Analog Integrated Circuits and Signal Processing, 13, 195–209 (1997)
© 1997 Kluwer Academic Publishers, Boston. Manufactured in The Netherlands.

Analog VLSI Stochastic Perturbative Learning Architectures*

GERT CAUWENBERGHS
gert@bach.ece.jhu.edu
Electrical and Computer Engineering, Johns Hopkins University, Baltimore, MD 21218

Received May 20, 1996; Accepted October 17, 1996

Abstract. We present analog VLSI neuromorphic architectures for a general class of learning tasks, which include supervised learning, reinforcement learning, and temporal difference learning. The presented architectures are parallel, cellular, sparse in global interconnects, distributed in representation, and robust to noise and mismatches in the implementation. They use a parallel stochastic perturbation technique to estimate the effect of weight changes on network outputs, rather than calculating derivatives based on a model of the network. This “model-free” technique avoids errors due to mismatches in the physical implementation of the network, and more generally allows to train networks of which the exact characteristics and structure are not known. With additional mechanisms of reinforcement learning, networks of fairly general structure are trained effectively from an arbitrarily supplied reward signal. No prior assumptions are required on the structure of the network nor on the specifics of the desired network response.

Key Words: Neural networks, neuromorphic engineering, reinforcement learning, stochastic approximation

1. Introduction

Carver Mead introduced “neuromorphic engineering” [1] as an interdisciplinary approach to the design of biologically inspired neural information processing systems, whereby neurophysiological models of perception and information processing in living organisms are mapped onto analog VLSI systems that not only emulate their functions but also resemble their structure [2].

Essential to neuromorphic systems are mechanisms of adaptation and learning, modeled after neural “plasticity” in neurobiology [3], [4]. Learning can be broadly defined as a special case of adaptation whereby past experience is used effectively in readjusting the network response to previously unseen, although similar, stimuli. Based on the nature and availability of a training feedback signal, learning algorithms for artificial neural networks fall under three broad categories: unsupervised, supervised and reward/punishment (reinforcement). Physiological experiments have revealed plasticity mechanisms in biology that correpond to Hebbian unsupervised learning [5], and classical (pavlovian) conditioning [6], [7] characteristic of reinforcement learning.

Mechanisms of adaptation and learning also provide a means to compensate for analog imperfections in the physical implementation of neuromorphic systems, and fluctuations in the environment in which they operate. Examples of early implementations of analog VLSI neural systems with integrated adaptation and learning functions can be found in [8]. While off-chip learning can be effective as long as training is performed with the chip “in the loop”, chip I/O bandwidth limitations make this approach impractical for networks with large number of weight parameters. Furthermore, on-chip learning provides autonomous, self-contained systems able to adapt continuously in the environment in which they operate.

On-chip learning in analog VLSI has proven to be a challenging task for several reasons. The least of problems is the need for local analog storage of the learned parameters, for which adequate solutions exist in various VLSI technologies [34]–[39]. More significantly, learning algorithms that are efficiently implemented on general-purpose digital computers do not necessarily map efficiently onto analog VLSI hardware. Second, even if the learning algorithm supports a parallel and

*This work was supported by ARPA/ONR under MURI grant N00014-95-1-0409. Chip fabrication was provided through MOSIS.

scalable architecture suitable for analog VLSI implementation, inaccuracies in the implementation of the learning functions may significantly affect the performance of the trained system. Third, the learning can only effectively compensate for inaccuracies in the network implementation when their physical sources are contained directly inside the learning feedback loop. Algorithms which assume a particular model for the underlying characteristics of the system being trained are expected to perform poorer than algorithms which directly probe the response of the system to external and internal stimuli. Finally, the nature of typical learning algorithms make assumptions on prior knowledge which place heavy constraints on practical use in hardware. This is particularly the case for physical systems of which the characteristics nor the optimization objectives are properly defined.

This paper addresses these challenges by using stochastic perturbative algorithms for model-free estimation of gradient information [9], in a general framework that includes reinforcement learning under delayed and discontinuous rewards [10]–[14]. In particular, we extend earlier work on stochastic error-descent architectures for supervised learning [15] to include computational primitives implementing reinforcement learning. The resulting analog VLSI architecture retains desirable properties of a modular and cellular structure, model-free distributed representation, and robustness to noise and mismatches in the implementation. In addition, the architecture is applicable to the most general of learning tasks, where an unknown "black-box" dynamical system is adapted using a external "black-box" reinforcement-based delayed and possibly discrete reward signal. As a proof of principle, we apply the model-free training-free adaptive techniques to blind optimization of a second-order noise-shaping modulator for oversampled data conversion, controlled by a neural classifier. The only evaluative feedback used in training the classifier is a discrete failure signal which indicates when some of the integrators in the modulation loop saturate.

Section 2 reviews supervised learning and stochastic perturbative techniques, and presents a corresponding architecture for analog VLSI implementation. The following section covers a generalized form of reinforcement learning, and introduces a stochastic perturbative analog VLSI architecture for reinforcement learning. Neuromorphic implementations in analog VLSI and their equivalents in biology are the subject of section 4. Finally, section 5 concludes the findings.

2. Supervised Learning

In a metaphorical sense, supervised learning assumes the luxury of a committed "teacher", who constantly evaluates and corrects the network by continuously feeding it target values for all network outputs. Supervised learning can be reformulated as an optimization task, where the network parameters (weights) are adjusted to minimize the distance between the targets and actual network outputs. Generalization and overtraining are important issues in supervised learning, and are beyond the scope of this paper.

Let $\mathbf{y}(t)$ be the vector of network outputs with components $y_i(t)$, and correspondingly $\mathbf{y}^{\text{target}}(t)$ be the supplied target output vector. The network contains adjustable parameters (or weights) $\mathbf{p}$ with components p_k, and state variables $\mathbf{x}(t)$ with components $x_i(t)$ (which may contain external inputs). Then the task is to minimize the scalar error index

$$\mathcal{E}(\mathbf{p}; t) = \sum_i |y_i^{\text{target}}(t) - y_i(t)|^\nu. \qquad (1)$$

in the parameters p_i, using a distance metric with norm $\nu > 0$.

2.1. Gradient Descent

Gradient descent is the most common optimization technique for supervised learning in neural networks, which includes the widely used technique of backpropagation (or "dynamic feedback") [16] for gradient derivation, applicable to general feedforward multilayered networks.

In general terms, gradient descent minimizes the scalar performance index $\mathcal{E}$ by specifying incremental updates in the parameter vector $\mathbf{p}$ according to the error gradient $\nabla_{\mathbf{p}}\mathcal{E}$:

$$\mathbf{p}(t+1) = \mathbf{p}(t) - \eta\, \nabla_{\mathbf{p}}\mathcal{E}(t). \qquad (2)$$

One significant problem with gradient descent and its variants for on-line supervised learning is the complexity of calculating the error gradient components $\partial\mathcal{E}/\partial p_k$ from a model of the system. This is especially so for complex systems involving internal dynamics in the state variables $x_j(t)$:

$$\frac{\partial \mathcal{E}}{\partial p_k} = \sum_{i,j} \frac{\partial \mathcal{E}(t)}{\partial y_i} \cdot \frac{\partial y_i(t)}{\partial x_j} \cdot \frac{\partial x_j(t)}{\partial p_k} \qquad (3)$$

where derivation of the dependencies $\partial x_j/\partial p_k$ over time constitutes a significant amount of computation that typically scales super-linearly with the dimension of the network [15]. Furthermore, the derivation of the gradient in (3) assumes accurate knowledge of the model of the network ($\mathbf{y}(t)$ as a function of $\mathbf{x}(t)$, and recurrence relations in the state variables $\mathbf{x}(t)$). Accurate model knowledge cannot be assumed for analog VLSI neural hardware, due to mismatches in the physical implementation which can not be predicted at the time of fabrication. Finally, often a model for the system being optimized may not be readily available, or may be too complicated for practical (real-time) evaluation. In such cases, a black-box approach to optimization is more effective in every regard. This motivates the use of the well-known technique of stochastic approximation [17] for blind optimization in analog VLSI systems. We apply this technique to supervised learning as well as to more advanced models of "reinforcement" learning under discrete delayed rewards. The connection between stochastic approximation techniques and principles of neuromorphic engineering will be illustrated further below, in contrast with gradient descent.

