MULTIPLE AND TIME-VARYING DYNAMIC MODELLING CAPABILITIES OF RECURRENT NEURAL NETWORKS

Andrew D. Back
Brain Information Processing Group, Frontier Research Program
The Institute of Physical and Chemical Research (RIKEN)
2-1 Hirosawa, Wako–shi, Saitama 351-01 Japan

Abstract

In this paper, we propose some theories regarding the dynamical system representational capabilities of recurrent neural networks with real-valued inputs and outputs. It is shown that multiple nonlinear dynamic systems can be approximated within a single nonlinear model structure. A relationship is identified between this class of recurrent network, hybrid models and agent based systems.

1 Introduction

Recurrent neural networks are very general models and have been proven to offer significant computational capabilities such as Turing equivalence with linear slowdown [20]. There has been some interest in applying recurrent networks to dynamical systems and control problems. It has been shown that such models possess universal dynamic approximation capabilities [11, 21].

Recently, it has been shown by Feldkamp et. al. [8–10], that a class of recurrent networks can learn to model several dynamical systems and switch between them, depending on the characteristics of the input signal. Instead of learning to model just one system, the network was able to learn several models with very different properties.

This type of modelling phenomenon is not unique to the above example however. There appear to be a number of interrelated methods in the literature which have been studied mostly independently, see for example: [1, 2, 4, 12–14, 18, 19].

In this paper, we examine this phenomema further, and propose a theory which explains how recurrent neural networks can possess the capability of modelling a number of dynamical systems simultaneously. Examples are also given.

2 Preliminaries

Function approximation is the task of approximating a mapping given by the function $F_1 : \mathcal{R}^m \to \mathcal{R}^n$, where $\mathcal{R}$ is the usual Euclidean space. *Functional approximation* is the task of approximating a mapping $F_2 : \mathcal{C}(K) \rightarrow \mathcal{R}$ where $\mathcal{C}(K)$ is a Banach space of continuous functions on a compact set K, defined by the norm $\|f\|_{\mathcal{C}(K)} = \max_{x\in K}|f(x)|$. The implication here, is that we have a mapping F_2 which maps the past inputs[1] to the current output (i.e. a variable $\zeta \in \mathcal{R}$) [3]. *Operator approximation* is the task of approxiating a mapping $F_3 : \mathcal{C}(K) \rightarrow \mathcal{C}(K)$. Thus, in this case, we seek to approximate a mapping of an input sequence $x(t) \in \mathcal{C}(K)$ to an output sequence $y(t) \in \mathcal{C}(K)$.

We term $\mathcal{C}_o(K_o)$ the space of operators on a compact set K_o. The mapping from one operator space $\mathcal{C}_o(K_o)$ to another operator space $\mathcal{C}_o(K_o)$ is called, for lack of a better name, an *operational map*.

Definition 1 *An operational map H is defined by*

$$H : \mathcal{C}_o(K_o) \rightarrow \mathcal{C}_o(K_o) \tag{1}$$

Definition 2 *A recurrent network (RNN) is defined by*

$$\mathbf{x}(t+1) = \mathbf{f}(\mathbf{Ax}(t) + \mathbf{Bu}(t) + \mathbf{Ey}(t)) \tag{2}$$
$$\mathbf{y}(t) = \mathbf{Cx}(t) + \mathbf{Du}(t) \tag{3}$$

where $\mathbf{x}(t) = [x(t)x(t-1)...x(t-n_x)]^T$ *is an* $n(n_x+1)\times 1$ *vector,* $\mathbf{u}(t) = [u(t)u(t-1)...u(t-n_u)]^T$, $\mathbf{y}(t-1) = [y(t-1)...y(t-n_y)]^T$, $\mathbf{f}$ *is* $n(n_x+1)\times 1$ *vector-valued function with sigmoid elements, typically defined as* $\{f(x)\} = \tanh(x)$, $\mathbf{A} \in \mathcal{R}^{n(n_x+1)\times n(n_x+1)}$, $\mathbf{B} \in \mathcal{R}^{n(n_x+1)\times mn_u}$, $\mathbf{C} \in \mathcal{R}^{pn_y\times n(n_x+1)}$, $\mathbf{D} \in \mathcal{R}^{pn_y\times m(n_u+1)}$ *and* $\mathbf{E} \in \mathcal{R}^{n(n_x+1)\times pn_y}$. *The bias terms are not explicitly shown, but are included within* $\mathbf{u}(t)$ *as a fixed-value input.*

