PERGAMON

Neural Networks

Neural Networks 14 (2001) 933–939

www.elsevier.com/locate/neunet

2001 Special issue

A spiking neural network architecture for nonlinear function approximation

Nicolangelo Iannella*, Andrew D. Back

Brain Science Institute, RIKEN, 2-1 Hirosawa, Wako-shi, Saitama 351-0198, Japan

Received 13 October 2000; accepted 19 April 2001

Abstract

Multilayer perceptrons have received much attention in recent years due to their universal approximation capabilities. Normally, such models use real valued continuous signals, although they are loosely based on biological neuronal networks that encode signals using spike trains. Spiking neural networks are of interest both from a biological point of view and in terms of a method of robust signaling in particularly noisy or difficult environments. It is important to consider networks based on spike trains. A basic question that needs to be considered however, is what type of architecture can be used to provide universal function approximation capabilities in spiking networks? In this paper, we propose a spiking neural network architecture using both integrate-and-fire units as well as delays, that is capable of approximating a real valued function mapping to within a specified degree of accuracy. © 2001 Elsevier Science Ltd. All rights reserved.

Keywords: Spiking neurons; Integrate-and-fire units; Delays; Neural architecture; Nonlinear function approximation

1. Introduction

Multilayer perceptrons (MLPs) have received significant attention in recent years. The main reason for this is the fact that MLPs are capable of approximating any continuous $\mathbf{F} \in \mathscr{C}(\mathscr{R}^m, \mathscr{R}^n)$ arbitrarily well by means of spatially-local basis functions. Note that $\mathscr{C}(\mathscr{R}^m, \mathscr{R}^n)$ denotes the space of continuous functions or mappings $\mathbf{F} : \mathscr{R}^m \to \mathscr{R}^n$. This property is termed *universal approximation* (Cybenko, 1989; Funahashi, 1989; Hornik, Stinchcombe & White, 1989). This means that given a function $\mathbf{F}(\mathbf{x})$, a model can be constructed as

$$G(\mathbf{x}) = \mathbf{B}\sigma(\mathbf{A}\mathbf{x} + \varphi), \tag{1}$$

where $\mathbf{x} \in \mathscr{R}^m$ is the input, $\mathbf{A} \in \mathscr{R}^n \times \mathscr{R}^m$ are the input layer weights, $\mathbf{B} \in \mathscr{R}^n$ are the output layer weights, $\varphi \in \mathscr{R}^n$ are the bias inputs and σ are some appropriately chosen basis functions, that can be shaped, offset and scaled,[1] to approximate $\mathbf{F}(\mathbf{x})$ to an arbitrary degree of accuracy.

The results for universal approximations are normally made using real valued inputs, outputs and weights. However, in real biological systems, signals are encoded using spike trains, not real values. This raises the question of how function approximation (whether 'universal' or not) can be achieved in spiking neural networks (SNNs). This issue is of interest for the development of spiking neural network models for practical applications as well as obtaining an understanding of possible biological computational mechanisms.

Neural coding refers to the scheme by which we attach meaning to the spiking signals. It is, therefore, useful to examine the various methods of neural coding that have been developed. Neural coding has been the subject of much debate and various spike train information encoding methods have been proposed in the literature (Bialek, Rieke, de Ruyter van Steveninck & Warland, 1991, 1997; de Ruyter van Steveninck & Bialek, 1995; DeWeese, 1996). Once we have explored the various methods of neural coding schemes, it is then possible to clearly see which of the various methods are best suited to our goal of function approximation.

In this paper, we propose a spiking network model based on a particular coding method. In Section 2 we review some of the more well known methods of neural coding. This review does not seek to be comprehensive, but sets the foundation for the remainder of this paper. In Section 3, we describe a network architecture and indicate how it is capable of approximating arbitrary function mappings. In Section 4, we describe an integrate-and-fire network architecture, based upon our general architecture. Section 5 gives some experimental results that show the performance of the

* Corresponding author.
E-mail address: angelo@postman.riken.go.jp (N. Iannella).

