Hardware-Amenable Structural Learning for Spike-based Pattern Classification using a Simple Model of Active Dendrites

Shaista Hussain

School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore.

Shih-Chii Liu

Institute of Neuroinformatics, University of Zurich and ETH Zurich, Switzerland.

Arindam Basu

arindam.basu@ntu.edu.sg; Corresponding author

School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore.

Abstract

This paper presents a spike-based model which employs neurons with functionally distinct dendritic compartments for classifying high dimensional binary patterns. The synaptic inputs arriving on each dendritic subunit are nonlinearly processed before being linearly integrated at the soma, giving the neuron a capacity to perform a large number of input-output mappings. The model utilizes sparse synaptic connectivity - where each synapse takes a binary value. The optimal connection pattern of a neuron is learned by using a simple hardware-friendly, margin enhancing learning algorithm inspired by the mechanism of structural plasticity in biological neurons. The learning algorithm groups correlated synaptic inputs on the same dendritic branch. Since the learning results in modified connection patterns, it can be incorporated into current event-based neuromorphic systems with little overhead. This work also presents a branch-specific spike-based version of this structural plasticity rule. The proposed model is evaluated on benchmark binary classification problems and its performance is compared against that achieved using Support Vector Machine (SVM) and Extreme Learning Machine (ELM) techniques. Our proposed method attains comparable performance while utilizing $10 - 50\%$ less computational resources than the other reported techniques.

1   Introduction

Neuromorphic systems (Mead, 1990) aim to emulate neuronal function by implementing the structure of the biological system. By using analog subthreshold Very Large

Scale Integration (VLSI) circuits, these systems can implement a power-efficient model of neuronal processing. This approach is particularly useful for testing ideas about brain function that span across the levels of ion channels to neural micro-circuits and systems. There has already been significant effort in the past in this realm (Arthur and Boahen, 2007; Bartolozzi and Indiveri, 2009; Basu et al., 2008; Hynna and Boahen, 2006; Liu and Boahen, 1996; Liu et al., 2001; Mahowald, 1994; Hasler and Marr, 2013). The latest neuromorphic hardware systems are event-driven real-time implementations of energy-efficient, spike-based intelligent processing systems. Two key advantages of these systems are as follows: One, the event-driven way of processing leads to minimal power dissipation during long inactive periods (O'Connor et al., 2013; Neil and Liu, 2014) and two, the systems are low-power because the transistors are operated in the sub-threshold region where currents are small. Usable event-driven sensors such as the Dynamic Vision Sensor retina (Lichtsteiner et al., 2008; Serrano-Gotarredona and Linares-Barranco, 2013; Posch et al., 2011) and the AEREAR2 cochlea (Liu et al., 2013) that produce asynchronous spike information about stimulus features in real-time environments are already available. There is a need to interface these sensors to event-based multi-neuron classifier systems that generate information only when necessary. These classifiers can benefit from the larger computational power of spiking neural networks (Maass and Schmitt, 1999) and can also be used in novel applications related to brain-machine interfaces where the incoming spikes are generated by biological neurons (Nicolelis, 2003).

Reconfigurable spiking neural networks composed of neurons and synapses have been fabricated in VLSI. The networks are usually mixed hybrid analog-digital circuits on the chip while the connectivity information of the network is stored in an off-chip

digital memory. An asynchronous communication protocol called the Address-Event Representation (AER) (Boahen, 2000; Serrano-Gotarredona et al., 2009; Choi et al., 2005; Vogelstein et al., 2007) is typically used to route the neuronal spikes to and from the chip on a shared 'fast' digital bus. Since the connectivity information is virtual, the user has full flexibility in the network reconfiguration. Synaptic weights are typically implemented using on-chip digital to analog converters (DAC) that are shared across synapses. Very few spike-based multi-neuron systems have reported solving real-world pattern classification or memorization problems comparable with state-of-the-art systems using algorithmic approaches. One major challenge in attaining a classification accuracy similar to a software implementation of a system on a computer is that statistical variations in VLSI devices reduce the accuracy of the synaptic weights. This problem is exacerbated by the fact that device mismatch is increasing with progressively shrinking transistor sizes. Hence, it has become increasingly important to design networks and algorithms which either exhibit robustness to variations or utilize low-resolution synapses which are better suited for robust implementation in VLSI systems. A spiking network classifier was implemented on a neuromorphic hardware system in which population coding was used to achieve tolerance against variability (Schmuker et al., 2014). This system however uses a large number of synapses, which need to have high resolution weights to attain the performance reported in the study. This neuromorphic network achieved performance comparable with a standard machine learning linear classifier. However, nonlinear classifiers like the Support Vector Machine (SVM) can give far better results.

In another study (Neftci et al., 2014), software simulation of a spiking neuromor-

phic system consisting of Restricted Boltzmann Machines (RBMs) was proposed with a learning rule based on Spike Timing Dependent Plasticity (STDP) for changing synaptic weights. These RBMs composed of leaky integrate and fire neurons were shown to be robust to the use of finite-precision synaptic weights with less than 3% drop in performance for an MNIST recognition task when using 8-bit vs 5-bit precision weights. However, RBMs use a high number of synapses, almost two million recurrent connections, which makes its hardware implementation quite costly. In contrast to this, several studies have used binary synapses inspired from the evidence that biological synapses exist in only a small number of states (Petersen et al., 1998; O'Connor et al., 2005). The use of binary synapses also provides a simple and robust solution to the variability due to device mismatch. A spike-based STDP learning algorithm using bistable synapses was implemented in Brader et al. (2007) with VLSI implementations in Chicca et al. (2003), Badoni et al. (2006) and Mitra et al. (2009). This supervised learning algorithm was used to train a network to classify handwritten characters from the MNIST database (Brader et al., 2007). The classification accuracy was improved by having a pool of neurons that compute in parallel. However, this again leads to the problem of increased number of synapses by a factor equal to the number of neurons in the pool.

In this work, we present a bio-plausible structural learning rule for a neuron with nonlinear dendritic branches and suitable for implementation on neuromorphic chips. The paper is organized as follows: the full spike-based classification model of neurons with dendritic nonlinearity is presented in Section 2.1, followed by the reduced model for faster training (RM) in Section 2.2 and a margin-based reduced model (RMWM) in Section 2.3. The theoretical capacities of the RM and RMWM provide an upper bound

on the number of patterns that can be correctly classified, as discussed in Section 2.4. The learning rules for the RM and RMWM are based on the structural plasticity mechanism and are presented in Section 2.5. Further, a branch-specific spike-timing-based implementation of the structural learning rule (BSTDSP) is presented in Section 2.6. The models were tested on a 2-class classification task (Sections 3.1-3.3) and the performance of the RMWM was compared against the SVM and ELM methods on a set of real-world binary databases in Section 3.4. The biological and computational relevance of our work is discussed in Section 4 followed by the conclusion and discussion of future work in Section 5.

2 Model of Nonlinear Dendrites & Structural Learning

Various computational modeling studies have shown that neuronal cells having nonlinear dendrites (NLD) can perform complex classification tasks much more accurately than their counterparts with linear subunits when the inputs are high-dimensional binary patterns (Poirazi and Mel, 2001; Hussain et al., 2013). The presence of NLD provides a neuron with many nonlinear functions to better approximate the decision boundary of a complex classification problem. In this work we extend the NLD model of Poirazi and Mel (2001) to perform supervised classification of binary patterns consisting of rate encoded spike train inputs or patterns of single spikes.

We consider two types of spike inputs - mean rate and single spike patterns. These spiking inputs encode binary patterns of numbers denoted as $x \in R^d$ where each component $x_i \in \{0, 1\}$. The first type is binary patterns of mean rate inputs which are
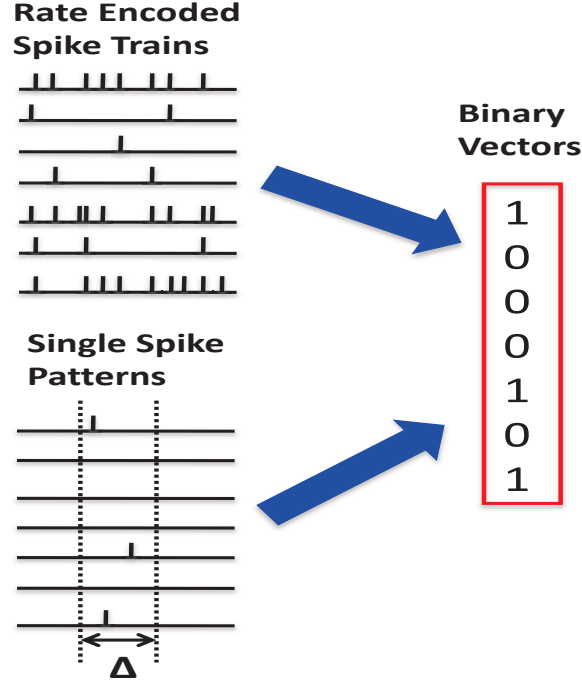
6

Figure 1: Conversion of binary patterns of spike inputs into binary vectors. An example of how rate encoded spike trains (top left) and single spikes (bottom left) are mapped to the same binary vector (right) is shown.

defined such that for a spike train $s(t) = \sum_{t_k} \delta(t - t_k)$ arriving at an afferent, where $\delta(t)$ is a delta function and spike times $\{t_k\} \in [0, T]$ (T is the duration of the pattern), if the mean firing rate of the spike train (averaged over T) is higher than a threshold, the corresponding x_i is set to 1; else it is set to 0 (Figure 1). The second type of spike inputs are binary patterns of single spikes in which $x_i = 1$ if the ith afferent fires a single spike and $x_i = 0$ if it fails to fire. Furthermore, all spikes arrive within a small time window (Δ), i.e. $t_k \in [T_{syn} - \frac{\Delta}{2}, T_{syn} + \frac{\Delta}{2}]$ where T_{syn} is a fixed time between $[0, T]$. The binary inputs x are referred to as **binary vectors** in the rest of the paper. To reduce the training time, we have adopted the strategy of training the model on binary
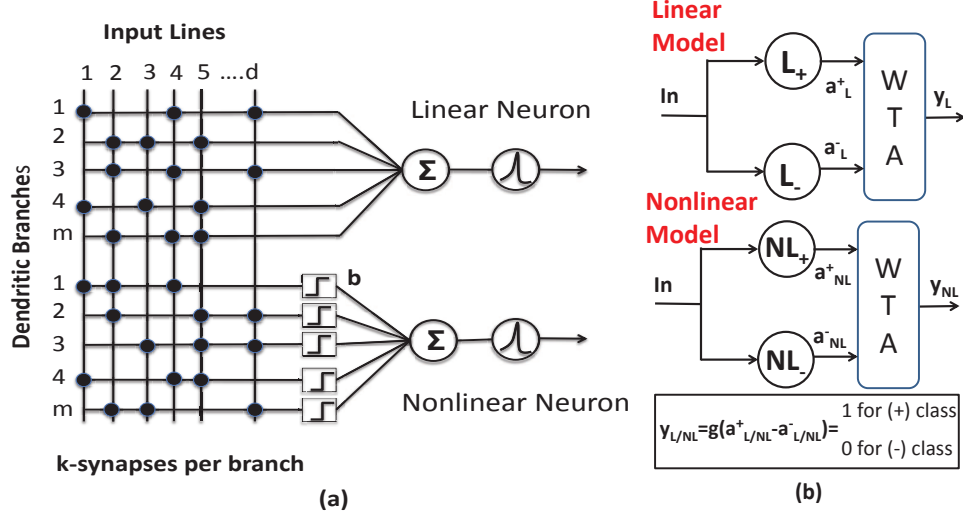
Figure 2: (a) The major difference between a traditional linearly summing neuron and our neuron with nonlinear dendrites is the lumped nonlinear function $b()$ for every dendritic branch. (b) Architecture of the 2-class pattern classifier that consists of a WTA to compare the outputs of two neurons trained to respond to patterns of classes (+) and (-) respectively by using the Heaviside function $g()$ to generate the output of the model.

vectors x and testing on the original spike train patterns. We shall next describe the full spike-based model in Section 2.1 followed by its reduced versions.

2.1 Spike-Based Model

We previously proposed a model of neuronal cells having a lumped dendritic nonlinearity in Hussain et al. (2013). The nonlinear neuron (NL-neuron) consists of a simplified neuron with m dendritic branches and k excitatory synapses on each branch (Figure 2(a)). Each synapse is driven by one of the d components of the input. We enforce the constraint $k << d$ in our model for several reasons: first, choosing the best 'k' connections out of 'd' possibilities is the basis of our learning rule; second, sparse connections

8

reduce synaptic resources and third, this provides a means of regularization by reducing the number of free parameters.

Let us consider the input spike train at the ith synapse of the jth branch given by:

$$s_{ij}(t) = \sum_{t_{ij}^s} \delta(t - t_{ij}^s) \tag{1}$$

where t_{ij}^s denotes the times at which spikes arrive at this synapse. We use the leaky integrate and fire neuron model to generate the spike output using the following differential equation for the membrane voltage:

$$C\frac{dV_m(t)}{dt} + \frac{V_m(t)}{R} = I_{in}(t) \tag{2}$$

$$\text{If } V_m(t) \geq V_{thr}, V_m(t) \rightarrow 0;$$

$$n_{spk} \rightarrow n_{spk} + 1$$

where V_{thr} is the threshold for firing and n_{spk} is the spike count, which is initialized to 0 before every pattern presentation. The input current to the neuron, $I_{in}(t)$ is the sum of all dendritic currents, given by

$$I_{in}(t) = \sum_j I_{b,out}^j(t) \tag{3}$$

$$I_{b,out}^j(t) = b(I_{b,in}^j(t)) \tag{4}$$

$$I_{b,in}^j(t) = \sum_i w_{ij}\left(\sum_{t_{ij}^s < t} K(t - t_{ij}^s)\right) \tag{5}$$

Here $I_{b,out}^j(t)$ is the branch output current, $I_{b,in}^j(t)$ is the branch input current given by the sum of currents generated from all synapses on the jth branch, w_{ij} is the weight of the ith synapse formed on the jth branch, $i \in \{1, \cdots d\}$, $j \in \{1, \cdots m\}$ such that $\sum_i w_{ij} = k$ where $w_{ij} \in Z_+$ which is the set of non-negative integers and b denotes the
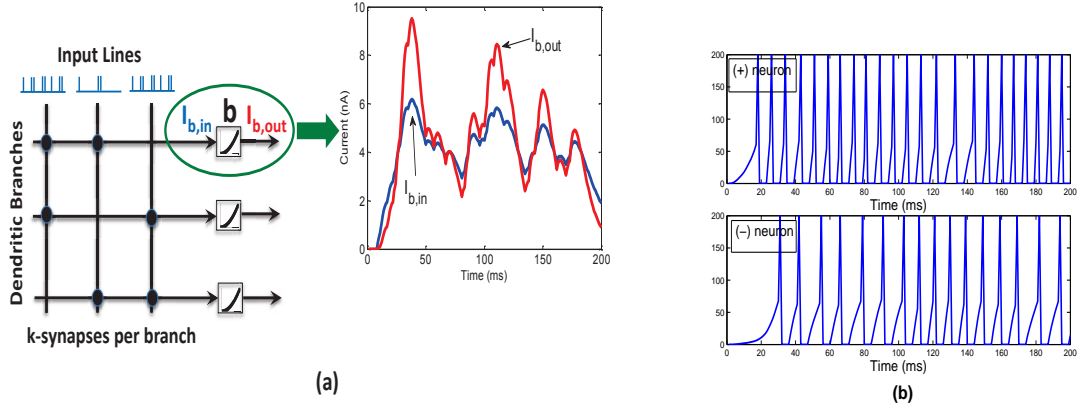
9

Figure 3: (a) The input and output of a dendrite (shown within ellipse) when a rate encoded spike pattern is presented. Time waveform of the input current, $I_{b,in}$ and the output current, $I_{b,out}$ of the branch are shown, where branch nonlinearity $b()$ is a quadratic function. (b) For an input pattern in class (+), the (+) neuron has a higher spiking rate. These output spike trains are input to a WTA which decides the winning neuron.

nonlinear activation function of the dendritic branch. The postsynaptic current (PSC) generated by the input spikes at an afferent is given by $K(t - t_{ij}^s) = K(t) \star s_{ij}(t)$ where the kernel function $K(t)$ (Gutig and Sompolinsky, 2006) is given by:

$$K(t) = I_0(exp(-t/\tau_f) - exp(-t/\tau_r)) \tag{6}$$

where I_0 is the normalization constant; and τ_f and τ_r are the fall and rise time constants of the PSC respectively.

