Dynamical Systems*, 3(4):671–700, 2004.

[2] Fu-Sheng Tsung and Garrison W. Cottrell. Phase-space learning for recurrent networks. Technical Report CS93-285, Dept. Computer Science and Engineering, University of California, San Diego, 1993.

[3] Takeshi Mori, Yutaka Nakamura, Masa-Aki Sato, and Shin Ishii. Reinforcement learning for cpg-driven biped robot. *Nineteenth National Conference on Artifical Intelligence*, pages 623–630, 2004.

[4] Guo-Zheng Sun, Hsin-Hsiung Chen, and Yee-Chun Lee. Green's function method for fast on-line learning algorithm of recurrent neural networks. In John E. Moody, Stephen Jose Hanson, and Richard Lippmann, editors, *Advances in Neural Information Processing Systems (NIPS)*, volume 4, pages 333–340. Morgan Kaufmann, 1991.

[5] Nicol N. Schraudolph and Thore Graepel. Towards stochastic conjugate gradient methods. In *Proceedings of the 9th International Conference on Neural Information Processing (ICONIP '02), Singapore*, 2002.

[6] Robert Haschke, Jochen J. Steil, and Helge Ritter. Controlling oscillatory behavior of a two neuron recurrent neural network using inputs. In *Proc. of the Int. Conf. on Artificial Neural Networks (ICANN)*, Wien, Austria, 2001.

[7] Peter F. Rowat and Allen I. Selverston. Learning algorithms for oscillatory networks with gap junctions and membrane currents. *Network*, 2:17–41, 1991.

[8] Kenji Doya. Bifurcations of recurrent neural networks in gradient descent learning. *IEEE Transactions on neural networks*, 1994.