[1] These operations correspond to the transformations made by the hidden layer weights, bias weights and output layer weights, respectively.

network. In Section 6 we qualitatively compare the proposed network with the Maass et al. model. Conclusions are given in Section 7.

2. A brief overview of neural coding schemes

The earliest neural coding method proposed is the mean or *temporally averaged firing rate* for encoding information. More recently, there has been growing recognition that the traditional view of mean firing encoding is often inadequate. Experiments from the fly visual system (Bialek et al., 1991, 1997) and studies of the middle temporal (MT) area of the monkey have indicated that the precise timing of spikes can be used to encode information, such a scheme is called *temporal coding*. *Population coding* is another scheme in which information is encoded in the activity of a given population of neurons firing within a small temporal window. Within this framework a notion of *rate coding* can be extracted as a (weighted) instantaneous average measure of firing activity in a given neuron population.

Function approximation methods have been demonstrated for several encoding schemes. Sanger (1998) demonstrated that function approximation can be performed by a population of spiking neurons based on a **rate coding** scheme, that assumes that information is encoded by the instantaneous average firing probability, but avoids the need to compute this rate explicitly. He showed that a network of spiking neurons can be constructed such that the output neurons' firing probability is a desired smooth function of the input neurons' firing probability. This result suggested a duality relationship between the firing nature of spiking cells and the deterministic smooth behavior of their tuning functions. Furthermore, investigation of both supervised and unsupervised learning resulted in adaptive algorithms that were similar to those rules used in standard artificial neural network theory. This allows classical neural network approximation and learning methods, based on continuous variables, to be implemented within populations of spiking neurons without the need to calculate the intermediate cell firing rates.

Maass et al. (Maass, 1997; Ruf, 1997) gave the first illustration that using a simple mathematical model of the biological neuron, a spiking neuron can compute a linear weighted sum in temporal coding. The proof took advantage of the mechanism that both excitatory (EPSPs) and inhibitory (IPSPs) post synaptic potentials can shift the firing time of a spiking neuron and that the initial phases of these potentials are approximately linear. Furthermore, it was demonstrated that a spiking neuron, that receives EPSP and IPSP inputs from a population of presynaptic excitatory and inhibitory neurons can emulate a sigmoidal neuron with a piecewise linear gain function in temporal coding. This led to a theorem that any feedforward or recurrent analog neural network, for example a multilayered perceptron, consisting of sigmoidal neurons that employ a piecewise linear gain function, can be simulated arbitrarily closely by a network of spiking neurons with analog inputs and output encoded by temporal delays of spikes. This holds even in the presence of noise. Maass (1997) also illustrated that from computational complexity theory, not only could SNNs emulate sigmoidal networks in principle, but also that SNNs were computationally as powerful or more powerful than sigmoidal neural networks. Thus, Maass (1997) showed that spiking neural networks that employ temporal coding are theoretically capable of universal function approximation.

In this paper, our interest is not driven from a theoretical point of view in the usual sense of trying to prove universal approximation, although this is a very important network property. Rather, our interest is in the question of *how* arbitrary functions can be approximated in a simple and practical manner using spiking networks. In this paper, we propose a method of constructing temporally coded, spiking neural networks that can be used for function approximation in a robust manner, to be defined precisely later in the paper.

3. A spiking network architecture

3.1. Neural encoding method

In this section, we derive a general framework for a class of spiking neural network models capable of forming arbitrary nonlinear function mappings (not necessarily universal) between an input and an output. To achieve this, we firstly define the neural encoding method used and then determine a neural architecture capable of forming the function mapping.

Biological neurons receive many spikes that form a representation of the information received by the neuron. In our work, we have chosen to use a temporal encoding scheme that is defined such that the *inter spike interval* (ISI), the temporal distance between two successive spikes, represents a real-valued quantity. The relationship between real valued inputs x_i and the ISI is typically defined as follows,

$$x_i = T_i(2) - T_i(1) = \Delta_i,$$

where x_i is the effective real value of the i-th input, $T_i(1)$ and $T_i(2)$ correspond to the arrival times of the first and second spikes, respectively, and Δ_i is used to denote the ISI.

