5 V 02/02/00

INTERNATIONAL JOURNAL OF NEURAL SYSTEMS

2

Volume 9, Number 3
June 1999

World Scientific
Singapore • New Jersey • London • Hong Kong

Redes Neurais

Volume 9 • Number 3 • June 1999

CONTENTS

International Journal of Neural Systems, Vol. 9, No. 3 (June, 1999) 235–242
Special Issue on Neural Networks in Brazil: V Brazilian Symposium on Neural Networks

Unsupervised Learning and Recall of Temporal Sequences: An Application to Robotics

GUILHERME DE A. BARRETO & ALUIZIO F. R. ARAÚJO
DEPARTMENT OF ELECTRICAL ENGINEERING, UNIVERSITY OF SÃO PAULO
C.P. 359, 13560-970, SÃO CARLOS, SP, BRAZIL
{GBARRETO, ALUIZIOA}@SEL.EESC.SC.USP.BR

This paper describes an unsupervised neural network model for learning and recall of temporal patterns. The model comprises two groups of synaptic weights, named competitive feedforward and Hebbian feedback, which are responsible for encoding the static and temporal features of the sequence respectively. Three additional mechanisms allow the network to deal with complex sequences: context units, a neuron commitment equation, and redundancy in the representation of sequence states. The proposed network encodes a set of robot trajectories which may contain states in common, and retrieves them accurately in the correct order. Further tests evaluate the fault-tolerance and noise sensitivity of the proposed model.

1 Introduction

It is well-known that artificial neural network (ANN) models have been successfully applied to a wide range of static pattern processing tasks, such as pattern recognition, categorization and feature detection. However, many real world tasks demand the ability of natural, or artificial neural systems to process patterns in which the information content depends not only on static or spatial features, but also on the temporal order of the input patterns. Such a set of temporally-ordered patterns are commonly referred to as spatio-temporal sequences [1].

Spatio-temporal patterns are often represented as a discrete finite sequence of feature vectors, also called items, components or states, consecutively connected. In such sequences, the variable *time* plays a fundamental role and is usually encoded by the presentation order of the sensory inputs, and by the relative time duration between each two consecutive sequence items.

Handling of context, short-term memory models, and the processing of multiple sequences are important issues regarding temporal patterns. Contextual information usually refers to the prior knowledge necessary to identify unambiguously a component of a temporal sequence [2]. A short-term memory (STM) model is basically a retention mechanism which aims at storing, for some period of time, information about past components of a given input sequence [1]. Such a mechanism of temporary storage allows the network to establish temporal associations between consecutive states of a sequence. Multiple sequence processing is concerned with the encoding of several sequences that may share components. Dealing with multiple sequences has two challenging aspects: the first is the so-called *catastrophic interference* [3] whereby later network training destroys vestiges of former training. The second is the ambiguity that results during recall of shared states.

Most of the ANN models for processing of spatio-temporal sequences are based on either a multilayer Perceptron trained with temporal versions of gradient-based learning algorithms [4]-[7] or the Hopfield model [8], [9]. Both cases consider a temporal sequence as associations of consecutive items, and the network should learn these associations as input-output pairs.

More recently, unsupervised neural network models have been proposed and successfully applied to a wide range of sequence processing tasks [2], [10]-[15]. Such models extract temporal information from the input stimuli without the need of an external teacher or signal to indicate the correct answer or relation to be pursued. In this sense, one can say that the learning process is stimuli-driven and governed by principles of self-organization.

This paper is primarily concerned with the proposal of an unsupervised neural network model and its application to a difficult problem in robotics: learning of multiple robot trajectories.

The paper is organized as follows. In Section 2, we present some concepts regarding unsupervised neural learning in robotics and the advantages in adopting such an approach. In Section 3, we develop our model discussing in detail all its components. In Section 4, we evaluate the performance of the model through computer simulations and discuss the main results. We conclude the paper in Section 5 with a summary of the network features, and also present possible directions for further developments.