We investigate the use of this model in a supervised classification of patterns belonging to two classes denoted as (+) and (-). Figure 2(b) shows the classifier that uses a two-input winner take all (WTA) (Lazzaro et al., 1988; Yuille and Geiger, 1995) circuit to compare the outputs of a pair of one excitatory (+) and one inhibitory (-) neuron trained to preferentially respond to patterns of class (+) or (-) respectively. The WTA

10

circuit is used to select the winning neuron by generating the binary-valued output of the classifier, y as follows:

$$\text{If } n_{spk}^{+} > n_{spk}^{-}, y = 1$$

$$\text{else } y = 0 \tag{7}$$

Figure 3(a) shows the input and output of a dendrite (ellipse) when a rate encoded spike pattern is presented. Sample waveforms denote the time evolution of $I_{b,in}(t)$ and $I_{b,out}(t)$ for the dendrite for a case where $b()$ is a squaring function. Figure 3(b) shows an example case where the firing rate of the neuron trained on mean rate encoded spike patterns of class (+) is higher for an input belonging to that class, i.e. $n_{spk}^{+} > n_{spk}^{-}$. These output spike trains of the (+) and (-) neurons are used as inputs to the WTA which gives the classification output.

2.2   Reduced Model (RM)

For the two specific cases of spike trains considered here (mean rate inputs and single spike patterns), we can reduce the above model to reduce the time required for the training process. The reduced model was briefly introduced in Hussain et al. (2013) and its validity is discussed in Section 5.2. First, we present the details of the RM and then we describe the improved modified form proposed in this work. We first define the average activation level ($z_{syn,ij}$) at a synapse as:

$$z_{syn,ij} = \frac{1}{T} \int_0^T \sum_{t_{ij}^s} K(t - t_{ij}^s) dt \tag{8}$$

11

where T is the duration of a pattern. Therefore, the average input current to a neuron is given by:

$$\tilde{I}_{in} = \sum_j b(\sum_i w_{ij} z_{syn,ij})$$ (9)

Here $z_{syn,ij}$ is directly proportional to x_{ij} which is the input arriving at the ith synapse on the jth branch, and can be written as $z_{syn,ij} = K_{syn} x_{ij}$. Therefore, the RM gives the average activation level of a NL-neuron as $a_{NL} = \hat{f}(\tilde{I}_{in})$, where $\hat{f}$ is the current-to-frequency function of the neuron (e.g. threshold linear functions can be used to approximate the expected firing rate of integrate and fire neurons (Gerstner and Kistler, 2002)). For simplicity, we have used a function with unity slope and zero threshold, leading to $\hat{f}(\tilde{I}_{in}) = \tilde{I}_{in}$ (choosing other values of slope will not affect the result of classification where only a comparison is made between output of the two neurons). Therefore, the activation level a_{NL} at the soma is given by:

$$a_{NL} = \sum_j b(\sum_i w_{ij} z_{syn,ij})$$ (10)

Similarly, for a linear neuron (L-neuron), the output of the soma is defined as $a_L = \sum_j \sum_i w_{ij} z_{syn,ij}$. In Figure 2(a), the L-neuron is drawn in a manner similar to the NL-neuron to emphasize that the only difference between the two models is the absence or presence of the nonlinearity $b()$. Also, in all simulations both the linear and nonlinear neurons have the same total number of synapses (or synaptic resource), $s = m \times k$ so that we can compare performance of the two neuron types. Finally we can calculate the output of the classifier by modeling the WTA as an ideal comparator to get:

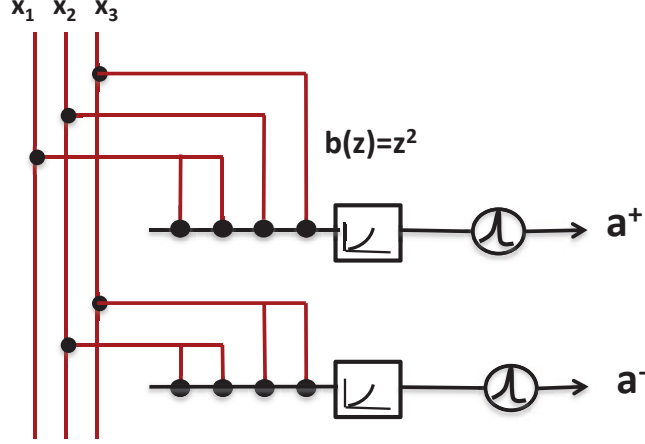$$y_{L/NL} = g(a^+_{L/NL} - a^-_{L/NL})$$ (11)

12

Figure 4: An example of (+) and (-) neurons with one dendrite each and a squaring nonlinearity. Connections on the branches with 4 synapses each are shown for binary input vector $\boldsymbol{x} = [x_1 \ x_2 \ x_3]^T$.

where g is a Heaviside step function that gives an output 1 if the argument $(a^+_{L/NL} - a^-_{L/NL})$ is positive and 0 otherwise. The subscript L/NL of the variables denotes the corresponding output of a linear/nonlinear neuron. For convenience, we will drop this subscript in the remainder of the paper with the understanding that y or a refers to y_L/y_{NL} or a_L/a_{NL} in the context of linear/nonlinear neurons.

The nonlinear dendritic subunits in the RM act as higher-order units which can capture nonlinear input-output relationships as in higher-order networks (Giles and Maxwell, 1987; Rumelhart et al., 1986; Ghosh and Shin, 1992). These nonlinear mappings can be realized by polynomial functions which are useful nonlinearities used to represent correlations among input components. We have used a squaring function for the dendritic nonlinearity, $b(z_j) = z_j^2/x_{thr}$, where $z_j = \sum_i w_{ij} z_{syn,ij}$ is the total synaptic activation on the jth branch. With this branch function, we can write the output of a NL-neuron as $a_{NL} = \frac{K_{syn}^2}{x_{thr}} \sum_j (\sum_i w_{ij} x_{ij})^2$. This nonlinearity is chosen for its easy implementation

13

in hardware using current-mode circuits (Liu et al., 2002; Basu et al., 2010). For our case of binary input vectors and binary synapses, setting $x_{thr} \geq 2$ ensures that at least two synapses on a NLD need to be active ($z_j \geq 2$) to evoke a supra-linear response. The use of a quadratic nonlinearity yields first and second-order terms in the decision function of the classifier. This is illustrated in the example shown in Figure 4, which shows (+) and (-) neurons with one dendrite each and input connections as shown. For the binary input vector $\boldsymbol{x} = [x_1 \; x_2 \; x_3]^T$ where $x_i \in 0, 1$, the two dendritic functions can be written as $(2x_1 + x_2 + x_3)^2$ and $(2x_2 + 2x_3)^2$, which generate cross-correlation terms $x_1 x_2$, $x_2 x_3$ and $x_1 x_3$. Finally, the discriminant function $h(\boldsymbol{x})$ can be written as the difference of the outputs of (+) and (-) neurons.

$$h(\boldsymbol{x}) = (2x_1 + x_2 + x_3)^2 - (2x_2 + 2x_3)^2$$

$$= 4x_1^2 - 3x_2^2 - 3x_3^2 + 4x_1 x_2 + 4x_1 x_3 - 6x_2 x_3$$

where $h(\boldsymbol{x}) = 0$ gives the decision boundary in the 3-dimensional pattern space. If the dendritic nonlinearity is a polynomial of any order, n, it will generate terms like $x_i^p x_j^q$ in each dendritic output function, where p and q are positive integers. Since for binary inputs $x_i^p = x_i$ and $x_i^p x_j^q = x_i x_j$, therefore the connection matrix learned by the neurons (or equivalently the decision boundary) with any polynomial dendritic nonlinearity reflects the correlations present in the inputs. In other words, the (+) neuron should make connections reflecting those correlations present in patterns belonging to class (+) but not in class (-) and vice versa. This will be shown later in Section 3.1.1.

The use of a squaring nonlinearity for each dendritic branch will result in unrealistically large values for large inputs and large power dissipation in the corresponding analog VLSI hardware. Hence, we introduce a saturation level such that for $b(z_j) \geq b_{sat}$,
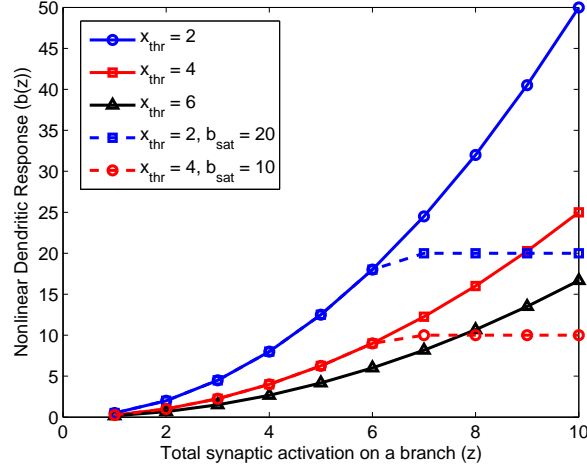
Figure 5: Dendritic nonlinear function $b(z)$ for different values of x_{thr} and b_{sat}.

$b(z_j) = b_{sat}$. Figure 5 shows the nonlinear function $b(z)$ for different x_{thr} values, with (dashed) and without (solid) the b_{sat} term. This saturating dendritic activation function also conforms well with the electrophysiological measurements of synaptic summation within dendrites of pyramidal neurons (Polsky et al., 2004). Fits using saturating polynomial functions on the experimental data measured in pyramidal neurons in the cited study is presented in Section 4.1.

2.3   Reduced Model with Margin (RMWM)

We have shown in Hussain et al. (2013) that when RM is tested with noisy spike inputs, the classification error is about 5 times higher than the training error. Therefore, we describe modifications to this model with the objective of enhancing the robustness of the model to noisy inputs. This method uses a modified margin function $g_{margin}()$ function (Figure 6(b)) during training. We can define $g_{margin}() : \Re \to \Re$, where $\Re$ is
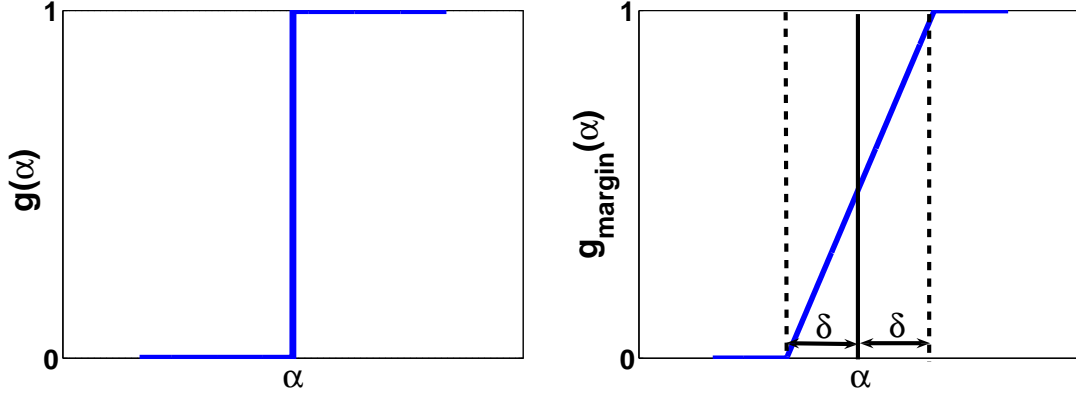
Figure 6: (a) Heaviside step function $g()$. (b) g_{margin} function to enforce a margin δ around the classification boundary.

the set of real numbers, as given below:

$$g_{margin}(\alpha) = 1 \text{ if } \alpha \geq \delta$$

$$= 0 \text{ if } \alpha \leq -\delta$$

$$= \frac{0.5}{\delta}\alpha + 0.5 \text{ otherwise} \tag{12}$$

where α denotes $a^+ - a^-$. The Heaviside step function $g()$ (Figure 6(a)) is only used during the testing phase. This $g_{margin}()$ function enforces a margin δ around the classification boundary and as shown in Section 3.2.2, this margin improves the generalization performance similar to margin maximization in SVMs (Haykin, 1994).

We found that the choice of δ depends on the size of the neuron measured by the number of branches and the number of synapses on each branch. The parameter δ can also be varied to achieve a trade-off between the training and the generalization performance, as shown in Brader et al. (2007). A larger δ will give better generalization ability and reduce training performance while smaller values of δ will give optimal performance on the training set at the cost of reduced generalization performance. There-

16

fore, an adaptive method was used to set the value of δ by assigning an initial high value δ_0 and then reducing it gradually as discussed in Section 2.5.2.

Next, we propose a modification to the dendritic nonlinearity used in RM. This was motivated by the saturating nature of the $b()$ function and the fact that branch outputs can get saturated (Figure 5) leading to reduced classification performance of the model. This problem becomes more severe as the number of synaptic connections on each branch increases, which increases the average input to $b()$. Hence, we modify $b()$ so that a constant signal proportional to the mean synaptic activation of a branch is subtracted from the total excitatory synaptic activation of each branch. The constant signal is similar to the signal from the inhibitory population in the model proposed by Brader et al. (2007). The new dendritic nonlinearity $b_{leak}()$ is defined as:

$$b_{leak}(z_j) = g(z_j - z_{leak,j})\frac{(z_j - z_{leak,j})^2}{x_{thr}} \text{ if } b_{leak}(z_j) < b_{sat}$$

$$= b_{sat} \text{ otherwise} \tag{13}$$

where g is the Heaviside step function and $z_{leak,j}$ is the average synaptic activation on the jth branch corresponding to the initial random connections, given by $z_{leak,j} = P(x_i = 1) \times z_{syn,ij} \times k$, where $P(x_i = 1)$ is the probability that $x_i = 1$ and k is the number of synapses on every branch of the neuron. Since, we consider a neuron with all the branches having same number of synapses, therefore, average activation on a branch before learning, $z_{leak,j}$ is same for all branches. Hence, we drop the branch subscript j and denote $z_{leak,j}$ as z_{leak} in the rest of the paper. Here, z_{leak} can be regarded as an inhibitory signal to balance the excitation on each dendritic branch.

2.4 Theoretical Capacity Predictions

For both the RM and RMWM, we predict the theoretical capacities of the L- and NL-neurons by estimating the number of distinct input-output functions that can be expressed by these neurons as a function of the total number of synapses, s, and the number of distinct input lines, d. We wanted to see if the added flexibility provided by the nonlinear dendrites allows us to relax the resolution of weights used. To do this, we use binary weights such that $w_{ij} = 1$ if a connection exists and 0 otherwise. Therefore, the capacity of these neurons can be given by the combinatorial expressions to count the number of input-output mappings, i.e. all possible ways in which input lines can connect to synapses on dendrites resulting in distinct memory fields. These expressions were derived in Poirazi and Mel (2001) and are provided here for reference. For a L-neuron, it is given by the number of ways in which s synaptic sites can be drawn from d input lines with replacement, equal to $\binom{s+d-1}{s}$. The spatial location of the s synapses is not considered. For a NL-neuron, we first count the number of distinct branch functions available by drawing k synapses from d afferents with replacement, given by $f = \binom{k+d-1}{k}$ and then the number of ways in which m branches of a NL-neuron can be drawn from f possible functions, calculated as $\binom{f+m-1}{m}$, where total number of synapses, $s = m \times k$ is kept the same as that for L-neuron to allow direct comparison of capacities. The theoretical capacity calculated from these expressions is compared with the classification errors measured by training linear and nonlinear neurons on binary classification tasks as described in Section 3.1.1.

2.5   Learning Algorithm

In this section, we describe a learning rule which attempts to achieve classification performance in agreement with the theoretical capacity of a neuron, i.e. for a fixed number of synapses, minimum classification error is obtained when the predicted capacity is high. This learning rule is motivated by the feasibility of the equivalent hardware implementation. The goal of the learning process is to train the L- and NL-neurons so that the classifier output is $y = 1$ and $y = 0$ for input samples from the positive and negative classes respectively. A binary teacher signal, o ($= 1$ or 0) guides the learning process by indicating the desired output during the training phase. The learning process can be regarded as a combinatorial search problem of selectively forming synaptic connections on each dendritic subunit. This learning rule is a supervised or error modulated form of Hebbian learning, which is based on the change in the connection strength of a synapse as a function of the presynaptic and postsynaptic activities. However, our learning algorithm involves the formation and elimination of synapses instead of weight modification, since we use binary weights.

2.5.1   Learning Rule for RM

To guide the learning process, a fitness function c_{ij} is calculated for each synapse as the correlation between synaptic input and the output of the branch on which the synapse is located and modulated by the classification error. This correlation function is based on the fitness function used by Poirazi and Mel (2001), which was derived from a stochastic gradient descent learning rule. Since, (+) and (-) neurons learn to respond preferentially to patterns belonging to class (+) and (-) respectively, the learning for the two

neurons is complementary to each other. The fitness values for each pattern, Δc_{ij} are computed using the following equations:

$$\text{For (+) class neuron, for every pattern } \Delta c_{ij}^+ = x_{ij} b_j sgn(o - y) \tag{14}$$

$$\text{For (-) class neuron, for every pattern } \Delta c_{ij}^- = -x_{ij} b_j sgn(o - y) \tag{15}$$

where b_j denotes the output of the jth branch given by $b_j = b(z_j)$ and $sgn()$ is the signum function with a value of 1 for $o > y$, -1 for $o < y$ and 0 for $o = y$. Note that the error term $sgn(o - y)$ only modulates the sign of Δc_{ij}. The c_{ij} values are then calculated over an epoch as:

$$c_{ij}^+ = < \Delta c_{ij}^+ > \tag{16}$$

$$c_{ij}^- = < \Delta c_{ij}^- > \tag{17}$$

where $< . >$ indicates averaging over the entire training set.