3.2. A network architecture

In this section, we describe the characteristics of a proposed network architecture, capable of approximating any nonlinear map. Our interest is to seek a network that could be used to form the arbitrary mapping $\mathbf{F}: R^m \rightarrow R^n$. In addition, obtaining such a network raises the issue of parameterization. How can a spiking network be trained to approximate some arbitrary nonlinear map $\mathbf{F}$, given some training set of (multidimensional) data $\mathscr{S}:\{\mathbf{U_i}, \mathbf{V_i}, \mathbf{i} = 1, \ldots, N\}$, where $\mathbf{U_a} = [u_{a1}\ u_{a2}\ \cdots\ u_{an}]$ and $\mathbf{V_a} = [v_{a1}\ v_{a2}\ \cdots\ v_{am}]$, that represent

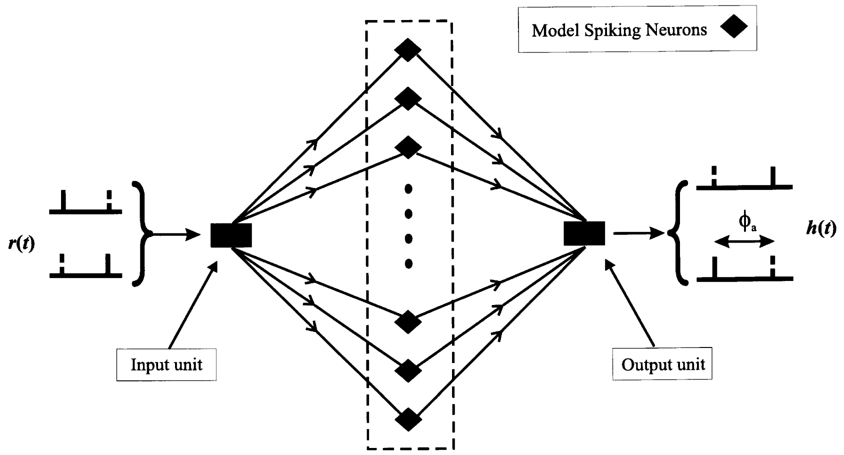

Fig. 1. The proposed network architecture capable of forming arbitrary nonlinear maps in a piecewise fashion. Both the input and output units, represented by rectangles, simply transmit input and output spikes, to and from the layer of spiking neurons. Our model spiking neurons are represented by diamonds.

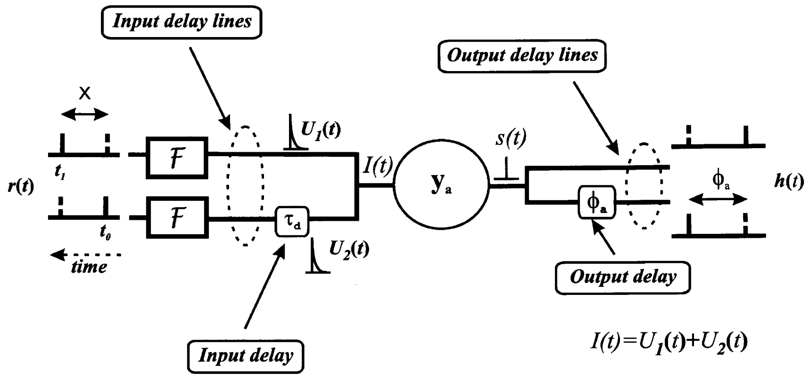

Fig. 2. The proposed model spiking unit consists of two input lines, an activation unit and two output lines. Its basic operation, when it fires, is to transform two incoming spikes to another spike pair of spikes with some desired ISI of ϕ.

observations of the mapping **F**? For reduced complexity, we will consider the case of one dimensional nonlinear maps.

The network architecture we propose is based on the idea of massive parallelism observed in biological nervous systems. In simple terms, the methodology we propose is to build a large network that temporally and spatially separates incoming spikes into different regions of the network, that in turn generate corresponding new spike pairs. These new output spikes may have any desired ISI. Hence, the network can form a piecewise mapping that can approximate any real valued function $f: R \rightarrow R$.

