J. Indian Inst. Sci., Sept.–Oct. 2006, **86**, 465–479

A novel multiplicative neural network architecture motivated by spiking neuron model

DEEPAK MISHRA, ABHISHEK YADAV AND PREM K. KALRA
Department of Electrical Engineering, Indian Institute of Technology Kanpur, Kanpur 208 016, India
email: dkmishra@iitk.ac.in.

Received on August 11, 2005; Revised on September 6, 2006.

Abstract

In this paper, learning algorithm for a multiplicative neural network motivated by spiking neuron model (MSN) is proposed and tested for various applications where a multilayer perceptron (MLP) neural network is conventionally used. It is observed that the inclusion of a few more biological phenomena in the formulation of artificial neural network models make them more prevailing. Several benchmark and real-life problems of classification and function-approximation are illustrated.

Keywords: Neuron models, artificial neural network, spiking neuron, multilayer perceptron.

1. Introduction

Researchers have proposed several neuron models for artificial neural networks. Although these models are primarily inspired from biological neuron, there is still a gap between philosophies used in neuron models for neuroscience studies and those used for artificial neural networks (ANN). Some of these models exhibit a close correspondence with their biological counterparts while others do not. Freeman [1] has pointed out that while brains and neural networks share certain structural features, such as massive parallelism, biological networks solve complex problems easily and creatively, but existing neural networks do not. He discussed the issues related to the similarities and dissimilarities between biological and artificial neural systems of the present day. The main focus in the development of a neuron model for artificial neural networks is not only its ability to represent biological activities with its maximum intricacy, but also some mathematical properties, e.g. its capability as a universal function approximator. However, it can be advantageous for artificial neural networks if we can bridge the gap between biology and mathematics by investigating the learning capabilities of biological neuron models for the applications of classification, time-series prediction, function approximation, etc.

The first artificial neuron model was proposed by McCulloch and Pitts [2] in 1943. They developed this model based on the fact that the output of neuron is 1 if the weighted sum of its inputs is greater than a threshold value, and 0, otherwise. In 1949, Hebb [3] proposed a learning rule that became initiative for ANNs. He postulated that the brain learns by changing its connectivity patterns. Widrow and Hoff [4] in 1960 presented the most analyzed and

*Author for correspondence.

most applied learning rule known as least mean square rule. Later in 1985, Widrow and Sterns [5] found that this rule converges in the mean square to the solution that corresponds to least mean square output error if all input patterns are of the same length. A single neuron of the above and many other neuron types proposed by several scientists and researchers are capable of linear classification [6]. Yadav *et al.* [7] incorporated various aggregation functions to model the nonlinear input–output relationships. In 2004, Mishra *et al.* [8] investigated the chaotic behavior in neural networks that represent biological activities in terms of firing rate. Scholles *et al.* [9] discussed biologically inspired artificial neurons and Feng and Li [10] introduced neuronal models with current inputs. Training the integrate-and-fire model with the Informax principle was discussed by Feng's groups [11, 12].

Recently, spiking neural networks have been the subject of significant research reflecting the view that spikes have a key role in biological information processing [13, 14]. New advances in neurophysiology have found that the difference in firing times could convey information about the input stimuli and that the relative order of firing times could be used as an alternative to rate coding [15, 16]. The first supervised training for this new computational paradigm was suggested by Bohte *et al*. [17]. However, in this model, a large number of parameters are to be adjusted. In our experiments, we found that the performance of this network is very much dependent on the initial values of these parameters. Kalra and colleagues [18, 19] used a single neuron for classification and function approximation. This model is inspired from the *f*/*I* characteristics of integrate-and-fire neuron model. His group also presents a comparison between the performances of multilayer perceptron and single multiplicative spiking neuron-based artificial neuron [20]. It has been found that for many benchmark problems a single multiplicative spiking neuron and single integrate-and-fire type neuron model-based learning is sufficient.

To solve difficult problems, a neural network model for function approximation and classification is proposed and discussed in the present work. The functioning of the proposed model is motivated by the activity of spiking neuron model. The proposed model considers a modified aggregation methodology at network units. This modification accounts for nonlinear aggregation operation at dendrites. Moreover, probability of spike is used as an output instead of time of spike. We found that with such modifications, the learning performance is drastically improved. These modifications provide an opportunity to add more biological features in forming an artificial neural network for solving problems like function approximation and classification.

The rest of the paper is organised as follows. In Section 2, a brief discussion of the spiking neuron model is presented. Inspired from the relationship between timings of incoming spikes and dynamics of internal state variables, a learning algorithm is proposed in Section 3. The comparison of the proposed model with classical multilayer perceptron (MLP) is discussed in Section 4. In Section 5, we conclude our work with a brief discussion.