The error term ensures that only wrongly classified patterns contribute to c_{ij}. This implies that the connection changes based on the input correlations, have minimal effect on the patterns already learned, which agrees with the basic adaptive learning principle of *minimal disturbance* (Widrow and Lehr, 1990). The connection changes are done by using the following logic: a synapse with the smallest c_{ij} value corresponds to a poorly-performing synapse and is a candidate for replacement. Since we operate with fixed synaptic resources, this synapse needs to be replaced by another one. To do this, a set of silent synapses are first formed on the chosen branch as possible candidates for the new connection. The silent synapses are randomly selected from the set of input lines in addition to the existing synapses and do not contribute to the dendritic activity. The silent synapse with the highest c_{ij} is chosen as the replacement. Hence, we modify the

20

grouping of connections on a dendrite guided by a Hebbian-like learning rule. This type of synaptic clustering has been observed experimentally in Kleindienst et al. (2011) showing activity-dependent clustering of synaptic inputs on developing hippocampal dendrites. Moreover, our calculation of c_{ij} can be implemented easily in hardware since it needs only local information while the global error signal is a binary value.

The input connections for a neuron with m branches and k synaptic contacts per branch were initialized by randomly selecting afferents from one of the d input lines with weight $w = 1$. We trained our model on P input pattern vectors $(\boldsymbol{x})$ belonging to (+) and (-) classes. The learning process comprises the following steps in every iteration:

(1) The activation of each dendrite, z_j is determined and then the sum of all branch responses is calculated as in equation (10) for both (+) and (-) neurons, giving $a^+(\boldsymbol{x})$ and $a^-(\boldsymbol{x})$ for all P patterns. The overall output of the classifier is computed according to equation (11).

(2) The error is calculated by averaging the absolute error $|o - y|$ over the entire batch of P patterns, which we describe as mean absolute error (MAE). Since in our case, o and y are binary-valued, the absolute error is always 1 for every misclassified pattern and hence, the MAE is equal to the fraction of misclassified patterns.

(3) A random set T consisting of n_T ($< s$) synapses is selected as candidates for replacement.

(4) The synapse with the lowest c_{ij} in T is replaced by the best-performing (maxi-

mum c_{ij}) synapse in a random replacement set R consisting of n_R ($< d$) synapses. The set R was created by placing n_R 'silent' synapses from d input lines on the branch with the lowest c_{ij} synapse. They do not contribute to the calculation in step (1).

(5) Synaptic connections are modified if the replacement led to either a reduction or no change in MAE. If the MAE increased with the replacement, a new replacement set R is created. If the MAE does not decrease after repeating this step n_{ch} ($= 100$) times, we assume a local minimum is encountered. We then do a replacement with the last choice of R, even if it increases MAE in an attempt to escape the local minimum. The connection matrix corresponding to the local minimum that gave the least error is saved.

(6) Steps (1) to (5) are repeated until either all the patterns have been memorized or n_{min} ($= 100$) local minima are encountered. At this point, the learning process is stopped. The connection patterns corresponding to the best minimum become the final learned synaptic connections of the neuron.

The fraction of synapses to be replaced in each iteration is similar to a learning rate parameter in the traditional weight modification algorithm. A minimum of 1 out of randomly selected n_T out of all s synapses is replaced in each iteration for smooth convergence to ideal topologies. The largest possible size (n_R) of the replacement set is equal to d, the dimension of the input. Using $n_R = d$ should give better optimization results but this increases computational cost. Therefore we chose n_R as a fraction of d which is a good trade-off between the computational cost and quality of the solution

to the optimization problem. Finally, the RM trained on binary input vectors is tested on spiking inputs using the equivalent spike-based model and the connection matrix after training. Here, we wanted to test how well our reduced abstract learning rule can transfer the learned performance to classify the original spike patterns or the noisy spike inputs when the temporal dynamics of a spike-based model are introduced.

2.5.2 Learning Rule for RMWM

The learning algorithm for the RMWM is the same except that the fitness values are calculated as:

$$\text{For (+) class neuron, for every pattern } \Delta c_{ij}^{+} = x_{ij} b_{leak,j} sgn(o - y) \qquad (18)$$

$$\text{For (-) class neuron, for every pattern } \Delta c_{ij}^{-} = -x_{ij} b_{leak,j} sgn(o - y) \qquad (19)$$

where, $y = g_{margin}(\alpha)$. The fitness values c_{ij} are calculated by taking the average of Δc_{ij} for all patterns as shown in equations (16) and (17). The value of margin δ for the $g_{margin}()$ function as given in equation (12) is determined using the following steps:

(1) Training is first performed using the $g()$ *function* and then the corresponding spike inputs are presented to the model.

(2) The values of α are recorded for all cases in which the network misclassified the spiking inputs.

(3) A margin of δ_0 given by the highest value of α for a wrongly classified pattern is introduced around the classification boundary to reduce the misclassified cases.

(4) δ value is reduced to 80% of δ_0 if the learning algorithm gets stuck in the same

local minimum for 5 consecutive times and is reduced to 80% of its present value everytime this condition is encountered.

After the completion of training, the testing is done in the same way as in the case of RM, on spike inputs using the spike-based model.

2.6 Spike-Time Based Learning

Our abstract models (RM and RMWM) have used the c_{ij} values to guide structural plasticity; however, no specific spike time based mechanism to achieve this computation has been shown so far. In other words, all learning is done on an abstract model while only the testing has involved spike trains. We present in this section a possible spike time based rule that enables online calculation of c_{ij} using an anti-Hebbian rule for excitatory synapses. Such "reverse STDP" (RSTDP) rules have also been widely reported in biological measurements (Letzkus et al., 2006; Froemke et al., 2005; Sjostrom and Hausser, 2006). Our spike-based learning rule is inspired from a recent study in which RSTDP in concert with hyperpolarization of the postsynaptic neuron or spike-frequency adaptation was used to realize variants of the perceptron learning rule (Albers et al., 2013; D'Souza et al., 2010) by modifying synaptic weights. The algorithm enables associative learning where a teacher signal is present during the learning process. In our case, we use a similar concept but in contrast to Albers et al. (2013), we do not modify the weights but instead calculate the change in c_{ij}. Another major difference in our work compared to Albers et al. (2013) is that the learning rule for a synapse is modulated by its dendritic branch activation level and is hence termed branch-specific. This concept of branch specific plasticity has also been observed to

a certain degree in biology (Makara et al., 2009; Losonczy et al., 2008) and has been used to model feature binding in a single neuron (Legenstein and Maass, 2011). Finally, there is a question of what physical variable in biology might represent c_{ij}? There is evidence that calcium concentration in spines is a correlation sensitive, spike-time dependent signal and has been implicated as a guide for structural plasticity (Helias et al., 2008). We compute the correlations in a branch-specific manner as discussed next in our branch-specific spike time dependent structural plasticity (BSTDSP) model.

2.6.1 BSTDSP: Model Details

The basic architecture is similar to the nonlinear model shown in Figure 2(b) consisting of (+) and (-) neurons, and the outputs of the neurons connected to a WTA circuit. The dynamics of the membrane potential $V_m(t)$ is as follows:

$$\tau_V \frac{dV_m}{dt} = (u - V_m) + I_{in}(t) \tag{20}$$

$$\tau_u \frac{du}{dt} = -u \tag{21}$$

$$\text{If } V_m \geq V_{thr}, V_m \rightarrow V_{reset};$$

$$u \rightarrow u_{reset}$$

$$u_{reset} = V_{reset} < 0$$

$$n_{spk} \rightarrow n_{spk} + 1$$

where u denotes a hyperpolarization variable which relaxes back to 0 with a time constant τ_u and is set to u_{reset} after a postsynaptic spike; $I_{in}(t)$ is the input current given by equations (3)-(5); and τ_V and τ_u are the time constants governing the fast and slow dynamics of the membrane voltage and hyperpolarization respectively. Note that the

differential equation for $V_m(t)$ in equation 2 only had one state variable while equation 20 has two. This allows us to separate the time scale for synaptic current based depolarization and post spike hyperpolarization i.e. recovery from hyperpolarization, which is much slower. It will be shown later in Section 2.6.2 that this hyperpolarization allows us to build a margin for classification. The output of the model is computed as:

$$y = g(n_{spk}^+ - n_{spk}^-) \tag{22}$$

The presynaptic spike train $s_{ij}(t)$ is given by equation (1) and the postsynaptic spike train is denoted by:

$$r(t) = \sum_{t_{post}} \delta(t - t_{post}) \tag{23}$$

where t_{post} is the postsynaptic spike time. These pre- and postsynaptic spikes also drive exponentially decaying memory traces–the presynaptic trace $\bar{s}$ and the postsynaptic trace $\bar{r}$ given by:

$$\bar{s} = exp(-t/\tau_{pre}) \tag{24}$$

$$\bar{r} = exp(-t/\tau_{post}) \tag{25}$$

2.6.2   BSTDSP: Learning Rule and Parameter Choice

To understand the learning process intuitively, let us consider the case of 2-class synchronous single spike inputs such that each afferent either fails to fire or fires a spike at a fixed time T_{syn}. During learning for $o = 1$, a teacher signal forcing a postsynaptic spike (at time $t = 1$ ms) is present for the (+) neuron and absent for the (-) neuron. For $o = 0$, vice versa conditions for the teacher signal exist.

Our structural plasticity rule used to calculate the correlation value c_{ij}, is analogous

to the RSTDP rule such that if a postsynaptic spike arrives before a presynaptic spike, then the correlation value for that synapse is increased. The condition for decreasing correlations does not use postsynaptic spike–instead, the relevant event is when the membrane voltage $V_m(t)$ crosses a subthreshold voltage V_{st} from below, where $0 < V_{st} < V_{thr}$. These subthreshold crossing events occurring at times t_{st} are denoted by $r_{st}(t) = \sum_{t_{st}} \delta(t - t_{st})$. Depression happens when presynaptic spike time t_{ij}^s occurs before t_{st}. The reason for using $r_{st}(t)$ instead of $r(t)$ is to enforce a margin as will be explained later in this section. These plasticity conditions are used in a branch-specific manner to compute the $\Delta c_{ij}(t)$ values for each pattern presentation, which are utilized to change connections on each branch. The learning rule can be written as:

For (+) class neuron, for every pattern $\Delta c_{ij}^+(t) = I_{b,out}^{j+}(t)\bar{r}^+(t)s_{ij}^+(t) - \gamma I_{b,out}^{j+}(t)\bar{s}_{ij}^+(t)r_{st}^+(t)$

$$(26)$$

For (-) class neuron, for every pattern $\Delta c_{ij}^-(t) = I_{b,out}^{j-}(t)\bar{r}^-(t)s_{ij}^-(t) - \gamma I_{b,out}^{j-}(t)\bar{s}_{ij}^-(t)r_{st}^-(t)$

$$(27)$$

where γ is a constant used to balance potentiation and depression and ensure that $c_{ij} = 0$ when a pattern is learned. The term $I_{b,out}^j(t)$ (from equation (4)) indicates branch specificity and is one major difference from (Albers et al., 2013). The correlation values averaged over all the patterns can be written as: $c_{ij}^+ = <\Delta c_{ij}^+(t)>$ and $c_{ij}^- = <\Delta c_{ij}^-(t)>$. The connection changes based on c_{ij} values are done as discussed in equations (14) and (15) in Section 2.5.1.
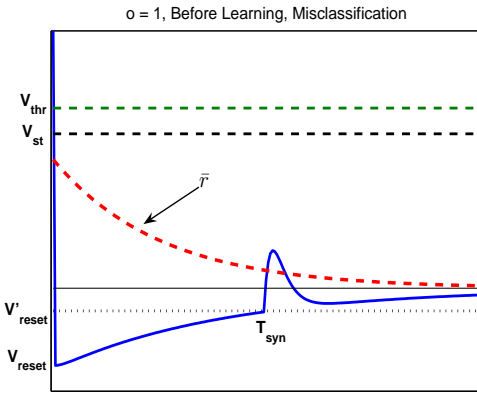
In order to understand how the c_{ij} values computed for the BSTDSP rule approximate the c_{ij} values of the RM learning (equations (14) and (15)), let us look at examples of all the cases – when the model predicts the correct or wrong output (y) for $o = 1$ or
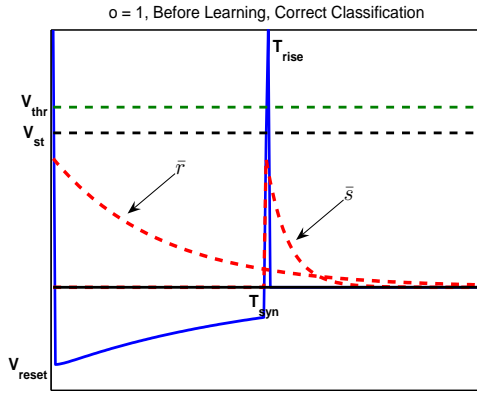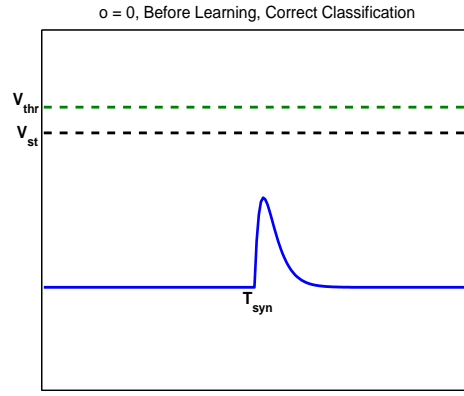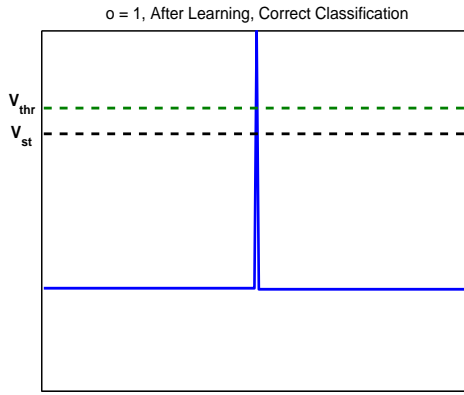
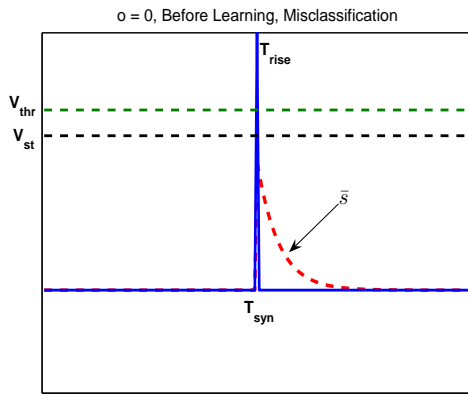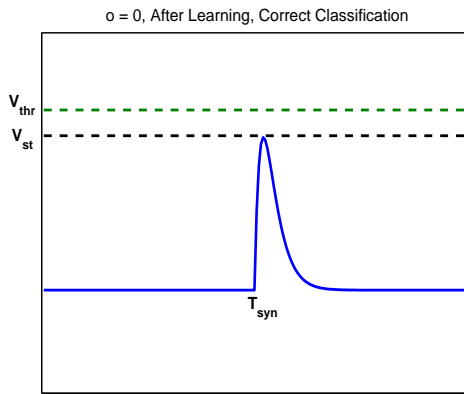Figure 7: Spike-based learning for patterns belonging to the two classes: membrane

voltage $(V_m(t))$ trace of the (+) neuron is shown for correctly classified patterns for

$o = 1$ (a) and $o = 0$ (b) and for misclassified patterns before learning (c and e) and

$o = 0$ class of input pattern. Figure 7 shows the membrane voltage $V_m(t)$ of the (+) neuron under these different conditions. We consider cases where $x_{ij} = 1$ since $\Delta c_{ij} = 0$ for both neurons when $x_{ij} = 0$. Further, we assume that there are n active synapses out of k synapses on the jth dendrite ($z_j = n$) for each of the cases considered. Therefore, for $o = 1$ and $y = 1$, the correlation values can be written as:

$$\Delta c_{ij}^+ = I_{b,out}^j(T_{syn})\bar{r}(T_{syn}) - \gamma I_{b,out}^j(T_{rise})\bar{s}_{ij}(T_{rise})$$

$$\Delta c_{ij}^- = 0 \tag{28}$$

where the presynaptic spike arrives at time T_{syn} and $V_m(t)$ crosses V_{st} at time T_{rise}; $I_{b,out}^j(T_{syn}) = b(z_j) = b(n)$. We assume that $\tau_{pre} \gg \tau_V$ which lets us write $I_{b,out}^j(T_{rise}) \approx I_{b,out}^j(T_{syn}) = b(n)$. Therefore, Δc_{ij} for this and other cases can be simplified as shown in Table 1.

