Neural Networks 122 (2020) 273–278

Contents lists available at ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet

Perceptrons from memristors

Francisco Silva [a,*], Mikel Sanz [b], João Seixas [a,c,d,e], Enrique Solano [b,f,g], Yasser Omar [a,c]

[a] Instituto de Telecomunicações, Physics of Information and Quantum Technologies Group, Portugal
[b] Department of Physical Chemistry, University of the Basque Country UPV/EHU, Apartado 644, E-48080 Bilbao, Spain
[c] Instituto Superior Técnico, Universidade de Lisboa, Portugal
[d] CeFEMA, Instituto Superior Técnico, Universidade de Lisboa, Portugal
[e] Laboratório de Instrumentação e Física Experimental de Partículas (LIP), Lisbon, Portugal
[f] IKERBASQUE, Basque Foundation for Science, Maria Diaz de Haro 3, 48013 Bilbao, Spain
[g] International Center of Quantum Artificial Intelligence for Science and Technology (QuArtist), and Department of Physics, Shanghai University, 200444 Shanghai, China

ARTICLE INFO

Article history:
Received 26 December 2018
Received in revised form 26 September 2019
Accepted 17 October 2019
Available online 2 November 2019

Keywords:
Perceptron
Memristor
Backpropagation
Delta rule
Neural network

ABSTRACT

Memristors, resistors with memory whose outputs depend on the history of their inputs, have been used with success in neuromorphic architectures, particularly as synapses and non-volatile memories. However, to the best of our knowledge, no model for a network in which both the synapses and the neurons are implemented using memristors has been proposed so far. In the present work we introduce models for single and multilayer perceptrons based exclusively on memristors. We adapt the delta rule to the memristor-based single-layer perceptron and the backpropagation algorithm to the memristor-based multilayer perceptron. Our results show that both perform as expected for perceptrons, including satisfying Minsky–Papert's theorem. As a consequence of the Universal Approximation Theorem, they also show that memristors are universal function approximators. By using memristors for both the neurons and the synapses, our models pave the way for novel memristor-based neural network architectures and algorithms. A neural network based on memristors could show advantages in terms of energy conservation and open up possibilities for other learning systems to be adapted to a memristor-based paradigm, both in the classical and quantum learning realms.

1. Introduction

The perceptron, introduced by Rosenblatt (1958), was one of the first models for supervised learning. More generally, artificial neural networks such as the multilayer perceptron (MLP) have proven extremely useful in solving a wide variety of problems (Devlin et al., 2014; Ercal, Chawla, Stoecker, Lee, & Moss, 1994; Rowley, Baluja, & Kanade, 1998), but they have thus far mostly been implemented in digital computers. This means that we are not profiting from some of the advantages that these networks could have over traditional computing paradigms, such as very low energy consumption and massive parallelization (Jain, Mao, & Mohiuddin, 1996). Keeping these advantages is, of course, of utmost interest, and this could be done if a physical neural network was used instead of a simulation on a digital computer. In order to construct such a network, a suitable building block must be found, with the memristor being a good candidate due to its memory and inherent non-linearity.

Besides these energy considerations, exploring the fact that MLPs are universal function approximators, i.e. they can approximate any smooth function to arbitrary accuracy (Cybenko, 1989), our proposal of MLPs based only on memristors implies that memristive circuits are also universal function approximators.

Memristive behaviours were observed as early as 1968 (Argall, 1968), but the first time the connection was made between such behaviours and the theoretical formulation of memristors was in 2008 at HP Labs (Strukov, Snider, Stewart, & Williams, 2008), which led to a new boom in memristor-related research (Prodromakis & Toumazou, 2010). In particular, there have been proposals of how memristors could be used in Hebbian learning systems (Pershin & Di Ventra, 2010), in the simulation of fluid-like integro-differential equations (Barrios, Retamal, Solano, & Sanz, 2019), in the construction of digital quantum computers (Pershin & Di Ventra, 2012) and of how they could be used to implement non-volatile memories (Ho, Huang, & Li, 2009).

The pinched current–voltage hysteresis loop inherent to memristors endows them with intrinsic memory capabilities, leading

* Corresponding author.
E-mail addresses: francisco.horta.ferreira.da.silva@tecnico.ulisboa.pt (F. Silva), mikel.sanz@ehu.eus (M. Sanz), joao.seixas@tecnico.ulisboa.pt (J. Seixas), enr.solano@gmail.com (E. Solano), yasser.omar@lx.it.pt (Y. Omar).

https://doi.org/10.1016/j.neunet.2019.10.013
0893-6080/

to the belief that they might be used as a building block in neural computing architectures (Traversa & Di Ventra, 2015; Yang, Strukov, & Stewart, 2013). Furthermore, the relatively small dimension of memristors, the fact that they can be laid out in a very dense manner and their non-volatile nature may lead to highly parallel, energy efficient neuromorphic hardware (Indiveri, Linares-Barranco, Legenstein, Deligeorgis, & Prodromakis, 2013; Jeong, Kim, Kim, Choi, & Hwang, 2016; Strachan, Torrezan, Medeiros-Ribeiro, & Williams, 2011; Taha, Hasan, Yakopcic, & McLean, 2013).