For this case, the network consists of an input unit, an output unit and a single layer of spiking neurons, as depicted in Fig. 1. Our proposed spiking neuron model (represented by a diamond in Fig. 1) is composed of three 'sections', two input lines one of which possesses a delay τ_d, an activation unit capable of firing a spike and two output lines (one possessing delay ϕ) as depicted in Fig. 2. Our model is based on the fact that a biological neuron is composed of three basic parts: a dendrite, a soma and an axonal tree. Traditionally, the neuron in a neural network was effectively treated as a 'point particle' modeled by the dynamics of the soma, and the connections were treated as transmission lines. In the model we propose, the effect of a neuron firing an action potential is considered in terms of the multiple resultant spikes that emanate via the axonal tree.

We consider that input spikes enter our model neuron via the two input lines. Mathematically, our proposed model is defined by the following set of equations. Consider a neural unit a with activation $y_a(t)$ at time t.[2] This activation occurs as a result of two inputs $U_\alpha(t)$ and $U_\beta(t-\tau_d)$ that are delayed with respect to each other by a time τ_d, and passed through some operator $\mathscr{G}$, where

$$y_a(t) = \mathscr{G}[U_\alpha(t) + U_\beta(t-\tau_d)]. \quad (2)$$

The total input entering the activation unit is designated by $\mathscr{I}(t)$ where

[2] In a biologically-oriented model, y_a represents the total somatic membrane potential, given by the summation of all postsynaptic potentials. In biological neurons, this summation is spatio-temporally nonlinear. Currently, we only consider linear summation, since the addition of nonlinearities would simply increase the difficulty in achieving function mappings.

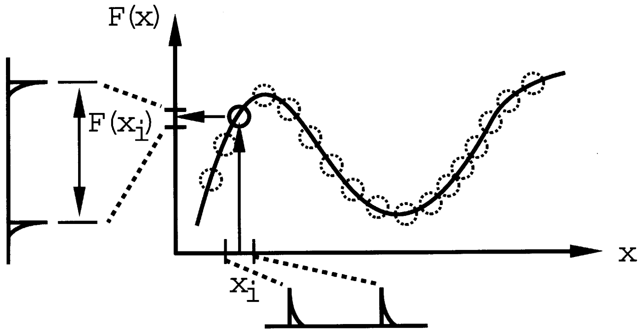

Fig. 3. The input space is subdivided into regions, with signals corresponding to each region being passed into a particular processing unit.

$$\mathscr{I}(t) = U_\alpha(t) + U_\beta(t - \tau_d). \tag{3}$$

Let the inputs U_α, U_β be obtained from the input lines as filtered versions of input spike trains $r_\alpha(t)$ and $r_\beta(t)$, respectively,[3] where

$$U_\alpha(t) = \mathscr{F} r_\alpha(t) \tag{4}$$

$$U_\beta(t - \tau_d) = \mathscr{F} r_\beta(t - \tau_d) \tag{5}$$

where $\mathscr{F}$ is a low-pass filter. If the activation y_a exceeds a threshold level θ at time t_f, the activation unit 'fires' to produce an output spike $s(t)$,

$$s(t) = \begin{cases} \delta(t - t_f) & \text{if } y_a(t) \geq \theta \\ 0 & \text{otherwise} \end{cases}. \tag{6}$$

The spike $s(t)$ then exits our activation unit via the two output lines generating a new spike pair having an ISI of ϕ seconds and denoted by

$$\mathbf{h}(t) = [s(t), s(t - \phi)].$$

Conceptually, the model is easy to understand. The equations above describe one input and one output, so to construct a network capable of function approximation, it is necessary to use many such units, each with their own associated τ_d and ϕ values. Each processing unit corresponds to each segment[4] as depicted in Fig. 3.