2.2. Stochastic Approximation Techniques

Stochastic approximation algorithms [17] have long been known as effective tools for constrained and unconstrained optimization under noisy observations of system variables [18]. Applied to on-line minimization of an error index $\mathcal{E}(\mathbf{p})$, the algorithms avoid the computational burden of gradient estimation by directly observing the dependence of the index $\mathcal{E}$ on randomly applied perturbations in the parameter values. Variants on the Kiefer-Wolfowitz algorithm for stochastic approximation [17], essentially similar to random-direction finite-difference gradient descent, have been formulated for blind adaptive control [19], neural networks [9],[20] and the implementation of learning functions in VLSI hardware [21], [22], [15], [23].

The broader class of neural network learning algorithms under this category exhibit the desirable property that the functional form of the parameter updates is "model-free", *i.e.*, independent of the model specifics of the network or system under optimization. The model-free techniques for on-line supervised learning are directly applicable to almost any observable system with deterministic, slowly varying, and possibly unknown characteristics. Parallel implementation of the stochastic approximation algorithms results into efficient and modular learning architectures that map well onto VLSI hardware. Since those algorithms use only direct function evaluations and no derivative information, they are functionally simple, and their implementation is independent of the structure of the system under optimization. They exhibit robust convergence properties in the presence of noise in the system and model mismatches in the implementation.

A brief description of the stochastic error-descent algorithm follows below, as introduced in [15] for efficient supervised learning in analog VLSI. The integrated analog VLSI continous-time learning system used in [24], [25] forms the basis for the architectures outlined in the sections that follow.

2.3. Stochastic Supervised Learning

Let $\mathcal{E}(\mathbf{p})$ be the error functional to be minimized, with $\mathcal{E}$ a scalar deterministic function in the parameter (or weight) vector $\mathbf{p}$ with components p_i. The stochastic algorithm specifies incremental updates in the parameters p_i as with gradient descent (2), although using a stochastic approximation to the true gradient

$$\frac{\partial\mathcal{E}(t)}{\partial p_i}^{\text{est}} = \pi_i(t)\cdot\hat{\mathcal{E}}(t) \tag{4}$$

where the differentially perturbed error

$$\hat{\mathcal{E}}(t) = \frac{1}{2\sigma^2}\left(\mathcal{E}(\mathbf{p}(t)+\boldsymbol{\pi}(t)) - \mathcal{E}(\mathbf{p}(t)-\boldsymbol{\pi}(t))\right) \tag{5}$$

is obtained from two direct observations of $\mathcal{E}$ under complementary activation of a parallel random perturbation vector $\boldsymbol{\pi}(t)$ with components $\pi_i(t)$ onto the parameter vector $\mathbf{p}(t)$. The perturbation components $\pi_i(t)$ are fixed in amplitude and random in sign, $\pi_i(t) = \pm\sigma$ with equal probabilities for both polarities. The algorithm essentially performs gradient descent in random directions in the parameter space, as defined by the position of the perturbation vector.

As with exact gradient descent, iteration of the updates using (4) converges in the close proximity of a (local) minimum of $\mathcal{E}$, provided the perturbation amplitude σ is sufficiently small. The rate of convergence is necessarily slower than gradient descent, since every observation (5) only reveals scalar information about the gradient vector in one dimension. However, the amount of computation required to compute the gradient at every update may outweigh the higher convergence rate offered by gradient descent, depending on the model complexity of the system under optimization. When applied to on-line supervised learning in recurrent dynamical systems, the stochastic algorithm provides a net computational efficiency rivaling that of exact gradient descent. Computational efficiency is defined in terms of the total number of operations required to converge, *i.e.*, reach a certain level of $\mathcal{E}$. A formal derivation of the convergence properties is presented in [15].

2.4. *Supervised Learning Architecture*

While alternative optimization techniques based on higher-order extensions on gradient descent will certainly offer superior convergence rates, the above stochastic method achieves its relative efficiency at a much reduced complexity of implementation. The only global operations required are the evaluations of the error function in (5), which are obtained from direct observations on the system under complementary activation of the perturbation vector. The operations needed to generate and apply the random perturbations, and to perform the parameter update increments, are strictly local and identical for each of the parameter components. The functional diagram of the local parameter processing cell, embedded in the system under optimization, is shown in Figure 1. The complementary perturbations and the corresponding error observations are performed in two separate phases on the same system, rather than concurrently on separate replications of the system. The sequential activation of the complementary perturbations and corresponding evaluations of $\mathcal{E}$ are synchronized and coordinated with a three-phase clock:

$$\begin{aligned} \phi^0 &: \mathcal{E}(\mathbf{p}, t) \\ \phi^+ &: \mathcal{E}(\mathbf{p} + \boldsymbol{\pi}, t) \\ \phi^- &: \mathcal{E}(\mathbf{p} - \boldsymbol{\pi}, t). \end{aligned} \tag{6}$$

This is represented schematically in Figure 1 by a modulation signal $\phi(t)$, taking values $\{-1, 0, 1\}$. The extra phase ϕ^0 ($\phi(t) = 0$) is not strictly needed to compute (5)—it is useful otherwise, *e.g.* to compute finite difference estimates of second order derivatives for dynamic optimization of the learning rate $\eta(t)$.

The local operations are further simplified owing to the binary nature of the perturbations, reducing the multiplication in (4) to an exclusive-or logical operation, and the modulation by $\phi(t)$ to binary multiplexing. Besides efficiency of implementation, this has a beneficial effect on the overall accuracy of the implemented learning system, as will be explained in the context of VLSI circuit implementation below.

2.5. *Supervised Learning in Dynamical Systems*

In the above, it was assumed that the error functional $\mathcal{E}(\mathbf{p})$ is directly observable on the system by applying the parameter values p_i. In the context of on-line supervised learning in dynamical systems, the error functional takes the form of the average distance of norm ν between the output and target signals over a moving time window,

$$\mathcal{E}(\mathbf{p}; t_i, t_f) = \int_{t_i}^{t_f} \sum_i |y_i^{\text{target}}(t') - y_i(t')|^\nu \mathrm{d}t', \tag{7}$$

which implicitly depends on the training sequence $\mathbf{y}^{\text{target}}(t)$ and on initial conditions on the internal state variables of the system. An on-line implementation prohibits simultaneous observation of the error measure (7) under different instances of the parameter vector $\mathbf{p}$, as would be required to evaluate (5) for construction of the parameter updates. However, when the training signals are periodic and the interval $T = t_f - t_i$ spans an integer multiple of periods, the measure (7) under constant parameter values is approximately invariant to time. In that case, the two error observations needed in (5) can be performed in sequence, under complementary piecewise constant activation of the perturbation vector.

Stochastic error-descent with the dynamic error index in (7) has been used in [24], [25] to implement the supervised learning functions in an integrated network of six fully interconnected continuous-time neurons with 42 parameters. The analog VLSI chip, dissipating 1.2 mW from a 5 V supply, learned to generate given periodic signals at two of the outputs, representing a quadrature-phase oscillator, in less than 1500 training cycles of 60 msec each.