2. Biological neurons

2.1. *Architecture of a biological neuron*

Networks of biological neurons compute with the help of fast traveling pulses called action potentials. A neuron is the fundamental building block of biological neural networks. A

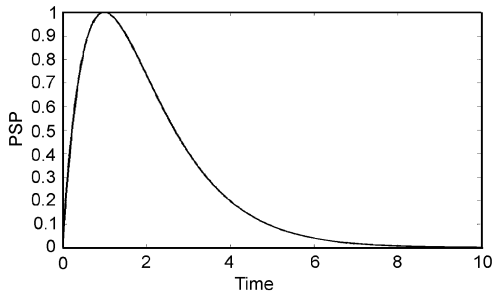

FIG. 1. A typical shape of the post-synaptic potentials (PSP). It models the changes in the membrane potential of the target neuron owing to the arrival of a single spike as a function of time-since-impact.

typical neuron has three major parts: soma, axon, and dendrites. Dendrites form a dendritic tree which is a very fine bush of thin fibers around the neuron's body. Dendrites receive information from neurons through axons. An axon is a long cylindrical connection that carries impulses from the neuron. The end part of an axon splits into a fine arborization which terminates in a small end-bulb almost touching the dendrites of neighboring neurons. The axon-dendrite contact organ is called synapse. Details of the biological neuron can be found in Koch [21], and Tuckwell [22]. Modeling such a complex system is difficult task and the model often has to be drastically simplified in order to make both the computation and analysis somewhat tractable. In the following subsection, we will discuss the functioning of *spiking neuron model* that mimics the biological activities of single neuron.

2.2. *Spiking neuron model*

A spiking neuron is represented by voltage across its cell membrane and a threshold [15]. The status of a neuron is determined by integration of its excitatory and inhibitory post-synaptic potentials (PSP). When its membrane potential reaches a certain threshold, the neuron generates a spike which is propagated to other neurons. The synapse is responsible for transforming the spike into a PSP. Typical shape of a PSP is shown in Fig. 1. A spike takes certain time, called synaptic delay, to reach the post-synaptic neuron. We assume that a neuron has n number of immediate predecessors called presynaptic neurons and receives a set of spikes with firing times t_i, $i = 1, 2, \ldots n$. At most one spike is generated by each neuron during the simulation interval (the presentation of an input pattern), and fires when internal state variable reaches a threshold. Dynamics of the internal state variable $x(t)$ is determined by the impinging spikes, whose impact is described by the spike-response function $\varepsilon(t)$ weighted by the synaptic efficacy or weight w_i [17].

$$x(t) = \sum_{i=1}^{n} w_i \varepsilon(t - t_i), \tag{1}$$

where w_i is positive for excitatory synapse and negative for inhibitory synapse. The spike-response function $\varepsilon(t - t_i)$, which is referred to as PSP, models how the arrival of a single spike changes the membrane potential of the target neuron as a function of time-since-impact. The height of PSP is modulated by synaptic weight w_i to obtain the effective PSP at the target neuron due to a spike from neuron-i. The commonly used spike response function is given in Bohte *et al.* [17] and can be expressed by eqn (2).

$$\varepsilon(t) = \frac{t}{\tau} e^{1-\frac{t}{\tau}}. \tag{2}$$

As an extension of this model, we can consider synaptic delay δ_i, associated with each input. Moreover, time-constant τ can be considered different for different inputs. Thus,

$$x(t) = \sum_{i=1}^{n} w_i \varepsilon(t - t_i - \delta_i),$$

$$x(t) = \sum_{i=1}^{n} w_i \left(\frac{t - t_i - \delta_i}{\tau_i} \right) e^{1-\left(\frac{t-t_i-\delta_i}{\tau_i}\right)}. \tag{3}$$

Thus, the state variable x, at a specified time instant t_0 is

$$x_0 = \sum_{i=1}^{n} \left((a_i + b_i t_i)\, e^{(c_i + d_i t_i)} \right), \tag{4}$$

where,

$$a_i = w_i \left(\frac{t_0 - \delta_i}{\tau_i} \right),$$

$$b_i = \frac{-w_i}{\tau_i},$$

$$c_i = 1 - \left(\frac{t_0 - \delta_i}{\tau_i} \right),$$

$$d_i = \frac{1}{\tau_i}.$$

In generalized form, eqn (4) can be represented by:

$$x_{net} = \sum_{i=1}^{n} \left((a_i + b_i t_i)\, e^{(c_i + d_i t_i)} \right). \tag{5}$$

Considering multiplicative aggregation at dendrites, as discussed in later section, eqn (5) can be written as:

$$x_{net} = \prod_{i=1}^{n} (a_i + b_i x_i)\, e^{\sum_{i=1}^{n} (c_i + d_i x_i)}. \tag{6}$$

The expanded form of this equation is

$$x_{net} = \prod_{i=1}^{n} \left((a_i + b_i x_i)\, e^{(c_i + d_i x_i)} \right)$$

$$= \left[(a_1 + b_1 x_1) e^{(c_1 + d_1 x_1)} \right] \times \left[(a_2 + b_2 x_2) e^{(c_2 + d_2 x_2)} \right] \times \cdots \times \left[(a_n + b_n x_n) e^{(c_n + d_n x_n)} \right]$$
$$= \left[(a_1 + b_1 x_1) \times (a_2 + b_2 x_2) \times \cdots \times (a_n + b_n x_n) \right] \times \left[e^{(c_1 + d_1 x_1)} \times e^{(c_2 + d_2 x_2)} \times \cdots \times e^{(c_n + d_n x_n)} \right]$$
$$= \left[\prod_{i=1}^{n} (a_i + b_i x_i) \right] \times \left[e^{\sum_{i=1}^{n} (c_i + d_i x_i)} \right]. \quad (7)$$

In this work, we concentrate mainly on the timing of spikes in *spiking neuron model* to derive an aggregation function for the learning of a neural network. The details of the proposed model are explained in the following sections.