We can equate the Δc_{ij} values of the two models for the correctly classified cases to compute the value of γ as given by:

$$\gamma = \frac{exp(-T_{syn}/\tau_{post})}{exp(-\bar{T}_{rise}/\tau_{pre})} \tag{29}$$

Here, we can estimate $\bar{T}_{rise}$ by taking the average of all the times at which $V_m(t)$ crosses V_{st} for different patterns. From the misclassified cases, we can write that:

$$[\Delta c_{ij}]_{BSTDSP} = exp(-T_{syn}/\tau_{post})[\Delta c_{ij}]_{RM} \tag{30}$$

Therefore, c_{ij} values for spike-based learning are smaller than the c_{ij} values calculated for the RM. Since connection changes are done on the basis of the ranking of c_{ij}, therefore equation (30) suggests that for the two models, the learning process comprising modifying connections should be the same for a given set of input patterns and for the

Table 1: Comparison of Δc_{ij} for the BSTDSP rule and RM.

Cases	BSTDSP		RM	
	Δc_{ij}^+	Δc_{ij}^-	Δc_{ij}^+	Δc_{ij}^-
$o=1, y=1$ (Figure 7(a))	$b(n)exp(-T_{syn}/\tau_{post})-\gamma b(n)exp(-T_{rise}/\tau_{pre})$	0	0	0
$o=0, y=0$ (Figure 7(b))	0	$b(n)exp(-T_{syn}/\tau_{post})-\gamma b(n)exp(-T_{rise}/\tau_{pre})$	0	0
$o=1, y=0$ (Figure 7(c))	$b(n)exp(-T_{syn}/\tau_{post})$	$-\gamma b(n)exp(-T_{rise}/\tau_{pre})$	$b(n)$	$-b(n)$
$o=0, y=1$ (Figure 7(e))	$-\gamma b(n)exp(-T_{rise}/\tau_{pre})$	$b(n)exp(-T_{syn}/\tau_{post})$	$-b(n)$	$b(n)$

same initial connections. This will remain true if the assumptions used to derive the above relationship are satisfied, i.e. $\tau_{pre} \gg \tau_V$ and that the estimate of T_{rise} is close to the actual value of T_{rise} for each pattern.

Figures 7(d) and 7(f) show the correct classification outputs after learning where teacher signals are not available. Hence for $o=1$, in the absence of initial hyperpolarization, $V_m(t)$ starts from 0 and can easily cross V_{thr} thereby providing robustness against noise. During learning, $V_m(t)$ is encouraged to cross a lower threshold V_{st},

which ensures a margin for learning class (+) patterns, given by:

$$\delta^+_{spike} = (V_{st} - V'_{reset}) - V_{thr} \tag{31}$$

where V'_{reset} is the voltage at the time T_{syn} when an initial postsynaptic spike is present. For class (-) patterns, $V_m(t)$ does not cross V_{thr} after learning while during learning $V_m(t)$ is reduced below V_{st} providing a desired margin of:

$$\delta^-_{spike} = (V_{thr} - V_{st}) \tag{32}$$

For an unbiased classifier, the margins for the two classes are equal. Using this condition, the values of V_{st} and V_{reset} can be found. The value of the desired margin $\delta_{spike} = \delta^{+/-}_{spike}$ is determined by using the spike equivalent value of margin, δ (Section 2.3) after learning. This is done by computing the difference in the membrane voltage of the (+) and (-) neurons (analogous to $a^+ - a^-$) when a single synapse is activated. This value multiplied by δ gives δ_{spike} since in the binary input case the synaptic strength is normalized to 1. Since δ is a function of m and k, we have also used different values of δ_{spike}, which entails changing the values of parameters V_{st} and V_{reset} for different neuron configurations.

3   Results

Firstly, we discuss the procedure for training the RM and RMWM, followed by the method used to test the performance of these models.

(1) Training on binary vectors: To demonstrate the learning process of the model, we used random patterns generated in a manner similar to Poirazi and Mel (2001).

31

The input patterns were chosen from a d_o ($= 40$) dimensional spherical Gaussian distribution. These patterns were randomly assigned to the positive ($o = 1$) and negative ($o = 0$) classes. The d_o-dimensional input patterns were mapped into a high d ($= 400$) dimensional space to incorporate sparse neural coding by using $n_{rf} = 10$ non-overlapping binary-valued receptive fields (RFs) along each dimension. Hence, we generated binary vectors $x \in \{0, 1\}^d$ such that the probability of any one RF neuron being active is 0.1, i.e. $P(x_i = 1) = p = 0.1 \ \forall i$. The binary patterns thus generated were used to train either the RM (Section 2.2) or the RMWM (Section 2.3). The connection matrix learned after training process is used for testing the model. This method has the advantage of shorter simulation times as compared to the learning on spike inputs. It can be used in practice as well (if the desired input patterns are known) where the learned connection matrix can be downloaded to a neuromorphic chip.

(2) Testing on noisy spike inputs: We have tested the noise sensitivity of the RM and RMWM on noisy spiking versions of the binary input vectors used to train the models. Henceforth testing the model will refer to testing the noise tolerance of the model except for the classification of benchmark datasets, where the generalization performance is evaluated. We introduce noise in the inputs by using Poisson spike trains or jittered single spikes. We used Poisson spike trains because we wanted to model a realistic case where our classifier receives inputs from a noisy spiking sensor or from biological neurons in the case of brain-machine interfaces. The two types of spiking inputs used for testing are generated as: 1) for rate encoded spike patterns, binary values of 1 and 0 are mapped to Poisson spike trains

32

with mean firing rates of f_{high} and f_{low} respectively, similar to Mitra et al. (2009) and 2) single spike patterns are generated by mapping binary input '$x_i = 1$' to a single spike arriving randomly within $[T_{syn} - \Delta/2, T_{syn} + \Delta/2]$ and '$x_i = 0$' to no spike, where T_{syn} is a fixed time within the stimulus duration T. The testing on spike patterns utilizes the spike-based model (Section 2.1) with the learned connection matrix.

The results obtained are organized as follows:

(1) Firstly, we discuss the classification performance of RM. These results were presented earlier in Hussain et al. (2013), where we showed that the performance of RM on noisy spike trains is much worse than the training performance on binary vectors. Hence, RM suffers from poor noise tolerance.

(2) Secondly, we show that the margin-based RMWM leads to improvements in both the training performance and the noise sensitivity.

(3) Next, the classification results from the BSTDSP rule are presented, where we also compare the performance of the abstract RMWM with that of our bio-realistic BSTDSP model.

(4) Finally, the performance of RMWM is compared against SVM and ELM methods on 3 different databases.

Simulation results are presented for $P = 500$, 700 and 1000 input patterns using $m = 10$, 20 and 50 dendrites and $k = 5$, 10, 15, 25 and 50 synapses on each dendrite, unless stated otherwise. The learning parameters are $n_R = n_T = 25$. The parameters

for testing on spike patterns using the spike-based model and the parameters for the BSTDSP model are summarized in Tables 2 and 3 respectively. These parameter values were chosen in accordance with the validity of the RM as discussed in Section 5.2. All the results are obtained by averaging across 5 simulation trials.

Table 2: Parameters for Spike-Based Model

f_{high}	f_{low}	τ_r	τ_f	I_0	V_{thr}	R	C	T
250 Hz	1 Hz	2 ms	8 ms	2.12	10 mV	10 MΩ	5 nF	200 ms

Table 3: Parameters for BSTDSP Model

τ_V	τ_u	τ_{pre}	τ_{post}	T_{syn}
4 ms	80 ms	10 ms	50 ms	100 ms

3.1 Performance of RM

First, we present the results of training the RM with binary vectors and then testing its robustness to noisy spike inputs.

3.1.1 Training Performance

(I) **Comparison of L- and NL-neurons**

In the first experiment, we compare the performance of L- and NL-neurons in a binary classification task. Both the classifiers were trained on $P = 500$ and 1000 input patterns and their performances were compared. As shown in Figure
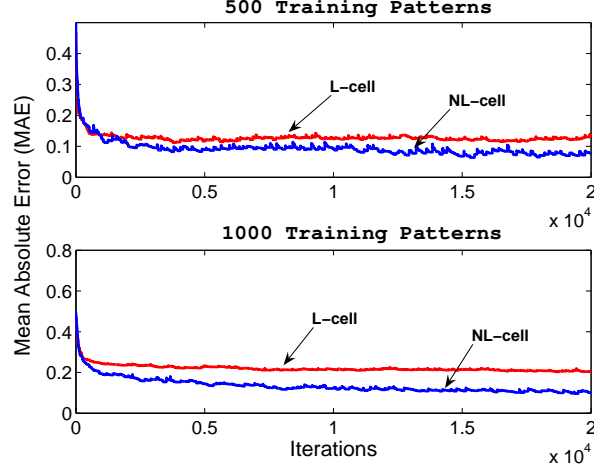
34

Figure 8: Classification error as measured by the MAE of the L-neuron and NL-neuron. The plot shows the MAE as function of learning iteration for 500 and 1000 training patterns, number of dendrites, $m = 20$.

8, the MAEs decrease for both neurons as learning progresses. However, the NL-neuron achieves a lower error than its linear counterpart and this difference in error grows as the classifier is trained on larger number of patterns as seen from the performance on 1000 versus 500 training patterns.

In order to understand the effect of learning on the connection matrix, we investigated whether the input connections learned by a neuron are able to capture the correlations present in the input patterns. If $\boldsymbol{x}_m^+$ and $\boldsymbol{x}_m^-$ denote the mth binary vector belonging to class (+) and (-) respectively, then $[(\boldsymbol{x}_m^+).(\boldsymbol{x}_m^+)^T]$ computes the second-order correlations present in the (+) pattern. $R = [\sum_{m=1}^{P^+}(\boldsymbol{x}_m^+).(\boldsymbol{x}_m^+)^T]/P^+ - [\sum_{m=1}^{P^-}(\boldsymbol{x}_m^-).(\boldsymbol{x}_m^-)^T]/P^-$ is a $d \times d$ matrix for d-dimensional patterns with d^2 entries of which the $d(d+1)/2$ values in the upper triangle and the diagonal are unique. P^+ and P^- denote the total number of patterns in class (+) and (-) re-
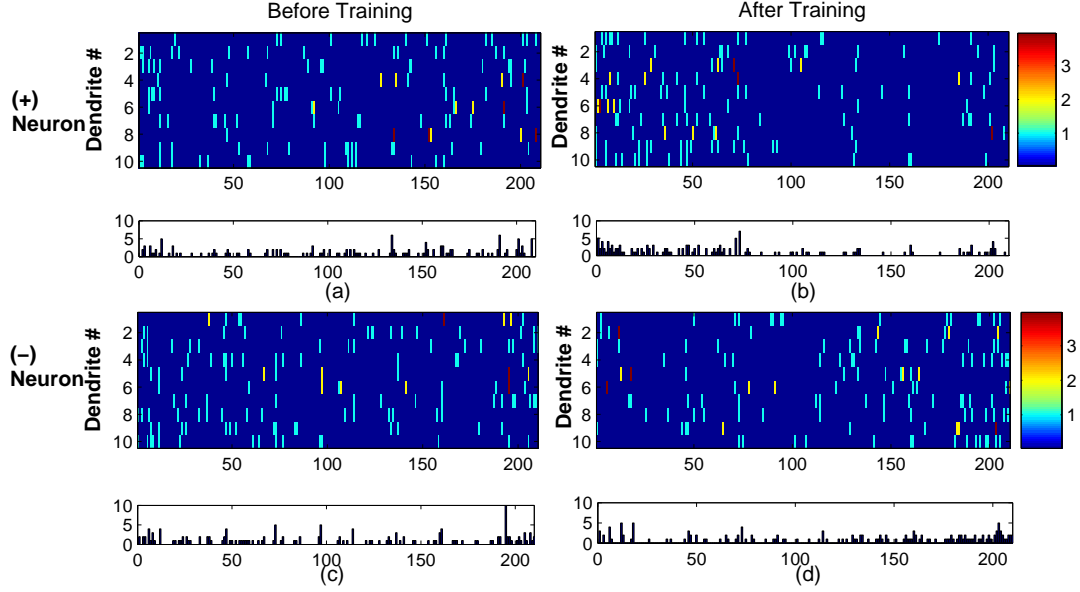
35

Figure 9: R_W values of the weight correlation matrix (x-axis) for (+) and (-) neurons with $m = 10$ dendrites (y-axis) before and after learning. The histograms plotted by summing the correlation values for each input pair for all the dendrites are shown at the bottom of each R_W matrix.

spectively. The entries in R can be ordered according to decreasing magnitude which provides a unique ranking $O = \{(r_1, c_1), (r_2, c_2), \ldots\}$ of the $d(d+1)/2$ entries in the upper triangle and the diagonal, where r_i and c_i denote row and column index respectively. Lower ranks correspond to high positive value entries in R, which denote $x_p x_q$ combinations that are specific to pattern class (+) and not in pattern class (-). Similarly, high ranks correspond to high negative value entries in R for those $x_p x_q$ combinations that uniquely define pattern class (-). Similarly, $[(\boldsymbol{W}_j^+).(\boldsymbol{W}_j^+)^T]$ gives the correlation matrix for the weights on a dendrite of the (+) neuron, where $\boldsymbol{W}_j^+$ is the d-dimensional weight vector of the jth dendrite, $\boldsymbol{W}_j^+ \in \mathbb{Z}_+^d$, where Z_+^d is a d-dimensional set of non-negative integers.

Figure 9 shows how the input correlations learned by the (+) and (-) neurons match with those present in the input patterns. In this example, neurons with $m = 10$ dendrites and $k = 5$ synapses per dendrite were trained to classify $P = 100$ binary patterns where $d = 20$. R_{Wj} is obtained by ordering the entries in the correlation matrix $[(\boldsymbol{W}_j).(\boldsymbol{W}_j)^T]$ according to O and is plotted for each dendrite of the (+) and (-) neurons before and after training. Each pixel shows the correlation value between a particular pair of input connections formed on a dendrite. For $d = 20$, the total number of unique $x_p x_q$ combinations is equal to 210, and hence x-axis ranges from $1 - 210$ and y-axis ranges from $1 - 10$ for 10 dendrites in each neuron. We have also shown the histograms obtained by plotting the sum of correlations for each connection pair for all the dendrites below each correlation matrix. Figures 9(a) and 9(c) show that the initial connections formed on the (+) and (-) neurons pick up the input correlations randomly, as evident from the correlation values evenly distributed along the x-axis. As seen in Figure 9(b), the (+) neuron has a much higher clustering of connections corresponding to the lower ranked $x_p x_q$ combinations unique to (+) patterns while it learns very few of the correlations $x_p x_q$ with high negative values (higher ranked) unique to the (-) patterns. Similarly, the clustered appearance toward the right end of x-axis in Figure 9(d) indicates that the (-) neuron learns to capture more of the correlations present in the (-) patterns and fewer of those in (+) patterns. Hence, the proposed learning algorithm enables the connection matrix to pick out unique correlations in the data. For the binary classifier consisting of a pair of neurons, we have shown that each neuron prefers patterns belonging to a particular class
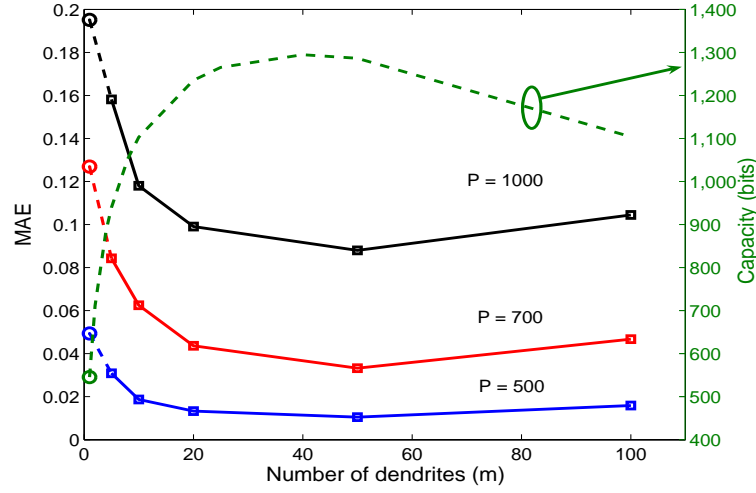
Figure 10: MAE as a function of the number of dendrites, m. The L-neuron (circles) has higher classification errors than its nonlinear counterpart (squares). The optimal choice of m for total number of synapses, $s = 200$ is $m = 50$. The dashed curve shows the theoretical capacity of the neuron for the same values of m and s.

by mimicking the input correlations present in that class.

