Practical Gradient-Descent for Memristive Crossbars

Manu V Nair
School of Electrical and Electronic Engineering
The University of Manchester, United Kingdom
manu.nair@postgrad.manchester.ac.uk

Piotr Dudek
School of Electrical and Electronic Engineering
The University of Manchester, United Kingdom
p.dudek@manchester.ac.uk

***Abstract*— This paper discusses implementations of gradient-descent based learning algorithms on memristive crossbar arrays. The Unregulated Step Descent (USD) is described as a practical algorithm for feed-forward on-line training of large crossbar arrays. It allows fast feed-forward fully parallel on-line hardware based learning, without requiring accurate models of the memristor behaviour and precise control of the programming pulses. The effect of device parameters, training parameters, and device variability on the learning performance of crossbar arrays trained using the USD algorithm has been studied via simulations.**

There is a significant interest in using memristive devices for computation, in particular in the context of neuromorpic systems [1] and artificial neural networks [2-7]. Memristors are typically fabricated in the form of highly-dense crossbar arrays, which naturally lend themselves to the vector-matrix multiplications that are at the core of the neural network algorithms. Memristor-based hardware implementations, while promising low-power high-speed computation, need to address several challenges, such as extreme device variability, complex state-dependent behaviours, or difficulty in integrating active devices within the crossbar. In this paper, we discuss an approximate gradient-descent based learning algorithm, called the Unregulated Step Descent (USD) [8] that addresses these hardware issues, and provides a practical method for training large crossbar arrays in machine learning applications.

We consider a hardware implementation of an optimization process, where the parameters $\mathbf{w}$ of the cost function $F(\mathbf{w})$ to be minimized are implemented as conductances of memristors in a crossbar array. A typical example of this is the supervised learning of weights in an artificial neural network. In the simplest case, that of a linear classifier (perceptron), the inputs can be applied as voltages to the rows of the crossbar array, and the output is obtained by applying a sigmoid nonlinearity to the column current. This basic hardware block can be extended to larger networks, as typically used in deep learning systems [7]. The goal of the optimization is to minimize the cost function expressing the difference between the actual outputs of the network, and the desired outputs, across the dataset. In this scenario, the training is achieved by delivering the programming pulses to the memristor devices, to change the weights. We are also interested in *in situ* training, where the crossbar array itself is used to implement the learning algorithm, as opposed to *ex situ* training, where the weights are determined outside the array, and then uploaded to the memristors. *In situ* training has many advantages, the most important of which is in allowing the array to "learn around" any defects and hardware variability issues [3-5].

The standard gradient descent rule prescribes an iterative training procedure where the weights $\mathbf{w}$ are updated as per the equation:

$$\mathbf{w} := \mathbf{w} - \alpha \nabla F(\mathbf{w}) \tag{1}$$

where $\nabla F(\mathbf{w})$ is the gradient of the cost function, and α is the learning rate. In order to minimize the cost function, in each iteration, the weights are updated in the direction of the "steepest slope" of the cost function. The magnitude of update of each weight, i.e. the size of a *step* towards the local minimum, depends on the corresponding partial derivative of the cost function, and the chosen learning rate.

More complex rules for determining a size/direction of each update step, to improve the convergence rate of this process, are well known in the optimization literature. However, these imply a control of the size of update for each weight separately. Such algorithms are not practical for hardware implementation on crossbar arrays, as they would require elaborate hardware to determine the size of each update step, in addition to sequential programming of the crossbar array required to deliver unique programming pulses to each of the memristor devices. This complicates the hardware interfacing to the array, and most importantly, makes the programming process much slower than a parallel programming scheme (where multiple memristors are updated at the same time), potentially obliterating any speed advantages gained from the improved convergence rate. On the other hand, it has been shown [9] that a simple fixed-step rule:

$$\mathbf{w} := \mathbf{w} - \alpha \cdot \text{sign}(\nabla F(\mathbf{w})) \tag{2}$$

where only a sign of the gradient is used to determine the magnitude of the update in each direction, can be also effectively used to train the weights. From the hardware implementation point of view, this is particularly attractive, as the sign of the gradient can be easily obtained - for example, in the linear classifier training, where the cost function describes the sum of squared differences between the training samples and the network outputs, the direction of each weight update is simply the sign of the error multiplied by the sign of the input. Furthermore, it appears that the desired weight changes could be then applied to all memristors in a crossbar array in parallel using a 4-phase scheme [2].