The threshold value θ requires a method of parameterization. This could be set simply at $\theta = 2L$, where

$$2L = \max(y_a) = \max(\mathscr{G}[U_\alpha(T_1) + U_\beta(T_0 - \tau_d)]),$$

corresponding to the maximum of the activation y_a, provided that $\tau_d = T_0 - T_1$, for a pair of spikes. However, this method would not be robust enough in practical circumstances and would easily fail if the spikes were anything less than exactly coincident. A solution is, therefore, to set θ to some lower value, but not too low so that firing occurs for any spikes. We consider below such a method of setting θ to allow some latitude in the input ISI for each unit.

3.3. Threshold calculation

Here, we show that it is possible to set the firing threshold for the unit so that there is an allowable tolerance in the timing between the input spikes, below which the unit will still fire. Thus, perfect coincidence is not required, and in this sense, the system is robust enough to variations in the ISI (that could be attributable to any number of physical factors and is a desirable feature to have).

Let us look at the simplest case where the activation is given by

$$y_a = U_\alpha(T_1) + U_\beta(T_0 - \tau_d)$$

and the inputs U_α, U_β are expressed by

$$U_\Gamma(t) = L\exp(-(t - t_i)/\tau)H(t - t_i),$$

for $\Gamma = \alpha, \beta$ and $t_i = T_1, T_0 - \tau_d$, where L is some arbitrary positive constant and $H(t)$ is the Heaviside step function. We specify a tolerance value ρ and derive θ such that a given unit produces an output spike pair if

$$|(T_1 - T_0) - \tau_d| \leq \rho \tag{7}$$

For the purpose of this paper, we specify the time decay characteristics τ of the input spikes. Choosing a threshold value of $1.5L$ gives

$$1.5L = L + L e^{-\rho/\tau} \tag{8}$$

and hence

$$\tau = \frac{\rho}{\ln(2)}, \tag{9}$$

thus ensuring that the neuron will fire when the time difference between spikes is ρ or less. Hence, the proposed network model is capable of robust performance by allowing for some tolerance in the range of input ISIs that will evoke an output. Accepting that biological systems possess a degree of robustness, then it is also reasonable to consider

[3] This corresponds well with biological spiking neurons where it is well known that the dendritic and axonal branches have transmission delays that primarily depend on their respective length and diameter. Furthermore, each spike arriving at different points on the dendritic tree evokes a current, that is transmitted up the dendrite to the soma and whose maximum (or minimum) is delayed by some amount. For simplicity, it is assumed that spikes occur at the same loci on the dendritic tree.

[4] This need not be the case in practice. Recently, there has been some suggestion that biological networks may actual perform a type of 'multiplexing'.

mechanisms of introducing these characteristics in a model as well.

4. An integrate-and-fire network

We now look at a simple specialization of the model described in Subsection 3.2. In particular, it is interesting to consider networks whose unit activation y_a is modeled by the leaky integrate-and-fire (IAF) model. In this case, we identify y_a as the membrane potential.

Let the input spike signal pair $\mathbf{r}(t)$ be expressed as follows

$$\mathbf{r}(t) = \begin{pmatrix} r_\alpha(t) \\ r_\beta(t) \end{pmatrix} = \begin{pmatrix} \delta(t - t^a_\alpha) \\ \delta(t - t^b_\beta) \end{pmatrix},$$

where $t^b_{\alpha,}\ t^a_\beta$ are the spike arrival times at the different input lines, where $t^b_\alpha < t^a_\beta$. Let us express U_α and U_β as

$$U_\alpha(t) = \mathscr{F} r_\alpha(t) = \exp\left(-\frac{t - t_1}{\tau_s}\right)\mathcal{H}(t - t_1)$$

and

$$U_\beta(t - \tau_d) = \mathscr{F} r_\beta(t - \tau_d) = \exp\left(-\frac{t - t_0 - \tau_d}{\tau_s}\right)\mathcal{H}(t - t_0 - \tau_d),$$

respectively. The total input $\mathscr{I}$(t) to our activation unit is

$$\mathscr{I}(t) = U_\alpha(t) + U_\beta(t - \tau_d) = \exp\left(-\frac{t - t_1}{\tau_s}\right)\mathcal{H}(t - t_1) + \exp\left(-\frac{t - t_0 - \tau_d}{\tau_s}\right)\mathcal{H}(t - t_0 - \tau_d),$$