The possibility of using memristors as synapses in neural networks has been extensively studied (Adhikari, Yang, Kim, & Chua, 2012; Bayat, Prezioso, Chakrabarti, Kataeva, & Strukov, 2017; Demin et al., 2015; Duan, Hu, Dong, Wang, & Mazumder, 2015; Emelyanov et al., 2016; Hasan & Taha, 2014; Negrov et al., 2017; Prezioso et al., 2015; Soudry, Di Castro, Gal, Kolodny, & Kvatinsky, 2015; Wang, Duan, & Duan, 2013; Wen, Xie, Yan, Huang, & Zeng, 2018; Wu, Wen, & Zeng, 2012; Yakopcic & Taha, 2013), with two major architectures being put forward, one based on memristor crossbars and another on memristor arrays.

Despite all these results, and to the best of our knowledge, all existent proposals use memristors exclusively as synapses, with the networks' neurons being implemented by some other device. The main goal of this work is thus to introduce a memristor-based perceptron, i.e., a single-layer perceptron (SLP) in which both synapses and neurons are built from memristors. It will be generalized to a memristor-based multilayer perceptron (MLP) and we will also introduce learning rules for both perceptrons, based on the delta rule for the SLP, and on the backpropagation algorithm for the MLP.

Recently the universality of memristors has been studied for Boolean functions (Lehtonen, Poikonen, & Laiho, 2010) and as a memcomputing equivalent of a Universal Turing Machine (Universal Memcomputing Machine (Traversa & Di Ventra, 2015)). However, to the best of our knowledge, it has not yet been shown that the memristor is a universal function approximator. This result will come as a consequence of the introduction of the aforementioned memristor-based MLP.

Algorithm 1 Delta rule for Single-layer Perceptron

Initialization
Set the bias current I_b to 0.
Initialize the weights w_1, w_2, w_b.
Set the internal state variables γ_1, γ_2, γ_3 to w_1, w_2 and w_b, respectively.
for d in data **do**
Forward Pass
Compute the net input to the perceptron:

$$I = w_1 x_1 + w_2 x_2. \tag{2}$$

Compute the perceptron's output:

$$V = g(I, \gamma_1, \gamma_2, \gamma_3). \tag{3}$$

Backward Pass
Compute the difference Δ between the target output and the actual output:

$$\Delta = T - V. \tag{4}$$

Compute the derivative of the activation function with respect to the net input, g'.
for i in internal variables **do**
if $\Delta \geq 0$ **then**
Set the bias $I_b = I_{\gamma_i}$.
else
Set the bias $I_b = -I_{\gamma_i}$.
end if
Update γ_i by inputting $I = \Delta x_i g' + I_b$.
end for
Update the weights by setting them to the updated values of the internal state variables.
Set the bias $I_b = 0$.
end for

2. The memristor as a dynamical system

In general, a current-controlled memristor is a dynamical system whose evolution is described by the following pair of equations (Chua, 1971)

$$\begin{cases} V = R(\vec{\gamma}, I)I & \text{(a)}, \\ \dot{\vec{\gamma}} = \vec{f}(\vec{\gamma}, I) & \text{(b)}. \end{cases} \tag{1}$$

The first one is Ohm's law and relates the voltage output of the memristor V with the current input I through the memristance $R(\vec{\gamma}, I)$, which is a scalar function depending both on I and on the set of the memristor's internal variables $\vec{\gamma}$. This dependence of the memristance on the internal variables induces the memristor's output dependence on past inputs, i.e., this is the mechanism that endows the memristor with memory. The second equation describes the time-evolution of the memristor's internal variables by relating their time derivative, $\dot{\vec{\gamma}}$, to an n-dimensional vector function $\vec{f}(\vec{\gamma}, I)$, depending on both previous values of the internal variables and the input of the memristor.

2.1. Memristor-based single-layer perceptron

Our goal is to implement a perceptron and an adaptation of the delta rule to train it using only a memristor. To this end, we use the memristor's internal variables to store the SLP's weights and the learning rate, a hyperparameter controlling how much the weights of the network are adapted with respect to the gradient of the cost function. Eq. (1b) allows us to control the evolution of the memristor's internal variables and implement a learning rule. If, for example, we want to implement a SLP with two inputs we need a memristor with four internal variables, two of them to store the weights of the connections between the inputs and the SLP, a third one to store the SLP's bias weight and another for the learning rate. An equivalent memristor with four internal variables can be obtained by coupling memristors whose behaviours are described by fewer internal variables. An in-depth explanation of how this can be done is given in Barrios et al. (2019), particularly in section *Equivalent Memristors for Enhancing Simulations*.