In the context of on-line supervised learning in dynamical systems, the requirement of periodicity on the training signals is a major limitation of the stochastic error-descent algorithm. Next, this requirement will be

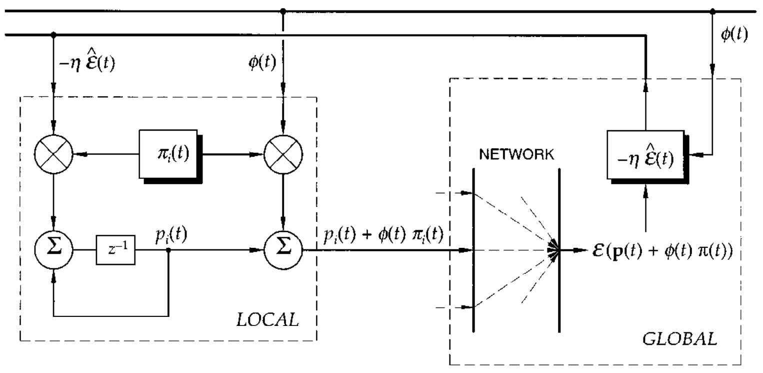

$-\eta\,\hat{\mathcal{E}}(t)$
$\phi(t)$
$\phi(t)$
$\pi_i(t)$
z^{-1}
$p_i(t)$
$p_i(t) + \phi(t)\,\pi_i(t)$
LOCAL
NETWORK
$-\eta\,\hat{\mathcal{E}}(t)$
$\mathcal{E}(\mathbf{p}(t) + \phi(t)\,\pi(t))$
GLOBAL

Fig. 1. Architecture implementing stochastic error-descent supervised learning. The learning cell is locally embedded in the network. The differential error index is evaluated globally, and communicated to all cells.

relaxed, along with some more stringent assumptions on the nature of supervised learning. In particular, a target training signal will no longer be necessary. Instead, learning is based on an external reward signal that conveys only partial and delayed information about the performance of the network.

3. Learning From Delayed and Discrete Rewards

Supervised learning methods rely on a continuous training signal that gives constant feedback about the direction in which to adapt the weights of the network to improve its performance. This continuous signal is available in the form of target values $\mathbf{y}^{\text{target}}(t)$ for the network outputs $\mathbf{y}(t)$. More widely applicable but also more challenging are learning tasks where target outputs or other continuous teacher feedback are not available, but instead only a non-steady, delayed reward (or punishment) signal is available to evaluate the *quality* of the outputs (or actions) produced by the network. The difficulty lies in assigning proper credit for the reward (or punishment) to actions that where produced by the network in the past, and adapting the network accordingly in such a way to *reinforce* the network actions leading to reward (and conversely, avoid those leading to punishment).

3.1. Reinforcement Learning Algorithms

Several iterative approaches to dynamic programming have been applied to solve the credit assignment problem for training a neural network with delayed rewards [10]–[14]. They all invoke an "adaptive critic element" which is trained along with the network to predict the future reward signal from the present state of the network. We define "reinforcement learning" essentially as given in [11], which includes as special cases "time difference learning" or TD(λ) [12], and, to some extent, Q-learning [13] and "advanced heuristic dynamic programming" [14]. The equations are listed below in general form to clarify the similarity with the above supervised perturbative learning techniques. It will then be shown how the above architectures are extended to allow learning from delayed and/or impulsive rewards.

Let $r(t)$ be the discrete delayed reward signal for state vector $\mathbf{x}(t)$ of the system (components $x_j(t)$). $r(t)$ is zero when no signal is available, and is negative for a punishment. Let $y(t)$ be the (scalar) output of the network in response to an input (or state) $\mathbf{x}(t)$, and $q(t)$ the predicted future reward (or "value function") associated with state $\mathbf{x}(t)$ as produced by the adaptive critic element. The action taken by the system is determined by the polarity of the network output, $\text{sign}(y(t))$. For example, in the pole balancing experiment of [11], $y(t)$ is hard-limited and controls the direction of the fixed-amplitude force exterted on the moving cart. Finally, let $\mathbf{w}$ and $\mathbf{v}$ (components w_i and v_i) be the weights of the network and the adaptive critic element, respectively. Then the weight updates are given by

$$\begin{aligned}\Delta w_i(t) &= w_i(t+1) - w_i(t) \qquad (8)\\ &= \alpha\,\hat{r}(t)\cdot e_i(t)\\ \Delta v_i(t) &= v_i(t+1) - v_i(t) \qquad (9)\\ &= \beta\,\hat{r}(t)\cdot d_i(t)\end{aligned}$$

where the "eligibility" functions $e_i(t)$ and $d_i(t)$ are updated as

$$
\begin{aligned}
e_i(t+1) &= \delta e_i(t) + (1-\delta)\,\mathrm{sign}(y(t))\frac{\partial y(t)}{\partial w_i} \qquad (10)\\
d_i(t+1) &= \lambda d_i(t) + (1-\lambda)\frac{\partial q(t)}{\partial v_i}
\end{aligned}
$$

and the reinforcement $\hat{r}(t)$ is given by

$$\hat{r}(t) = r(t) + \gamma q(t) - q(t-1). \qquad (11)$$

The parameters δ and λ define the time span of credit assigned by $e_i(t)$ and $d_i(t)$ to actions in the past, weighting recent actions stronger than past actions:

$$
\begin{aligned}
e(t) &= (1-\delta)\sum_{t'=-\infty}^{t-1}\delta^{t-t'-1}\,\mathrm{sign}(y(t'))\frac{\partial y(t')}{\partial w_i} \qquad (12)\\
d(t) &= (1-\lambda)\sum_{t'=-\infty}^{t-1}\lambda^{t-t'-1}\frac{\partial q(t')}{\partial v_i}.
\end{aligned}
$$

Similarly, the parameter γ defines the time span for the prediction of future reward by $q(t)$, at convergence:

$$q(t) \approx \sum_{t'=t+1}^{\infty}\gamma^{t'-t-1}r(t'). \qquad (13)$$

For $\gamma = 1$ and $y \equiv q$, the equations reduce to TD(λ). Convergence of TD(λ) with probability one has been proven in the general case of linear networks of the form $q = \sum v_i x_i$ [26].

Learning algorithms of this type are neuromorphic in the sense that they emulate classical (pavlovian) conditioning in pattern association as found in biological systems [6] and their mathematical and cognitive models [27], [7]. Furthermore, as shown below, the algorithms lend themselves to analog VLSI implementation in a parallel distributed architecture which, unlike more complicated gradient-based schemes, resembles the general structure and connectivity of biological neural systems.

3.2. Reinforcement Learning Architecture

While reinforcement learning does not perform gradient descent of a (known) error functional, the eligibility functions $e_i(t)$ and $d_i(t)$ used in the weight updates are constructed from derivatives of output functions to the weights. The eligibility functions in equation (10) can be explicitly restated as (low-pass filtered) gradients of an error function

$$\mathcal{E}(t) = |y(t)| + q(t) \qquad (14)$$

with

$$
\begin{aligned}
e_i(t+1) &= \delta\, e_i(t) \;+\; (1-\delta)\,\frac{\partial \mathcal{E}(t)}{\partial w_i} \qquad (15)\\
d_i(t+1) &= \lambda\, d_i(t) \;+\; (1-\lambda)\,\frac{\partial \mathcal{E}(t)}{\partial v_i}.
\end{aligned}
$$

Rather than calculating the gradients in (15) from the network model, we can again apply stochastic pertubative techniques to estimate the gradients from direct evaluations on the network. Doing so, all properties of robustness, scalability and modularity that apply to stochastic error descent supervised learning apply here as well. As in (4), stochastic approximation estimates of the gradient components in (15) are

$$
\begin{aligned}
\frac{\partial \mathcal{E}(t)}{\partial w_i}^{\mathrm{est}} &= \omega_i(t)\,\hat{\mathcal{E}}(t) \qquad (16)\\
\frac{\partial \mathcal{E}(t)}{\partial v_i}^{\mathrm{est}} &= \upsilon_i(t)\,\hat{\mathcal{E}}(t)
\end{aligned}
$$

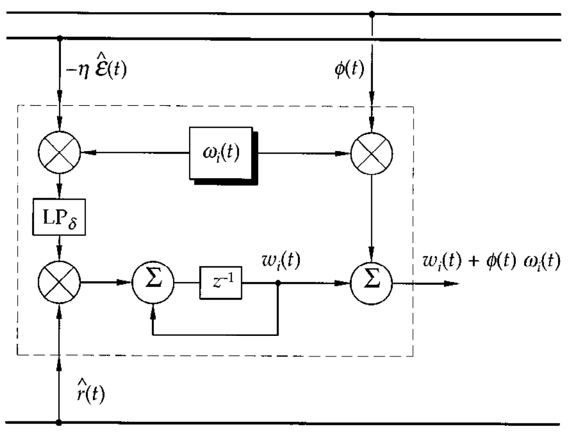
where the differential perturbed error

$$
\begin{aligned}
\hat{\mathcal{E}}(t) = \frac{1}{2\sigma^2}\,&(\mathcal{E}(\mathbf{w}+\boldsymbol{\omega}, \mathbf{v}+\boldsymbol{\upsilon}, t)\\
&- \mathcal{E}(\mathbf{w}-\boldsymbol{\omega}, \mathbf{v}-\boldsymbol{\upsilon}, t)) \qquad (17)
\end{aligned}
$$

