Proc. 1997 Int. Conf. on Neural Inf. Proc and Intelligent Inf. Syst., Vol. I, pp. 326-329, Springer-Verlag, 1998.

Approximation of Hybrid Systems by Neural Networks

Andrew D. Back[1] and **Tian-Ping Chen**[2]

[1]Brain Information Processing Group, The Institute of Physical and Chemical Research (RIKEN), Wako-shi, 351-01, Japan.

[2]Department of Mathematics, Fudan University, Shanghai, P.R. China

Abstract

In this paper it is shown that hybrid systems can be approximated arbitrarily well by recurrent neural networks. The results indicate that the newly emerging field of hybrid systems can be considered in terms of the architectures and learning algorithms developed for neural network models. Examples are given of the types of architectures that can be developed.

1 Introduction

Recently, it has been observed in experimental work, that a class of recurrent network is capable of learning to model several different dynamical systems within a fixed structure with constant parameters (Feldkamp, Puskorius and Moore 1996). Depending on the characteristics of the input signal, one of several models could be selected. Such *multiple modelling* capabilities are potentially useful in applications where fast adaptive behaviour to nonstationary systems is required. To date however, there does not appear to have been any clear explanations of a mechanism by which this multiple modelling behaviour occurs. Multiple model approaches for control and system identification have been considered by various authors. A common theme of this field is that the state space is decomposed in some manner and individual models are assigned to different regions. Early work in the field is that of Lainiotis who proposed the partitioning method (Lainiotis 1976a, Lainiotis 1976b). More recently, Narendra et. al. have proposed a general framework for multiple model control which includes multiple observers, multiple controllers, and a method of switching between the models (Narendra, Balakrishnan and Ciliz 1995). This work is also related to LTSTM (long term short term memory) networks (Schmidhuber 1992), multiple agent control (Nerode and Kohn 1993), mixtures of experts (Jacobs, Jordan, Nowlan and Hinton 1991) and local model networks (Ronco and Gawthrop 1995).

The capability of modelling multiple systems is similar to that observed in hybrid systems. Hybrid systems are usually defined as having both continuous-state and discrete-state or symbolic systems (Branicky, Borkar and Mitter 1994) where the discrete-state system permits jumps to occur between a set of continuous-state dynamical systems. In this paper, we seek to delineate the relationship between recurrent networks and hybrid systems. We are particularly interested in giving a more precise understanding to the problem than simply referring to the universal approximation capabilities of recurrent networks, which is of course an obvious, but not so helpful statement in terms of synthesizing or analysing such models.

2 Main Results

2.1 Definitions

It is well known that nonlinear dynamical systems can be treated as functionals and operators (Sandberg 1991). Accordingly, a multiple model G can be characterized as an operator, which itself, consists of multiple operators F. Our main results are based on the notion of "operational maps", which give a specific framework for mappings between operators, which describe in a general manner, the function of dynamic models. Operational maps generalize the notion of an operator. An operator can vary the functional mapping at every sequential instant. An operational map can vary the operator mapping at every sequential instant.

Definition 1 *An operational map $H \subseteq \mathcal{C}(K_o)$ is defined by*

$$H : \mathfrak{F}_a \to \mathfrak{F}_a \tag{1}$$

where we define the space of operators as existing on a compact set K_o,$F_a \in \mathfrak{F}_a \subseteq K_o$, $F_b \in \mathfrak{F}_b \subseteq K_o$.

For a definition of recurrent networks, see for example (Sontag 1992).

The hybrid model is a combination of nonlinear ordinary differential equations (ODEs) and discrete event dynamical systems (DEDs). A suitable definition of a hybrid system is given as follows (Brockett 1993).