2 Robotics and Unsupervised Learning

Robot learning problems are usually characterized by: (i) a real-world system that tightly integrates perception, decision making and execution; (ii) complex domains, yet the expense of using actual robotic hardware often prohibits the collection of large amounts of training data; and (iii) real-time requirements in which decisions must be made within critical time constraints. These characteristics present challenges to learning systems, and motivate considerably the search for good neural models.

The research in ANN and its application to distinct domains make it possible to investigate solutions for complex problems in robotics following different learning paradigms. In particular, unsupervised learning has appealing characteristics for use in robotics and temporal sequence processing. Behavior in unsupervised neural networks emerges by means of a self-organization process, which substantially reduces the robot programming burden [16].

Another suitable property is the ability to generalize the learned material that enables the robot controller to respond to unexpected situations. Moreover, the structure of neural networks allows massive parallel processing [17] which enables the network to respond quickly in generating real-time control actions. In particular, unsupervised models are often fast, encouraging their use in incremental and on-line learning.

The robot task we are interested in is the so-called *trajectory tracking*, in which the robot has to follow a prescribed path [18]. Such a task can be handled within the framework of ANN models, since trajectories can be seen as spatio-temporal sequences. In this case, the neural network must learn to associate consecutive states of a trajectory and store these transitions for total or partial reproduction of the memorized trajectory. For the purpose of recall, the network receives as input the current state of the robot

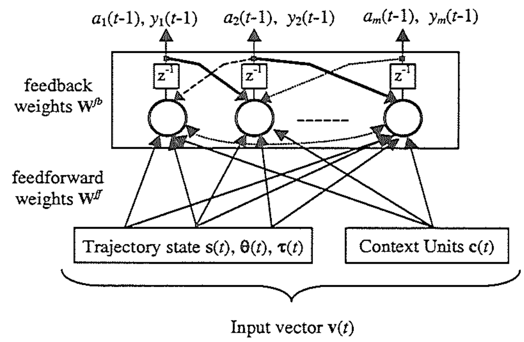

Figure 1. Topology of the proposed unsupervised network

and responds with the next state until the goal state has been reached.

Despite the appealing properties, few unsupervised ANN models have been proposed. Among them we can refer the readers to [19], [20], and [21]. However, such references have not directly addressed tracking and learning of multiple trajectories, and some properties of great interest such as tolerance to fault and noise. In this paper, we aim at proposing an unsupervised ANN model to deal with complex robotics problems.

3 Presenting the Model

The basic topology of the proposed network is shown in Figure 1. The input units broadcast the state vector, $\mathbf{v}(t) \in \mathbf{R}^n$, which comes from sensor readings and is defined as:

$$\begin{aligned} \mathbf{v}(t) &= \{\mathbf{s}(t), \theta(t), \tau(t), \mathbf{c}(t)\} \\ \mathbf{s}(t) &= (x(t), y(t), z(t)) \\ \theta(t) &= \{\theta_1(t), \ldots, \theta_{dof}(t)\},\ \theta_i(t) \in \mathbf{R} \\ \tau(t) &= \{\tau_1(t), \ldots, \tau_{dof}(t)\},\ \tau_i(t) \in \mathbf{R} \\ \mathbf{c}(t) &= (x_g, y_g, z_g) \end{aligned} \qquad (1)$$

where $\mathbf{s}(t)$ is the spatial location of the robot end-effector, $\theta(t)$ is the joint-angle vector, $\tau(t)$ is the applied-torque vector at time step t, and *dof* is the number of degrees of freedom of the robot arm. The context vector $\mathbf{c}(t)$ is set to the target spatial position of the robot end-effector, which is given as a task specification.

The network has two groups of synaptic connections, namely: (i) competitive feedforward weights, $\mathbf{W}^{ff}(t) = [w_{ij}^{ff}(t)]_{mxn}$, and (ii) Hebbian feedback weights, $\mathbf{W}^{fb}(t) = [w_{jr}^{fb}(t)]_{mxm}$. Feedforward weights encode the spatial features of the sequence (static information), while feedback weights store the state transitions (temporal information). At the beginning of training, they are initialized as $w_{ji}^{ff}(0) = rand[0, 1]$ and $w_{jr}^{fb}(0) = 0$.