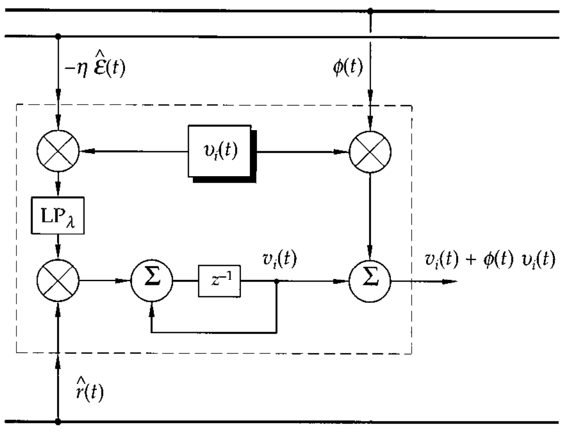
is obtained from two-sided parallel random perturbation $\mathbf{w} \pm \boldsymbol{\omega}$ simultaneous with $\mathbf{v} \pm \boldsymbol{\upsilon}$ ($|\omega_i| = |\upsilon_i| = \sigma$).

A side benefit of the low-pass filtering of the gradient in (15) is an improvement of the stochastic gradient estimate (16) through averaging. As with stochastic error descent supervised learning, averaging reduces the variance of the gradient estimate and produces learning increments that are less stochastic in nature, although this is not essential for convergence of the learning process [15].

Figure 2 shows the block diagram of a reinforcement learning cell and an adaptive critic cell, with stochastic perturbative estimation of the gradient according to (16). LP$_\delta$ and LP$_\lambda$ denote first-order low-pass filters (15) with time constants determined by δ and λ. Other than the low-pass filtering and the global multiplicative factor $\hat{r}(t)$, the architecture is identical to that of stochastic error descent learning in Figure 1. As before, the estimation of $\hat{\mathcal{E}}(t)$ does not require separate instances of perturbed and non-perturbed networks shown in Figure 2, and can be computed sequentially



(a)



(b)

Fig. 2. General architecture implementing reinforcement learning using stochastic gradient approximation. *(a)* Reinforcement learning cell. *(b)* Adaptive critic cell.

by evaluating the output of the network and adaptive critic in three phases for every cycle of t:

$$\begin{aligned} \phi^0 &: \mathcal{E}(\mathbf{w}, \mathbf{v}, t) \\ \phi^+ &: \mathcal{E}(\mathbf{w} + \boldsymbol{\omega}, \mathbf{v} + \boldsymbol{\upsilon}, t) \\ \phi^- &: \mathcal{E}(\mathbf{w} - \boldsymbol{\omega}, \mathbf{v} - \boldsymbol{\upsilon}, t). \end{aligned} \tag{18}$$

In systems with a continuous-time output response, we assume that the time lag between consecutive observations of the three phases of $\mathcal{E}$ is not an issue, which amounts to choosing an appropriate sampling rate for t in relation to the bandwidth of the system.

Similarities between the above cellular architectures for supervised learning and reinforcement learning are apparent: both correlate local perturbation values π_i, ω_i or υ_i with a global scalar index $\hat{\mathcal{E}}$ that encodes the differential effect of the perturbations on the output, and both incrementally update the weights p_i, w_i or v_i accordingly. The main difference in reinforcement learning is the additional gating of the correlate product with a global reinforcement signal $\hat{r}$ after temporal filtering. For many applications, the extra overhead that this implies in hardware resources is more than compensated by the utility of the reward-based credit assignment mechanism, which does not require an external teacher. An example is given below in the case of oversampled A/D conversion.

3.3. Simulation Results

We evaluated both exact gradient and stochastive perturbative embodiments of the reinforcement learning algorithms on an adaptive neural classifier, controlling a higher-order noise-shaping modulator used for oversampled A/D data conversion [30]. The order-n modulator comprises a cascade of n integrators $x_i(t)$ operating on the difference between the analog input $u(t)$ and the binary modulated output $y(t)$:

$$\begin{aligned} x_0(t+1) &= x_0(t) + a\,(u(t) - y(t)) \\ x_i(t+1) &= x_i(t) + a\,x_{i-1}(t), \quad i = 1, \ldots n-1 \end{aligned} \tag{19}$$

where $a = 0.5$. The control objective is to choose the binary sequence $y(t)$ such as to keep the excursion of the integration variables within bounds, $|x_i(t)| < x_{\mathrm{sat}}$ [29].

For the adaptive classifier, we specify a one-hidden-layer neural network, with inputs $x_i(t)$ and output $y(t)$. The network has m hidden units, a tanh(.) sigmoidal nonlinearity in the hidden layer, and a linear output layer. For the simulations we set $n = 2$ and $m = 5$. The case $n = 2$ is equivalent to the single pole-balancing problem [11].

The only evaluative feedback signal used during learning is a failure signal which indicates when one or more integration variables saturate, $|x_i(t)| \geq x_{\mathrm{sat}}$. In particular, the signal $r(t)$ counts the number of integrators in saturation:

$$r(t) = -b \sum_i H(|x_i(t)| - x_{\mathrm{sat}}) \tag{20}$$

where $b = 10$, and where $H(.)$ denotes a step function ($H(x) = 1$ if $x > 0$ and 0 otherwise). The adaptive critic $q(t)$ is implemented with a neural network of

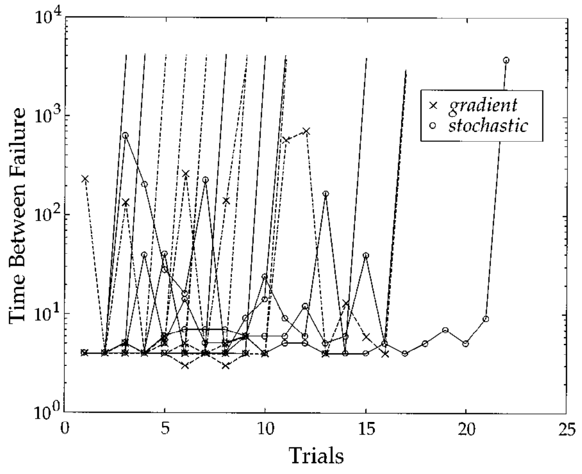


Fig. 3. Simulated performance of stochastic perturbative (○) and gradient-based (×) reinforcement learning in a second-order noise-shaping modulator. Time between failures for consecutive trials from zero initial conditions.

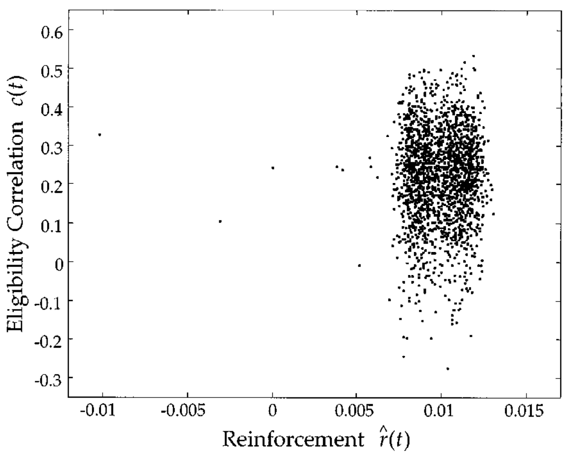


Fig. 4. Effect of stochastic perturbative gradient estimation on the reinforcement learning. $c(t)$ quantifies the degree of conformity in the eligibilities $e_i(t)$ and $d_i(t)$ between exact and stochastic versions.

identical structure as for $y(t)$. The learning parameters in (8), (10) and (11) are $\delta = 0.8$, $\lambda = 0.7$, $\gamma = 0.9$, $\alpha = 0.05$ and $\beta = 0.001$. These values are consistent with [11], adapted to accommodate for differences in the time scale of the dynamics (19). The perturbation strength in the stochastic version is $\sigma = 0.01$.