Let us then consider a memristor with four internal state variables, from now on labelled by $\vec{\gamma} = (\gamma_1, \gamma_2, \gamma_3, \gamma_4)$. It could be difficult to externally control multiple internal variables. However, a possible solution is to use several memristors with the chosen requirements and with an externally controlled internal variable each.

In order to understand the form of these functions, we must remember that we expect different behaviours from the perceptron depending on the stage of the algorithm. In the forward propagation stage, the weights must remain constant to obtain the output for a given input. In this phase the internal variables must not change. On the other hand, in the backpropagation stage, we want to update the perceptron's weights by changing the internal variables. However, it may happen that the update is

different for each of the weights, so we need to be able to change only one of the internal variables without affecting the others.

There are thus three different possible scenarios in the backpropagation stage: we want to update γ_1, while γ_2 and γ_3 should not change; we want to update γ_2, while γ_1 and γ_3 should not change, and we want to update γ_3, while γ_1 and γ_2 should not change. To conciliate this with the fact that a memristor takes only one input, we propose the use of threshold-based functions, as well as a bias current I_b, for the evolution of the internal variables

$$V(t) = g(I, \gamma_1, \gamma_2, \gamma_3, \gamma_4), \tag{5}$$

$$\begin{aligned}\dot{\gamma}_i = &(I - I_b)\left(\theta(I - I_{\gamma_i}) - \theta\left(I - \left(I_{\gamma_i} + a\right)\right)\right)\\ &+ (I + I_b)\left(\theta(-I - I_{\gamma_i}) - \theta(-I - (I_{\gamma_i} + a))\right),\end{aligned} \tag{6}$$

where g is an activation function, θ is the Heaviside function, I_{γ_i} is the threshold for the internal variable γ_i and a is a parameter that determines the dimension of the threshold, i.e., the range of current values for which the internal variables are updated. The first term of the update function can only be non-zero if the input current is positive, whereas the second term can only be non-zero if the input current is negative, allowing us to both increase and decrease the values of the internal variables. If I_{γ_1}, I_{γ_2} and I_{γ_3} are sufficiently different from each other and from zero, we can reach the correct behaviour by choosing the memristor's input appropriately. The thresholds and the a parameter are thus hyperparameters that must be calibrated for each problem. In the aforementioned construction in which our memristor with three internal variables is constructed as an equivalent memristor, we can also use an external current or voltage control to keep the internal variable fixed. In fact, this is how it is usually addressed experimentally (Budhathoki, Sah, Adhikari, Kim, & Chua, 2013; Xia et al., 2009; Yang et al., 2013; Yu, Iu, Liang, Fernando, & Chua, 2015). Therefore, we can assume that this construction is possible. It is important to note that, in an experimental implementation, this threshold system does not need to be based on the input currents. It can, for instance, be based on the use of signals of different frequencies for each of the internal variables or in the codification of the signals meant for each of the internal variables in AC voltage signals.

We are now ready to present a learning algorithm for our SLP based on the delta rule, which is described in Algorithm 1. In case one wants to generalize this procedure to an arbitrary number of inputs n, this can be trivially achieved by using a memristor with $n + 2$ internal variables and adapting Algorithm 1 accordingly.

2.2. Memristor-based multilayer perceptron

In this model, memristors are used to emulate both the connections and the nodes of a MLP. In principle, the nodes could be emulated by non-linear resistors, but using memristors allows us to take advantage of their internal variable to implement a bias weight, which in some cases proves fundamental for a successful network training.

The equations describing the evolution of the memristor at each node in this model are the same as in the seminal HP Labs paper (Strukov et al., 2008). We have chosen the experimentally tested set

$$V(t) = \left(R_{\text{ON}}\frac{\gamma(t)}{D} + R_{\text{OFF}}\left(1 - \frac{\gamma(t)}{D}\right)\right) I(t), \tag{7}$$

$$\dot{\gamma} = \begin{cases} \mu_V \frac{R_{\text{ON}}}{D} I(t) - I_\gamma & \text{if } \mu_V \frac{R_{\text{ON}}}{D} I(t) > I_\gamma, \\ 0 & \text{o.w.} \end{cases} \tag{8}$$

Here, R_{ON} and R_{OFF} are, respectively, the doped and undoped resistances of the memristor, D and μ_V are physical memristor