Definition 2 *A continuous output discrete time hybrid system is defined by*

$$\begin{aligned} x(t+1) &= \alpha\left(x(t), u(t), z\lfloor p \rfloor\right) & (2)\\ p(t+1) &= \beta\left(x(t), u(t), z\lfloor p \rfloor\right) & (3)\\ z\lceil p \rceil &= \gamma\left(y(t_p), z\lfloor p \rfloor, v\lfloor p \rfloor\right) & (4)\\ y(t) &= \zeta\left(x(t), z\lfloor p \rfloor\right) & (5) \end{aligned}$$

where $x(t) \in \mathcal{R}^n$ is the state, $u(t) \in \mathcal{R}^m$ is the control input, $z\lfloor p \rfloor \in \mathcal{Z}$ is the symbolic input, where $\mathcal{Z}$ is isomorphic to a subset of $\mathcal{N}^+$, $y(t) \in \mathcal{R}^p$ is the output, $p(t) \in \mathcal{R}$, $\lfloor p \rfloor \in \mathcal{Z}$ is a scalar given by the largest integer less than or equal to p,

$\lceil p \rceil$ is the smallest integer greater than p, t_p is the time t for which p last passed an integer value (i.e. the last time step of the symbolic dynamics), $v\lfloor p \rfloor \in \mathcal{V} \subset \mathcal{Z}_+$ is a symbolic control input $\alpha : \mathcal{R}^n \times \mathcal{R}^m \times \mathcal{Z} \to \mathcal{R}^n$, $\beta : \mathcal{R}^n \times \mathcal{R}^m \times \mathcal{Z} \to \mathcal{R}$, $\gamma : \mathcal{R}^n \times \mathcal{Z} \times \mathcal{V} \to \mathcal{Z}$, and $\zeta : \mathcal{R}^n \times \mathcal{Z} \to \mathcal{R}^p$.

We shall state some pertinent theoretical results below. Due to space constraints, proofs will be omitted.

2.2 Constructive Existence Result

The existence of a physical model which can implement an operational map is given by the following lemma.

Lemma 1 *A model G with continuous, real-valued input input and outputs $x(t) \in X \subseteq \mathcal{C}(K)$, $y(t) \in Y \subseteq \mathcal{C}(K)$, respectively, consisting of 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*

$$\begin{aligned} y &= F(x(t); \theta(x)) \qquad x(t) = x_0 \; t \leq 0 &(6)\\ \theta(x) &= M_c(x) &(7) \end{aligned}$$

where $t \in \mathcal{R}_+$, and $\theta(x)$ is the m-dimensional parameter vector of F is capable of implementing a general operational map $H(\theta) : \mathfrak{F}_a \to \mathfrak{F}_a$, where $F_a \in \mathfrak{F}_a \subseteq K_o$ and $F_b \in \mathfrak{F}_b \subseteq K_o$.

2.3 Universal Approximation of Operational Maps

If a model can approximate any operational map arbitrarily well, then it can approximate any multiple model arbitrarily well. Such a property may be defined as universal approximation of an operational map. We give some of the main theoretical results below without proof.

Theorem 2 *There exists a parameterized model $G(\theta) : x(t), \hat{y}(t-1) \to \hat{y}(t)$, and a system $H : x(t), y(t-1) \to y(t)$ such that*

$$|y(t) - \hat{y}(t)| < \epsilon \qquad t \in [0, T] \qquad (8)$$

where where $x \in X \subseteq \mathcal{C}(K)$, $\hat{y} \in \hat{Y} \subseteq \mathcal{C}(K)$, $y \in Y \subseteq \mathcal{C}(K)$, $\epsilon > 0$ and T is some finite time.

Theorem 3 *There exists a parameterized model $G(\hat{\theta}) : x(t), \hat{y}(t-1) \to \hat{y}(t)$ given by the interconnection of a universal operator $F(\theta) : X \times \hat{Y} \times V \to \hat{Y}$ and a single-input single-output universal functional $M_v : X \times \hat{Y} \to V$ according to*

$$\begin{aligned} \hat{y}(t) &= F\left(x(t), \hat{y}(t-1), v; \hat{\theta}\right) &(9)\\ v &= M_v(x(t), \hat{y}(t)) &(10) \end{aligned}$$

where $x \in X \subseteq \mathcal{C}(K)$, $v \in V \subseteq \mathcal{C}(K)$, $\hat{y} \in \hat{Y} \subseteq \mathcal{C}(K)$, $\hat{\theta} \in \hat{\Theta} \subseteq \mathcal{R}^m$ and a system $H : x(t), y(t-1) \to y(t)$ which implements an operational map such that

$$|y(t) - \hat{y}(t)| < \epsilon \qquad t \in [0, T] \qquad (11)$$

where $y \in Y \subseteq \mathcal{C}(K)$, $\epsilon > 0$ and T is some finite time.