Figure 3 shows the learning performance for several trials of both versions of reinforcement learning, using exact and stochastic gradient estimates. During learning, the input sequence $u(t)$ is random, uniform in the range $-0.5 \dots 0.5$. Initially, and every time failure occurs ($r(t) < 0$), the integration variables $x_i(t)$ and eligibilities $e_k(t)$ and $d_k(t)$ are reset to zero. Qualitative differences observed between the exact and stochastic versions in Figure 3 are minor. Further, in all but one of the 20 cases tried, learning has completed (*i.e.*, consequent failure is not observed in finite time) in fewer than 20 consecutive trial-and-error iterations. Notice that a non-zero $r(t)$ is only generated at failure, *i.e.*, less than 20 times, and no other external evaluative feedback is needed for learning.

Figure 4 quantifies the effect of stochastic perturbative estimation of the gradients (15) on the quality of reinforcement learning. The correlation index $c(t)$ measures the degree of conformity in the eligibilities (both $e_i(t)$ and $d_i(t)$) between stochastic and exact versions of reinforcement learning. Correlation is expressed as usual on a scale from -1 to 1, with $c(t) = 1$ indicating perfect coincidence. While $c(t)$ is considerably less than 1 in all cases, $c(t) > 0$ about 95 % of the time, meaning that *on average* the sign of the parameter updates (8) for exact and stochastic versions are consistent in at least 95 % of the cases. The scatterplot $c(t)$ vs. $\hat{r}(t)$ also illustrates how the adaptive critic produces a positive reinforcement $\hat{r}(t)$ in most of the cases, even though the "reward" signal $r(t)$ is never positive by construction. Positive reinforcement $\hat{r}(t)$ under idle conditions of $r(t)$ is desirable for stability. Notice that the failure-driven punishment points (where $r(t) < 0$) are off-scale of the graph and strongly negative.

We tried reinforcement learning on higher-order modulators, $n = 3$ and higher. Both exact and stochastic versions were successful for $n = 3$ in the majority of cases, but failed to converge for $n = 4$ with the same parameter settings. On itself, this is not surprising since higher-order delta-sigma modulators tend to become increasingly prone to unstabilities and sensitive to small changes in parameters with increasing order n, which is why they are almost never used in practice [30]. It is possible that more advanced reinforcement learning techniques such as "Advanced Heuristic Dynamic Programming" (AHDP) [14] would succeed to converge for orders $n > 3$. AHDP offers improved learning efficiency using a more advanced, gradient-based adaptive critic element for prediction of reward, although it is not clear at present how to map the algorithm efficiently onto analog VLSI.

The above stochastic perturbative architectures for both supervised and reinforcement learning support common "neuromorphs" and corresponding analog

VLSI implementations. Neuromorphs of learning in analog VLSI are the subject of next section.

4. Neuromorphic Analog VLSI Learning

Neuromorphic engineering [1] combines inspiration from neurobiology to pursue human-engineered human-like VLSI information processing systems at all levels of the design: device physics and technology, circuit topologies, and the overall architecture organizing the flow of information. We address these levels in the context of learning as defined above.

4.1. Subthreshold MOS Technology

MOS transistors operating in the subthreshold region [31] are attractive for use in medium-speed, medium-accuracy analog VLSI processing, because of the low current levels and the exponential current-voltage characteristics that span a wide dynamic range of currents [32] (roughly from 100 fA to 100 nA for a square device in 2 μm CMOS technology). Futhermore, subthreshold MOS transistors provide a clear “neuromorph” [1], since their exponential I-V characteristics closely resemble the carrier transport though cell membranes in biological neural systems, as governed by the same Boltzman statistics [33]. The exponential characteristics provide a variety of subthreshold MOS circuit topologies that serve as useful computational primitives (such as nonlinear conductances, sigmoid nonlinearities, etc.) for compact analog VLSI implementation of neural systems [2]. Of particular interest are translinear subthreshold MOS circuits, derived from similar bipolar circuits [32]. They are based on the exponential nature of current-voltage relationships, and offer attractive compact implementations of product and division operations in VLSI.

4.2. Adaptation and Memory

Learning in analog VLSI systems is inherently coupled with the problem of storage of analog information, since after learning it is most often desirable to retain the learned weights for an extended period of time. The same is true for biological neural systems, and mechanisms of plasticity for short-term and long-term synaptic storage are not quite understood. In VLSI, analog weights are conveniently stored as charge or voltage on a capacitor. A capacitive memory is generically depicted in Figure 5. The stored weight charge is preserved when brought in contact with the gate of an MOS transistor, which serves as a buffer between weight storage and the implementation of the synaptic function. An adaptive element in contact with the capacitor updates the stored weight in the form of discrete charge increments

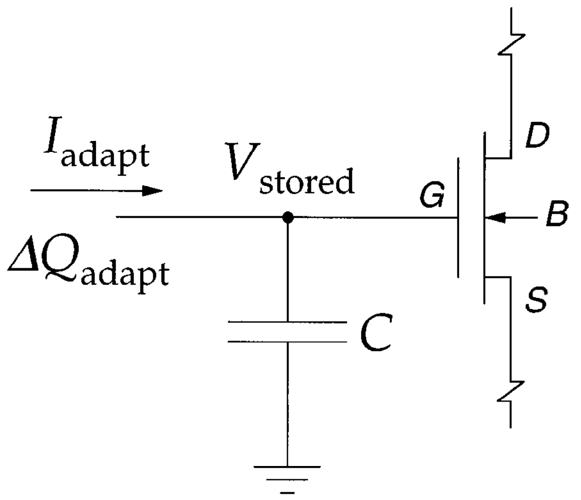


Fig. 5. Adaptation and memory in analog VLSI: storage cell with charge buffer.

$$V_{\text{stored}}(t+1) = V_{\text{stored}}(t) + \frac{1}{C}\Delta Q_{\text{adapt}}(t) \tag{21}$$

or, equivalently, a continuous current supplying a derivative

$$\frac{d}{dt}V_{\text{stored}}(t) = \frac{1}{C}I_{\text{adapt}}(t) \tag{22}$$

where $\Delta Q_{\text{adapt}}(t) = \int_t^{t+1} I_{\text{adapt}}(t')dt'$.

On itself, a floating gate capacitor is a near-perfect memory. However, leakage and spontaneous decay of the weights result when the capacitor is in volatile contact with the adaptive element, such as through drain or source terminals of MOS transistors. Non-volatile memories contain adaptive elements that interface with the floating gate capacitor by capacitive coupling across an insulating oxide. Charge transport through the oxide is controlled by tunneling and hot electron injection [34] or UV-activated conduction [35], [36]. Volatile memories are adequate for short-term storage, or re-

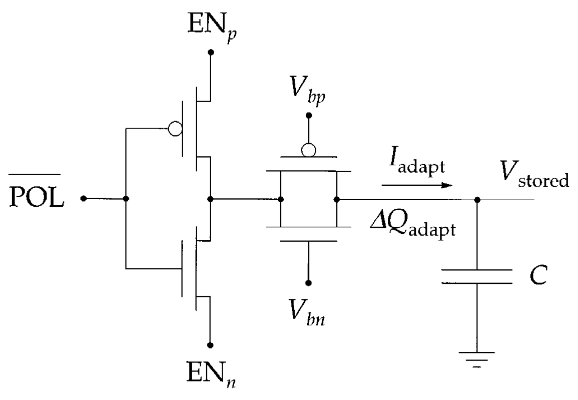


Fig. 6. Charge-pump adaptive element implementing a volatile synapse.

quire an active refresh mechanism for long-term storage [37], [38].

4.3. Adaptive Circuits

Charge-Pump Adaptive Element Figure 6 shows the circuit diagram of a charge-pump adaptive element implementing a volatile synapse. The circuit is a simplified version of the charge pump used in [38] and [25]. When enabled by EN_n and EN_p (at GND and V_{dd} potentials, respectively), the circuit generates an incremental update (21) or (22) of which the polarity is determined by $\overline{\mathrm{POL}}$. The amplitude of the current supplying the incremental update is determined by gate voltages V_{bn} and V_{bp}, biased deep in subthreshold to allow fine (sub-fC) increments if needed. The increment amplitude is also determined by the duration of the enabled current, controlled by the timing of EN_n and EN_p. When both EN_n and EN_p are set midway between GND and V_{dd}, the current output is disabled. Notice that the switch-off transient is (virtually) free of clock feedthrough charge injection, because the current-supplying transistors are switched from their source terminals, with the gate terminals being kept at constant voltage [38].