There is an activation value, $a_j(t) \in [0,1]$, $j = 1, \ldots, m$, and an output value $y_j(t) \in \mathbf{R}$, $j = 1, \ldots, m$, associated with each neuron in the output layer. The activation indicates the winners of the current competition, and the output indicates the neuron that encodes the next state of the trajectory. These variables are initialized as $a_j(0) = y_j(0) = 0$, for all j, every time an input is observed.

It is worth emphasizing that by a discrete time step t, we mean a period that begins with the presentation of an input pattern $\mathbf{v}(t)$, and ends with the calculation of the output signal $y_j(t)$. Similarly, the instant $(t-1)$ corresponds to the learning or recalling period of the last observed input, and $(t+1)$ refers to the next one.

3.1 Training Feedforward Weights

In our model, the number of winners is set to K, hence each state of the input trajectory is encoded by K different neurons. Thus, neurons are ranked according to the proximity of their weight vectors and the current input as follows:

$$\begin{aligned} p_1(t) &= \arg\min_j \{\|\mathbf{v}(t) - \mathbf{w}_j^{ff}(t)\| + e_j(t)\} \\ p_2(t) &= \arg\min_{j \notin \{p_1(t)\}} \{\|\mathbf{v}(t) - \mathbf{w}_j^{ff}(t)\| + e_j(t)\} \\ &\vdots \qquad\qquad \vdots \qquad\qquad \vdots \\ p_K(t) &= \arg\min_{j \notin \bigcup_{i=1}^{K} p_i(t)} \{\|\mathbf{v}(t) - \mathbf{w}_j^{ff}(t)\| + e_j(t)\} \end{aligned} \qquad (2)$$

where $p_i(t)$ is the index of the $i-th$ winner. Then, activations of the winners are calculated by the following equation:

$$a_{p_i}(t) = \begin{cases} a_0 \phi^{i-1} & \text{for } i = 1, \ldots, K \\ 0 & \text{for } i > K \end{cases} \qquad (3)$$

where $a_0 \geq 1$ and $0 < \phi < 1$. Equation 3 means that the activation levels of $p_1(t), \ldots, p_K(t)$ decrease as their distances to $\mathbf{v}(t)$ increase.

The feedforward weights are trained with a simple competitive learning rule, defined as in [22]:

$$\Delta \mathbf{w}_j^{ff}(t) = \alpha a_j(t)[\mathbf{v}(t) - \mathbf{w}_j^{ff}(t)] \qquad (4)$$

where $0 < \alpha \leq 1$ is the feedforward learning rate. Equation 4 indicates that only neurons with $a_j(t) \neq 0$ have their weights modified. If we set $\alpha \simeq 1$ the input vector can be learned in a single pass.

Context is stored together with the state of the robot arm. As we handle multiple trajectories, context helps the network to make the right decision in case of ambiguity by identifying the trajectory to which a shared state belongs. Several trajectories are learned sequentially, as a single long trajectory, and context units change accordingly every time a new trajectory starts. For recall purposes, Equation 4 is omitted, and we set K=1 (WTA behavior).

A neuron is not chosen as a winner for more than one state of the input trajectory. This is accomplished through the definition of a *commitment equation* as follows:

$$e_j(t+1) = e_j(t) + \beta a_j(t) \qquad (5)$$

where $\beta \gg 1$. At the beginning of training and testing, we set $e_j(0) = 0$, for all j. The effect of Equation 5 is the following: once a neuron is committed to an input vector, its function $e_j(t)$ assumes a high value to remove this neuron from subsequent competitions. Similar mechanisms were proposed by [23] and [24].

3.2 Training Feedback Weights

In this work, we propose a very simple learning rule to deal with temporal dependencies. The premise considered here is that, for learning temporal order, feedback connections between neurons should depend not only on current activation $a_j(t)$, but also upon the previous activation $a_r(t-1)$ as well. A mathematical formulation takes the form of a time-delayed Hebb-like learning rule [25], defined as:

$$\Delta w_{jr}^{fb}(t) = \lambda a_j(t) a_r(t-1) \qquad (6)$$

or, in matrix form as:

$$\Delta \mathbf{W}^{fb}(t) = \lambda \mathbf{a}(t) \mathbf{a}^T(t-1) \qquad (7)$$

where $0 < \lambda \leq 1$ is the feedback learning rate. Equation 6 learns the order of a state transition between two consecutive competitions. Thus, non-zero connections are established from winners at $t-1$ to winners at time t. The states themselves are stored in the feedforward weights through Equation 4.