Algorithm 2 Backpropagation for Multilayer Perceptron

INITIALIZATION
Set the bias current I_b to 0.
Initialize the weights $\{w_{ij}\}$ and $\{w_{b_k}\}$.
Set the internal variable γ_{ij} of each connection memristor ij to the respective connection weight w_{ij}.
Set the internal variable γ_k of each connection memristor k to the respective bias weight w_{b_k}.
for d in data **do**
FORWARD PASS
for l in layers **do**
Compute the output of each connection memristor ij in layer l:

$$V_{ij}(w_{ij}, I) = w_{ij} I. \tag{9}$$

Sum the outputs of the connection memristors connected to each node memristor k in layer l

$$\text{in}_k = \sum I_{ik} \tag{10}$$

Compute the node memristor's output:

$$V_k = R_{\text{OFF}}\left(1 - \frac{\gamma_{b_k}}{D} + \frac{R_{\text{ON}}}{R_{\text{OFF}}}\frac{\gamma_{b_k}}{D}\right)\text{in}_k.$$

end for
end for
for d in data **do**
BACKWARD PASS
for k in output layer **do**
Compute the difference Δ between the target output and the actual output of the node memristor:

$$\Delta_k = T_k - V_k. \tag{11}$$

Compute the local gradient of the node memristor using Eq. (19a).
end for
for layer in hidden layers **do**
for node in layer **do**
Compute the local gradient of node memristor l in layer using Eq. (19b).
end for
end for
for connection in connections **do**
Compute the weight update.
Set the bias current: $I_b = I_{\gamma_{ij}}$.
Update the connection memristor's internal variable by inputting $I = \Delta w_{ij} + I_b$ to it.
Update the connection's weight by setting it to the updated value of the respective internal variable.
end for
for node in nodes **do**
Compute the bias weight update according to Equation Eq. (20).
Set the bias current: $I_b = I_{\gamma_b}$.
Update the node memristor's internal variable by inputting $I = \Delta w_k + I_b$.
Update the bias weight by setting it to the updated value of the respective internal variable.
end for
end for

parameters, namely the thickness of its semiconductor film and its average ion mobility, and I_γ is a threshold current playing the same role as the $I_{\bar{\gamma}}$ in the model for the memristor-based SLP introduced above. Eq. (7) can be approximated by

$$V(t) = R_{\text{OFF}}\left(1 - \frac{\gamma(t)}{D}\right) I(t), \tag{12}$$

since we have that $\frac{R_{\text{ON}}}{R_{\text{OFF}}} \approx \frac{1}{100}$. If, for instance, we impose a constant current input $I_0 + I_\gamma$ to the memristor and we integrate Eq. (8), we get:

$$\gamma(t) = \mu_V \frac{R_{ON}}{D} I_0 t. \tag{13}$$

Subbing this equation back into Eq. (7):

$$V(t) = R_{OFF}\left(1 - \frac{\mu_V R_{ON}}{D^2} I_0 t\right) I_0 \tag{14}$$

For the memristor described in Strukov et al. (2008), we have $D \approx 10^{-8}$, $\mu_V \approx 10^{-14}$ and $R_{ON} \approx 1$, all in SI units. It thus follows that $\mu_V R_{ON}/D^2 \approx 100$, so we can approximate the above equation to obtain:

$$V(t) \approx \frac{\mu_V R_{ON} R_{OFF}}{D^2} I_0^2 t. \tag{15}$$

This finally leads us to the desired relation between voltage output and current input:

$$V(t) \propto -I^2 t. \tag{16}$$

It is then possible to implement non-linear activation functions starting from Eq. (7), which is an important condition for the universality of neural networks (Hornik, 1991).

In deriving this result, we have made some assumptions on the parameters describing the memristor. These assumptions are correct for the memristor presented in Strukov et al. (2008), but as we have discussed, there is an extensive literature on memristors, with a wide range of physical parameter values. However, one can couple memristors to obtain an effective or equivalent memristor with any parameters one desires. This can be seen in detail in Barrios et al. (2019), particularly in the section entitled *Equivalent memristors for enhancing simulations*. Taking this into account, it becomes clear that the assumptions we made do not jeopardize the applicability of our models.

Looking now at synaptic memristors, their evolution is described by

$$V(t) = \gamma(t) I(t), \tag{17}$$

$$\dot{\gamma} = \left(\mu_V \frac{R_{\text{ON}}}{D} I(t) - I_\gamma\right) \theta\left(\mu_V \frac{R_{\text{ON}}}{D} I(t) - I_\gamma\right). \tag{18}$$

In synaptic memristors, the internal variable γ is used to store the weight of the respective connection, whereas in node memristors the internal variable is used to store the node's bias weight.

As explained before, the node memristors are chosen to operate in a non-linear regime, which allows us to implement non-linear activation functions. On the other hand, we choose a linear regime for synaptic memristors, which allows us to emulate the multiplication of weights by signals. It is important to note that this does not imply that different memristors must be used. In fact, these different behaviours can be attained by coupling memristors, with the result being an equivalent composite memristor with another behaviour. This process is detailed in section *Equivalent Memristors For Enhancing Simulations* of Barrios et al. (2019).