3. The proposed learning scheme

In this section, we propose a learning scheme that will be used for the task like function approximation and classification. The formulation of the proposed scheme is inspired from the functioning of *multiplicative spiking neuron model* whose details have been discussed in the previous section. The proposed learning scheme is firstly formulated for doing single neuron computation and further it is extended for forming the network of neurons.

Inspired from the relationship between the timings of incoming spikes t_i and state variable x (eqns (3) and (4)) for spiking neuron model, following aggregation function is assumed:

$$x_{net} = \left[\prod_{i=1}^{n} (a_i + b_i x_i) \right] \times \left[e^{\sum_{i=1}^{n} (c_i + d_i x_i)} \right] \quad (8)$$

where n is the number of inputs. x_i is analogous to the input spike time t_i and x_{net} is analogous to the state variable x at a specified time t. It is to be noted that parameters a_i, b_i, c_i and d_i have been used to represent the constant values that depend upon τ_i, δ_i, w_i and the specified time t at which the value of the state variable is to be calculated. It is to be noted here that x_i in eqn (8) is analogous to the input spike time; therefore, it is incorporating the fundamental property of the spiking neuron. Besides this, it is also incorporating the threshold variability by imposing the probabilistic firing.

In view of evidences in support of the presence of multiplicative operations in the nervous system [23], multiplication of net inputs to the activation function is considered. Instead of time of spike in the post-synaptic neuron, we considered the spiking probability y till a specified time t as output. y is a nonlinear function of x_{net} and should have the following properties:

1. It is a monotonically increasing function as the probability of spike is increased for increasing x_{net}.
2. It is almost zero for low values of x_{net} as there is negligible probability of spike if x_{net} is small. This is true even if threshold variability is taken into account.
3. It is almost unity for high values of x_{net} as spike is almost certain for large values of x_{net}. This is also true even if threshold variability is taken into account.

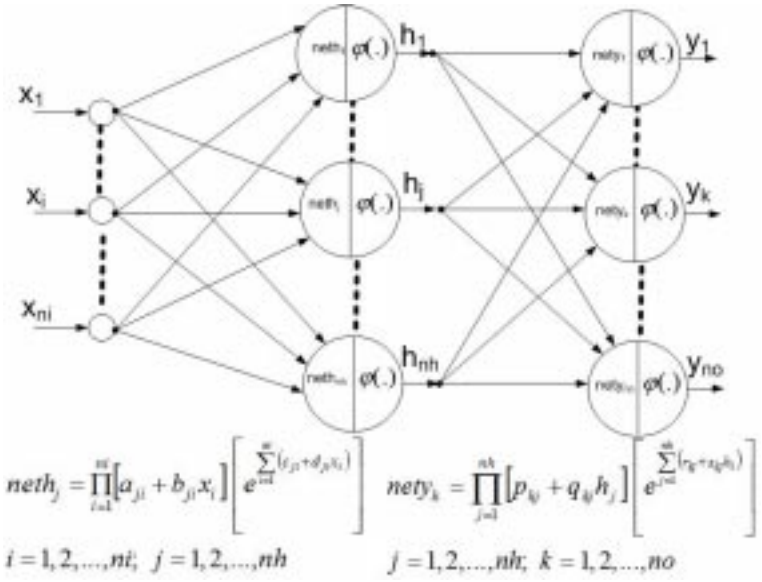

FIG. 2. Sketch of the proposed multiplicative neural network architecture motivated by spiking neuron model.

Assumption of constant threshold would lead to a special case of this function. In that case, the probability of spike would either be zero or one. Incorporation of threshold variability in our model is achieved by assuming this function as the sigmoid function. The threshold variability causes probability of spike generation between 0 and 1 and if we will not consider the threshold variability then probability of spike generation is either 0 or 1. Hence, to incorporate threshold variability aggregated input is passed through the sigmoid function. This is a continuous and differentiable nonlinear function given by

$$y = \frac{1}{1+e^{-x_{net}}}. \tag{9}$$

For single neuron computation, input–output relation is derived in eqns (8), and (9). When we consider this neuron in network like architecture (Fig. 2), the input-output expressions are modified as eqns (10).

The MSN model is inspired from the fact that the actual shape of action potential does not contain any neuronal information. It is the timing of spikes that matters. A substantial body of evidence supports the presence of multiplicative-like operations in nervous system [23]. Physiological and behavioral data strongly suggest that the optomotor response of insects to moving stimuli is mediated by a correlation-like operation [21]. Another instance of a multiplication-like operation in the nervous system is the modulation of receptive field location of neurons in the posterior parietal cortex by the eye and head positions of monkey [21].