(II) **Effect of number of dendrites**

Next, we tested the performance of the model as the number of dendrites, m is varied. The total number of synaptic resources was kept constant at $s = 200$. Figure 10 shows the MAE versus m. The circles represent the errors for the L-neuron with $m = 1$ and the NL-neuron errors are denoted by squares ($m > 1$). The performance of both types of neurons decreases with larger number of input patterns. However, the L-neuron performs much worse than the NL-neuron with classification errors of 19% against 9% for 1000 patterns. Further the NL-neuron errors decrease as the number of branches increases, which can be explained by the fact that as m increases, a larger number of dendritic functions can be selected from

the f possible functions obtained by drawing k synapses from d input lines (Section 2.4). These nonlinear branch functions can well approximate the nonlinear classification boundary of the problem as explained in the example in Section 2.2. As the number of dendrites continues to increase, the number of branch functions that can be drawn from f functions reduces if m increases beyond a certain value. As seen in Figure 10, the optimal value of m is 50 for $s = 200$. The theoretical capacity of the neuron is also plotted on the right-hand y-axis (dashed), calculated by taking the logarithm of the combinatorial expressions given in Section 2.4. As the capacity increases, more patterns can be learned and can be correctly classified. Hence, classification errors should decrease, which is in accordance with the empirical results. Similar to the measured classification errors, the theoretical capacity also peaks around $m = 50$ and further reduces as m increases. Therefore, as the number of dendrites grow, the classification performance of a NL-neuron with fixed synaptic resources (s) first increases to a peak value and then falls with further addition of dendrites.

(III) **Effect of number of synapses per dendrite**

We also determined the performance of NL-neuron as a function of number of synapses per branch (k) since it is a free parameter for an AER-based hardware system that can be easily adjusted. The reciprocal of theoretical capacity of a neuron as a function of k is plotted in Figure 11. It shows that for a neuron with a fixed number of dendrites, the inverse capacity decreases and therefore capacity increases as the number of synapses per dendrite is increased and as more dendrites are added, the neuron can learn more patterns. Figure 12 shows
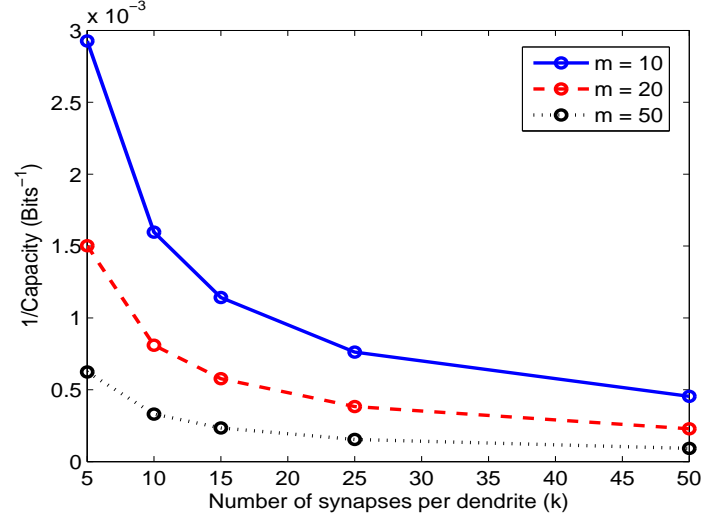
Figure 11: Reciprocal of capacity of a neuron as a function of number of synapses per branch (k) for varying number of branches (m). Capacity increases as the total number of synapses $m \times k$ increases.

classification errors measured empirically. The errors decrease as k increases for a fixed value of m, which is due to the presence of higher number of total synapses $s = m \times k$. The larger number of patterns are more difficult to train and hence lead to higher errors. As larger number of dendrites are used, the errors decrease further and for a fixed number of synapses per branch, for example $k = 25$, the classification error for 1000 patterns reduces from 11.2% for $m = 10$ by a factor of 2 for $m = 20$ and of 6.5 for $m = 50$. Therefore, the empirical results agree with the theoretical capacity predictions that is, capacity increases with higher number of synapses, more patterns can be correctly classified.
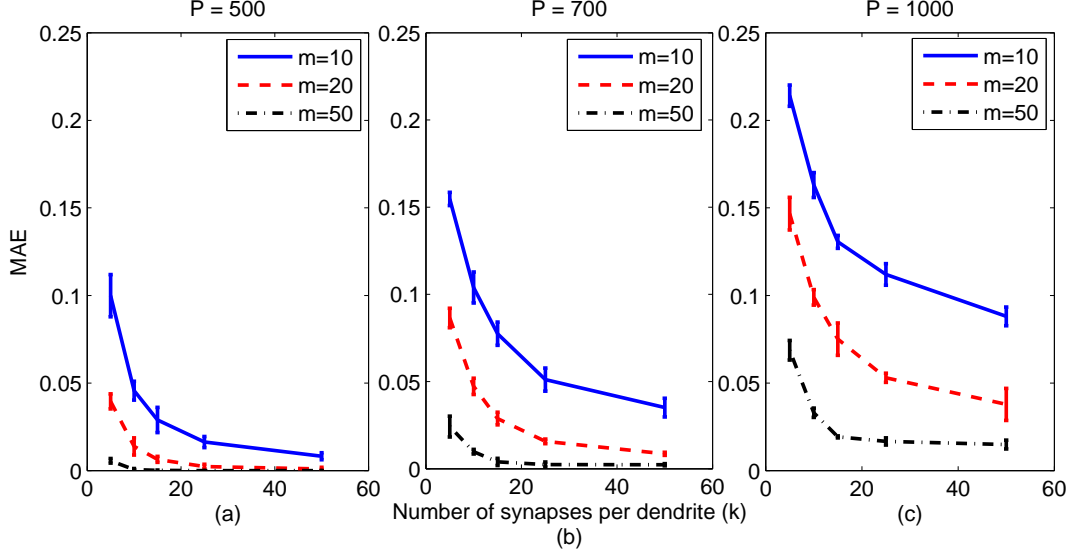
Figure 12: MAE as a function of k, the number of synapses per branch in a NL-neuron for binary inputs. 500, 700 and 1000 patterns were trained on neurons with $m = 10, 20$ and 50 branches. The errors decrease as the number of synapses increases.

3.1.2 Noise Sensitivity Performance

The spike-based model with the same connection matrix was used to classify spike patterns of mean firing rates. The classification performance of RM on spike patterns was measured as a function of k. Figure 13 shows that the classification errors for spiking inputs are much higher than that for the mean rate inputs on which the model was trained. It can be seen that the testing errors on noisy spike train inputs are $2 - 5$ times larger than that of the training errors for binary input vectors (Figure 12(c)). In contrast to the training errors decreasing with k, we observe that the testing errors increase as k is increased.

The increased errors can be attributed to the fact that as number of branches (m) increases, the linear summation of dendritic branch outputs leads to a very high average
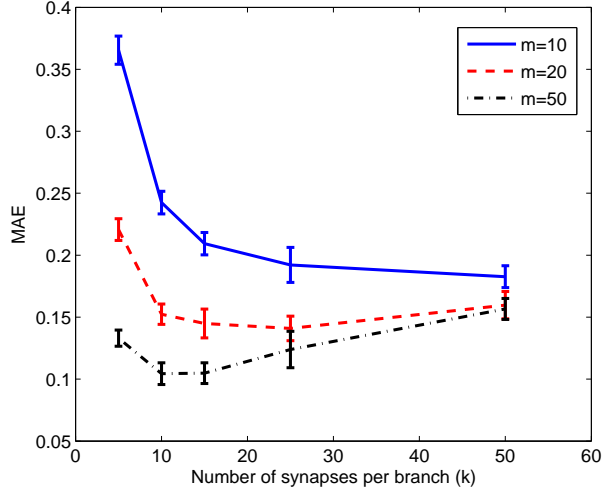
Figure 13: Testing performance for spike patterns as a function of k. The errors are much higher than that for the corresponding binary input vectors (Figure 12(c)).

value of the input currents I_{in}^{+} and I_{in}^{-} ($I_{in} = \sum_{j=1}^{m} I_{b,out}^{j} = \sum_{j=1}^{m} I_{b,in}^{2}/I_{thr}$) to the (+) and (-) neurons respectively. This causes the small differences between I_{in}^{+} and I_{in}^{-} to become indistinguishable for the (+) and (-) neurons due to the finite precision of the spike-based model used for testing as compared with the infinite precision of the abstract RM used for training. We also see from Figure 13 that the errors increase as the number of synapses per branch increases. This behaviour of the model can be explained by the fact that the use of higher dendritic threshold I_{thr} to prevent branch saturation at larger k reduces the I_{in}^{+} and I_{in}^{-} currents due to a spike pattern and therefore leads to decreased discriminability of the input currents to the spike-based (+) and (-) neurons. Next, we investigate how these issues are rectified using RMWM.

3.2 Performance of RMWM

Here, we present the results of training the RMWM with binary vectors and then testing its noise sensitivity.

3.2.1 Training Performance with $g_{margin}()$ function

We present the results of applying the method discussed in Section 2.5.2 to determine the value of δ for the $g_{margin}()$ function. The model was trained on $P = 500$ binary input patterns and then tested on the spike versions of these patterns. The α values for all the misclassified inputs were recorded. This experiment was repeated for NL-neurons with different number of branches and number of synapses per branch ranging from $k = 5$ to 50. The probability density distribution (pdf) of α as a function of number of synapses on each dendrite is shown in Figure 14 for $m = 20$ and three values of k. We can see that the peak of the pdf of α varies with both k and m (Effect of 'm' not shown in the Figure). Therefore, for a neuron with 20 branches and around $5 - 50$ synapses per branch, δ_0 can be set to around 20 and then gradually reduced during training. Similar δ values can be used for training the NL-neurons with the same values of m and k on a classification dataset consisting of binary input patterns.

Next we utilized the $g_{margin}()$ function (Figure 6(b)) to test if the performance of the classifier improves. The model was trained on 1000 patterns for different values of k. The $g_{margin}()$ function was used during the training process with $\delta_0 = 25$. After the training was completed, the memorized patterns were recalled using $g()$ function. As shown in the Figure 15, the classification errors reduce for all neuron configurations. As compared to training errors of RM (Figure 12(c)), the errors corresponding to $g_{margin}()$
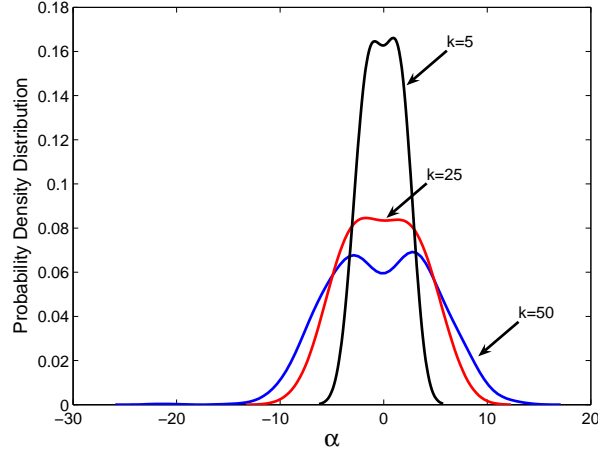
Figure 14: Determination of δ for $g_{margin}()$ function. The distribution of α for misclassified spiking inputs. $P = 500$ and $m = 20$.

are smaller by factors of $2-5$ and for large k values (> 25), the errors reduce drastically by factors of more than $10 - 20$. Hence, the improved classification performance is observed because the model is trained to generate a decision boundary with a margin δ. We will also present the results of improved noise tolerance achieved using the $g_{margin}()$ function.

3.2.2 Improving Noise Sensitivity

Here, we utilize the RMWM to improve the noise sensitivity by rectifying the problems discussed in Section 3.1.2 in a step-wise manner.

(a) In order to balance the effect of high average values of the input currents for large values of m, we adopted the following strategy. The current differences $(I_{in}^+ - I_{in}^-)$ and $(I_{in}^- - I_{in}^+)$ are used as inputs to the two neurons (Figure 16). As shown in Figure 17(a), the reduction in classification error is most significant, by a factor
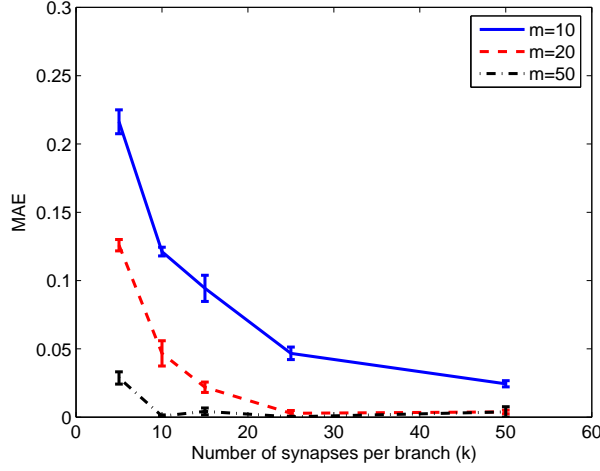
44

Figure 15: MAE as a function of k for the $g_{margin}()$ function. The training was done with δ_0 set to 25 and the input patterns were tested using $g()$ function. The performance improves for increasing k and m.
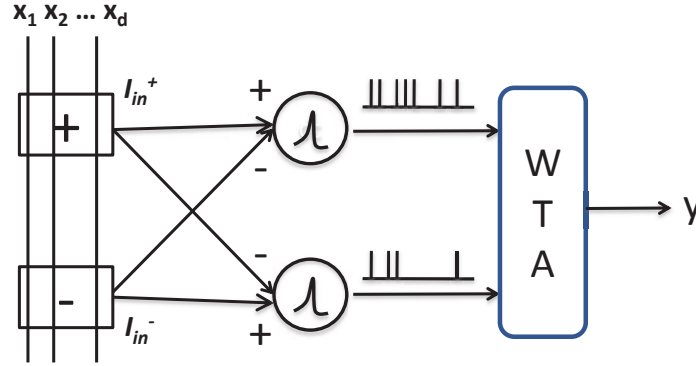
Figure 16: Architecture showing two neurons generating the output of the model in response to current differences $(I_{in}^{+} - I_{in}^{-})$ and $(I_{in}^{-} - I_{in}^{+})$.

of 1.2 for $m = 10$ as compared to when testing results were obtained without the differential input (Figure 13). The errors reduce for smaller k while for larger k, the performance still drops with increasing k. The dashed plots correspond to the training performance of the RMWM as shown in Figure 15. As described next,

45

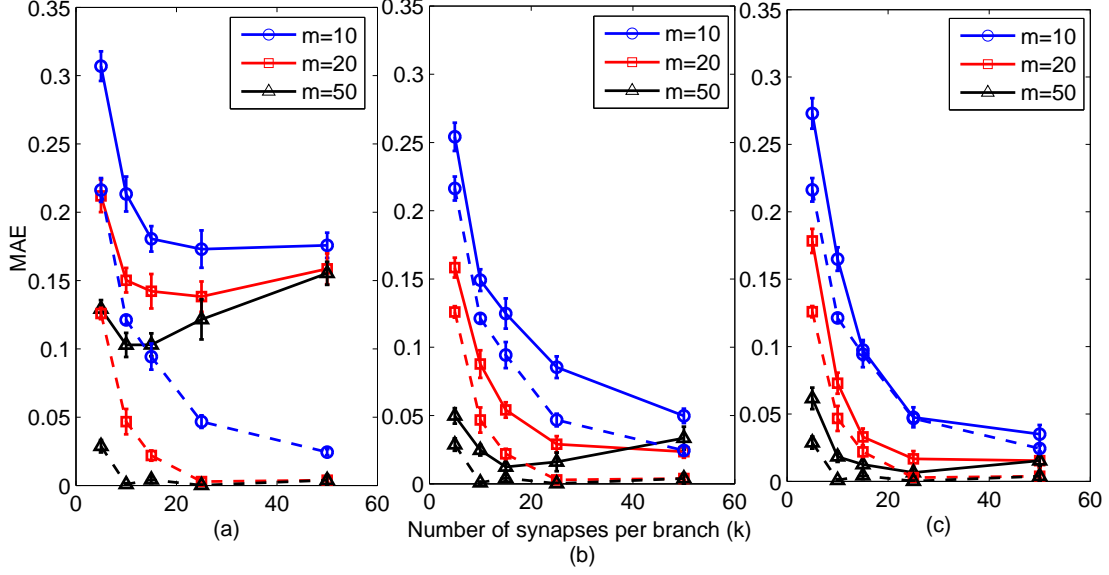Figure 17: Performance on spike patterns improves as (a) differences in I_{in}^{+} and I_{in}^{-} used as input currents for the integrate and fire neurons to counter the high 'm' effect; (b) $g_{margin}()$ used to improve the noise tolerance of the learning algorithm; and (c) inclusion of a dendritic nonlinear function $b_{leak}()$ improves the performance at larger k values. $P = 1000$, dashed plots correspond to the errors for binary vectors trained using the $g_{margin}()$ function.

the RMWM proposes to achieve noise tolerance to spiking inputs which is close to this optimal training performance obtained by the model.

(b) Next, we used the $g_{margin}()$ function to train the model on binary input vectors and then tested the performance on spiking inputs. This approach was used to improve the noise tolerance to the spike train versions. The errors computed using a NL-neuron are shown in Figure 17(b). As seen, the classification errors reduce for all combinations of m and k by factors of $1.5 - 6.5$ of that of errors obtained using RM and the performance levels match with the training performance (dashed

plots) more closely than in Figure 17(a). These results suggest that training the model with $g_{margin}()$ function improves the robustness to noise by introducing a margin around the classification boundary. As seen in Figure 17(b), for $m = 50$, the errors tend to rise with increasing k as observed in the earlier results. Next, we discuss the approach used to solve this problem.