It must be mentioned that Eq. (8) is only valid for $\gamma \in [0, D]$. If we were to store the network weights in the internal variables using only a rescaling constant A, i.e., $w = A\gamma$, then the weights would all have the same sign. Although convergence of the standard backpropagation algorithm is still possible in this case (Dickey & DeLaurentis, 1993), it is usually slower and more difficult, so it is convenient to redefine the variable (Strukov et al., 2008) $D \to D'$ so that the interval of the internal variable in which Eq. (8) is valid becomes $[-D'/2, D'2]$. Using a rescaling constant B, the network weights can then be in the interval $[-BD'/2, BD'/2]$.

The new learning algorithm is an adaptation of the backpropagation algorithm, chosen due to its widespread use and robustness. In our case, the activation function of the neurons is the function that relates the output of a node memristor with its input, as seen in Eq. (7). The local gradients of the output layer and hidden layer neurons are respectively given by:

$$\begin{cases} \text{Output: } \delta_k = T_k \phi'\left(\sum_i V_{ik}\right), & \text{(a)} \\ \text{Hidden: } \delta_k = \phi'\left(\sum_i V_{ik}\right) \sum_l \delta_l w_{kl}. & \text{(b)} \end{cases} \tag{19}$$

In Eq. (19a), T_k denotes the target output for neuron k in the output layer. In Eqs. (19a) and (19b), ϕ' is the derivative of the neuron's activation function with respect to the input to the neuron $\sum_i V_{ik}$. Finally, in Eq. (19b), the sum $\sum_l \delta_l w_{kl}$ is taken over the gradients of all neurons l in the layer to the right of the neuron that are connected to it by weights w_{kl}. The update to the bias weight of a node memristor is given by:

$$\Delta w_k = \eta \delta_k, \tag{20}$$

where η is the learning rate. The connection weight w_{ij} is updated using $\Delta w_{ij} = \eta \delta_j V_i$, where δ_j is the local gradient of the neuron to the right of the connection, and V_i is the output of the neuron to the left of the connection.

We count now with all necessary elements to adapt the backpropagation algorithm for our memristor-based MLP, as described in Algorithm 2.

3. Simulation results

In order to test the validity of our SLP and MLP, we tested their performance on three logical gates: OR, AND and XOR. The first two are simple problems which should be successfully learnt by SLP and MLP, whereas only the MLP should be able to learn the XOR gate, due to Minsky–Papert's theorem.

The Glorot weight initialization scheme (Glorot & Bengio, 2010) was used for all simulations, as it has been shown to bring faster convergence in some problems when compared to other initialization schemes. In this scheme the weights are initialized according to $\mathcal{U}(-1, 1)$, weighed by $\sqrt{\frac{6}{n_{in}+n_{out}}}$, where n_{in} and n_{out} are the number of neurons in the previous and following layers, respectively. The data sets used contain 100 randomly generated labelled elements, which were shuffled for each epoch, and the cost function is:

$$E = \frac{1}{2}(T - O)^2, \tag{21}$$

where T is the target output and O the actual output.

3.1. Single-layer perceptron simulation results

For the SLP, a learning rate of 0.1 was used for all tested gates, a value set by trial and error. The metric we used to evaluate the evolution of the network's performance on a given problem was its total error over an epoch, which is given by Eq. (22).

$$E_{\text{total}} = \sum_j E_j = \frac{1}{2} \sum_j (T_j - O_j)^2, \tag{22}$$

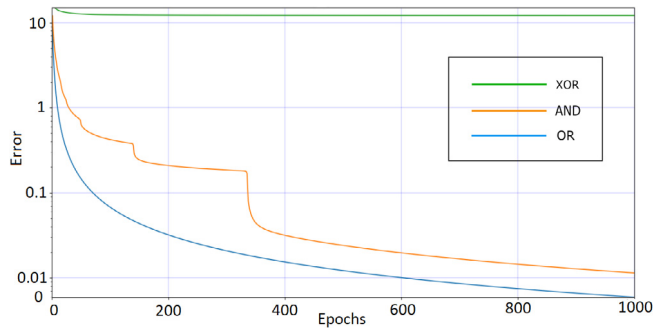

Fig. 1. Evolution of the learning progress of our single-layer perceptron (SLP), quantified by its total error, given by Eq. (22), for the OR, AND and XOR gates over 1000 epochs. The total error of our SLP for the OR and AND gates goes to 0 very quickly, indicating that our SLP successfully learns these gates. The same is not true for the XOR gate, which our SLP is incapable of learning, in accordance with Minsky–Papert's theorem (Minsky, Papert, & Bottou, 2017).