In all the conventional neural network models, simple summation is necessary for them to work. However, such a sum cannot distinguish among the individual contributions to it. For the neurons to respond strongly to correlations among particular input pairs or groups,

one must include multiplicative terms and then sum over the product. The multiplicative operation can be used to implement second- and higher-order polynomial relations among a set of inputs. Mel [24] and Durbin and Rumelhart [25] suggest that networks based on sigma-pi units are more powerful and have many advantages with respect to the traditional threshold-based networks. Koch and Poggio [23] discuss the relevance of multiplicative operations, particularly in computation underlying motion perception and learning. Multiplication in neurons often occurs in dendritic trees with voltage-dependent membrane conductances [26]. Arcas *et al.* [27] provide a linear subspace-based approach to model the computation abilities of a neuron. Linearity is believed to be sufficient in capturing the passive properties of dendritic membrane where synaptic inputs are currents. However, synaptic inputs can interact nonlinearly when synapses are colocalized on patches of dendritic membrane with specific properties. Hence, an artificial neuron model then should be capable of including this inherent nonlinearity in the mode of aggregation. Multiplication, being the most basic of all nonlinearities, has been a natural choice of models trying to include nonlinearity in artificial neuron model. Poggio [28] explains the relevance of using multiplication as a computationally powerful and biologically realistic possibility of synthesizing high-dimensional Gaussian radial basis function [29] from low dimensionality. The role of multiplication is explained in the computation underlying motion perception and learning in pairs of individual synapses to a small set of neurons. The nonlinear capability of neuron is usually modeled through a stationary nonlinearity. This however is not sufficient to capture the possible nonlinear association among the inputs to the single neuron systems. In view of these evidences, we incorporate multiplication operation while aggregating inputs to the activation function.

3.1. *Development of the training algorithm*

The ANN motivated from the multiplicative spiking neuron model is shown in Fig. 2. In this network, all the inputs in each layer are aggregated according to eqn (8) to generate the net output. This net is fed through a log sigmoid nonlinearity to create the final output in each layer. This kind of neuron itself looks complex in the first instance but when used to solve a complicated problem it needs reasonably less number of parameters as compared to the existing conventional models. The gradient descent approach has been used to develop the learning algorithm for the network. The learning rules for various weights of a feedforward (FF) network of the proposed multiplicative neuron model can be given by the following set of equations.

3.1.1. *Forward pass*

In forward pass, the net summation and output at each neuron are calculated as per the following equations

$$neth_j = \left[\prod_{i=1}^{ni}\left(a_{ji} + b_{ji}x_i\right)\right] \times \left[e^{\sum_{i=1}^{ni}\left(c_{ji}+d_{ji}x_i\right)}\right]$$

$$h_j = \frac{1}{1+e^{-neth_j}}$$

$$nety_k = \left[\prod_{j=1}^{nh}\left(p_{kj} + q_{kj}h_j\right)\right] \times \left[e^{\sum_{j=1}^{nh}\left(r_{kj}+s_{kj}h_j\right)}\right]$$

$$y_k = \frac{1}{1+e^{-nety_k}}. \tag{10}$$

3.1.2. *Backward pass*

After the forward pass, the total error of the network is given by eqn (11). A simple steepest descent method is applied to minimize this error function.

$$dy_k = (y_k - t_k)$$

$$e = \frac{1}{2}\sum_{k=1}^{n_0}(dy_k)^2 \tag{11}$$

where t_k is target (not time) and y_k is actual output of kth output neuron, e, a function of parameters a_{ji}, b_{ji}, c_{ji}, d_{ji}, p_{kj}, q_{kj}, r_{kj}, and s_{kj} (i = 1, 2, … ni, j = 1, 2, … , nh, k = 1, 2, … , no). Therefore, the parameter update rule (weight update rule) can be expressed by eqns (12)–(15).

$$\begin{aligned}
\frac{\partial e}{\partial p_{kj}} &= dy_k \times y_k \times (1-y_k) \times \frac{nety_k}{p_{kj} + q_{kj}h_j} \\
\frac{\partial e}{\partial q_{kj}} &= \frac{\partial e}{\partial p_{kj}} \times h_j \\
\frac{\partial e}{\partial r_{kj}} &= dy_k \times y_k \times (1-y_k) \times nety_k \\
\frac{\partial e}{\partial s_{kj}} &= \frac{\partial e}{\partial r_{kj}} \times h_j
\end{aligned} \tag{12}$$

$$\begin{aligned}
\frac{\partial e}{\partial a_{ji}} &= \sum_{k=1}^{no}\left(dy_k \times y_k \times (1-y_k) \times nety_k \times \left(S_{ki} + \frac{q_{kj}}{(p_{kj} + q_{kj}h_j}\right)\right) \\
&\quad \times h_j(1-h_j) \times \frac{neth_j}{(a_{ji} + b_{ji}x_i)} \\
\frac{\partial e}{\partial b_{ji}} &= \frac{\partial e}{\partial a_{ji}} \times x_i \\
\frac{\partial e}{\partial c_{ji}} &= \sum_{k=1}^{no}\left(dy_k \times y_k \times (1-y_k) \times nety_k \times \left(s_{kj} + \frac{q_{kj}}{(p_{kj} + q_{kj}h_j}\right)\right) \\
&\quad \times h_j \times (1-h_j) \times neth_j \\
\frac{\partial e}{\partial d_{ji}} &= \frac{\partial e}{\partial c_{ji}} \times x_i.
\end{aligned} \tag{13}$$