Hence, these results indicate that the universal approximation result for recurrent networks (Sontag 1992) are applicable, and therefore, there exists a constant recurrent network which is capable of approximating multiple systems $\mathcal{S}_i$ $i = 1, 2, ..., m$. Moreover, it is clear that such approximations may be made individually to an arbitrary degree of accuracy. This follows from consideration of the increased dimensions of approximation space of the network. These theorems give a mechanism by which multiple modelling can take place in neural networks

3 Hybrid Systems and Neural Networks

3.1 Architectures

The results given in section 2 indicate that novel *modular* neural network architectures can be used to construct operational maps. This leads us to obtain the following results concerning the relationship between hybrid systems and operational maps, and hence neural networks and hybrid[1] systems. From (5) and Theorem 3 we obtain the following result.

Theorem 4 *A hybrid system is an operational map.*

Theorem 5 *There exists a parameterized model $G(\hat{\theta}) : x(t) \to \hat{y}(t)$ given by the interconnection of a universal operator $F(\theta) : X \times V \to \hat{Y}$ and a single-input single-output universal functional $M_v : X \to V$ according to*

$$\begin{aligned} \hat{y} &= F\left(x(t), v; \hat{\theta}\right) &(12)\\ v &= M_v(x(t), u) &(13) \end{aligned}$$

where $x \in X \subseteq \mathcal{C}(K)$, $v \in V \subseteq \mathcal{C}(K)$, $u \in U \subseteq \mathcal{C}(K)$ or $u \in U \subseteq \mathcal{R}$, $\hat{y} \in \hat{Y} \subseteq \mathcal{C}(K)$, $\hat{\theta} \in \hat{\Theta} \subseteq \mathcal{R}^m$ and a system $H : x(t) \to y(t)$ which implements an operational map such that

$$|y(t) - \hat{y}(t)| < \epsilon \qquad (14)$$

where $y \in Y \subseteq \mathcal{C}(K)$ is the output of a hybrid model $\overline{\{\alpha, \beta, \gamma, \zeta\}}$, with $\overline{\{\alpha, \zeta\}} : x(t) \to y(t)$ and $\epsilon > 0$.

Corollary 6 *A hybrid model can be approximated by a recurrent neural network.*

The implication of these results, is that it is possible for a neural network to approximate a hybrid system with arbitrary accuracy. This result shows that it is possible to treat hybrid models from a neural network perspective. Intuitively, one may have suspected this to be the case, since it is known that recurrent neural networks can approximate finite state automata[2] (Giles, Miller, Chen, Chen, Sun and Lee 1992), a commonly

[1] Due to the many different definitions for hybrid systems, we do not seek to prove the results for all known classes of hybrid systems, but rather we have chosen to show the results for one relatively general model. From these results it is straightforward to extend the results to other classes of hybrid systems (Branicky et al. 1994).

[2] Note that a recurrent network implementing a finite automata can be used in place of the submodel M_v, however more general models are clearly possible.

used component in hybrid systems, and in general, they can approximate the solutions of various ODEs. The results in this paper make it clear that recurrent networks possess very general, universal approximation capabilities for hybrid systems. The results given here can also be interpreted in terms of Sontag's PL map framework (Sontag 1996) which we do not go into here due to lack of space.

4 Synthesis of Neural Network Hybrid Models

In this section, we give some practical aspects and examples of synthesizing hybrid models based on neural network structures.

There are numerous methods in which models can be derived based on the framework presented in this paper. Here we shall only attempt to give a brief summary of some of those methods and examples of each. Clearly this will not be exhaustive, but representative of some techniques only.