(c) We test whether the use of the dendritic nonlinearity, $b_{leak}()$ defined in equation (13), improves the performance for spike patterns at larger k values. As shown in Figure 17(c), the errors at larger k ($15 - 50$) are smaller by factors of $1.3 - 2.5$ as compared to errors obtained with the nonlinearity $b()$ (Figure 17(b)). However, there is no significant change in the performance for smaller k values. This method further improves the test performance on noisy spiking inputs and the overall reduction in the classification errors for RMWM is by a factor of about $1.5 - 10$ of that of test errors for RM.

3.2.3 Performance on Single Spike Patterns

We also tested the performance of the model on patterns of single spikes. The input connections were learned using the RMWM on binary input vectors. The testing was done on 1000 single spike patterns arriving within a time window of $\Delta = 0.5\tau_f, \tau_f, 2\tau_f$ and $3\tau_f$, where τ_f is the fall time constant of the postsynaptic waveform (equation (6)) and $T_{syn} = 100$ ms. The results shown in Figure 18 suggest that the model generalizes well on single spike patterns with testing errors about $1.2 - 4$ times that of the training errors (dashed) when $\Delta \leq \tau_f$, where $\tau_f = 8$ ms. We found that this condition is satisfied for different values of $\tau_f = 8$, 16 and 24 ms, the results for $\tau_f = 16$ and 24
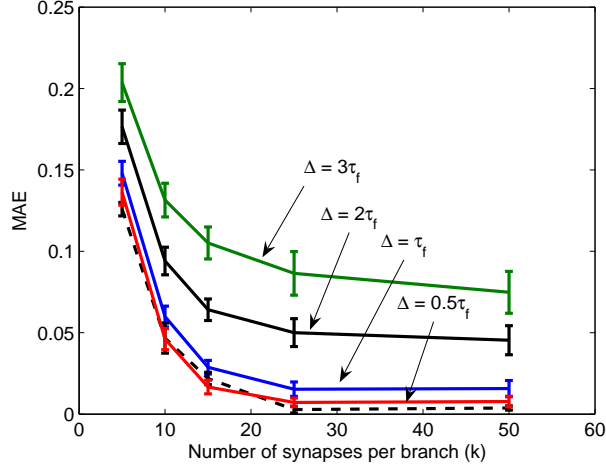
Figure 18: Testing performance for single spike patterns ($P = 1000$) as a function of k for $m = 20$. The training (dashed) errors are about 2% less than the testing errors for $\Delta \leq \tau_f$.

ms are not shown in Figure 18. As Δ increases, the testing errors increase because the spikes arriving at the synapses on the same dendrite are now more spread out in time leading to a reduced dendritic response as compared to the response learned with binary vectors.

3.3 Performance of BSTDSP Model

We have evaluated the performance of our BSTDSP rule discussed in Section 2.6 on a two-class classification of spike patterns. The architecture used here is similar to the two neurons model shown in Figure 16 where the input current into each neuron is the difference of currents with opposite signs. For training, we used random binary patterns of single spikes with $\Delta = 0$. In this case, we chose a low value of V_{thr} ($= 0.1$ mV) such that for the initial random connections, a neuron evokes a spike for about 65% of
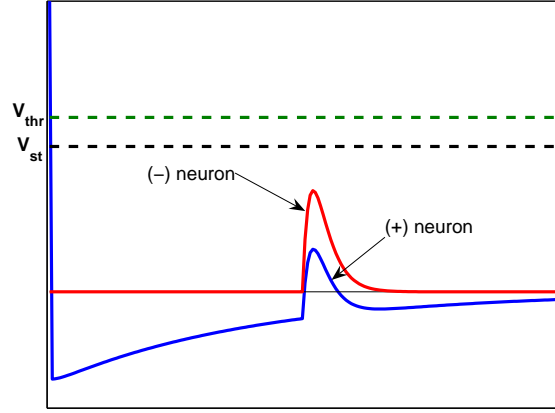
Figure 19: Membrane voltage $V_m(t)$ of (+) and (-) neurons when a class (+) pattern is presented. The pattern is not classified correctly since the (+) neuron fails to fire.

the input patterns. The reason for choosing a low V_{thr} value can be understood from Table 1 where for the RM, any misclassification leads to Δc_{ij} values (last two rows of Table 1) of opposite signs which contribute to the connection change in both (+) and (-) neurons after an epoch. This strategy of modifying both neurons limits the changes required in each neuron as compared to the case when only one neuron contributes at a time. Figure 19 shows a case where a (+) pattern is not classified correctly. However, only (+) neuron has non-zero Δc_{ij} (Δc_{ij}^{+} value for $o = 1$ and $y = 0$ in Table 1) while $\Delta c_{ij}^{-} = 0$ because the (-) neuron does not fire a spike due to high V_{thr}. The use of a lower V_{thr} will result in both Δc_{ij}^{+} and Δc_{ij}^{-} to be non-zero and of opposite signs.

The classification performance of the spike-based algorithm is shown in Figure 20. The training errors are plotted as a function of k for $P = 500$ patterns. The model was also tested on noisy versions of the patterns of single spikes by adding a uniform random noise within the time window $\Delta = \tau_f$ around the spike time T_{syn}. Figure 20(a)
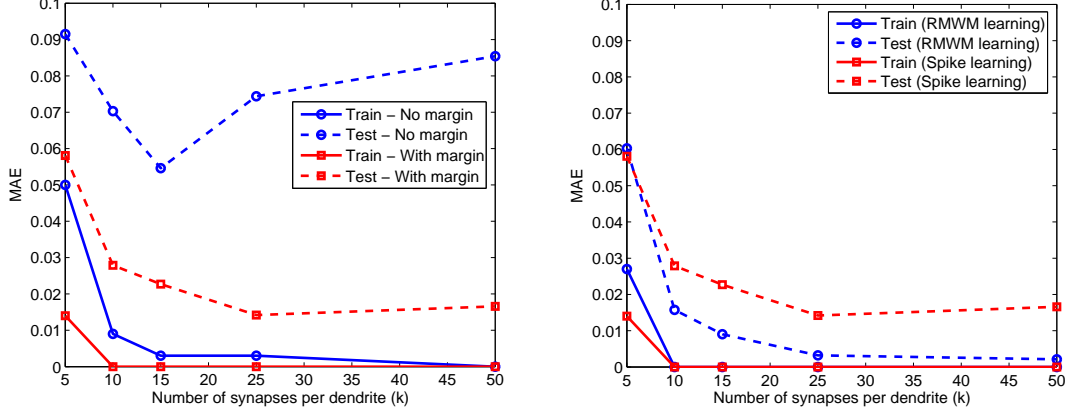
Figure 20: Classification performance of the BSTDSP algorithm. (a) Spike learning results when trained with (squares) and without (circles) margin. $P = 500$ patterns of single spikes, $m = 20$ (b) Comparison with RMWM learning (circles). The training errors are shown in solid and testing errors with dashed lines.

shows the performance when margin was not introduced, i.e. $\delta_{spike} = 0$ (circles) and when margin was introduced by using $\delta_{spike} > 0$ (squares), corresponding to different k values. We can see that both training and noise testing performances improve significantly as a result of margin-based learning, similar to the results obtained for RMWM learning (Section 3.2). The errors for noisy spike inputs with margin classifier reduce by up to a factor of 5 as compared to learning without margin.

We also compared margin-based learning using spike inputs with the RMWM learning on mean rate inputs as shown in Figure 20(b). The results show that the spike-timing based learning yields training errors (solid, squares) which are similar to that of training errors for RMWM (solid, circles). The test performance on noisy patterns of single spikes for the two instances of learning is shown with dashed plots. The test errors for spike learning are higher than the errors for RMWM learning by about 2.5 times and

this factor increases with k. These differences in the performance of BSTDSP from RMWM can be attributed to the differences in Δc_{ij} calculations for the two schemes, which are not exactly same. As discussed in Section 2.6.2, the value of γ as calculated in equation (29) is based on the assumption that $\tau_{pre} \gg \tau_V$ which is not perfectly true in our case. This also results in the estimate of T_{rise} used for calculating γ to be different from the actual T_{rise} for each pattern. Hence, the values of Δc_{ij} are slightly different from the ideal values as shown in Table 1. Another discrepancy between the learning for BSTDSP and RMWM can arise from the fact that the margin setting is not exact. As discussed in Section 2.6.2, the margin for spike-based learning was set using $\delta_{spike} = \eta\delta$ where a wrong estimate of the constant η will mean that δ_{spike} and δ are not truly analogous to each other, leading to different results. However, we can see that even with these differences in the learning for the two models, our proposed bio-realistic spike-based learning can achieve comparable errors with the more precise learning in the abstract model presented earlier.

3.4 Performance on Benchmark Datasets

Finally, we tested the performance of the proposed algorithm on real-world benchmark classification tasks and compared the results with the performance of the SVM and ELM methods. The binary datasets include datasets on breast cancer (BC), heart disease (HEART) and ionosphere (ION). These datasets (Table 4) are taken from the UCI Machine Learning Repository (Bache and Lichman, 2013). The performance of our improved RMWM with structural learning rule was evaluated for both binary vector patterns as well as spiking inputs, where each sample corresponding to a feature was

mapped into ten non-overlapping density-matched RFs, which were generated from the probability distribution of all samples of that particular feature. The resulting binary samples were then converted into spike patterns by using the method discussed in Section 3.

Table 4: Specifications of Benchmark Binary Datasets.

Datasets	# features	# training samples	# testing samples
BC	9	222	383
HEART	13	70	200
ION	34	100	251

The performance measures like number of neurons, number of weights and classification accuracy for different algorithms are given in Table 5 to compare the classification performance as a function of the computational complexity of the network used. The results for SVM and ELM are taken from Babu and Suresh (2013). The number of nonlinear dendrites of a neuron is analogous to the number of support vectors of the SVM and the number of hidden neurons for the ELM. The performance of different classifiers is also compared on the basis of synaptic resources used. The total number of weights in an N-input network (SVM and ELM) using M neurons is given by $N \times M + M$. The total number of weights in the dendritic model with m dendrites and k synapses per dendrite for each neuron in the classifier is given by $s = m \times k$. The classification accuracy obtained in this work for spike inputs is slightly less $(0.1 - 0.8\%)$ than that for binary input vectors. The reduced accuracy is probably due to the noise in

Table 5: Performance Comparison of SVM, ELM and Dendrite Algorithms on Benchmark Datasets.

Datasets	SVM			ELM			Dendrite Model			
	# neurons	# wts	Acc.	# neurons	# wts	Acc.	# dendrites	# wts	Acc. (Binary)	Acc. (Spike)
BC	24	240	96.61	66	660	96.35	20	204	96.01	95.93
HEART	42	588	75.5	36	504	76.5	10	104	75.3	74.53
ION	43	1505	91.24	32	1120	89.64	50	404	89.22	88.96

the Poisson spike trains and the effects of the integration and fire process of the neuron. As the results suggest the proposed model uses a similar number of dendritic processing units as the number of neurons used by SVM and ELM and achieves generalization performance within $1 - 2\%$ of that of these algorithms. Moreover, our method achieves this performance by utilizing $10 - 50\%$ fewer binary synapses, particularly for HEART and ION datasets, therefore rendering the model feasible for hardware implementation.

4 Discussion

Here, we discuss the biological and machine learning perspectives of models which compute a higher-order function of the inputs, particularly polynomial functions. We

also present neurobiological and computational evidences to suggest that these higher-order units can be mapped to active dendrites consisting of clustered coactive synapses and the fact that formation of such synaptic clusters is supported by the mechanism of structural plasticity. Finally, we compare the features of our model with that of other machine-learning classifier models based on similar concepts as ours.

4.1 Sum-of-Products: Biological Perspective

Our proposed model employs a static quadratic dendritic nonlinearity as approximation to the nonlinear dendritic processing due to voltage-gated ion channels. This simplification leads to an abstract neuron model in which the neuron output can be expressed as a sum of squaring subunit responses which can be further written as the sum of products of synaptic inputs on a dendritic subunit. The idea that the input-output behaviour of a complex dendritic tree can be reduced to a simple algebraic formula consisting of sum-of-squares is supported by an earlier compartmental model for disparity tuning in complex cells of primary visual cortex (Archie and Mel, 2000). This model consisting of simple-cell-like subunits mapped onto active dendrites, behaves like a hypothetical cell with squaring subunits. In another study, the firing rate of a complex biophysical pyramidal cell model was consistent with the mean rate predicted by an abstract two-layer neural network with sigmoidal subunits representing the thin terminal dendrites (Poirazi et al., 2003). Hence, these and other studies (Mel, 1991, 1993) suggest that from a biophysical perspective, a pyramidal cell with varying density, spatial distribution and kinetics of voltage-gated ion channels can be well approximated by an abstract model of sum of nonlinear subunits.

To investigate the particular form of nonlinear function which might represent the input-output behaviour of a dendrite, we look at the experimental study supporting the two-layer model of sum of subunits (Polsky et al., 2004). In this study confocal imaging was combined with focal synaptic stimulation to examine the rules of subthreshold synaptic integration in thin dendrites of pyramidal neurons. Figure 21 shows the measured and expected peak responses when two nearby sites on the same branch were stimulated simultaneously, where the expected response is the arithmetic sum of the responses to individual stimulations. The solid plot marked with circles corresponds to the measured responses. This curve has been reproduced from the data presented in Figure 4 in Polsky et al. (2004) for single-pulse stimulation. Therefore, experimental evidence suggests that the dendritic response to synaptic inputs on that branch is governed by a saturating expansive nonlinearity. In order to determine the form of subunit nonlinearity, we have fit polynomials to the experimental data because it is easier to implement polynomial functions in a hardware system. We have used the polynomials $y = \beta x^2$, $y = \beta x^3$ and $y = \beta x^5$ with saturating effects, to fit the data, where the constant β was chosen to get the best fit. As shown in Figure 21, the quadratic function provides the best fit as compared with degree 3 and degree 5 polynomials tested here. Hence, for the few data points obtained from biological measurements, we can consider the squaring function as a valid choice for the dendritic nonlinearity.

Further, we addressed this question from the perspective of performance in machine learning tasks. We tested the different dendritic polynomial nonlinearities used above in the random pattern binary classification task for our proposed model. As above, we compared the polynomials of degree 2, 3 and 5 under the conditions when saturation
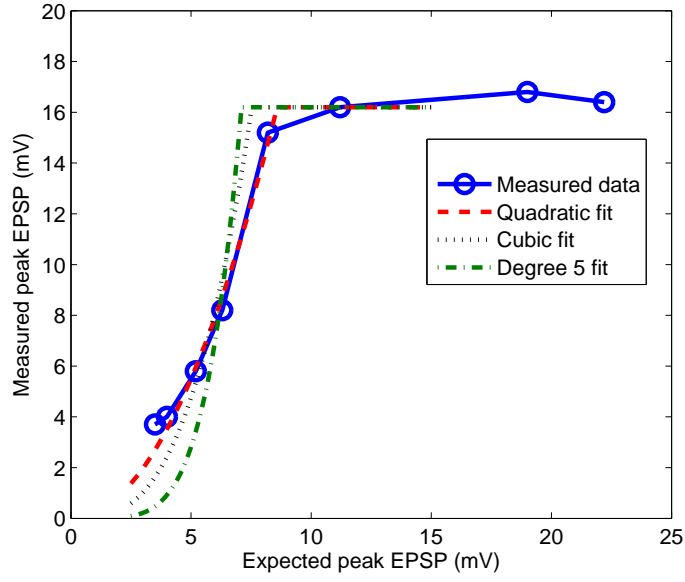
55

Figure 21: Expected vs actual peak voltage responses to stimulation of synaptic sites on the same dendrite (Reproduced from Figure 4 in Polsky et al. (2004)) shown with circles. Rest of the curves show polynomial fits to the experimental data.

is either enforced or not. The exact polynomial function used is given in the equation for $b_{leak}()$ (equation (13)), where the exponent is 2, 3 or 5. Moreover, the value of margin δ used in the $g_{margin}()$ function is determined for each form of nonlinearity used according to the method discussed in Section 2.5.2. The results shown in Figure 22(a) suggest that when there is no saturation on the nonlinearity, all three polynomials result in similar performance, that is improvement comes with more synapses on each dendrite. This indicates that the particular form of nonlinearity does not have any influence over the classification performance. This result can be explained by the fact that from the combinatorial expressions used to predict the storage capacity of the nonlinear neurons (Section 2.4), we can see that the purpose of branch nonlinearity is to partition the multiple dendrites into independent computational units and therefore the type
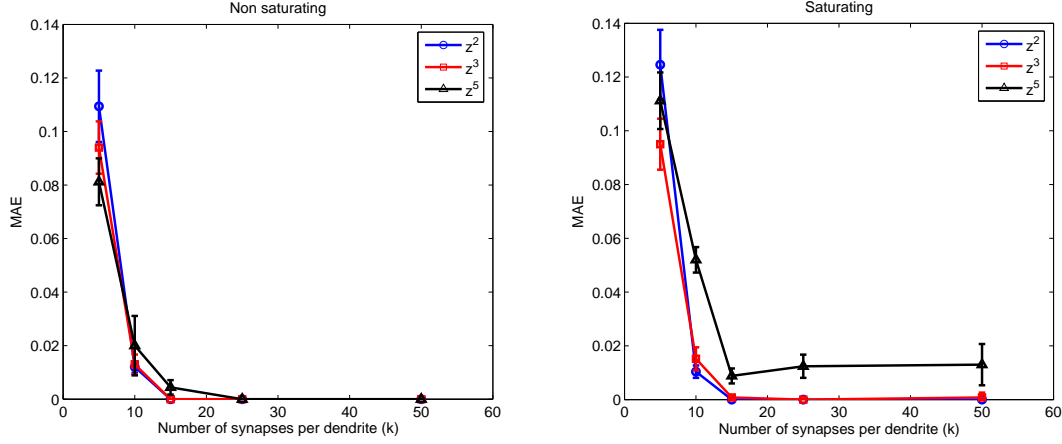
Figure 22: (a) Polynomial functions of different degrees used as branch nonlinearity $b()$. Saturation condition is not used here for learning $P = 500$ patterns (b) Saturating polynomial nonlinearities used as $b()$ function.

of nonlinearity used should not have an effect. Figure 22(b) shows that when a fixed saturation level is applied, lower degree polynomials (z^2, z^3) yield better classification performance than a higher degree polynomial (z^5), which is due to more branches getting saturated faster in case of a higher power branch function. Finally, our choice of quadratic nonlinearity is supported by easier implementation in hardware, biological plausibility and better performance in a classification task.