where t_i represent spike arrival times, τ_s is the (synaptic) time constant, τ_d is the (dendritic) delay and $\mathcal{H}(t)$ is the Heaviside step function. Our activation y_a is described by the leaky IAF model,

$$\frac{d}{dt} y_a(t) = -\frac{1}{\tau_m} y_a(t) + \mathscr{I}(t), \tag{10}$$

where $y_a(t)$ represents the membrane potential, τ_m is the membrane time constant. When the membrane potential reaches threshold θ, a spike is produced and transmitted via the output lines. The output spikes $h(t, \phi)$ have an ISI of ϕ.

In this case, to ensure that our model neuron fires with a prespecified tolerance ρ, the threshold parameter θ is determined as follows. We select ρ and τ_d as before, and initially we set $\tau_s = \rho/\ln(2)$. Noting that

$$\rho = |\tau_d - \Delta|, \tag{11}$$

and using $t_1 = t_0 + \tau_d$, an analytical expression for the total membrane potential can be easily obtained by solving Eq. (10). Thus, the required threshold is obtained as a function of time constants and the tolerance range, and is given by

$$\theta(\rho) = \frac{1}{(\tau_m/\tau_s - 1)}\left[X_r^{\tau_s/(\tau_m - \tau_s)}(e^{\rho/\tau_m} + 1) - X_r^{\tau_s/(\tau_m - \tau_s)}(e^{\rho/\tau_s} + 1)\right],$$

$$X_r = \frac{\tau_s}{\tau_m}\left(\frac{1 + e^{\rho/\tau_m}}{1 + e^{\rho/\tau_s}}\right). \tag{12}$$

This method of threshold determination is used in the next section to show the performance of the network architecture.

5. Experimental results

5.1. Function approximation

Here, we present some simulation results to show the behavior of the proposed model. In the experimental results given here, we approximate three nonlinear functions:

$$g_1(x) = 1 + \sin(4\pi x)$$

$$g_2(x) = (x - 1.6)^3 - x + 4$$

$$g_3(x) = (2x - 1.6)^3 - 2x + 4$$

Fig. 4 shows the approximation results for segment sizes is g_1, g_2: $\Delta x = 0.1$ units, and g_3: $\Delta x = 0.05$ units. It can be observed that the model approximates the functions as required. In order to show the results more clearly, a low resolution is used with large segment sizes. However, by increasing the resolution with a larger number of units and smaller segment sizes, a more accurate approximation would be obtained.

5.2. Gradient descent learning

In this section, we demonstrate that the proposed architecture does not require a complex learning algorithm to learn an arbitrary mapping. In fact, we show that it is possible to utilize a simple gradient descent learning algorithm to train the network.

A gradient descent learning algorithm can be derived as follows to determine the value for ϕ_a to approximate some desired function $F(x_i)$. For each ϕ_a, we have simply

$$\Delta\phi_a = \gamma(F(x_i) - \phi_a)$$

$$\phi_a = \phi_a + \Delta\phi_a.$$

The delay ϕ_a is updated for the unit that has fired.

An example of one simple trial is given below to demonstrate the effectiveness of gradient descent learning. Values for ϕ_a were initially selected at random between [0, 3] and $\gamma = 0.025$. Gradient descent training was applied to the network until an error of 0.01 was achieved for all the ϕ_as.

In Fig. 5, gradient descent learning is applied to the function $g_3(x) = (2x - 1.6)^3 - 2x + 4$. Fig. 5a shows the initial