4.1 $M_{\{c,v\}}$ Subsection

The $M_{\{c,v\}}$ model subsection implements the discrete state dynamics. For convenience, we will refer to this subsection as M_c. The basic idea is that M_c implements the symbolic level processing in a neural network structure. For example, this may be a decision tree, a DFA (Giles et al. 1992), or some more complex computational device (Siegelmann and Sontag 1991).

For convenience, it is useful to break M_c into three further modules: M_{c1}, M_{c2} and M_{c3}. Depending on the quantization or classification performed by M_{c1}, the symbolic processing in M_{c2}, and the decoding performed in M_{c3}, F will switch between different models.

4.2 F Subsection

The F model subsection is a real-valued continuous state model. One of the simplest examples is the linear filter given by

$$y(t) \quad = \quad H(\theta; z)x(t) \tag{15}$$

Clearly, F can also be implemented as a more complex nonlinear operator. Using $\theta = M_c x(t)$ allows M_c to switch the base model F into a number of discrete states. As indicated by Theorems 3 and 5, another approach is to use M_c to drive the input of F giving a similar result. These methods are termed the *parameter replacement* and *augmented input* form of coupling M_c and F.

4.3 Augmented Input Networks: Hybrid 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.

The fundamental approximation capabilities of neural network models come about due to the ability to sum basis functions comprised of sigmoids, which are local in the input space. That is, the flatness of the sigmoids' tail implies that we can add as many sigmoids as required to form an approximation and the effect can be made local. Accordingly, we may deduce the following constructive method.

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

$$G(x, y, v) = m_v(x, y, v), \ v = 1, ...p \tag{16}$$

defined by

$$m_v(x, y) = \sum_{i=1}^{N} c_{vi} g\left(\sum_{j=1}^{m} \xi_{vij} x, y + \phi_{vi}\right) \tag{17}$$

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

The implication of this is that by appropriate biases offsets in the different groups of units, it is possible to obtain universal approximators in each subgroup of units over a finite range. This finite range is not a problem however, since we can always make r arbitrarily large in order to keep the individual mappings separated. Each of the subgroups may define unique mappings, activated by the appropriate input v.

We term this constructive method of synthesizing multiple models *bias shifting*. This method can be used as a means of switching between different models.

5 Examples

5.1 Bias-shifted MLP

As a simple example of the bias shifting method, consider a multiple model feedforward network[3]. The network is a two-weight layer network in which there are two groups of hidden units. The mappings pertaining to each group of weights are obtained simply by providing different inputs to the network are shown in Fig. 1. In other words, the network 'contains' two quite different function approximations. According to the value of the input signal, it is possible to select which of the two mappings will be used.

5.2 A Recurrent Network Hybrid Model

In this example, we propose a hybrid model based on the recurrent network. An interesting aspect of this model is that it is structurally similar to the usual pole-zero linear model, but the actual pole-zero positions are determined by nonlinear functions. The model is described as follows.

$$\begin{aligned} y(t) &= F(x, t, v) && (18) \\ &= \frac{q^{n-m} \prod_{i=1}^{m_1} (q - M_{ci}(u_{\beta ri}))}{\prod_{i=1}^{n_1} (q - M_{di}(u_i))} x(t) && (19) \end{aligned}$$

where $\{M_{\{c,d\}i}\}$ are the element-wise characteristic functions of the vector function equation $\theta = M_c(u)$, corresponding to the previously developed notation. Since the poles and zeros

[3] As a means of demonstrating the technique, we use a feedforward network only.