This characterization of a recurrent network gives a general framework from which many well known structures can be derived, e.g. [20–22].

Definition 3 *A* multiple model $\mathcal{M}_1(\mathcal{S}_1, \theta_1)$ *with fixed structure* $\mathcal{S}_1$ *and parameter vector* θ_1 *exhibits a set of characteristic properties* $\mathcal{P}_1^{'}$*, where*

$$\mathcal{P}_1^{'} = \begin{cases} \mathcal{P}_{1i} & i = 1, 2, ... \quad \textit{discrete multiple model} \\ \mathcal{P}_{1v} & v \in \mathcal{R}. \quad \textit{continuous multiple model} \end{cases} \tag{4}$$

3 Multiple Model Representation Using Operational Maps

A multiple model G can be characterized as an operator, which itself, consists of multiple operators F. Consider a multiple model operator $\Lambda : x \rightarrow y$, $x \in X \subseteq \mathcal{C}(K)$, $y \in Y \subseteq \mathcal{C}(K)$ describing the input-output functional relationship. Additionally, Λ contains an operator F subject to, for example, $\Xi : F_a \rightarrow F_b$ $F_a \in \mathfrak{F}_a \subseteq \mathcal{C}_o(K_o)$, $F_b \in \mathfrak{F}_b \subseteq \mathcal{C}_o(K_o)$. Now, instead of considering Λ directly, we are interested in the existence of, and mechanisms by which there may arise mappings of the form $H : F_i \rightarrow F_{i+1}$, where

[1] A sequence $x(t)$, which we may sample at discrete points $t = 0, 1, ...$ is given by $x_s(t) = [x(0), x(1), \cdots]^t$ and is a function in $\mathcal{C}(K)$.

$i = 1, 2, ...$ is the index of the discrete functionals in the discrete multiple model case, or $H : F_a \to F_b$ corresponding to the continuous multiple model case. In each case, $H \subseteq \mathcal{C}_\Omega(K_o)$ where $\mathcal{C}_\Omega(K_o)$ is the space of operational maps.

Hence, in addressing the issue of multiple models, this implies that we seek to answer the following question: 'Is it possible to find a nonlinear model which approximates an operational map $H : \mathcal{C}_o(K_o) \to \mathcal{C}_o(K_o)$?' We consider this question below.

The existence of a general operational map is made clear by the following theorem.

Theorem 1 *An operational map H given by*

$$F_\theta, x(t) \longmapsto H(F) : x(t) \to y(t) \tag{5}$$

can be obtained by the interconnection of a parameterized operator $F_\theta : \mathcal{C}(K) \to \mathcal{C}(K)$, and functional $M_c : \mathcal{C}(K) \to \mathcal{R}^m$ according to

$$\begin{aligned} y &= F_\theta(x;\theta) && (6)\\ \theta &= M_c(x) \qquad \theta = \theta_0 \; t \leq 0 && (7) \end{aligned}$$

where $x(t), y(t) \in \mathcal{C}(K)$ are continuous, real-valued input and output functions respectively of $t \in \mathcal{R}_+$, θ is the m-dimensional parameter vector of F_θ and θ_0 is the initial parameter vector.