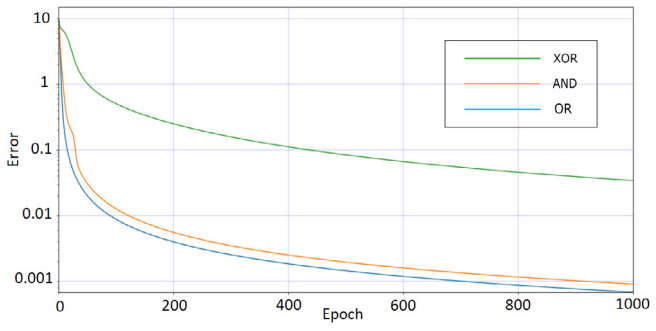

Fig. 2. Evolution of the learning progress of our multilayer perceptron (MLP), quantified by its total error, given by Eq. (22) for the OR, AND and XOR gates over 1000 epochs. As can be seen, the total error of our MLP for these gates approaches 0, indicating that it successfully learns all three gates.

where the sum is taken over all elements in the training set. In Fig. 1, the evolution of the total error over 1000 epochs, averaged over 100 different realizations of the starting weights, is plotted.

We observe that our SLP successfully learns the gates OR and AND, with the total error falling to 0 within 200 epochs, as expected from a SLP. However, the total error of our SLP for the XOR gate does not go to zero, which means that it is not able to learn this gate, in accordance with Minsky–Papert's theorem.

3.2. Multilayer perceptron simulation results

The structure of the network was chosen following Walczak and Cerpa (1999). There, a network with one hidden layer of two neurons is recommended for the case of two inputs and one output. As noted in Walczak and Cerpa (1999), networks with only one hidden layer are capable of approximating any function, although in some problems, adding extra hidden layers improves the performance. However, the results obtained by employing only one hidden layer are satisfactory, thus there is no need for a more complex network structure. There is also the matter of how many neurons must be employed in the hidden layer. In this case, there is a trade-off between speed of training and accuracy. A network with more neurons in the hidden layer counts with more free parameters, so it will be able to output a more accurate fit, but at the cost of a longer time required to train the network. A rule of thumb for choosing the number of neurons in the hidden layer is to start with an amount that is between the number of inputs and the number of outputs and adjust according to the results obtained. This leads to two neurons for the hidden layer and, similarly to what happened with the number of hidden layers, the results obtained using two neurons in the hidden layer are sufficiently accurate, so there was no need to try other structures. The learning rates used, which we have chosen through trial and error, are 0.1 for the OR and AND gates, and 0.01 for the XOR gate. In Fig. 2, the evolution of the total error over 1000 epochs, averaged over 100 different realizations of the starting weights, is plotted.

As was the case for our SLP, our MLP successfully learns the OR and AND gates. In fact, it is able to learn them faster than our SLP, which is a consequence of the larger number of free parameters. Additionally, it is able to learn the XOR gate, indicating that it behaves as well as a regular MLP.

In summary, both memristor-based perceptrons behave as expected. Our SLP is able to learn the OR and AND gates, but not the XOR gate, so it is limited to solving linearly separable problems, just as any other single-layer neural network. However, our MLP is not subject to such a limitation and it is able to learn all three gates.

4. Conclusion

In this paper, we introduced models for single and multilayer perceptrons based exclusively on memristors. We provided learning algorithms for both, based on the delta rule and on the backpropagation algorithm, respectively. Using a threshold-based system, our models are able to use the internal variables of memristors to store and update the perceptron's weights. We also ran simulations of both models, which revealed that they behaved as expected, and in accordance with Minsky–Papert's theorem. Our memristor-based perceptron models have the same capabilities of regular perceptrons, thus showing the feasibility and power of a neural network based exclusively on memristors.

To the best of our knowledge, our neural network models are the first ones in which memristors are used as both the neurons and the synapses. Due to the Universal Approximation Theorem for multilayer perceptrons, this implies that memristors are universal function approximators, i.e., they can approximate any smooth function $f : \mathbb{R}^n \to \mathbb{R}^m$ to arbitrary accuracy, which is a novel result in their characterization as devices for computation.

Our models also pave the way for novel neural network architectures and algorithms based on memristors. As previously discussed, such networks could show advantages in terms of energy optimization, allow for higher synaptic densities and open up possibilities for other learning systems to be adapted to a memristor-based paradigm, both in the classical and quantum learning realms. In particular, it would be interesting to try to extend these models to the quantum computing paradigm, using a recently proposed quantum memristor (Pfeiffer, Egusquiza, Di Ventra, Sanz, & Solano, 2016), and its implementation in quantum technologies, such as superconducting circuits (Salmilehto, Deppe, Di Ventra, Sanz, & Solano, 2017) or quantum photonics (Sanz, Lamata, & Solano, 2018).