But, and this is what we are trying to make clear in this paper, even such a simple update rule is actually impractical to implement in all presently known memristor technologies. Any explicit weight update rule, e.g. as per equation (1) or (2), requires that the weights are updated precisely by the amount specified by second term on the RHS of the equations. This is difficult with memristive crossbar arrays for several reasons. Memristors are complex devices that display state-dependent transitions on application of a voltage. There are several technologies that exhibit the memristive effect, which can be attributed to a combination of complex physical processes [10]. Precise models to describe their behaviors do not exist, but even if such models were available, using complex models to predict the training pulses to be applied for a desired change in conductance would not be feasible for *in situ* training of large arrays, given the relatively large cost of additional computations involved, and the inevitable serialization of the programming pulses that would need to be delivered to individual devices.

Furthermore, the memristance change in response to a given programming pulse is a function of the internal state of the device, which may be not directly observable. And, the variability inherent in any fabrication process gives each individual memristor in a crossbar array a unique set of

parameters, which in current technologies can vary over orders of magnitude. Finally, stochastic behaviours [11] make the memristor's response to a programming pulse essentially unpredictable.

It is henceforth unrealistic to expect that feed-forward programming schemes, i.e. ones that rely on delivering precise programming pulses to individual devices, can actually implement an arbitrary learning algorithm. Some authors have even declared the gradient-descent in memristive crossbars outright impossible [6], although we think that it is their assumptions (especially the fixed, long pulsewidth) that have led to this somewhat overly pessimistic conclusion.

The alternative to feed-forward schemes is to apply feedback programming schemes [6,7], but again, this comes at a very large cost in terms of programming time, as not only all devices need to be programmed serially, but each programming instance involves an elaborate, time consuming procedure of the read-monitored write operation. Furthermore, active devices within the memrisor crossbar array are often required in these schemes [7].

Therefore, we postulate that a practical gradient-descent update rule has to forego any attempts at *precisely* regulating the size of the update step, and instead rely on the inherent ability of the iterative gradient descent procedure to converge towards a local minimum, even if the path towards the minimum is sub-optimal. Indeed, most of the gradient-descent schemes are already exploiting this principle, most notably the widely used stochastic gradient descent that trades-off the optimality of a single weight update step for the simplicity of the update rule and high frequency of the updates (e.g. after each training sample).

We dubbed such update rule an Unregulated Step Descent [8], and it can be generally expressed as:

$$\mathbf{w} := \mathbf{w} - \lambda(\beta, \mathbf{s}, \nabla F(\mathbf{w})) \qquad (3)$$

where $\lambda(\beta, \mathbf{s}, \nabla F(\mathbf{w}))$ denotes the changes in the weights (conductances) of the devices with internal state $\mathbf{s}$ when a training pulse β is applied to them. All of the devices in the crossbar arrays are subjected to training pulses β of the same width and magnitude, with just the sign of the applied voltage determined by the sign of the cost function gradient, analogous to the Manhattan rule. This generally results in unknown, state-dependent changes of individual conductances, but it is assumed that the direction of change is generally consistent with the direction of the cost function gradient. It should be noted that a similar rule has been used in a recent hardware demonstration of perceptrons in memristive crossbars [3]. The training pulse widths are proportional to the learning rate α, that could be held constant or reduced as the algorithm converges, according to some annealing schedule.

We carried out simulation experiments, using memristor models with build-in variability, and investigated the application of the USD rule to training various neural networks. We studied the effects of learning rate, the memristor on/off conductance ratio and amount of variability. The readers are directed to [8] for details of the experiments and results. As a highlight, Figure 1 shows the results of training a 20-dimensional linear classifier, with 1000 training samples. Data samples were generated randomly, with 5% probability to be on the "wrong" side of a dividing hyperplane, which results in <5% baseline error of an optimal linear classifier.