Stochastic Perturbative Learning Cell The circuit schematic of a learning cell implementing stochastic error descent is given in Figure 7, adapted from [24], [25] in simplified form. The incremental update $-\eta\pi_i\hat{\mathcal{E}}$ to be performed in (5) is first decomposed in amplitude and sign components. This allows for a hybrid digital-analog implementation of the learning cell, in which amplitudes of certain operands are processed in analog format, and their polarities implemented in logic. Since $|\pi_i| \equiv 1$, the amplitude $\eta|\hat{\mathcal{E}}|$ is conventiently communicated as a global signal to all cells, in the form of two gate voltages V_{bn} and V_{bp}. The (inverted) polarity $\overline{\mathrm{POL}}$ is obtained as the (inverted) exclusive-or combination of the perturbation π_i and the polarity of $\hat{\mathcal{E}}$. The decomposition of sign and amplitude ensures proper convergence of the learning increments in the presence of mismatches and offsets in the physical implementation of the learning cell. This is a consequence of the fact that the polarities of the increments are implemented accurately by logic-controlled circuitry, regardless of analog mismatches in the implementation.

The perturbation π_i is applied to p_i in three phases (5) by capacitive coupling onto the storage node C. The binary state of the local perturbation π_i selects one of two global perturbation signals to couple onto C. The perturbation signals (V_σ^+ and its complement V_σ^-) globally control the three phases ϕ^0, ϕ^+ and ϕ^- of (5), and set the perturbation amplitude σ. The simple configuration using a one-bit multiplexer is possible because each perturbation component can only take one of two values $\pm\sigma$.

Analog Storage Because of the volatile nature of the adaptive element used, an active refresh mechanism may be required if long-term local storage of the weight values after learning is desired [37]. A robust and efficient mechanism is "partial incremental refresh" [38], which can be directly implemented using the adaptive element in Figure 7 by driving $\overline{\mathrm{POL}}$ with a binary function of the weight value [39]. As in [39], the binary quantization function can be multiplexed over an array of storage cells, and can be implemented by retaining the LSB from A/D/A conversion [40] of the value to be stored.

A non-volatile equivalent of the adaptive element in Figure 6 is described in [34]. A non-volatile learning cell performing stochastic error descent can be obtained by substituting [34] as the core adaptive element in Figure 7. Likewise, more intricate volatile and non-volatile circuits implementing stochastic reinforcement learning can be derived from extensions on Figure 7 and [34].

Measurements The model-free nature of the stochastic perturbative learning algorithms does not impose any particular conditions on the implementation of computational functions required for learning. By far

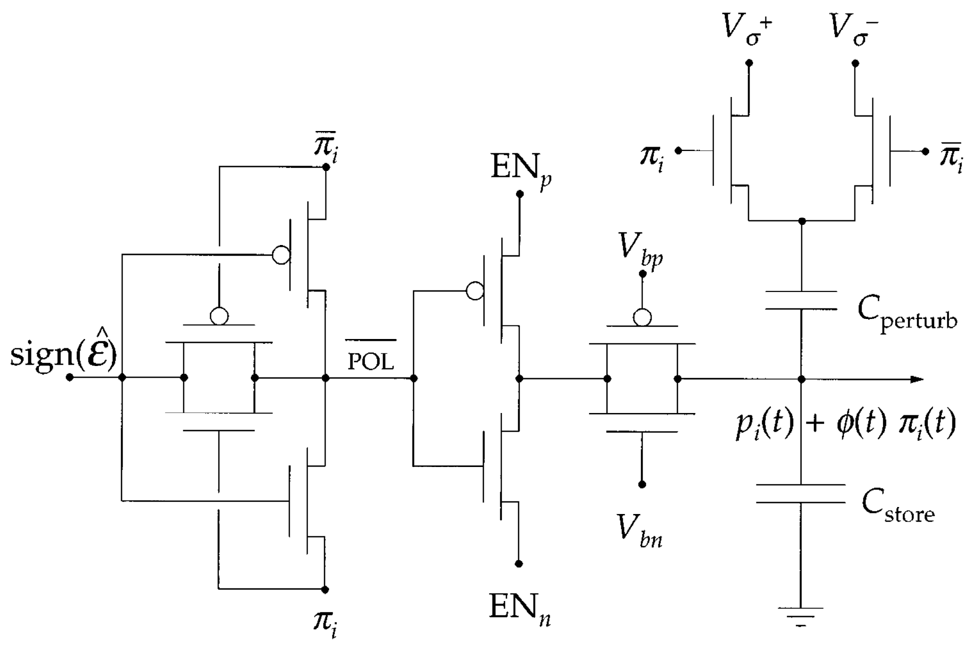


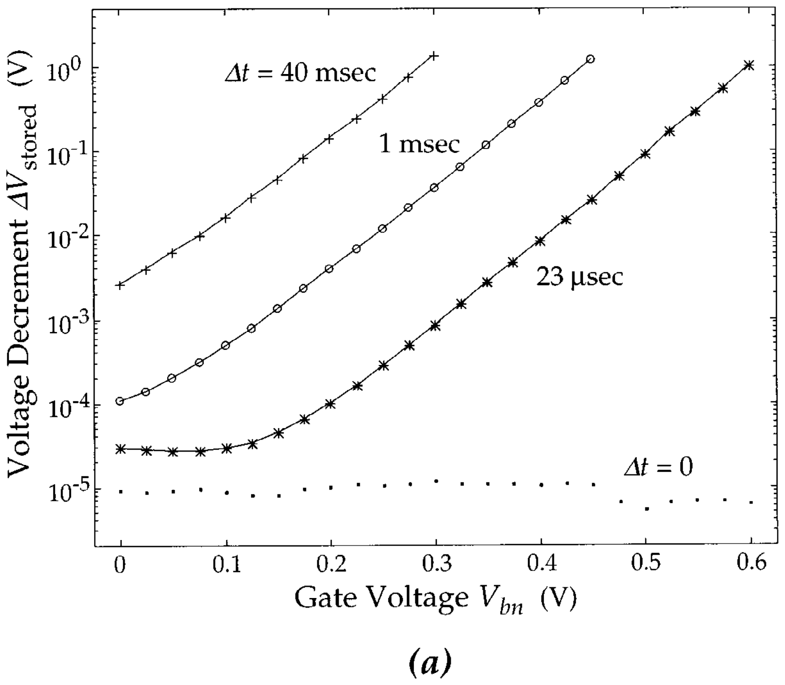
Fig. 7. Circuit schematic of a learning cell implementing stochastic error descent, using the charge pump adaptive element.

the most critical element in limiting learning performance is the quality of the parameter update increments and decrements, in particular the correctness of their polarity. Relative fluctuations in amplitude of the learning updates do not affect the learning process to first order, since their effect is equivalent to relative fluctuations in the learning rate. On the other hand, errors in the polarity of the learning updates might adversely affect learning performance even at small update amplitudes. Therefore, we investigate practical limits as imposed by the binary controlled charge-pump adaptive element in Figures 6, which controls the polarity and amplitude of the updates for learning as well as dynamic refresh.

Measurements on a charge pump with $C = 0.5$ pF fabricated in a 2 μm CMOS process are shown in Figure 8. Under pulsed activation of EN$_n$ and EN$_p$, the resulting voltage increments and decrements are recorded as a function of the gate bias voltages V_{bn} and V_{bp}, for both polarities of POL, and for three different values of the pulse width Δt (23 μsec, 1 msec and 40 msec). In all tests, the pulse period extends 2 msec beyond the pulse width. The exponential subthreshold characteristics are evident from Figure 8, with increments and decrements spanning four orders of magnitude in amplitude. The lower limit is mainly determined by junction diode leakage currents, as shown in Figure 8 (a) for $\Delta t = 0$ (0.01 mV per 2 msec interval at room temperature). This is more than adequate to accommodate learning over a typical range of learning rates. Also, the binary control POL of the polarity of the update is effective for increments and decrements down to 0.05 mV in amplitude, corresponding to charge transfers of only a few hundred electrons.