The following equations show the update rule which is based on steepest descent algorithm

$$p_{kj}^{new} = p_{kj}^{old} - \eta \times \frac{\partial e}{\partial p_{kj}}$$

$$q_{kj}^{new} = q_{kj}^{old} - \eta \times \frac{\partial e}{\partial q_{kj}}$$

$$r_{kj}^{new} = r_{kj}^{old} - \eta \times \frac{\partial e}{\partial r_{kj}}$$

$$s_{kj}^{new} = s_{kj}^{old} - \eta \times \frac{\partial e}{\partial s_{kj}} \tag{14}$$

$$a_{ji}^{new} = a_{ji}^{old} - \eta \times \frac{\partial e}{\partial a_{ji}}$$

$$b_{ji}^{new} = b_{ji}^{old} - \eta \times \frac{\partial e}{\partial b_{ji}}$$

$$c_{ji}^{new} = c_{ji}^{old} - \eta \times \frac{\partial e}{\partial c_{ji}}$$

$$d_{ji}^{new} = d_{ji}^{old} - \eta \times \frac{\partial e}{\partial d_{ji}}, \tag{15}$$

where ni = number of inputs applied to the network, nh = number of hidden layer neurons, no = Number of outputs in the data set, η = learning rate, a_{ji}, b_{ji}, c_{ji}, d_{ji}, p_{kj}, q_{kj}, r_{kj}, s_{kj} = Parameters of the proposed model. $i = 1, 2, \ldots, ni$; $j = 1, 2, \ldots, nh$; $k = 1, 2, \ldots, no$.

4. Illustrative examples

In this section, we present experimental details and comparison between the proposed (MSN) and conventional (MLP) neural architectures. Parameters of both the networks are randomly initialized in the range [–0.1 0.1]. The networks are simulated on a machine having Pentium (R) 4 CPU of 3.20 GHz with 512 MB of RAM. The validation set is used for deciding when to terminate the training. Training is continued as long as the performance on the validation set keeps on improving. When it ceases to improve, training is stopped. Training is also stopped as the stopping criterion was fulfilled or when a maximum of 1000 or 2000 epochs are reached. The test set performance is then computed for that state of network which has minimum error during the training process. The results shown in this paper are obtained after taking the average multiple runs.

4.1. *Classification problems*

4.1.1. *Pima Indian*

This UCI dataset was contributed by Sigillito. The patients in the dataset are females at least 21 years old of Pima Indian heritage living near Phoenix, Arizona, USA. The problem

is to predict whether a patient would test positive for diabetes given a number of physiological measurements and medical test results. There are two classes, seven numerical attributes, and 532 records. Out of 532 samples, 372 samples, are used for training, 160 for validation and 54 for testing. Performance of the proposed model with three hidden units is compared with the conventional MLP with five hidden units. The number of hidden units is chosen after several (around ten) runs for each case. Specified structures gave the best results. Table I presents the comparison between MLP and MSN. It is observed that the performance of the proposed model is better than the conventional neural network model (MLP).

4.1.2. *Iris data*

This classic data of Anderson and Fisher pertains to a four-input, three-class classification problem. The first four columns indicate petal and sepal widths and lengths of various Iris flowers, and the fifth column indicates the appropriate class (Setosa, Versicolor, and Virginia). For comparison of performance with MSN and MLP in case of *Iris data problem*, we considered MLP with five hidden units. This data set has total 187 samples out of which 105 samples are used for training, 45 for validation and 37 for testing purpose. It is observed that the performance of MSN is better than that of MLP. MSN with less number of parameters is capable of learning this relationship faster than an MLP. Moreover, we found less number of misclassified points with MSN model (Table I).

4.1.3. *Boston housing data*

This UCI dataset gives the housing values in Boston suburbs. There are three classes, 12 numerical attributes, one binary attribute, and 506 records. Out of 506 instants, 318 are used for training, 137 for validation, and 51 for testing purpose. The performance results for this data for all the three classes taken together are shown in Table I. It is found that the performance of MSN is slightly better than that of MLP.

4.1.4. *Congressional voting records data*

This UCI dataset gives the votes of each member of the U.S. House of Representatives of 98th congress on 16 key issues. The problem is to classify a Congressman as a Democrat or a Republican based on the 16 votes. There are two classes, 16 categorical attributes with three categories each and 435 records. Of the records, 435,274 instants are used for training, 117 for validation and 44 for testing purpose. It is found from Table I that the performance of MLP is slightly better than that of MSN.

4.1.5. *Wine recognition data*

The data are the results of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars. The analysis determined the quantities of 13 constituents found in each of the three types of wines. In a classification context, this is a well-posed problem. This classic data of Forina *et al.*, pertains to a four-input, three-class classification problem. For comparison of the performance with MSN and MLP in the case of *wine data problem*, we considered MLP with 5 hidden units and MSN with 3 hidden units. The data set contains 178 samples, of which 80 are used in training, 40 in validation and 58 for testing. It is observed from Table I that the performance of the proposed model is significantly better on this data set.