Proof Sketch. Let there exist an operator F_θ determined by the parameter vector θ and a functional map M_c capable of universal approximation in the sense of , for example, [17]. For the ith parameter within a given model F_θ, we have

$$s_i = \theta_i z_i \tag{8}$$

Now, let M_c be a functional. Therefore, for any input sequence $x \in X$, M_c results in any desired set of parameters $\tilde{\theta} \in \tilde{\Phi} \subseteq \mathcal{R}^m$, such that

$$\begin{aligned} s_i &= M_{ci}(x) z_i && (9)\\ &= \tilde{\theta} z_i && (10) \end{aligned}$$

where $M_c = [M_{c1} \cdots M_{cm}]^T$ is a vector function. Thus, the existence of $M_c : u \to \tilde{\theta}$ permits any arbitrary F_θ to be obtained due to the mapping M_c. The operational map $H = \{F_\theta, M_c\}$ is general in the sense that it is capable of the same approximation as F_θ, but can be varied arbitrarily for any sequence x.

4 Universal Operational Maps

From Theorem 1, we can obtain the following result.

Theorem 2 (Universal Operational Map-I) *There exists a parameterized model $G(\theta) : x(t) \to \hat{y}(t)$, where $x \in X \subseteq \mathcal{C}_o(K_o)$ and $\hat{y} \in Y \subseteq \mathcal{C}_o(K_o)$, such that*

$$|y(t) - \hat{y}(t)| < \epsilon \tag{11}$$

where $F_{\theta'}, x(t) \longmapsto H(F) : x(t) \to y(t)$ and $\epsilon > 0$.

Proof Sketch. Let $F_{\theta'}$ be a parameterized model capable of universally approximating any operator (e.g. a Chen network [7]). Let M_c be a time delay neural network or other structure having universal functional approximation characteristics [6, 17]. Then from Theorem 1, there exists a neural network G defined collectively by $F_{\theta'}$ and M_c, which can approximate, arbitrarily closely, some operational map $H = \{F_{\theta'}, M_c\}$, where $H : \mathcal{C}_o(K_o) \to \mathcal{C}_o(K_o)$.

Remarks

1. The implication of the theorem is that every weight in F is replaced by an additional network M_{ci}. This provides the means of approximating nonlinear operational maps. As noted earlier, related approaches have been considered in the literature (see, for example, [18, 19]).

2. It is possible to introduce any required type of model for M_c. Thus, a hybrid model can be elegantly obtained.

3. The model G is a sigma-pi network, but can also be interpreted as a modular structure.

4. A more general form for M_c is $s = M_{ci}(x, u)$. In this case, the output from M_c is not used as a parameter, but receives the previous parameter input u and gives the previous output s after the parameter. Hence, we can derive the following related theorem.

Theorem 3 (Universal Operational Map-II) *A universal operational mapping $F_\theta, x(t) \longmapsto H(F) : x(t) \to y(t)$ is given by the interconnection of a universal operator $F : (X, V) \to Y$ and a single-input single-output universal function map $M_{c1} : X \to V$ according to*

$$y = F(x, v; \theta) \tag{12}$$

$$v = M_{c1}(x) \tag{13}$$

where $x \in X \subseteq \mathcal{C}(K)$, $v \in V \subseteq \mathcal{C}(K)$, $y \in Y \subseteq \mathcal{C}(K)$, and $\theta \in \Theta \subseteq \mathcal{R}^m$.

Proof Sketch. The proof follows directly from Theorem 1. The operator $F(x, v)$ is a universal approximator as independent as required in each of its inputs. Therefore, for every distinct value of v, e.g. $v = 1, 2, ...$ a distinct universal approximation $F : \mathcal{C}(K) \to \mathcal{C}(K)$ may be obtained. Let M_{c1} prescribe some operator from the input x to the extra input v. Therefore, a distinct universal approximation F can be obtained as required for any given input x, hence a universal operational model is obtained.

Remarks

1. In the above theorems, for universal operational maps, it may be assumed that F is given by a network as described in [7].
2. F may also be given by a recurrent neural network, with the universal approximation properties in the sense of those shown by Sontag [21]. In this case, the resulting recurrent network H, possesses also multiple modelling capabilities[2].