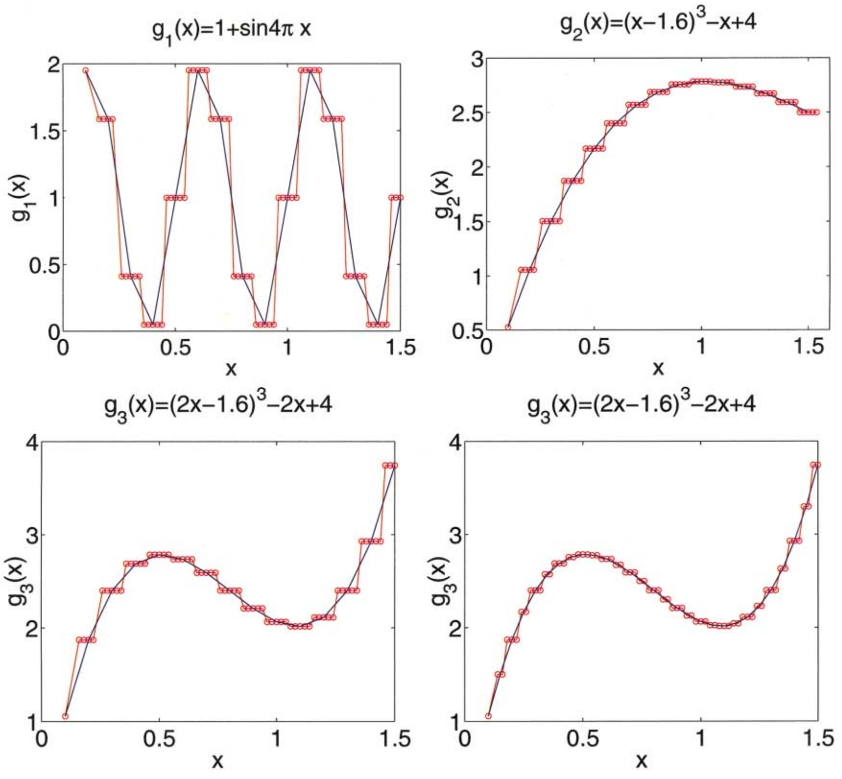

Fig. 4. Experimental results showing the approximation of nonlinear functions by the proposed network. The functions approximated are $g_1(x) = 1 + \sin(4\pi x)$, $g_2(x) = (x - 1.6)^3 = x + 4$, $g_3(x) = (2x - 1.6)^3 - 2x + 4$. The lower two plots demonstrate that by increasing the number of units and decreasing the segment size, a better approximation to the function $g_3(x)$ is obtained.

random selection of ϕ_as and the final learned values within an error of 0.01 and Fig. 5b shows the corresponding errors after training between the trained values of ϕ_a and the desired output $F(x_i)$.

6. A qualitative comparison of the proposed model with the Maass et al. network model

It is useful to compare the network we have proposed with the well established Maass (1997) network model that also uses a temporal coding scheme.

The Maass construction provides a possible explanation of the observed speed of fast information processing in the cortex. It was also shown that spiking neurons can compute linear functions in temporal coding, provided that the spiking neuron fires only once within a small (fixed) time interval and that the initial phases of both excitatory and inhibitory postsynaptic potentials are well described by linear functions. Using this approach, Maass et al. also

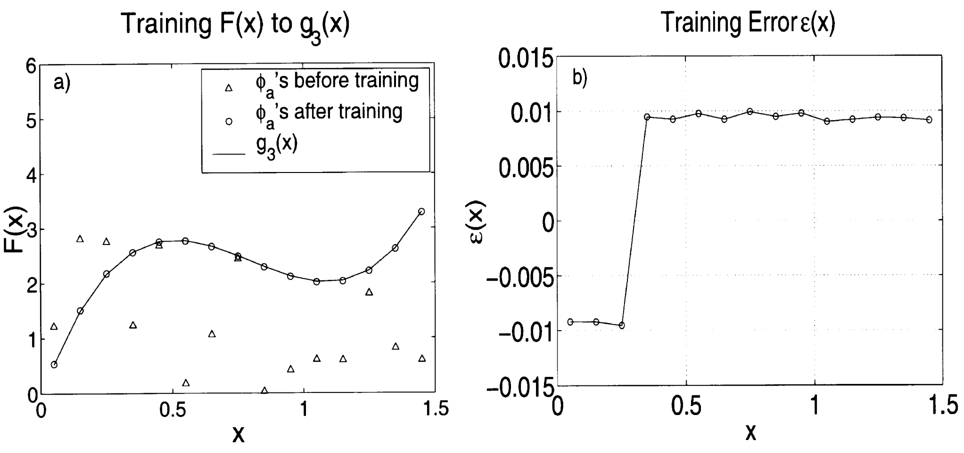

Fig. 5. Application of gradient descent learning to the proposed network: (a) initial selection of ϕ_as, the final trained ϕ_as and the desired function; (b) error between the values of the trained ϕ_as and the desired function.