In Equation 7, $\Delta \mathbf{W}^{fb}(t)$ is a *feedback memory matrix* corresponding to the learning of one state transition, and determined solely by the pair ($\mathbf{a}(t)$,$\mathbf{a}(t-1)$).

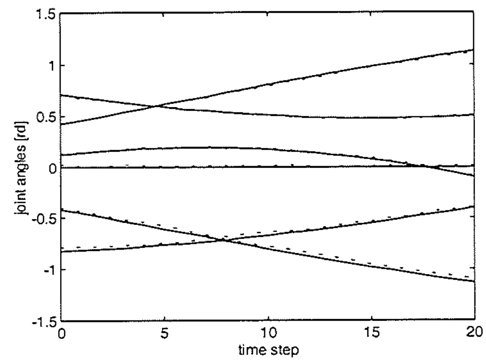

Figure 5. The desired (solid lines) and obtained (dashed lines) joint angles for trajectory s-21 in the presence of neuron faults.

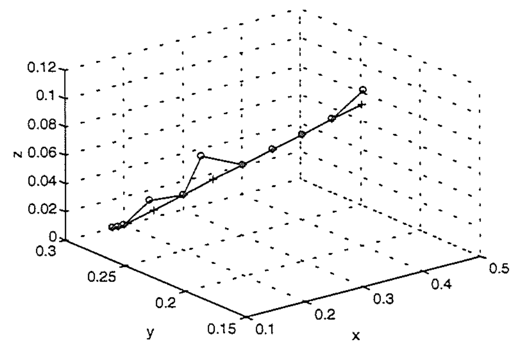

Figure 7. Evaluation of the redundancy mechanism for trajectory t-11 in the presence of noisy inputs.

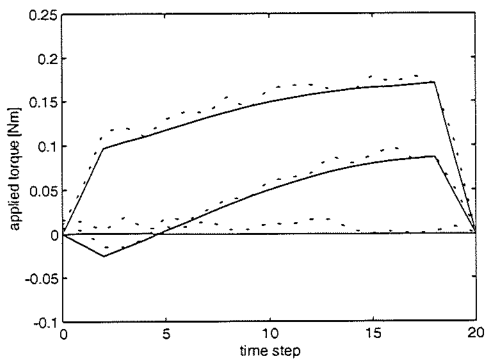

Figure 6. The desired and obtained applied torques at the three end-effector joints for trajectory s-21 in the presence of neuron faults.

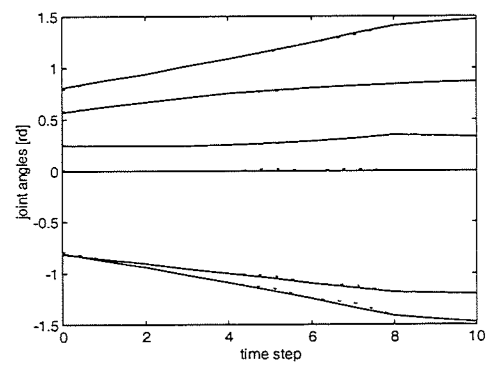

Figure 8. Retrieved joint angles of trajectory t-11 for the case of noisy inputs

the tests include zero-mean Gaussian stochastic noise ($\sigma^2 = 0.85$) which simulates measurement errors. A typical result obtained for the spatial positions is shown in Figure 7 for trajectory $t-11$ which share an intermediate position (0.3, 0.2, 0.0) with $t-15$ and $t-21$.

It can be noted from Figure 7 that trajectory s-11 was correctly recalled, with a resulting tracking error $MSE_{t-11} = 0.0000377$. Note that the presence of noise during the tests forces some states of the trajectory to be encoded by second-place winners.