Acknowledgements

Work by FS was supported in part by a New Talents in Quantum Technologies scholarship from the Calouste Gulbenkian Foundation. FS and YO thank the support from Fundação para a Ciência e a Tecnologia (Portugal), namely through programme POCH and projects UID/EEA/50008/2013 and IT/QuNet, as well as from the project TheBlinQC supported by the EU H2020 QuantERA ERA-NET Cofund in Quantum Technologies and by FCT, Portugal (QuantERA/0001/2017), from the JTF project NQuN (ID 60478),

and from the EU H2020 Quantum Flagship projects QIA (820445) and QMiCS (820505). MS and ES are grateful for the funding from Spanish MCIU/AEI/FEDER (PGC2018-095113-B-I00), Basque Government IT986-16, the projects QMiCS, Spain (820505) and OpenSuperQ, Spain (820363) of the EU Flagship on Quantum Technologies and the EU FET Open Grant Quromorphic (828826).

References

Adhikari, S. P., Yang, C., Kim, H., & Chua, L. O. (2012). Memristor bridge synapse-based neural network and its learning. *IEEE Transactions on Neural Networks and Learning Systems*, *23*(9), 1426–1435.

Argall, F. (1968). Switching phenomena in titanium oxide thin films. *Solid-State Electronics*, *11*(5), 535–541.

Barrios, G. A., Retamal, J. C., Solano, E., & Sanz, M. (2019). Analog simulator of integro-differential equations with classical memristors. *Scientific Reports*, *9*(1), 1–10.

Bayat, F. M., Prezioso, M., Chakrabarti, B., Kataeva, I., & Strukov, D. (2017). Memristor-based perceptron classifier: Increasing complexity and coping with imperfect hardware. In *Proceedings of the 36th international conference on computer-aided design* (pp. 549–554). IEEE Press.

Budhathoki, R. K., Sah, M. P., Adhikari, S. P., Kim, H., & Chua, L. (2013). Composite behavior of multiple memristor circuits. *IEEE Transactions on Circuits and Systems. I. Regular Papers*, *60*(10), 2688–2700.

Chua, L. (1971). Memristor-the missing circuit element. *IEEE Transactions on Circuit Theory*, *18*(5), 507–519.

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

Demin, V., Erokhin, V., Emelyanov, A., Battistoni, S., Baldi, G., Iannotta, S., et al. (2015). Hardware elementary perceptron based on polyaniline memristive devices. *Organic Electronics*, *25*, 16–20.

Devlin, J., Zbib, R., Huang, Z., Lamar, T., Schwartz, R., & Makhoul, J. (2014). Fast and robust neural network joint models for statistical machine translation. In *Proceedings of the 52nd annual meeting of the association for computational linguistics (volume 1: long papers)* (pp. 1370–1380).

Dickey, F., & DeLaurentis, J. (1993). Optical neural networks with unipolar weights. *Optics Communications*, *101*(5–6), 303–305.

Duan, S., Hu, X., Dong, Z., Wang, L., & Mazumder, P. (2015). Memristor-based cellular nonlinear/neural network: design, analysis, and applications.. *IEEE Transactions on Neural Networks and Learning Systems*, *26*(6), 1202–1213.

Emelyanov, A., Lapkin, D., Demin, V., Erokhin, V., Battistoni, S., Baldi, G., et al. (2016). First steps towards the realization of a double layer perceptron based on organic memristive devices. *AIP Advances*, *6*(11), 111301.

Ercal, F., Chawla, A., Stoecker, W. V., Lee, H.-C., & Moss, R. H. (1994). Neural network diagnosis of malignant melanoma from color images. *IEEE Transactions on Biomedical Engineering*, *41*(9), 837–845.

Glorot, X., & Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics* (pp. 249–256).

Hasan, R., & Taha, T. M. (2014). Enabling back propagation training of memristor crossbar neuromorphic processors. In *Neural networks (IJCNN), 2014 international joint conference on* (pp. 21–28). IEEE.

Ho, Y., Huang, G. M., & Li, P. (2009). Nonvolatile memristor memory: device characteristics and design implications. In *Computer-aided design-digest of technical papers, 2009. ICCAD 2009. IEEE/ACM international conference on* (pp. 485–490). IEEE.

Hornik, K. (1991). Approximation capabilities of multilayer feedforward networks. *Neural Networks*, *4*(2), 251–257.

Indiveri, G., Linares-Barranco, B., Legenstein, R., Deligeorgis, G., & Prodromakis, T. (2013). Integration of nanoscale memristor synapses in neuromorphic computing architectures. *Nanotechnology*, *24*(38), 384010.

Jain, A. K., Mao, J., & Mohiuddin, K. (1996). Artificial neural networks: A tutorial. *Computer*, *29*(3), 31–44.

Jeong, D. S., Kim, K. M., Kim, S., Choi, B. J., & Hwang, C. S. (2016). Memristors for energy-efficient new computing paradigms. *Advanced Electronic Materials*, *2*(9), 1600090.