These results offer a different viewpoint for modular networks. Previously, modular neural networks were used to approximate a mapping by spatial decomposition [15]. Here, operational maps perform a temporal decomposition.

The discrete multiple model can be interpreted as a hybrid system [5]. Moreover, a neural network may arbitrarily form such hybrid systems, even in the course of training, *without the user necessarily being aware of this phenomena occurring.*

Discrete multiple models are related to agent systems [2, 16]. The models F and M_c can each be viewed as agents cooperating together to produce a more complex mapping than either is capable of acting alone. The framework proposed here also includes models such as mixtures of experts [12].

5 Synthesis of Multiple Models by Bias Shifting

Here we present a simple constructive approach to show how multiple models may be synthesized in both feedforward and recurrent neural networks. The approach we propose is well suited for multiple models which are comprised of a set of discrete F models and is applicable to synthesis as well as analysis, though the latter case is not discussed here.

Theorem 4 *An MLP model $G(x, v)$ can form multiple unique function mappings*

$$G(x, v) = m_v(x), \; v = 1, ...p \tag{14}$$

defined by

$$m_v(x) = \sum_{i=1}^{N} c_{vi} g \left(\sum_{j=1}^{m} \xi_{vij} x + \phi_{vi} \right) \tag{15}$$

where the extra input v indexes the desired mapping $m_v(x)$.

Proof Sketch. Without loss of generality, let $N = np$. This specifies p subgroups within G, each of n hidden units. Therefore we have

$$G(x, v) = \sum_{i=1}^{N} c_{hi} g \left(\sum_{j=1}^{m} \xi_{hij} x + \theta_{hi} + rv \right) \quad h = 1, ...p \tag{16}$$

$$\theta_{hi} = \phi_{hi} - rh \tag{17}$$

[2]Though to be precise, one may wish to qualify the sense of the approximation in terms of the specific characteristics of universal approximation being performed, i.e. in the sense defined by Chen and Chen [7] or Sontag [21].

For sufficiently large r, if we consider approximations on the range $[a, b]$ where $|a, b| \ll r$, then setting $v = \{1, .., p\}$ results in

$$G(x, v) = \sum_{i=1}^{N} c_{vi} g\left(\sum_{j=1}^{m} \xi_{vij} x + \phi_{vi}\right) \tag{18}$$

$$= m_v(x) \tag{19}$$

as required.

Theorem 5 *An RNN model $G(x, y, v)$ can form multiple unique mappings*

$$G(x, y, v) = m_v(x, y), \; v = 1, ...p \tag{20}$$

where the extra input v indexes the desired mapping $m_v(x, y)$.

Proof Sketch. Omitted due to lack of space. The proof follows a similar procedure to that used for Theorem 4.

The implication of this theorem is that by appropriate biases offsets in the different groups of units, various units in the recurrent network can be “pushed” in and out of action. Note that this method can be used as a means of switching between different discrete models, or in a continuous sense, by setting r to an appropriately small value.

6 Examples

In this section, we give a number of examples, which indicate the idea of the multiple models discussed in the paper. In order to clarify the results, we use simple model structures.

6.1 General Examples

Here we consider some general examples to indicate some possible types of multiple models which may be synthesized.

1. Input Amplitude Dependent Model.

   This type of model is derived by using a characteristic function M_c given by

$$M_c(x) = \begin{cases} \theta_0 x & x^2 \leqslant r_0 \\ \theta_1 x & r_0 < x^2 \leqslant r_1 \\ \theta_2 x & x^2 > r_2 \end{cases} \tag{21}$$

   where $\{r\}$ are scalar values. The model behaviour is dependent on the instantaneous value of the input $x(t)$. In particular, the model will switch parameter sets when the amplitude of $x^2(t)$ crosses certain thresholds. This model can be considered as a multiple bilinear or state-dependent model.