The retrieved joint angles for the test with noise are shown in Figure 8. The retrieved applied torques are shown in Figure 9. Note that the retrieved and the desired curves in both figures are very similar.

From the tests carried out in this section, we can conclude that the proposed model is able to learn and recall robot trajectories unambiguously with very low tracking errors. In addition, it is robust, in the sense that it can handle neuronal failures and noise to some extent. If both these abilities are desirable, a balance between fault- and noise-tolerance must be found.

5 Conclusions and Further Work

In this work, we were particularly concerned with the problem of fast and accurate learning of single and multiple temporal sequences in the form of robot trajectories. An unsupervised context-based neural network algorithm is the learning strategy mainly because it is based on self-organization, a rather generic principle, that is employed in a wide range of application domains. For the proposed model, the correct temporal order of the states of a particular trajectory is the property that emerges as a result of the network's self-organizing nature.

The contribution of this work to the field of unsupervised neural networks are twofold: (i) use of a Hebb-like learning rule to process spatio-temporal patterns and (ii) application of such a model to control robots involved in tracking tasks. The viabil-

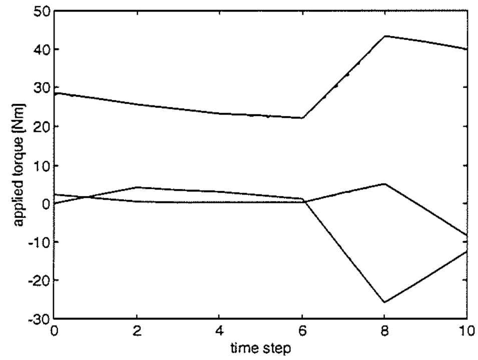

Figure 9. Retrieved applied torques of trajectory t-11 for the case of noisy inputs

ity of the proposed learning algorithm was evaluated through simulations of robot trajectories with reasonable complexity.

In summary, the proposed model is characterized as being simple, fast and accurate, handling ambiguity when dealing with multiple sequences, able to learn trajectories of different lengths, model-free for the development of a control law, fault and noise tolerant, and able to recall a trajectory from any intermediate point.

Some robot tasks may demand the storage of long sequences, the redundancy and commitment mechanisms could cause the use of a high number of output neurons, leading to a situation in which all of them were used and could not be allocated to encode new sequences. To deal with this drawback, further studies should develop mechanisms for alleviating memory requirements. We suggest: 1) the use of constructive algorithms [30] to include neurons when necessary, and/or 2) the development of context-sensitive neurons, to be responsible for storing shared or repeated states of the various sequences. During recall, such neurons would identify the sequence to which a shared state belongs.

Acknowledgements

The authors would like to thank FAPESP and PICDT/CAPES/UFC for their financial support.

References

[1] M. C. Mozer. Neural net architectures for temporal sequence processing. In A. Weigend and N. Gershenfeld, editors, *Predicting the Future and Understanding the Past*, pages 243–264. Addison-Wesley, Redwood City, CA, 1993.

[2] D.-L. Wang and M. A. Arbib. Complex temporal sequence based on short-term memory. *Proceedings of IEEE*, 78:1536–1543, 1990.

[3] D.-L. Wang and B. Yuwono. Anticipation-based temporal pattern generation. *IEEE Trans. on Systems, Man, and Cybernetics*, 25:615–628, 1995.

[4] M. I. Jordan. Attractor dynamics and parallelism in a connectionist sequential machine. In *8th Annual Conference of the Cognitive Science Society*, pages 531–546, Amherst, MA, 1986.

[5] J. L. Elman. Finding structure in time. *Cognitive Science*, 14:179–211, 1994.

[6] B. Pearlmutter. Gradient calculations for dynamic recurrent neural networks: a survey. *IEEE Transactions on Neural Networks*, 6(5):1212–1228, 1995.

[7] A. F. R. Araújo and H. D'Arbo Jr. Partially recurrent neural network to perform trajectory planning, inverse kinematics, and inverse dynamics. In *Proc. IEEE Int. Joint Conf. on System, Man, and Cybernetics*, pages 1784–1789, San Diego, CA, 1998.