Table I
Comparison of training and testing performance data for various examples

Sl no.	Parameter	Pima Indian problem		Iris data		Boston housing data		Congressional voting records data	
		MLP	MSN	MLP	MSN	MLP	MSN	MSN	MLP
1.	Minimum MSE for training data	0.0446	0.0432	0.0026	0.0018	0.0141	0.0156	0.0032	0.0093
2.	Minimum MSE for validation data	0.0610	0.0531	0.0024	0.0047	0.0177	0.0155	0.0097	0.0045
3.	MSE for testing data	0.0541	0.0402	0.0541	0.0018	0.0209	0.0184	0.0131	0.0184
4.	Iterations needed	500	475	1000	1000	1000	1000	1000	1000
5.	Correlation coefficient	0.5775	0.6693	0.9838	0.979	0.8594	0.9039	0.9123	0.9029
6.	Total training time (s)	140.6	160.90	545.4531	565.7500	615.1250	679.8281	588.9844	656.7188
7.	Total testing time (s)	2.13	2.53	1.5000	1.6250	4.4531	2.9063	3.1719	4.8750
8.	AIC (Akaikes information criterion)	–1392.0	–1661.9	–576.3466	–631.7954	-1.2151×10^3	-1.0914×10^3	-1.4059×10^3	-1.1791×10^3
9.	Number of hidden neurons	5	3	5	3	5	3	5	3
		Wine recognition data		Electroencephalogram data		Wolfer sun spot data		MackeyGlass (MG) time series data	
		MLP	MSN	MLP	MSN	MLP	MSN	MLP	MSN
1.	Minimum MSE for training data	0.0023	6.6274×10^{-4}	0.0033	0.0034	0.0027	0.0023	1.5339×10^{-4}	1.0150×10^{-4}
2.	Minimum MSE for validation data	0.0029	0.0024	0.0033	0.0034	0.0033	0.0026	1.7600×10^{-4}	1.2280×10^{-4}
3.	MSE for testing data	0.0033	7.0507×10^{-4}	0.0035	0.0044	0.0044	0.0044	2.0266×10^{-4}	1.4411×10^{-4}
4.	Iterations needed	1000	1000	1000	1000	1000	1000	1000	1000
5.	Correlation coefficient	0.9692	0.9939	0.8218	0.8199	0.8846	0.8950	0.9953	0.9978
6.	Total training time (s)	555.7813	550.4219	654.4844	706.5938	540.6250	523	654.7969	698.4063
7.	Total testing time (s)	3.6719	2.8281	0.0313	0.0313	4.6094	2.7656	0.0469	0.0469
8.	AIC (Akaikes information criterion)	–368.7272	–530.8070	-1.9541×10^3	-1.9645×10^3	–384.0096	–407.4649	-2.7165×10^3	-2.8666×10^3
9.	Number of hidden neurons	5	3	5	3	5	3	5	3

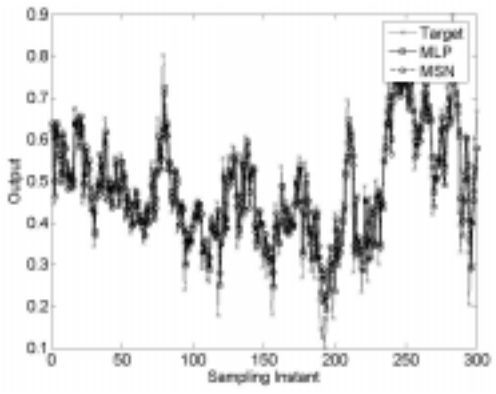

FIG. 3. Target and actual output of the proposed model and MLP for electroencephalogram data.

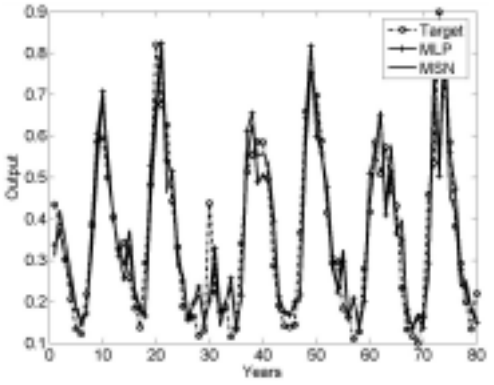

FIG. 4. Target and actual output of the proposed model and MLP for wolfer annual sunspot data.

4.2. *Function approximation problems*

4.2.1. *Electroencephalogram (ECG) data*

EEG data used here is taken from the website of ColoState University. The presence of randomness and chaos [8] in this data makes it interesting for neural network-related research. In this problem, four measurements $y(t-1)$, $y(t-2)$, $y(t-4)$ and $y(t-8)$ were used to predict $y(t)$. Figure 3 shows the comparison between MLP and MSN in terms of deviation of actual outputs from corresponding targets. The EEG data set contains 800 samples, of which 350 were used for training, 150 for validation and 300 for testing. It can be seen that the performance of MSN for training as well as testing data is identical to MLP, but the number of hidden units needed in MLP is more than that in MSN (Table I).