4.2   Sum-of-Products: Machine Learning Perspective

It was shown in Section 2.2 that our proposed classifier uses a discriminant function of higher-order terms to perform nonlinear mapping for classification. Several higher-order networks have been proposed to solve nonlinear classification problems. These models utilize higher-order combinations of some of the input components to solve the problem. A method to construct higher-order networks involves incremental tech-

nique in which the number of free parameters is gradually increased (Ghosh and Shin, 1992). A polynomial neural network utilizes this technique which is based on an evolutionary strategy of fitting a high-degree polynomial to the input components using a multilayered network (Misra et al., 2006). This approach employs polynomial theory (Ivakhnenko, 1971) to determine the polynomial description of optimum complexity which is necessary to achieve high prediction accuracies. Another method to generate higher-order functions of the input components involves a functional link network which estimates an unknown function as $\sum_i w_i \phi_i(\boldsymbol{x})$, where $\boldsymbol{x}$ is the input vector and $\phi_i(\boldsymbol{x})$ serve as basis functions such as products of inputs or sinusoidal functions. This method suffers from the need to use *a priori* knowledge in order to choose optimal basis functions (Ghosh and Shin, 1992).

A special case of this approach, when $\phi_i(\boldsymbol{x})$ are products of input components, is discussed next. This category includes several models which employ higher-order correlations among input components to perform nonlinear mapping. Such higher-order networks can be trained using fast learning schemes like Hebbian learning rule or perceptron type learning. A sigma-pi network is a type of high-order network consisting of a hidden layer which computes the product of inputs, followed by a summing unit (Rumelhart et al., 1986). The output of a "sigma-pi" unit is given by the following equation:

$$y = \sigma(w_0 + \sum_j w_j x_j + \sum_{j,k} w_{jk} x_j x_k + \sum_{j,k,l} w_{jkl} x_j x_k x_l + \cdots) \tag{33}$$

where $\sigma()$ is a sigmoid-like activation function, x_j is the jth component of input vector $\boldsymbol{x}$, $w_{jkl} \ldots$ is a weight which captures the correlation between product of input components x_j, x_k, x_l, $\cdots$ and the output of the unit; and w_0 is an adjustable threshold. The

drawback of higher-order networks consisting of such units is the combinatorial explosion of terms and hence weights needed to accommodate all possible 2^N correlation terms among N inputs, where input $\boldsymbol{x} \in R^N$. This exponential increase in the number of higher-order terms with the input dimension limits the applicability of the higher-order approach. Another higher-order network is the pi-sigma network (PSN) which computes the *products of sums* of input components instead of *sums of products* as in sigma-pi networks (Ghosh and Shin, 1992) and overcomes the problem of combinatorial increase in the number of weights. Durbin and Rumelhart (Durbin and Rumelhart, 1989) used product units which represent any generalized higher-order term and can learn which one to represent. These units can be incorporated in layered networks along with thresholded summing units.

4.3 Correlation-based Synaptic Clustering: Biological Relevance and Computational Modeling

Our hardware-friendly learning rule involves aggregation of correlated synapses on a dendrite, which is consistent with several experimental findings. These studies have demonstrated that clustering of synapses on dendritic branches might be a substrate for learning. This phenomenon suggests that single dendritic segments serve as functional subunits thereby expanding the neuronal properties significantly. This is in agreement with the predictions of the earlier computational studies. The mechanism of nonlinear dendritic integration leads to regenerative events including dendritic spikes, which have a stronger influence on neuronal firing than the activation of synapses located on different dendrites (Larkum and Nevian, 2008). This evidence suggests clustering

of coactive inputs within a dendritic subregion, which can lead to compartmentaliza-tion of various forms of signalling in dendrites, like electrical, biochemical and cel-lular (Branco and Hausser, 2010). Based on these results, it has been proposed that at the single cell level, dendritic branches act as basic functional units for long-term memory storage (Govindarajan et al., 2011). One such study has reported that long-term potentiation (LTP) at a single spine can modulate the threshold for plasticity at neighbouring spines (Harvey and Svoboda, 2007). This 'crosstalk' between plasticity at nearby synapses results in dendritic segments having similar properties, which could allow the memory traces to preferentially form at synapses clustered within these den-dritic subunits (Govindarajan et al., 2011). Gordon et al. (2006) showed that synapses innervating the proximal and distal regions of basal dendrites are governed by different rules for the induction of LTP. This domain-specific plasticity mechanism may support different learning functions.

The first demonstration that synaptic clustering might be a substrate for learning was given in the barn owl auditory system (McBride et al., 2008). The authors reported that behavioural experience drives changes in the clustering of axo-dendritic contacts. A related study has shown that new spines emerge in clusters along dendrites as a result of the repetitive activation of the cortical circuitry during motor learning (Fu et al., 2012). Synaptic clustering was also reported from a calcium imaging study which demon-strated that locally convergent inputs in spontaneously active networks lead to frequent synchronization of adjacent spines thereby facilitating dendritic compartmentalization (Takahashi et al., 2012). In addition to the structural clustering of synapses on dendrites, a novel form of plasticity at the dendritic level is provided by the excitability of the

dendritic branches. A study by Makara et al. (2009) showed that experience in enriched environments can lead to compartmentalized increase in dendro-somatic coupling. A recent study showed activity-dependent fine-tuning of synaptic inputs having temporally correlated activity in a developing neuron (Kleindienst et al., 2011). This pattern of connectivity was predicted by a computational model (Poirazi and Mel, 2001) based on theoretical capacity measures. This neuron model consisted of multiple dendritic subunits, which acted as independent computational units.

Several other computational models have hypothesized earlier that synapses on a dendrite are activated in clusters (Rumelhart et al., 1986; Mel and Koch, 1989; Mel, 1991, 1993). In a sigma-pi model, the multiplicative interactions among neighboring synaptic inputs could underlie local dendritic computations such that these local dendritic regions act as product units whose outputs are then summed at the soma (Mel and Koch, 1989). In this study, it was argued that the output of a cluster of synaptic inputs on a dendrite can be AND-like, i.e. selective for co-activation of all the inputs. Mel and his group later presented several computational studies to elucidate the role of dendrites which can be incorporated in neuron models rendering them computationally more powerful (Poirazi and Mel, 2001). A proposed abstract model neuron called the "clusteron" maps low-order polynomial functions of inputs onto a dendritic tree receiving synaptic contacts from a set of afferent axons (Mel, 1991). Later, it was hypothesized that a single neuron acts like a powerful multilayered network which performs nonlinear operations within spatially separated dendritic branches (Mel, 1994). In a more recent study, a two-layer dendrite model was proposed (Poirazi et al., 2003) in which different sets of inputs targeting functionally distinct dendrites are first processed

nonlinearly and then the dendritic outputs are linearly integrated at the soma.

4.4 Structural Plasticity: Biological Relevance

The learning process in our proposed neuron model with nonlinear dendrites involves formation and elimination of synapses to enable clustering of coactive synapses on a dendrite. Mel and Koch (1989) proposed that a tangle of axons within a dendrite's receptive field facilitates the formation of clusters of synapses on local regions of dendritic arbor. The process of constructing the optimal connection matrix involves alterations to the cortical 'wiring diagram', which requires structural plasticity at the axo-dendritic interface comprising synaptogenesis, axonal and dendritic growth and remodeling, and also retraction and reformation of dendritic spines (Chklovskii et al., 2004; Butz et al., 2009). Several experimental studies have provided evidence for the formation and elimination of synapses in the adult brain (Trachtenberg et al., 2002; Knott et al., 2006), including activity-dependent pruning of weaker synapses during early development (Be and Markram, 2006). Large-scale growth and remodeling of axonal and dendritic branches has been found to occur in adult brain within days (Darian-Smith and Gilbert, 1994). In vivo imaging studies have shown that synaptic rewiring mediated by re-routing of whole axonal branches to different postsynaptic targets takes place in the mature cortex (Stettler et al., 2006).

Based on the above studies, we can conclude that structural changes provide an alternative form of plasticity in addition to synaptic weight changes associated with a classical Hebbian scheme. However, these two modes of long-term memory might operate on different timescales, where changes in synaptic efficacy could happen in

seconds to hours while structural changes might operate over longer periods of days to months. The information storage capacity associated with structural plasticity lies in the ability to change wiring diagram in a sparsely connected network, which provides a large number of functionally distinct circuits available to encode information (Chklovskii et al., 2004).

4.5   Comparisons with Related Work

Some earlier studies have presented learning algorithms partly related to our work. Brader et al. (2007) used binary synapses and a stochastic spike-driven synaptic plasticity rule to train the network. Classification accuracy is obtained by a voting mechanism. The use of multiple output neurons for each class results in a high number of synapses. In comparison to this study, we use "sparsely" connected binary synapses and dendritic nonlinearity to get high accuracy. The sparsity of synapses helps in learning as well as reduces the number of synapses used.

Structural plasticity is used to modify binary synaptic connections on dendritic branches in Poirazi and Mel (2001). The authors also use silent synapses to search for the best possible connection, which can replace a poorly performing active synapse. Compared to this study, our method of calculating c_{ij} only needs values of inputs and dendritic branch outputs instead of derivatives as used by Poirazi and Mel (2001), rendering our model more feasible for hardware implementation. In their study, the learning rule involved $g'()$ and $b'()$ which are derivatives of $g()$ and $b()$ functions respectively, where $g()$ was a sharp sigmoid and $b()$ was a degree 10 polynomial, which is difficult to implement in hardware systems as compared to the quadratic function used in our work.

We compared the performance of our learning rule with that in Poirazi and Mel (2001) without the $g'()$ term since it is non-zero only in a small region around origin, resulting in only marginally misclassified patterns with non-zero fitness values contributing to connection changes. The results (not shown in the paper) suggest that using either $b()$ in our model or $b'()$ in Poirazi and Mel (2001) does not lead to a significant difference in the classification performance.

The unsupervised branch-specific STDP rule has been proposed by Legenstein and Maass (2011), where the weight change of a synapse depends on the local dendritic branch potential. This was shown to allow a neuron to solve the binding problem. The supervised BSTDSP rule we proposed also uses branch-specific spike-time based correlations but in contrast to Legenstein and Maass (2011), we use the correlation values to perform structural changes.

5   Conclusion and Future Work

This paper introduces two different learning schemes for spike-based representation in a nonlinear dendritic neuron model with the aim for implementation in neuromorphic hardware for which, we have (1) used a learning rule which is more amenable to hardware implementation; (2) tested the sensitivity of our models to noisy spike train inputs; (3) developed and tested a reduced model (RM) to minimize the training time and a margin-based learning method (RMWM) which led to a significant improvement in the noise sensitivity performance of our model; (4) proposed a branch-specific spike time dependent structural plasticity (BSTDSP) rule tying it to the well-known spike-

based learning protocols; and (5) finally, compared the generalization performance of our model with state-of-the-art nonlinear classifiers on real-world classification tasks and showed that our neurally-inspired model gives similar performance to these other methods.

In the future, we shall extend our spike-based algorithm which combines both structural learning with STDP, to incorporate precise spike timing information for learning both synaptic connections as well as the weights of these synapses on a dendritic branch. Also, we shall incorporate the effect of a specific spatial location of a synapse on the same dendritic branch by including location dependent delays as an extra tuning parameter (Hussain et al., 2012, 2014). We are also exploiting the silicon-friendly features of this algorithm to develop integrated circuits implementing the model described here for a spike-based smart sensor.

Acknowledgement

Appendix

5.1   Hardware Implementation

The classifier model presented is suited to VLSI implementation since it does not require high resolution weights; instead, it only requires the storage of a sparse connection matrix in memory. For applications where the statistics of the input are known, a con-

nectivity matrix obtained through offline training can be downloaded to the hardware. However, input statistics may not be known a priori in many cases, and the connections might have to be learned in the field. The c_{ij} values guiding this connection change will thus need to be computed on-chip and the current learning rule is designed to facilitate that. The dendritic output (b_j) can be computed in current mode using a translinear loop as shown in Hussain et al. (2013)–only one such circuit is needed per dendrite. Each synapse, implemented using a spike to current conversion circuit (Bartolozzi and Indiveri, 2007), needs to have an associated current mode multiplier to compute the product $x_{ij}b_j$, where x_{ij} is the current output of the synapse. The last part of the c_{ij} term can be implemented using switches controlled by digital circuits since $sgn(o - y)$ only takes 3 discrete values and both o and y are digital signals. To encode the 3 values of the sgn function, we need two digital wires such that 01, 10 and 00 represent the values -1, 1 and 0 respectively. These two wires can control two separate switches diverting the current encoding $x_{ij}b_j$ to a positive or a negative integrator for the 01 and 10 codes. Both switches are off for the 00 code thus not adding to the integral.

5.2   Validity of the Reduced Model

The RM and RMWM employ time-averaged synaptic activation, $z_{syn,ij}$ instead of the actual postsynaptic current generated by a synapse. This approximation can be used to describe the average current for the jth dendrite as in the following equation:

$$\tilde{I}_{b,out}^j = \left\{ \sum_i w_{ij} \left(\frac{1}{T} \int_0^T \left(\sum_{t_{ij}^s} K(t - t_{ij}^s) \right) dt \right) \right\}^2 / x_{thr} \tag{34}$$

where the dendritic nonlinearity is a quadratic function. However, the actual dendritic current is given by the following equation:

$$I_{b,out}^j(t) = \left\{ \sum_i w_{ij} \sum_{t_{ij}^s} K(t - t_{ij}^s) \right\}^2 / x_{thr} \tag{35}$$

Therefore, the reduced models are valid if the average dendritic current $\tilde{I}_{b,out}^j$ is equal to the average current computed using the spike-based model, i.e.

$$\tilde{I}_{b,out}^j = \frac{1}{T} \int_0^T I_{b,out}^j(t) dt$$

$$\Rightarrow \left\{ \sum_i w_{ij} (\frac{1}{T} \int_0^T (\sum_{t_{ij}^s} K(t - t_{ij}^s)) dt) \right\}^2 = \frac{1}{T} \int_0^T \left\{ \sum_i w_{ij} \sum_{t_{ij}^s} K(t - t_{ij}^s) \right\}^2 dt$$

$$\tag{36}$$

The equation (36) will hold true only if the postsynaptic current $\sum_{t_{ij}^s} K(t - t_{ij}^s)$ is a constant function of time. For the PSC waveform to be described by a constant function, both the frequency of a spike train arriving at a synapse (f_{high}) and synaptic current fall time constant (τ_f in equation (6)) are increased. Figure 23 shows that as $f_{high} \times \tau_f$ increases, the plot of average dendritic current calculated using the reduced models (y-axis) versus the actual values (x-axis) drifts toward the 'y=x' line, which indicates that the values predicted by the reduced model become equal to the actual values. The data points correspond to different number of synapses on a branch. These results have guided the choice of f_{high} and τ_f in this work.