2. Frequency Dependent Model Selection.
   M_c may be a function of the frequency of the incoming signal. For example,
$$M_c(x) = f_m\left(X(\omega)\right) \tag{22}$$
   where f_m is a function of $X(\omega) = FFT[x]$.
3. Sequential Model Selection
   Here, the model is based on some predetermined time-sequence and bears a close relationship with hybrid systems considered in robotics [5]. In this case, the input x is a sequence of symbolic binary values. $M_c(x)$ processes this symbolic binary input and upon recognition of a particular sequence, outputs a 1 and holds it for a specified period of time, otherwise the output is a zero.

6.2 A Recurrent Network Multiple Model

6.2.1 Network Architecture

Based on the multiple model framework presented in this paper, a recurrent network multiple model[3] can be proposed as follows.

$$y(t) = \frac{q^{n-m}\prod_{i=1}^{m_1}\left(q - g_{\beta ri}\left(u_{\beta ri}\right)\right)\prod_{j=1}^{m_2}\left(q - g_{\beta ci}\left(u_{\beta ci}\right)\right)\left(q - g^*_{\beta ci}\left(u_{\beta ci}\right)\right)}{\prod_{i=1}^{n_1}\left(q - g_{\alpha ri}\left(u_{\alpha ri}\right)\right)\prod_{j=1}^{n_2}\left(q - g_{\alpha ri}\left(u_{\alpha ri}\right)\right)\left(q - g^*_{\alpha ri}\left(u_{\alpha ri}\right)\right)} x(t) \tag{23}$$

where $\{g\}$ are the element-wise characteristic functions corresponding to M_c. $\{u\}$ are ancillary inputs. Since the poles and zeros in (23) are the outputs of nonlinear functions, they are termed nonlinear poles and zeros respectively and the model can also be considered as a *nonlinear pole-zero model*. The coefficient functions $\{g\}$ and ancillary inputs$\{u\}$ provide a wide scope for introducing a variety of models.

6.2.2 Example: An Input Amplitude-Dependent Multiple Model

A nonlinear pole-zero multiple model is synthesized in this example as follows. The model is described by

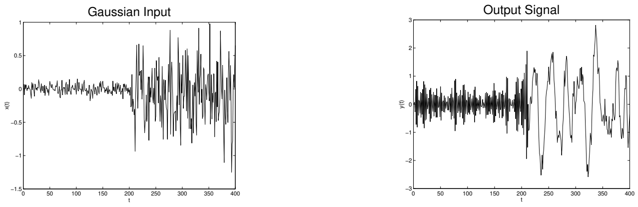
$$\begin{aligned} H_m(q) &= \frac{1}{\left(1 - G_1\left(u\right)q^{-1}\right)\left(1 - G_1^*(u)q^{-1}\right)} \qquad (24) \\ G_1\left(u\right) &= \alpha_1 u\left(t\right) + \alpha_2 \bar{u}\left(t\right) \qquad G_1^*\left(u\right) = \alpha_1^* u\left(t\right) + \alpha_2^* \bar{u}\left(t\right) \end{aligned}$$

where $G_1\left(u\right)$ and $G_1^*(u)$ provide complex conjugate outputs and $H_m(q)$ switches between two underlying linear models $H_1(q)$, $H_2(q)$, depending on the binary ancillary input signal $u\left(t\right)$.

$$H_i(q) = \frac{1}{1 + a_{i1}q^{-1} + a_{i2}q^{-2}} \tag{25}$$

(26)

[3]For convenience and clarity of the example, we have chosen to use linear models as a basis, however these may be arbitrary nonlinear models in practice. Since the behaviour of linear systems is well known, it is easier to visualize the differences between the two models in this case.