4.2.2. *Wolfer sun spot data*

Wolfer sun spot [30] numbers are used for time-series analysis. The data set, usually attributed to Rudolf Wolf, consists of means of daily relative numbers of sunspot sightings. The relative number for a day is given by $k(f + 10g)$, where g is the number of sunspot groups observed, f, the total number of spots within the groups and k, a scaling factor relating the observer and telescope to a baseline. The relative numbers are then averaged to give an annual figure. The averaged annual sun spot numbers used here are for 180 years (1749–1929). Among 180 points, 70 are used for training, 30 for validation and 80 for testing the capability of the proposed model. The performance of the proposed neuron model is compared with that of MLP in Table I. Figure 4 shows the plot of target and actual output values for MLP and MSN. It is evident from the table and plot that the performance of the proposed model is comparable to conventional neural network model.

4.2.3. *MackeyGlass (MG) time-series data*

The MackeyGlass (MG) time series [19] represents a model for white blood cell production in leukemia patients and has nonlinear oscillations. The MG-delay difference equation is given by eqn (16).

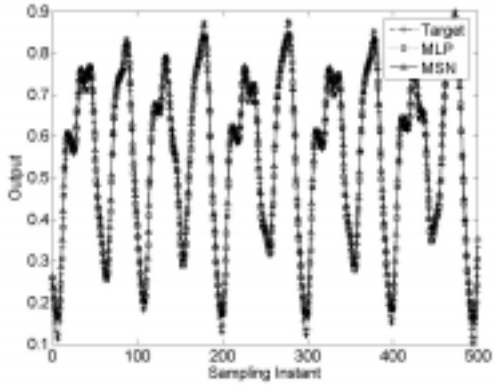

FIG. 5. Target and actual output of the proposed model and MLP for MackeyGlass (MG) time series data.

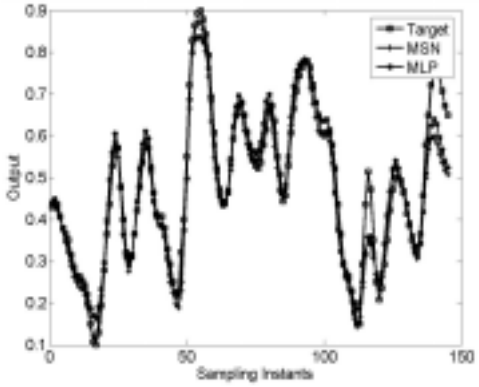

FIG. 6. Target and actual output of the proposed model and MLP for BoxJenkins gas furnace data.

$$y(t+1) = (1-b)\,y(t) + a\,\frac{y(t-\tau)}{1+y^{10}(t-\tau)}, \tag{16}$$

where $a = 0.2$, $b = 0.1$, and $\tau = 17$. The time delay τ is a source of complications in the nature of the time series. The objective of the modeling is to predict the value of the time series based on four previous values. Four measurements, $y(t-1)$, $y(t-7)$, $y(t-13)$ and $y(t-19)$, are used to predict $y(t)$. The training is performed on 315 samples; validation is performed on 135 samples and the model is tested on 500 time instants post training. Figure 5 shows the prediction results. In Table I, the performance of the proposed neuron model with one hidden layer having three nodes is compared with a multilayer network with one hidden layer having five nodes. It can be seen that the performance of MSN for training as well as testing data is much better than that of MLP. MSN is capable to learn this relationship faster than that in the case of MLP and its performance on seen as well as unseen data is significantly better.

4.2.4. *BoxJenkins gas furnace data*

The BoxJenkins gas furnace dataset reports the furnace input $u(t)$ as the gas flow rate and the furnace output $y(t)$ as the CO_2 concentration. In this gas furnace, air and methane were

Table II
Comparison of training and testing performance for BoxJenkins gas furnace data

Sl. no.	Parameter	MLP	MSN
1.	Minimum MSE for training data	2.0004×10^{-4}	2.0995×10^{-4}
2.	Minimum MSE for validation data	2.6728×10^{-4}	2.2219×10^{-4}
3.	MSE for testing data	0.0011	0.0012
4.	Iterations needed	1000	1000
5.	Total training time (s)	540.1563	549.7188
6.	Total testing time (s)	0.0156	0.0156
7.	AIC (Akaikes information criterion)	–842.2168	–837.3350
8.	Number of hidden neurons	3	3
9.	Correlation coefficient	0.9692	0.9693

combined in order to obtain a mixture of gases which contained CO_2. We model the furnace output $y(t)$ as a function of the previous output $y(t-1)$ and input $u(t-1)$. The training is performed on 101 samples; validation is performed on 44 samples and the model is tested on 145 samples. The generalization capability of the proposed model is shown in Fig. 6. Table II shows the comparison of training and generalization capabilities of the proposed neuron with MLP.

5. Conclusions

This paper presents a new approach towards the conceptualization of an artificial neural system motivated from the spiking neuron model with better learning and generalization capabilities. Idea of the spiking neuron model is inspired from the fact that the actual shape of action potential does not contain any neuronal information. It is the timing of spikes that matters. The proposed model incorporates multiplicative aggregation at dendrites and threshold variability in biological neurons. The training and testing results with different benchmark and real-life problems are discussed. It is found that the proposed artificial neural network with comparatively less number of hidden neurons is capable of performing classification and function approximation tasks as efficiently as a multilayer perceptron with several hidden neurons. Moreover, in many cases, its learning is significantly better than that of a conventional multilayer perceptron.