[8] H. Sompolinsky and I. Kanter. Temporal association in asymmetric neural networks. *Phys. Rev. Lett.*, 57:2861–2864, 1986.

[9] I. Guyon, L. Personnaz, J.-P. Nadal, and G. Dreyfus. Storage and retrieval of complex sequences in neural networks. *Phys. Rev. A*, 38:6365–6372, 1988.

[10] J. Kangas. Time-delayed self-organizing maps. In *Proc. IEEE Int. Joint Conf. on Neural Networks*, volume II, pages 331–336, San Diego, CA, 1990.

[11] J. G. Taylor. The temporal Kohonen map. *Neural Networks*, 6(3):441–445, 1993.

[12] K. Kopecz. Unsupervised learning of sequences on maps with lateral connectivity. In F. Fogelman-Soulié and P. Gallinari, editors, *Proc. ICANN'95, Int. Conf. on Artificial Neural Networks*, volume I, pages 449–454, Paris, 1995.

[13] M. Girolami and C. Fyfe. A temporal model of linear anti-Hebbian learning. *Neural Processing Letters*, 4(3):139–148, 1996.

[14] O. A. Carpinteiro. A hierarchical self-organizing map model for sequence recognition. In *Proc. ICANN'98, Int. Conf. on Artificial Neural Networks*, volume 2, pages 815–820. Springer, 1998.

[15] G. Wallis. Spatio-temporal influences at the neural level of object recognition. *Network: Computation in Neural Systems*, 9:265–278, 1998.

[16] J. Heikkonen and P. Koikkalainen. Self-organization and autonomous robots. In O. Omidvar and P. van der Smagt, editors, *Neural Systems for Robotics*, pages 297–337. Academic Press, 1997.

[17] P. C. Y. Chen, J. K. Mills, and K. C. Smith. Performance improvement of robot continuous-path operation through iterative learning using neural networks. *Machine Learning*, 23:75–104, 1996.

[18] J. J. Craig. *Introduction to robotics: mechanics and control*. Addison-Wesley, 2nd edition, 1989.

[19] M. Kuperstein and J. Rubistein. Implementation of an adaptive neural controller for sensory-motor coordination. *IEEE Control Systems Magazine*, 9(3):25–30, 1989.

[20] H. Ritter, T. Martinetz, and K. Schulten. *Neural computation and self-organizing maps: an introduction.* Addison-Wesley, Reading, MA, 1992.

[21] J. A. Walter and K. J. Schulten. Implementation of self-organizing networks for visuo-motor control of an industrial robot. *IEEE Transactions on Neural Networks*, 4(1):86–95, Jan 1993.

[22] J. Hertz, A. Krogh, and R. G. Palmer. *Introduction to the theory of neural computation.* Addison-Wesley, Redwood City, CA, 1991.

[23] D. L. James and R. Miikkulainen. SARDNET: a self-organizing feature map for sequences. In G. Tesauro, D. Touretzky, and T. Leen, editors, *Advances in Neural Information Processing Systems 7*, pages 577–584. MIT Press, 1995.

[24] D. Calvert, D. A. Stacey, and M. S. Kamel. A distance based network for sequence processing. In *Proc. IEEE Int. Joint Conf. on Neural Networks*, pages 2507–2512, Anchorage, AK, 1998.

[25] D. O. Hebb. *The organization of behavior.* Wiley, New York, 1949.

[26] M. Hagiwara. Time-delay ART for spatio-temporal patterns. *Neurocomputing*, 6(5-6):513–521, 1994.

[27] P. R. Montague and T. J. Sejnowski. The predictive brain: temporal coincidence and temporal order in synaptic learning mechanisms. *Learning & Memory*, 1:1–33, 1994.

[28] T. Kohonen. *Self-organizing maps.* Berlin, Heidelberg, Springer, 2nd extended edition, Springer Series in Information Sciences, Vol. 30, 1997.

[29] P. I. Corke. A robotics toolbox for MATLAB. *IEEE Robotics and Automation Magazine*, 3(1):24–32, 1996.

[30] B. Fritzke. Growing cell structures - a self-organizing network for unsupervised and supervised learning. *Neural Networks*, 7(9):1441–1460, 1994.