(a) (b)

Figure 1: The performance of a recurrent neural network multiple model. In this simple example, (a) is the input signal, and (b) shows the model output due to the input.

where, for the purposes of this example, we choose $a_{11} = 1.6$, $a_{12} = 0.73$, $a_{21} = -1.9$, $a_{22} = 0.925$ and α_i are the corresponding first order poles of $H_i(q)$. The input $u(t)$ can be obtained, for example, by

$$u(t) = \Gamma\left(E_s\left[x^2(t)\right]\right) \tag{27}$$

where $E_s[\cdot]$ denotes the short term expectation and

$$\Gamma(\varsigma) = \begin{cases} 1 & \varsigma \geqslant 0 \\ 0 & \varsigma < 0 \end{cases} \tag{28}$$

Therefore, when the magnitude of the short term average of the squared input signal reaches a certain threshold, defined by Γ, the model will change. The performance of this model is shown in Fig. 1, where the model characteristics due to the change in input can be seen.

6.2.3 Example: An RNN Modelling a Time-varying Linear System

This example we synthesize a recurrent neural network model which models a time-varying linear system, described by

$$H_m(q) = \frac{1}{1 - \zeta_1(u,t)\, q^{-1} + \zeta_2(u,t)\, q^{-2}} \tag{29}$$

The model varies as a function of the ancillary input signal u, and the coefficient functions $\zeta_i(t)$, $i = 1, 2$ are in this example, simple linear functions, given by

$$\begin{aligned} \zeta_i(t) &= c_{i0} + c_{i1}\zeta_i^{'}(t) & (30) \\ \zeta_i^{'}(t) &= \frac{b_{i0}u(t) + b_{i1}u(t-1)}{1 + a_{i1}q^{-1} + a_{i2}q^{-2}} & (31) \end{aligned}$$

For the purposes of this example, we choose the parameters[4] $a_{11} = 1.6$, $a_{12} = 0.9$, $a_{21} = -1.9$, $a_{22} = 0.925$, $b_{10} = 1.0$, $b_{11} = 0.5$, $b_{20} = 0.9$, $b_{21} = 0.5$, $c_{10} = 1.75$, $c_{11} = 0.04$, $c_{20} = 0.8$, $c_{21} = 0.02$ and $u(t) = x(t)$.

[4] The parameters were chosen on the basis of known, stable linear systems. In the case of this linear model, it is necessary to bound the pole positions as usual. However, the situation changes when normal recurrent networks with sigmoid functions are employeed. The signals are then bounded by the sigmoids.

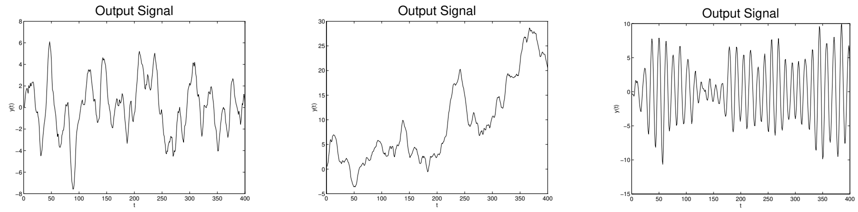


Figure 2: The time-varying recurrent network described in Section 6.2.3 is capable of exhibiting a variety of different behaviours as observed here.

The resulting multiple model, which is an extension of the usual bilinear structure, can be described by the difference equation form of nonlinear pole-zero model given by

$$y(t) = x(t) + \zeta_1(t)\, y(t-1) + \zeta_2(t)\, y(t-2) \quad (32)$$

$$\zeta_i(t) = c_{i1} + c_{i2}\left(b_0 x(t) + b_1 x(t) - a_{i1}\zeta_i(t-1) + a_{i2}\zeta_i(t-2)\right) \quad (33)$$

where $x(t)$ is the input and $y(t)$ is the output. Examples of the model behaviour are shown in Fig. 2, which indicate some of the 'richness' of the model's capabilities.

7 Conclusions

In this paper, we have given theories which indicate how nonlinear models, including feedforward and recurrent networks, can approximate systems known as multiple models. We have shown that such models can be considered in terms of operational maps. It was shown that there exist classes of neural networks which can universally approximate operational maps. The results provide an explanation for the experimental behaviour of some recurrent networks in being able to model multiple dynamic systems [8–10].