References

Albers, C., M. Westkott, and K. Pawelzik (2013). Perfect associative learning with spike-timing-dependent plasticity. In C. Burges, L. Bottou, M. Welling, Z. Ghahra-
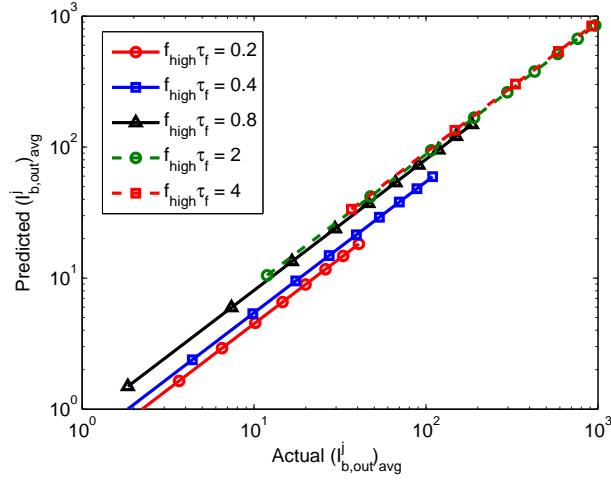
Figure 23: Average dendritic current predicted using the reduced model (y-axis) plotted against actual mean current (x-axis) for different values of $f_{high} \times \tau_f$ and different number of synapses on a dendrite.

mani, and K. Weinberger (Eds.), *Advances in Neural Information Processing Systems 26*, pp. 1709–1717. Curran Associates, Inc.

Archie, K. A. and B. W. Mel (2000). A model for intradendritic computation of binocular disparity. *Nature Neuroscience 3*(1), 54–63.

Arthur, J. and K. Boahen (2007). Synchrony in silicon: The gamma rhythm. *IEEE Transactions on Neural Networks 18*(6), 1815–1825.

Babu, G. and S. Suresh (2013). Sequential projection-based metacognitive learning in a radial basis function network for classification problems. *IEEE Transactions on Neural Networks and Learning Systems 24*(2), 194–206.

Bache, K. and M. Lichman (2013). UCI Machine Learning Repository. Internet : http://archive.ics.uci.edu/ml/.

Badoni, D., M. Giulioni, V. Dante, and P. D. Giudice (2006). An aVLSI recurrent network of spiking neurons with reconfigurable and plastic synapses. In *Proceedings of the International Symposium on Circuits and Systems*, pp. 1227–1230.

Bartolozzi, C. and G. Indiveri (2007). Synaptic dynamics in analog VLSI. *Neural Computation 19*(10), 2581–2603.

Bartolozzi, C. and G. Indiveri (2009). Selective attention in multi-chip address-event systems. *Sensors 9*, 5076–98.

Basu, A., S. Brink, C. Schlottmann, S. Ramakrishnan, C. Petre, S. Koziol, F. Baskaya, C. Twigg, and P. Hasler (2010). A floating-gate based field programmable analog array. *IEEE Journal of Solid-State Circuits 45*(9), 1781–94.

Basu, A., C. Petre, and P. Hasler (2008). Bifurcations in a silicon neuron. In *Proceedings of the International Symposium on Circuits and Systems*, pp. 428–31.

Be, J. V. L. and H. Markram (2006). Spontaneous and evoked synaptic rewiring in the neonatal neocortex. *Proceedings of the National Academy of Sciences of the United States of America 103*(35), 13214–13219.

Boahen, K. (2000). Point-to-point connectivity between neuromorphic chips using address events. *IEEE Transactions on Circuits and Systems II 47*(5), 416–34.

Brader, J., W. Senn, and S. Fusi (2007). Learning real-world stimuli in a neural network with spike-driven synaptic dynamics. *Neural Computation 19*, 2881–2912.

Branco, T. and M. Hausser (2010). The single dendritic branch as a fundamental functional unit in the nervous system. *Current Opinion in Neurobiology 20*, 494–502.

Butz, M., F. Worgotter, and A. V. Ooyen (2009). Activity-dependent structural plasticity. *Brain Research Reviews 60*(2), 287–305.

Chicca, E., D. Badoni, V. Dante, M. D'Andreagiovanni, G. Salina, L. Carota, S. Fusi, and P. D. Giudice (2003). A VLSI recurrent network of integrate-and-fire neurons connected by plastic synapses with long-term memory. *IEEE Transactions on Neural Networks 14*(5), 1297–1307.

Chklovskii, D. B., B. W. Mel, and K. Svoboda (2004). Cortical rewiring and information storage. *Nature 431*(7010), 782–788.

Choi, T., P. Merolla, J. Arthur, K. Boahen, and B. Shi (2005). Neuromorphic implementation of orientation hypercolumns. *IEEE Transactions on Circuits and Systems I 52*, 1049–60.

Darian-Smith, C. and C. D. Gilbert (1994). Axonal sprouting accompanies functional reorganization in adult cat striate cortex. *Nature 368*(6473), 737–740.

D'Souza, P., S.-C. Liu, and R. H. Hahnloser (2010). Perceptron learning rule derived from spike-frequency adaptation and spike-time-dependent plasticity. *Proceedings of the National Academy of Sciences of the United States of America 107*(10), 4722–4727.

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

Froemke, R., M. Poo, and Y. Dan (2005). Spike-timing-dependent synaptic plasticity depends on dendritic location. *Nature 434*, 221–225.

Fu, M., X. Yu, J. Lu, and Y. Zuo (2012). Repetitive motor learning induces coordinated formation of clustered dendritic spines in vivo. *Nature 483*, 92–95.

Gerstner, W. and W. Kistler (2002). *Spiking Neuron Models: Single neurons, Populations, Plasticity*. Cambridge University Press.

Ghosh, J. and Y. Shin (1992). Efficient higher-order neural networks for classification and function approximation. *International Journal of Neural Systems 3*(4), 323–350.

Giles, C. L. and T. Maxwell (1987). Learning, invariance, and generalization in high-order neural networks. *Applied Optics 26*(23), 4972–4978.

Gordon, U., A. Polsky, and J. Schiller (2006). Plasticity compartments in basal dendrites of neocortical pyramidal neurons. *The Journal of Neuroscience 26*(49), 12717–12726.

Govindarajan, A., I. Israely, S.-Y. Huang, and S. Tonegawa (2011). The dendritic branch is the preferred integrative unit for protein synthesis-dependent LTP. *Neuron 69*(1), 132–146.

Gutig, R. and H. Sompolinsky (2006). The tempotron: a neuron that learns spike timing-based decisions. *Nature Neuroscience 9*, 420–428.

Harvey, C. D. and K. Svoboda (2007). Locally dynamic synaptic learning rules in pyramidal neuron dendrites. *Nature 450*(7173), 1195–1200.

Hasler, J. and B. Marr (2013). Finding a roadmap to achieve large neuromorphic hardware systems. *Frontiers in Neuroscience 7*(118), 1–29.

Haykin, S. (1994). *Neural Networks: A Comprehensive Foundation*. MacMillan Publishing Company.

Helias, M., S. Rotter, M. Gewaltig, and M. Diesmann (2008, December). Structural plasticity controlled by calcium based correlation detection. *Frontiers in Computational Neuroscience 2*(7), 1–21.

Hussain, S., A. Basu, M. Wang, and T. Hamilton (2012). DELTRON: Neuromorphic architectures for delay based learning. In *IEEE Asia Pacific Conference on Circuits and Systems*, pp. 304–307.

Hussain, S., A. Basu, M. Wang, and T. Hamilton (2014). Delay learning architectures for memory and classification. *Neurocomputing*.

Hussain, S., R. Gopalakrishnan, A. Basu, and S.-C. Liu (2013). Morphological learning: increased memory capacity of neuromorphic systems with binary synapses exploiting AER based reconfiguration. In *Proceedings of the International Joint Conference on Neural Networks*.

Hynna, K. and K. Boahen (2006). Neuronal ion-channel dynamics in silicon. In *Proceedings of the International Symposium on Circuits and Systems*, pp. 3614–3617.

Ivakhnenko, A. G. (1971). Polynomial theory of complex systems. *IEEE Transactions on Systems Man and Cybernetics 1*(4), 364–378.

Kleindienst, T., J. Winnubst, C. Roth-Alpermann, T. Bonhoeffer, and C. Lohmann (2011). Activity-dependent clustering of functional synaptic inputs on developing hippocampal dendrites. *Neuron 72*, 1012–1024.

Knott, G. W., A. Holtmaat, L. Wilbrecht, E. Welker, and K. Svoboda (2006). Spine growth precedes synapse formation in the adult neocortex in vivo. *Nature Neuroscience 9*(9), 1117–1124.

Larkum, M. E. and T. Nevian (2008). Synaptic clustering by dendritic signalling mechanisms. *Current Opinion in Neurobiology 18*, 321–331.

Lazzaro, J., S. Ryckebusch, M. Mahowald, and C. Mead (1988). Winner-take-all networks of O(N) complexity. In *Proceedings of the Neural Information Processing Systems*, pp. 703–11. The MIT Press.

Legenstein, R. and W. Maass (2011). Branch-specific plasticity enables self-organization of nonlinear computation in single neurons. *The Journal of Neuroscience 31*(20), 10787–10802.

Letzkus, J., B. Kampa, and G. Stuart (2006). Learning rules for spike timing-dependent plasticity depend on dendritic synapse location. *The Journal of Neuroscience 26*(41), 10420–9.

Lichtsteiner, P., C. Posch, and T. Delbruck (2008). A 128x128 120 dB 15us latency asynchronous temporal contrast vision sensor. *IEEE Journal of Solid-State Circuits 43*(2), 566–76.

Liu, S.-C. and K. Boahen (1996). Adaptive retina with center-surround receptive field.

In *Proceedings of the Neural Information Processing Systems*, pp. 678–84. The MIT Press.

Liu, S.-C., G. Indiveri, T. Delbruck, and R. Douglas (2002). *Analog VLSI: Circuits and Principles*. Bradford Book, MIT Press, ISBN No. 0-262-12255-3.

Liu, S.-C., J. Kramer, G. Indiveri, T. Delbruck, T. Burg, and R. J. Douglas (2001). Orientation-selective aVLSI spiking neurons. *Neural Networks 14*, 629–43.

Liu, S.-C., A. V. Schaik, B. A. Minch, and T. Delbruck (2013). Asynchronous binaural spatial audition sensor with 2 x 64 x 4 channel output. *IEEE Transactions on Biomedical Circuits and Systems 8*(4), 453 – 464.

Losonczy, A., J. Makara, and J. C. Magee (2008). Compartmentalized dendritic plasticity and input feature storage in neurons. *Nature 452*, 436–441.

Maass, W. and M. Schmitt (1999). On the complexity of learning for spiking neurons with temporal coding. *Information and Computation 153*, 26–46.

Mahowald, M. (1994). Analog VLSI chip for stereocorrespondence. In *Proceedings of the International Symposium on Circuits and Systems*, London, pp. 347–350.

Makara, J. K., A. Losonczy, Q. Wen, and J. C. Magee (2009). Experience-dependent compartmentalized dendritic plasticity in rat hippocampal CA1 pyramidal neurons. *Nature Neuroscience 12*(12), 1485–1487.

McBride, T. J., A. Rodriguez-Contreras, A. Trinh, R. Bailey, and W. M. DeBello (2008). Learning drives differential clustering of axodendritic contacts in the barn owl auditory system. *The Journal of Neuroscience 28*(27), 6960–6973.

Mead, C. (1990). Neuromorphic electronic systems. *IEEE Proceedings 78*(10), 1629–36.

Mel, B. W. (1991). The clusteron: Toward a simple abstraction for a complex neuron. In *Proceedings of the Neural Information Processing Systems*, pp. 35–42. Morgan Kaufman, San Mateo, CA.

Mel, B. W. (1993). Synaptic integration in an excitable dendritic tree. *Journal of Neurophysiology 70*(3), 1086–1101.

Mel, B. W. (1994). Information processing in dendritic trees. *Neural Computation 6*(6), 1031–1085.

Mel, B. W. and C. Koch (1989). Sigma-pi learning: On radial basis functions and cortical associative learning. In *Proceedings of the Neural Information Processing Systems*, pp. 474–481. Morgan Kaufman, San Mateo, CA.

Misra, B. B., S. C. Satapathy, B. N. Biswal, P. K. Dash, and G. Panda (2006). Pattern classification using polynomial neural network. In *IEEE Conference on Cybernetics and Intelligent Systems*.

Mitra, S., S. Fusi, and G. Indiveri (2009). Real-time classification of complex patterns using spike-based learning in neuromorphic VLSI. *IEEE Transactions on Biomedical Engineering 3*(1), 32–42.

Neftci, E., S. Das, B. Pedroni, K. Kreutz-Delgado, and G. Cauwenberghs (2014). Event-driven contrastive divergence for spiking neuromorphic systems. *Frontiers in Neuroscience 7*(272), 1–14.

Neil, D. and S.-C. Liu (2014). Minitaur, an event-driven FPGA-based spiking network accelerator. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems PP:(99) 1.*

Nicolelis, M. (2003). Brain-machine interfaces to restore motor function and probe neural circuits. *Nature Reviews Neuroscience 4*, 417–422.

O'Connor, D. H., G. M. Wittenberg, and S. S.-H. Wang (2005). Graded bidirectional synaptic plasticity is composed of switch-like unitary events. *Proceedings of the National Academy of Sciences of the United States of America 102*(27), 9679–9684.

O'Connor, P., D. Neil, S.-C. Liu, T. Delbruck, and M. Pfeiffer (2013). Real-time classification and sensor fusion with a spiking deep belief network. *Frontiers in Neuroscience 7*, 1–13.

Petersen, C. C., R. C. Malenka, R. A. Nicoll, and J. J. Hopfield (1998). All-or-none potentiation at CA3-CA1 synapses. *Proc. Natl. Acad. Sci. USA 95*(8), 4732–4737.

Poirazi, P., T. Brannon, and B. Mel (2003). Pyramidal neuron as two-layer neural network. *Neuron 37*(6), 989–999.

Poirazi, P. and B. Mel (2001). Impact of active dendrites and structural plasticity on the memory capacity of neural tissue. *Neuron 29*, 779–96.

Polsky, A., B. Mel, and J. Schiller (2004). Computational subunits in thin dendrites of pyramidal cells. *Nature Neuroscience 7*(6), 621–627.

Posch, C., D. Matolin, and R. Wohlgenannt (2011). A QVGA 143 dB dynamic range

frame-free PWM image sensor with lossless pixel-level video compression and time-domain CDS. *IEEE Journal of Solid-State Circuits 46*(1), 259–275.

Rumelhart, D. E., G. E. Hinton, and J. L. McClelland (1986). A general framework for parallel distributed processing. In D. E. Rumelhart and J. L. McClelland (Eds.), *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1*, pp. 45–76. MIT Press Cambridge, MA, USA.

Schmuker, M., T. Pfeil, and M. P. Nawrot (2014). A neuromorphic network for generic multivariate data classification. *Proceedings of the National Academy of Sciences of the United States of America 111*(6), 20812086.

Serrano-Gotarredona, R., M. Oster, P. Lichtsteiner, A. Linarez-Barranco, R. Paz-Vicente, F. Goméz-Rodríguez, L. C. nas Mesa, R. Berner, M. Rivas-Pérez, T. Delbruck, S.-C. Liu, R. Douglas, P. Hafliger, G. Jiménez-Moreno, A. Ball-cels, T. Serrano-Gotarredona, A. Acosta-Jiménez, and B. Linares-Barranco (2009). CAVIAR: A 45k neuron, 5M synapse, 12G connects/s AER hardware sensory-processing-learning-actuating system for high-speed visual object recognition and tracking. *IEEE Transactions on Neural Networks 20*(9), 1417–38.

Serrano-Gotarredona, T. and B. Linares-Barranco (2013). A 128x128 1.5 % contrast sensitivity 0.9 % FPN 3ms latency 4mW asynchronous frame-free dynamic vision sensor using transimpedance preamplifiers. *IEEE Journal of Solid-State Circuits 48*(3), 827–838.

Sjostrom, P. and M. Hausser (2006). A cooperative switch determines the sign of synap-

tic plasticity in distal dendrites of neocortical pyramidal neurons. *Neuron 51*, 227–238.

Stettler, D. D., H. Yamahachi, W. Li, W. Denk, and C. D. Gilbert (2006). Axons and synaptic boutons are highly dynamic in adult visual cortex. *Neuron 49*(6), 877–887.

Takahashi, N., K. Kitamura, N. Matsuo, M. Mayford, M. Kano, N. Matsuki, and Y. Ikegaya (2012). Locally synchronized synaptic inputs. *Science 335*(6066), 353–356.

Trachtenberg, J. T., B. E. Chen, G. W. Knott, G. Feng, J. R. Sanes, E. Welker, and K. Svoboda (2002). Long-term in vivo imaging of experience-dependent synaptic plasticity in adult cortex. *Nature 420*, 788–794.

Vogelstein, R., U. Mallik, E. Culurciello, G. Cauwenberghs, and R. Etienne-Cummings (2007). A multichip neuromorphic system for spike-based visual information processing. *Neural Computation 19*, 1–20.

Widrow, B. and M. Lehr (1990). 30 years of adaptive neural networks: Perceptron, madaline and backpropagation. *Proceedings of the IEEE 78*(9), 1415–1442.

Yuille, A. L. and D. Geiger (1995). Winner-take-all mechanisms. In M. Arbib (Ed.), *The Handbook of Brain Theory and Neural Networks*, pp. 1056–1060. MIT Press.