4.4. Architecture

The general structure of neuromorphic information processing systems has some properties differentiating them from some more conventional human-engineered computing machinery, which are typically optimized for general-purpose sequential digital programming. Some of the desirable properties for neuromorphic architectures are: fault-tolerance and robustness to noise through a redundant distributed representation, robustness to changes in operating conditions through on-line adaptation, real-time bandwidth through massive parallelism, and modularity as a result of locality in space and time. We illustrate these properties in the two architectures for supervised and reinforcement learning in Figures 1 and 2. Since both architectures are similar, a distinction between them will not explicitly be made.



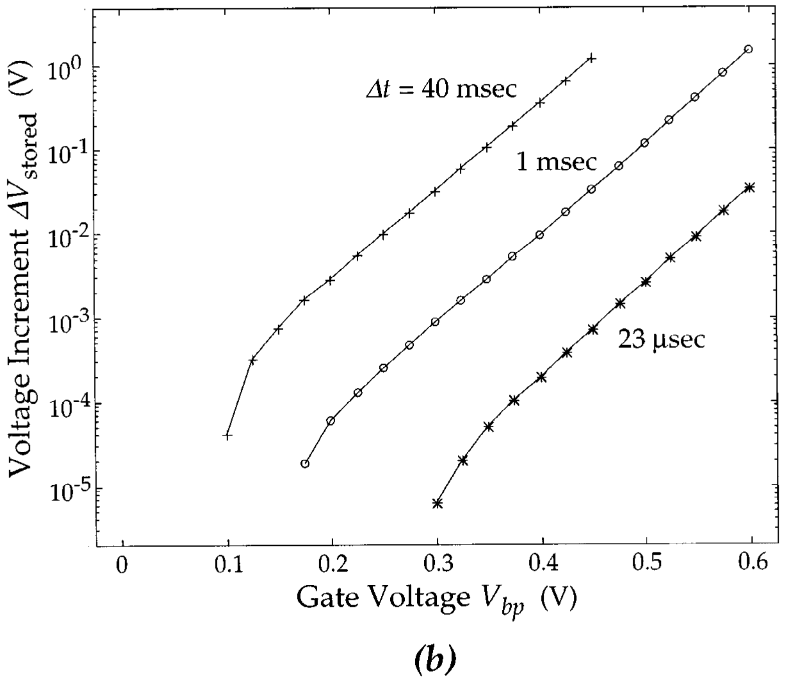


Fig. 8. Measured characteristics of charge-pump adaptive element in 2 μm CMOS with $C = 0.5$ pF. *(a)* n-type decrements (POL = 0); *(b)* p-type increments (POL = 1).

Fault-Tolerance Through Statistical Averaging in a Distributed Representation Direct implementation of gradient descent, based on an explicit model of the network, is prone to errors due to unaccounted discrepancies in the network model and mismatches in the physical implementation of the gradient. This is due to the localized representation in the computation of the gradient as calculated from the model, in which any discrepancy in one part may drastically affect the final result. For this and other reasons, it is unlikely that biology performs gradient calculation on complex systems such as recurrent neural networks with continuous-time dynamics. Stochastic error descent avoids errors of various kinds by physically probing the gradient onto the system rather than deriving it. Using simultaneous and uncorrelated parallel perturbations of the weights, the effect of a single error on the outcome is thus significantly reduced, by virtue of the statistical nature of the computation. However, critical in the accuracy of the implemented learning system is the precise derivation and faithful distribution of the global learning signals $\hat{\mathcal{E}}(t)$ and $\hat{r}(t)$. Stictly speaking, it is essential only to guarantee the correct *polarity* and not the exact amplitude of the global learning signals, as implemented in Figure 7.

Robustness to Changes in the Environment Through On-Line Adaptation This property is inherent to the on-line incremental nature of the studied supervised and reinforcement learning algorithms, which track structural changes in $\hat{\mathcal{E}}(t)$ or $\hat{r}(t)$ on a characteristic time scale determined by learning rate constants such as η or α and β. Learning rates can be reduced as convergence is approached, as in the popular notion in cognitive neuroscience that neural plasticity decreases with age and experience.

Real-Time Bandwidth Through Parallelism All learning operations are performed in parallel, with exception of the three-phase perturbation scheme (5) or (17) to obtain the differential index $\hat{\mathcal{E}}$ under sequential activation of complementary perturbations $\boldsymbol{\pi}$ and $-\boldsymbol{\pi}$. We note that the synchronous three-phase scheme is not essential and could be replaced by an asynchronous perturbation scheme as in [9] and [41]. While this probably resembles biology more closely, the synchronous gradient estimate (4) using complementary perturbations is computationally more efficient as it cancels error terms up to second order in the perturbation strength σ [17]. In the asynchronous scheme, one could envision the role of random noise naturally present in biological systems as a source of perturbations, although it is not clear how noise sources can be effectively isolated to produce the correlation measures necessary for gradient estimation.

Modular Architecture with Local Connectivity The learning operations are local in the sense that a need for excessive global interconnects between distant cells is avoided. The global signals are few in number and common for all cells, which implies that no signal interconnects are required *between* cells across the learning architecture, but all global signals are communicated uniformly *across* cells instead. This allows to embed the learning cells directly into the network (or adaptive critic) architecture, where they interface physically with the synapses they adapt, as in biological systems. The common global signals include the differential index $\hat{\mathcal{E}}(t)$ and reinforcement signal $\hat{r}(t)$, besides common bias levels and timing signals. $\hat{\mathcal{E}}(t)$ and $\hat{r}(t)$ are obtained by any global mechanism that quantifies the "fitness" of the network response in terms of teacher target values or discrete rewards (punishments). Physiological experiments support evidence of local (hebbian [5]) and sparsely globally interconnected (reinforcement [6]) mechanisms of learning and adaptation in biological neural systems [3], [4].

5. Conclusion

Neuromorphic analog VLSI architectures for a general class of learning tasks have been presented, along with key components in their analog VLSI circuit implementation. The architectures make use of distributed stochastic techniques for robust estimation of gradient information, accurately probing the effect of parameter changes on the performance of the network. Two architectures have been presented: one implementing stochastic error-descent for supervised learning, and the other implementing a novel stochastic variant on a generalized form of reinforcement learning. The two architectures are similar in structure, and both are suitable for scalable and robust analog VLSI implementation.

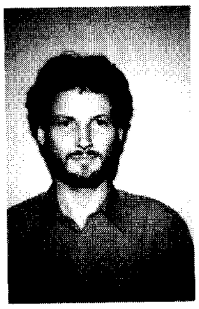
While both learning architectures can operate on (and be integrated in) arbitrary systems of which the characteristics and structure does not need to be known, the reinforcement learning architecture additionally supports a more general form of learning, using an arbitrary, externally supplied, reward or punishment signal. This allows the development of more powerful, generally applicable devices for "black-box" sensor-motor control which make no prior assumptions on the structure of the network and the specifics of the desired network response.

We presented simulation results that demonstrate the effectiveness of perturbative stochastic gradient estimation for reinforcement learning, applied to adaptive oversampled data conversion. The neural classifier controlling the second-order noise-shaping modulator was trained for optimal performance with no more evaluative feedback than a discrete failure signal indicating whenever any of the modulation integrators saturate. The critical part in the VLSI implementation of adaptive systems of this type is the precision of the *polarity*, rather than the amplitude, of the implemented weight parameter updates. Measurements on a binary controlled charge-pump in 2 μm CMOS have demonstrated voltage increments and decrements of precise polarity spanning four orders of magnitude in amplitude, with charge transfers down to a few hundred electrons.