References

1. W. J. Freeman, Why neural networks do not yet fly: inquiry into the neurodynamics of biological intelligence, *IEEE Int. Conf. on Neural Networks*, July 24–27, 1988, Vol. 2, pp. 1–7 (1988).
2. W. McCulloch, and W. Pitts, A logical calculus of the ideas immanent in nervous activity, *Bull. Math. Biophys.*, **5**, 115–133 (1943).
3. D. Hebb, *Organization of behavior*, Wiley (1949).
4. B. Widrow, and M. E. Hoff, *Adaptive switching circuits*, IREWESCON Connection Records, IRS, New York (1960).
5. B. Widrow, and S. Steams, *Adaptive signal processing*, Prentice-Hall (1985).
6. M. Sinha, D. K. Chaturvedi, and P. K. Kalra, Development of flexible neural network, *J. IE(I)*, **83** (2002).
7. R. N. Yadav, V. Singh, and P. K. Kalra, Classification using single neuron, *Proc. IEEE Int. Conf. on Industrial Informatics*, Banff, Alberta, Canada, Aug. 21–24, 2003, pp. 124–129.
8. D. Mishra, A. Yadav, and P. K. Kalra, Chaotic behavior in neural networks and FitzHugh-Nagumo neuronal model, *Proc. ICONIP-2004, LNCS 3316*, Dec. 2004, India, pp. 868–873.
9. M. Scholles, B. J. Hosticka, M. Kesper, P. Richert, and M. Schwarz, Biologically-inspired artificial neurons: modeling and applications, *Proc. 1993 Int. Jt Conf. Neural Networks*, IJCNN '93-Nagoya, Oct. 25–29, 1993, Vol. 3, pp. 2300–2303 (1993).
10. J. Feng, and G. Li, Neuronal models with current inputs, *J. Phys. A*, **24**, 1649–1664 (2001).
11. J. Feng, H. Buxton, and Y. C. Deng, Training the integrate-and-fire model with the Informax principle I, *J. Phys. A*, **35**, 2379–2394 (2002).
12. J. Feng, Y. Sun, H. Buxton, and G. Wei, Training integrate-and-fire neurons with the Informax principle II, *IEEE Trans. Neural Networks*, **14**, 326–336 (2003).
13. W. Gerstner, Time structure of the activity in neural network models, *Phys. Rev. E*, **51**, 738–758 (1995).

14. J. J. Hopfield, Pattern recognition computation using action potential timing for stimulus representation, *Nature*, **376**, 33–36 (1995).

15. W. Gerstner, and W. Kistler, *Spiking neuron models: Single neurons, populations, plasticity*, Cambridge University Press (2002).

16. W. Maass, Networks of spiking neurons: the third generation of neural network models, *Neural Networks*, **10**, 1659–1671 (1997).

17. S. M. Bohte, H. La Poutre, and J. N. Kok, SpikeProp: Error-backpropagation for networks of spiking neurons, *ESANN*2000, pp. 419–425 (2000).

18. A. Yadav, D. Mishra, R. N. Yadav, S. Ray, and P. K. Kalra, Learning with single integrate-and-fire neuron, *IEEE Int. Jt Conf. on Neural Network, IJCNN*-2005, Montreal, Canada, Vol. 4, pp. 2156–2161 (2005).

19. A. Yadav, D. Mishra, R. N. Yadav, S. Ray, and P. K. Kalra, Time-series prediction with single integrate-and-fire neuron, *Appl. Soft Computing*, **1** (in press).

20. D. Mishra, A. Yadav, and P. K. Kalra, A neural network using single multiplicative spiking neuron for function approximation and classification, *IEEE Int. Jt Conf. on Neural Network, IJCNN*-2006, Canada, pp. 396–403 (2006).

21. C. Koch, *Biophysics of computation: Information processing in single neurons*, Oxford University Press (1999).

22. H. C. Tuckwell, *Introduction to theoretical neurobiology*, Cambridge University Press (1988).

23. C. Koch, and T. Poggio, *Multiplying with synapses and neurons. Single neuron computation*, Academic Press, pp. 315–315 (1992).

24. B. W. Mel, and C. Koch, Sigma-pi learning: on radial basis functions and cortical associative learning, in *Advances in neural information processing systems*, Vol. 2 (D. S. Touretzky, ed.), Morgan Kaufmann, pp. 474–481 (1990).

25. R. Durbin, and D. E. Rumelhart, Product units: a computationally powerful and biologically plausible extension to back-propagation networks, *Neural Computation*, **1**, 133–142 (1989).

26. B. W. Mel, Information processing in dendritic trees, *Neural Computation*, **6**, 1013–1085 (1994).

27. B. A. Arcas, A. L. Fairhall, and W. Bialek, What can a single neuron compute?, *Proc. Adv. Neural Inf. System* (T. Leen, T. Dietterich, and V. Tresp, eds), Vol. 13, pp. 75–81, The MIT Press (2001).

28. T. Poggio, On optimal nonlinear associative recall, *Biol. Cybernetics*, **19**, 201–209 (1975).

29. S. Haykin, *Neural networks: A comprehensive foundation*, Pearson Education (2003).

30. A. J. Inzenman, J. R. Wolf, and H. A. Wolfer, An historical note on the Zurich sunspot relative numbers, *J. R. Stat. Soc., A*, **146**, 311–318 (1983).