Acknowledgments

The author acknowledges the support of the Frontier Research Program RIKEN and helpful discussions with S. Amari, T. Chen and L. Feldkamp.

References

[1] A.D. Back and A.C. Tsoi. A cascade neural network model with nonlinear poles and zeros. In *Proc of 1996 Int Conf on Neural Information Processing ICONIP'96*, volume 1, pages 486–491, Piscataway, NJ, 1996. IEEE.

[2] C.J. Bett and M.D. Lemmon. Bounded amplitude control using multiple linear agents, technical report ISIS-97-004. Technical report, ISIS Group, University of Notre Dame, March 1997.

[3] S.P. Boyd. *Volterra Series: Engineering Fundamentals*. PhD thesis, University of California, Berkeley, 1985.

[4] M. S. Branicky, V. S. Borkar, and S. K. Mitter. A unified framework for hybrid control: background, model, and theory. Technical Report LIDS-P-2239, Laboratory for Information and Decision Systems, Dept of Electrical Engineering and Computer Science, MIT, June 1994.

[5] R. Brockett. Hybrid models for motion control systems. In *Perspectives in Control*, pages 29–54, Boston: Birkhauser, 1993.

[6] T. Chen and H. Chen. Approximations of continuous functionals by neural networks with application to dynamic systems. *IEEE Trans. Neural Networks*, 4(6):910–918, 1993.

[7] T. Chen and H. Chen. Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems. *IEEE Trans. Neural Networks*, 6(4):911–917, 1995.

[8] L.A. Feldkamp. Personal communication, 1996.

[9] L.A. Feldkamp, G.V. Puskorius, and P.C. Moore. Adaptation from fixed weight dynamic networks. In *Proceedings of IEEE International Conference on Neural Networks, Washington (ICNN '96)*, pages 155–160, 1996.

[10] L.A. Feldkamp, G.V. Puskorius, and P.C. Moore. Adaptive behaviour from fixed weight dynamic networks. *Information Sciences*, 98:217–235, 1997.

[11] K. Funahashi and Y. Nakamura. Approximation of dynamical systems by continuous time recurrent neural networks. *Neural Networks*, 6:801–806, 1993.

[12] M.I. Jordan and R.A. Jacobs. Hierarchical mixtures of experts and the EM algorithm. *Neural Computation*, 6(2):181–214, 1994.

[13] D. Lainiotis. Partitioning: A unifying framework for adaptive systems I. estimation. *Proc. IEEE*, 64:1126–1143, 1976.

[14] K. Narendra, J. Balakrishnan, and M. Ciliz. Adaptation and learning using multiple models switching and tuning. *IEEE Control Systems Magazine*, 15(3):37–51, 1995.

[15] E. Ronco and P.J. Gawthrop. Modular neural networks: a state of the art. Technical Report CSC-95026, Center for System and Control, University of Glasgow, May 1995.

[16] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach,*. Prentice-Hall, Englewood Cliffs, NJ, 1995.

[17] I.W. Sandberg. Approximation theorems for discrete-time systems. *IEEE Trans. Circuits, Syst.*, 38(5):564–566, 1991.

[18] M. Schetzen. *The Volterra and Wiener Theory of Nonlinear Systems*. Wiley, (Reprint Ed. Krieger Publishing, 1989), New York, NY, 1980.

[19] J. Schmidhuber. Learning to control fast-weight memories: An alternative to dynamic recurrent networks. *Neural Computation*, 4(1):131–139, 1992.

[20] H.T. Siegelmann, B.G. Horne, and C.L. Giles. Computational capabilities of recurrent narx neural networks. *IEEE Trans. on Systems, Man and Cybernetics*, 26(6), 1996.

[21] E. Sontag. Neural nets as systems models and controllers. In *Proc. Seventh Yale Workshop on Adaptive and Learning Systems*, pages 73–79. Yale University, 1992.

[22] R.J. Williams and D. Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, 1:270–280, 1989.