demonstrated that upon reception of the correct amounts of inhibitory and excitatory inputs at specific times, a spiking neuron can in fact behave as a sigmoidal unit with a piecewise linear activation function. On this basis, it was then shown that a spiking network can achieve universal function approximation.

The SNN architecture we propose is able to temporally and spatially separate incoming spikes into different regions of the network, that in turn generate corresponding new spike pairs. These output spikes can have any desired ISI. Our basic processing unit does not require the condition that it must fire only once within a given small time interval. In addition, we allow some tolerance in the range of input ISIs that do evoke an output, thereby providing a network with robust properties in terms of spike timing.

In terms of function approximation characteristics, we suggest that it may not be necessary to simulate a sigmoidal neural network to achieve useful function approximation capabilities. As shown above, it is possible to employ a relatively simple architecture to model arbitrary functions.

Two drawbacks are evident with the proposed model. Firstly, the approximation of arbitrary nonlinear maps is piecewise and not continuous. This is an area to be examined in future work. Secondly, like most other models, the nonlinearity of the dendrite was not considered.

7. Conclusion

In this paper, we have considered the issue of function approximation in spiking networks. Such networks, although widely considered in the biological literature, are not as well represented in the signal processing and control community. However, for some practical applications spiking networks have particular merit.

From a bio-computational point of view, it is of particular interest to discover mechanisms of how spiking neural networks can approximate arbitrary nonlinear functions. Answers to this question may have some important implications for neurological computation.

We have proposed a simple spiking neural network architecture that has ability to transform incoming pairs of spikes to outgoing pairs with any desired ISI. Our architecture achieves this by spatially and temporally separating incoming spikes into different regions of the network, that in turn generate corresponding new spike pairs with any desired ISI. This is the underlying mechanism that allows our spiking neural network to approximate arbitrary nonlinear network architecture that is capable of approximating function mappings.

Acknowledgements

The authors wish to thank the support of RIKEN, Brain Science Institute and comments by all the referees.

References

Bialek, W., Rieke, F., de Ruyter van Steveninck, R., & Warland, D. (1991). Reading a neural code. *Science*, *252*, 1854–1857.

Bialek, W., Rieke, F., de Ruyter van Steveninck, R., & Warland, D. (1997). *Spikes: exploring the neural code*, Cambridge: MIT Press.

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, *2* (4), 303–314.

de Ruyter van Steveninck, R., & Bialek, W. (1995). Reliability and statistical efficiency of a blowfly movement-sensitive neuron. *Phil. Trans. R. Soc. Lond. Ser. B*, *348*, 321–340.

DeWeese, M. (1996). Optimization principles for the neural code. In D. S. Touretsky, M. C. Mozer & M. E. Hasselmo, *Advances in neural information processing systems* (pp. 281–287), vol. 8. Cambridge: MIT Press.

Funahashi, K. (1989). On the approximate realization of continuous mappings by neural networks. *Neural Networks*, *2* (3), 182–192.

Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, *2* (5), 359–366.

Maass, W. (1997). Fast sigmoidal networks via spiking neurons. *Neural Computation*, *9*, 279–304.

Ruf, B. (1997). *Computing functions with spiking neurons in temporal coding* In J. Mira, R. Moreno-Diaz & J. Cabestony (eds), Biological and artificial computation: from neuroscience to technology, volume 1240 of Lecture Notes on Computer Sciences, pp. 265–272. Berlin: Springer.

Sanger, T. D. (1998). Probability density methods for smooth function approximation and learning in populations of tuned spiking neurons. *Neural Computation*, *10*, 1567–1586.