Lehtonen, E., Poikonen, J., & Laiho, M. (2010). Two memristors suffice to compute all boolean functions. *Electronics Letters*, *46*(3), 230.

Minsky, M., Papert, S. A., & Bottou, L. (2017). *Perceptrons: An introduction to computational geometry*. MIT press.

Negrov, D., Karandashev, I., Shakirov, V., Matveyev, Y., Dunin-Barkowski, W., & Zenkevich, A. (2017). An approximate backpropagation learning rule for memristor based neural networks using synaptic plasticity. *Neurocomputing*, *237*, 193–199.

Pershin, Y. V., & Di Ventra, M. (2010). Experimental demonstration of associative memory with memristive neural networks. *Neural Networks*, *23*(7), 881–886.

Pershin, Y. V., & Di Ventra, M. (2012). Neuromorphic, digital, and quantum computation with memory circuit elements. *Proceedings of the IEEE*, *100*(6), 2071–2080.

Pfeiffer, P., Egusquiza, I., Di Ventra, M., Sanz, M., & Solano, E. (2016). Quantum memristors. *Scientific Reports*, *6*, 29507.

Prezioso, M., Merrikh-Bayat, F., Hoskins, B., Adam, G., Likharev, K. K., & Strukov, D. B. (2015). Training and operation of an integrated neuromorphic network based on metal-oxide memristors. *Nature*, *521*(7550), 61.

Prodromakis, T., & Toumazou, C. (2010). A review on memristive devices and applications. In *Electronics, circuits, and systems (ICECS), 2010 17th IEEE international conference on* (pp. 934–937). IEEE.

Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain.. *Psychological Review*, *65*(6), 386.

Rowley, H. A., Baluja, S., & Kanade, T. (1998). Neural network-based face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *20*(1), 23–38.

Salmilehto, J., Deppe, F., Di Ventra, M., Sanz, M., & Solano, E. (2017). Quantum memristors with superconducting circuits. *Scientific Reports*, *7*, 42044.

Sanz, M., Lamata, L., & Solano, E. (2018). Invited article: Quantum memristors in quantum photonics. *APL Photonics*, *3*(8), 080801. http://dx.doi.org/10.1063/1.5036596.

Soudry, D., Di Castro, D., Gal, A., Kolodny, A., & Kvatinsky, S. (2015). Memristor-based multilayer neural networks with online gradient descent training. *IEEE Transactions on Neural Networks and Learning Systems*, *26*(10), 2408–2421.

Strachan, J. P., Torrezan, A. C., Medeiros-Ribeiro, G., & Williams, R. S. (2011). Measuring the switching dynamics and energy efficiency of tantalum oxide memristors. *Nanotechnology*, *22*(50), 505402.

Strukov, D. B., Snider, G. S., Stewart, D. R., & Williams, R. S. (2008). The missing memristor found. *Nature*, *453*(7191), 80.

Taha, T. M., Hasan, R., Yakopcic, C., & McLean, M. R. (2013). Exploring the design space of specialized multicore neural processors. In *Neural networks (IJCNN), the 2013 international joint conference on* (pp. 1–8). IEEE.

Traversa, F. L., & Di Ventra, M. (2015). Universal memcomputing machines. *IEEE Transactions on Neural Networks and Learning Systems*, *26*(11), 2702–2715.

Walczak, S., & Cerpa, N. (1999). Heuristic principles for the design of artificial neural networks. *Information and Software Technology*, *41*(2), 107–117.

Wang, L., Duan, M., & Duan, S. (2013). Memristive perceptron for combinational logic classification. *Mathematical Problems in Engineering*, *2013*.

Wen, S., Xie, X., Yan, Z., Huang, T., & Zeng, Z. (2018). General memristor with applications in multilayer neural networks. *Neural Networks*, *103*, 142–149.

Wu, A., Wen, S., & Zeng, Z. (2012). Synchronization control of a class of memristor-based recurrent neural networks. *Information Sciences*, *183*(1), 106–116.

Xia, Q., Robinett, W., Cumbie, M. W., Banerjee, N., Cardinali, T. J., Yang, J. J., et al. (2009). Memristor- CMOS hybrid integrated circuits for reconfigurable logic. *Nano Letters*, *9*(10), 3640–3645.

Yakopcic, C., & Taha, T. M. (2013). Energy efficient perceptron pattern recognition using segmented memristor crossbar arrays. In *Neural networks (IJCNN), the 2013 international joint conference on* (pp. 1–8). IEEE.

Yang, J., Strukov, D. B., & Stewart, D. R. (2013). Memristive devices for computing. *Nature nanotechnology*, *8*(1), 13.

Yu, D., Iu, H. H.-C., Liang, Y., Fernando, T., & Chua, L. O. (2015). Dynamic behavior of coupled memristor circuits. *IEEE Transactions on Circuits and Systems. I. Regular Papers*, *62*(6), 1607–1616.