References

1. C. A. Mead, "Neuromorphic electronic systems." *Proceedings of the IEEE* 78(10), pp. 1629–1639, 1990.
2. C. A. Mead, *Analog VLSI and Neural Systems*. Addison-Wesley: Reading, MA, 1989.
3. G. M. Shepherd, *The Synaptic Organization of the Brain*. 3rd ed. Oxford Univ. Press: New York, NY, 1992.
4. P. S. Churchland and T. J. Sejnowski, *The Computational Brain*. MIT Press: Cambridge, MA, 1990.
5. S. R. Kelso and T. H. Brown, "Differential conditioning of associative synaptic enhancement in Hippocampal brain slices." *Science* 232, pp. 85–87, 1986.
6. R. D. Hawkins, T. W. Abrams, T. J. Carew, and E. R. Kandell, "A cellular mechanism of classical conditioning in *Aplysia*: activity-dependent amplification of presynaptic facilitation." *Science* 219, pp. 400–405, 1983.
7. P. R. Montague, P. Dayan, C. Person and T. J. Sejnowski, "Bee foraging in uncertain environments using predictive Hebbian learning." *Nature* 377(6551), pp. 725–728, 1996.
8. C. A. Mead and M. Ismail, Eds., *Analog VLSI Implementation of Neural Systems*. Kluwer: Norwell, MA, 1989.
9. A. Dembo and T. Kailath, "Model-free distributed learning." *IEEE Transactions on Neural Networks* 1(1), pp. 58–70, 1990.
10. S. Grossberg, "A neural model of attention, reinforcement, and discrimination learning." *International Review of Neurobiology* 18, pp. 263–327, 1975.
11. A. G. Barto, R. S. Sutton, and C. W. Anderson, "Neuronlike adaptive elements that can solve difficult learning control problems." *IEEE Transactions on Systems, Man, and Cybernetics* 13(5), pp. 834–846, 1983.
12. R. S. Sutton, "Learning to predict by the methods of temporal differences." *Machine Learning* 3, pp. 9–44, 1988.
13. C. Watkins and P. Dayan, "Q-Learning." *Machine Learning* 8, pp. 279–292, 1992.
14. P. J. Werbos, "A menu of designs for reinforcement learning over time," in *Neural Networks for Control* (W. T. Miller, R. S. Sutton and P. J. Werbos, eds.). MIT Press: Cambridge, MA, 1990, pp. 67–95.
15. G. Cauwenberghs, "A fast stochastic error-descent algorithm for supervised learning and optimization," in *Advances in Neural Information Processing Systems*, vol. 5. Morgan Kaufman: San Mateo, CA, 1993, pp. 244–251.
16. P. Werbos, "Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences," Ph.D. dissertation, 1974. Reprinted in P. Werbos, *The Roots of Backpropagation*. Wiley: New York, 1993.
17. H. J. Kushner, and D. S. Clark, *Stochastic Approximation Methods for Constrained and Unconstrained Systems*. Springer-Verlag: New York, NY, 1978.
18. H. Robins and S. Monro, "A stochastic approximation method." *Annals of Mathematical Statistics* 22, pp. 400–407, 1951.
19. J. C. Spall, "A stochastic approximation technique for generating maximum likelihood parameter estimates." *Proceedings of the 1987 American Control Conference*, Minneapolis, MN, 1987.
20. M. A. Styblinski and T.-S. Tang, "Experiments in nonconvex optimization: Stochastic approximation with function smoothing and simulated annealing." *Neural Networks* 3(4), pp. 467–483, 1990.
21. M. Jabri and B. Flower, "Weight perturbation: An optimal architecture and learning technique for analog VLSI feedforward and recurrent multilayered networks." *IEEE Transactions on Neural Networks* 3(1), pp. 154–157, 1992.
22. J. Alspector, R. Meir, B. Yuhas, and A. Jayakumar, "A parallel gradient descent method for learning in analog VLSI neural networks," in *Advances in Neural Information Processing Systems*, vol. 5. Morgan Kaufman: San Mateo, CA, 1993, pp. 836–844.
23. B. Flower and M. Jabri, "Summed weight neuron perturbation: An $\mathcal{O}(n)$ improvement over weight perturbation," in *Advances in Neural Information Processing Systems*, vol. 5, Morgan Kaufman: San Mateo, CA, 1993, pp. 212–219.
24. G. Cauwenberghs, "A learning analog neural network chip with continuous-recurrent dynamics," in *Advances in Neural Information Processing Systems*, vol. 6. Morgan Kaufman: San Mateo, CA, 1994, pp. 858–865.
25. G. Cauwenberghs, "An analog VLSI recurrent neural network learning a continuous-time trajectory." *IEEE Transactions on Neural Networks* 7(2), March 1996.
26. F. Pineda, "Mean-field theory for batched-TD(λ)," submitted to *Neural Computation*, 1996.
27. S. Grossberg and D. S. Levine, "Neural dynamics of attentionally modulated pavlovian conditioning: Blocking, interstimulus interval, and secondary reinforcement." *Applied Optics* 26, pp. 5015–5030, 1987.
28. E. Niebur and C. Koch, "A model for the neuronal implementation of selective visual attention based on temporal correlation among neurons." *Journal of Computational Neuroscience* 1, pp. 141–158, 1994.
29. G. Cauwenberghs, "Reinforcement learning in a nonlinear noise shaping oversampled A/D converter." To appear in *Proc. Int. Symp. Circuits and Systems*, Hong Kong, June 1997.

30. J. C. Candy and G. C. Temes, "Oversampled methods for A/D and D/A conversion," in *Oversampled Delta-Sigma Data Converters*, IEEE Press, 1992, pp. 1–29.
31. E. Vittoz and J. Fellrath, "CMOS analog integrated circuits based on weak inversion operation." *IEEE Journal on Solid-State Circuits* 12(3), pp. 224–231, 1977.
32. A. G. Andreou, K. A. Boahen, P. O. Pouliquen, A. Pavasovic, R. E. Jenkins, and K. Strohbehn, "Current-mode subthreshold MOS circuits for analog VLSI neural systems." *IEEE Transactions on Neural Networks* 2(2), pp. 205–213, 1991.
33. A. L. Hodgkin and A. F. Huxley, "Current carried by sodium and potassium ions through the membrane of the giant axon of *Loligo*." *Journal of Physiology* 116, pp. 449–472, 1952.
34. C. Diorio, P. Hassler, B. Minch and C. A. Mead, "A single-transistor silicon synapse." To appear in *IEEE Transactions on Electron Devices*.
35. C. A. Mead, "Adaptive retina," in *Analog VLSI Implementation of Neural Systems* (C. Mead and M. Ismail, eds.). Kluwer Academic Pub.: Norwell, MA, 1989, pp. 239–246.
36. G. Cauwenberghs, C. F. Neugebauer, and A. Yariv, "Analysis and verification of an analog VLSI outer-product incremental learning system." *IEEE Transactions on Neural Networks* 3(3), pp. 488–497, 1992.
37. Y. Horio, and S. Nakamura, "Analog memories for VLSI Nneurocomputing," in *Artificial Neural Networks: Paradigms, Applications, and Hardware Implementations* (C. Lau and E. Sanchez-Sinencio, eds.), IEEE Press, 1992, pp. 344–363.
38. G. Cauwenberghs, and A. Yariv, "Fault-tolerant dynamic multi-level storage in analog VLSI." *IEEE Transactions on Circuits and Systems II* 41(12), pp. 827–829, 1994.
39. G. Cauwenberghs, "Analog VLSI long-term dynamic storage," in *Proceedings of the International Symposium on Circuits and Systems*, Atlanta, GA, 1996.
40. G. Cauwenberghs, "A micropower CMOS algorithmic A/D/A converter." *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications* 42(11), pp. 913–919, 1995.
41. D. Kirk, D. Kerns, K. Fleischer, and A. Barr, "Analog VLSI implementation of gradient descent," in *Advances in Neural Information Processing Systems*, vol. 5. Morgan Kaufman: San Mateo, CA, 1993, pp. 789–796.



Gert Cauwenberghs received the engineer's degree in applied physics from the Vrije Universiteit Brussel, Belgium, in 1988, and the M.S. and Ph.D. degrees in electrical engineering from the California Institute of Technology in 1989 and 1994, respectively. In 1994, he joined Johns Hopkins University as an assistant professor in electrical and computer engineering. His research covers analog and digital VLSI circuits, systems and algorithms for parallel signal processing and adaptive neural computation.