Our experiments suggest, that the proposed, straightforward, sign-based, unregulated step descent convergences to a good solution, despite large device variability, providing a practical hardware-based method for training crossbar arrays. A recent

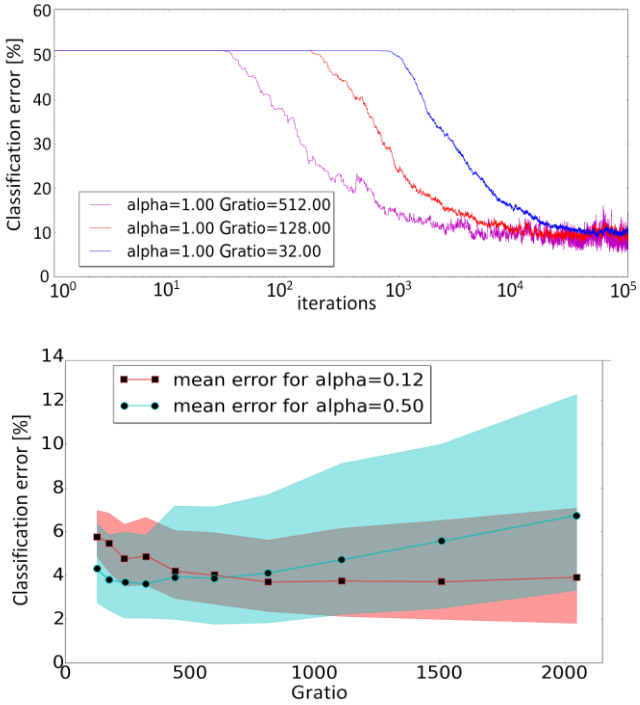

Figure 1: The effect of learning rate α ('alpha') and the ratio of on/off conductances ('Gratio') on convergence in the crossbar array (see text for details). Top plot shows classification error vs number of iterations; Bottom plot shows classification error (at convergence) for different 'alpha' and 'Gratio' values; the coloured bands show min-max values over 40 runs.

work [5] has demonstrated that fixed-step learning in a backpropagation network can produce successful results, albeit somewhat inferior to a variable-step (steepest descent) method. However, this has been done using specific idealized memristor models and ignoring variability issues other than point defects. We think that the global control of the learning rate α would make these differences even smaller. Nevertheless, given that the approximate control of the pulsewidth, in proportion to the error magnitude, should be relatively straightforward, we think that the unregulated step principle could be also successfully applied to approximate variable-step rules. In either case, while further research is needed to ascertain the practicality of simple memristor-based feed-forward training rules, especially regarding the extension to deep networks, we think that a simple update rules, that minimize the hardware overheads and allow fully parallel feed-forward weigh update, will be the ones that ultimately find practical application in on-line training of memristor arrays.

REFERENCES

[1] C. Zamarreno-Ramos et al., Frontiers in Neuroscience, 5 (20), 1-36. 2011

[2] F.Alibart, E.Zamanidoost and D.Strukov, Nature Communications., 4:2027, 2013

[3] M.Prezioso et al., Nature, vol 521, pp.61-64, 2015

[4] M.Hu et al. IEEE Trans. on Neural Networks and Learning Systems, vol.25 no.10, pp.1864-1878, 2014

[5] I. Kataeva et al. "Efficient Training Algorithms for Neural Networks Based on Memristive Crossbar Circuits", Proc. IJCNN 2015.

[6] F. Rothganger et al., "Training neural hardware with noisy components", Proc. IJCNN 2015

[7] C.Yakopcic, et al. "Memristor Based Neuromorphic Circuit for Ex-Situ Training of Multi-Layer Neural Network Algorithms", Proc. IJCNN 2015

[8] M.V.Nair and P.Dudek,"Gradient-descent based learning in memristive crossbar arrays", Proc. IJCNN 2015

[9] M.Riedmiller and H.Braun, "A direct adaptive method for faster backpropagation learning: The RPROP algorithm.", IEEE International Conference on Neural Networks, 1993

[10] J.J.Yang, D.B.Strukov and D.R.Steward, Nature Nanotechnology, Vol 8, pp.13-25, 2013

[11] Q Li, A Khiat, I Salaoru, H Xu, T Prodromakis, Nanoscale research letters, 9 (1), 1-5, 2014