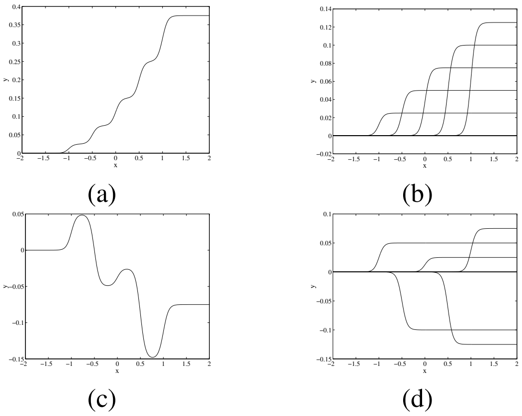

Figure 1: An example showing the use of bias-shifting to select different function approximations. The input-output mapping and hiddent unit outputs are shown for each case. Note that these mappings coexist within the one network, but one is "selected" for the inputs at any given time, due to the function M_c, which outputs the extra input bias v.

in (19) 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 $\{M_c\}$ and ancillary inputs$\{u\}$ provide a wide scope for introducing a variety of models. For example, M_c may be the discrete state automata as indicated earlier, implementing some linear or nonlinear function of time t, inputs $\{x\}$, and outputs $\{y\}$. Indeed, other auxiliary variables can also be naturally introduced at this point.

If $M_{\{c,d\}i}(u_i) = \theta_i$, then a linear model will result. However, by the appropriate choice of $\{M_{\{c,d\}i}\}$, (19) is capable of representing a range of time-varying linear models. Moreover, such models can generalize existing structures in an intelligent manner. Similar models have been recently considered in the literature (Narendra et al. 1995). It is interesting to note however, that while this model conforms to the general hybrid system framework, it can be viewed as an RNN with *nonlinear synapses*.

6 Conclusions

In this paper, we have given a new framework for understanding dynamic neural network models. Operational maps define mappings from one operator space to another operator space. Theorems showing the existence of neural network universal operational map approximations have been given. The framework of operational maps has been used to prove that a general class of hybrid models can be approximated arbitrarily closely by a recurrent neural network. We have given an indication of how such models may be synthesized. While we have not considered it in this paper, it would be interesting to devise a method, if one exists, which would allow the extraction of the subsystems described in this paper from a trained RNN structure. If this can be done, it may provide some useful insights into the function of trained recurrent networks, particularly in terms of nonlinear system identification models.

References

Branicky, M. S., Borkar, V. S. and Mitter, S. K. (1994). 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.

Brockett, R. (1993). Hybrid models for motion control systems, *Perspectives in Control*, Boston: Birkhauser, pp. 29–54.

Feldkamp, L., Puskorius, G. and Moore, P. (1996). Adaptation from fixed weight dynamic networks, *Proceedings of IEEE International Conference on Neural Networks, Washington (ICNN '96)*, pp. 155–160.

Giles, C., Miller, C., Chen, D., Chen, H., Sun, G. and Lee, Y. (1992). Learning and extracting finite state automata with second-order recurrent neural networks, *Neural Computation* **4**(3): 393–405.

Jacobs, R. A., Jordan, M. I., Nowlan, S. J. and Hinton, G. E. (1991). Adaptive mixtures of local experts, *Neural Computation* **3**(1): 79–87.

Lainiotis, D. (1976a). Partitioning: A unifying framework for adaptive systems I. estimation, *Proc. IEEE* **64**: 1126–1143.

Lainiotis, D. (1976b). Partitioning: A unifying framework for adaptive systems II. control, *Proc. IEEE* **64**: 1182–1197.

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

Nerode, A. and Kohn, W. (1993). Multiple agent hybrid control architecture, *Technical Report TR 93-12*, Mathematical Sciences Institute, Cornell University, Ithaca, New York 14850.

Ronco, E. and Gawthrop, P. (1995). Modular neural networks: a state of the art, *Technical Report CSC-95026*, Center for System and Control, University of Glasgow.

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

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

Siegelmann, H. T. and Sontag, E. D. (1991). Turing computability with neural nets, *Applied Mathematics Letters* **4**(6): 77–80.

Sontag, E. (1992). Neural nets as systems models and controllers, *Proc. Seventh Yale Workshop on Adaptive and Learning Systems*, Yale University, pp. 73–79.

Sontag, E. (1996). Interconnected automata and linear systems: A theoretical framework in discrete-time, *in* R. Alur, T. Henzinger and E. Sontag (eds), *Hybrid Systems III: Verification and Control*, Springer, Birkh"auser Verlag, Basel, NY, pp. 436–448.