Biol Cybern (2009) 100:147–158
DOI 10.1007/s00422-008-0288-z

ORIGINAL PAPER

Biological Cybernetics

Learning flexible sensori-motor mappings in a complex network

Eleni Vasilaki · Stefano Fusi · Xiao-Jing Wang · Walter Senn

Received: 31 July 2007 / Accepted: 10 December 2008 / Published online: 20 January 2009

Abstract Given the complex structure of the brain, how can synaptic plasticity explain the learning and forgetting of associations when these are continuously changing? We address this question by studying different reinforcement learning rules in a multilayer network in order to reproduce monkey behavior in a visuomotor association task. Our model can only reproduce the learning performance of the monkey if the synaptic modifications depend on the pre- and postsynaptic activity, and if the intrinsic level of stochasticity is low. This favored learning rule is based on reward modulated Hebbian synaptic plasticity and shows the interesting feature that the learning performance does not substantially degrade when adding layers to the network, even for a complex problem.

E. Vasilaki · W. Senn
Institute of Physiology, University of Bern,
Buehlplatz 5, 3012 Bern, Switzerland

E. Vasilaki (✉)
Laboratory of Computational Neuroscience, EPFL,
Station 15, 1015 Lausanne, Switzerland
e-mail: eleni.vasilaki@epfl.ch

S. Fusi
Institute of Neuroinformatics, ETH/University of Zurich,
Winthertihurerstrasse 190, 8057 Zurich, Switzerland

S. Fusi
Center for Neurobiology and Behavior,
Columbia University College of Physicians and Surgeons,
New York, NY 10032, USA

X.-J. Wang
Department of Neurobiology,
Kavli Institute for Neuroscience,
Yale University School of Medicine,
333 Cedar Street, New Haven, CT 06520, USA

Keywords Reward-modulated · Hebbian · Learning · Multilayer · Visuomotor task

1 Introduction

Experimental work of Asaad et al. (1998) has revealed mechanisms which might be involved in a visuomotor learning task performed by monkeys. In this task, the monkeys are trained to associate visual stimuli (pictures) with delayed saccadic movements, left or right, with associations being reversed from time to time. In Fusi et al. (2007), the authors hypothesize that learning and forgetting associations happens purely by synaptic modification mechanisms, on multiple time scales. To support this hypothesis, they reproduce the experimental results of Asaad et al. with a simplified one-layer network, and make further predictions which are also experimentally verified.

The objective of this paper is to investigate whether there are learning prescriptions which allow us to extend the results of Fusi et al. (2007) to the case of a multilayer network. One major limitation of the analysis of Fusi et al. (2007) was the assumption that the plastic input to the decision network was generated by a single layer of synaptic inputs. In fact, many algorithms result in a loss of performance as the number of hidden layers is increased, although this problem is often not explicitly addressed Fahlman and Lebiere (1991), Bak and Chialvo (2001), Hinton and Salakhutdinov (2006); Bengio and LeCun (2007). We therefore ask: what are the appropriate synaptic mechanisms which allow us to reproduce the learning performance observed in this experiment within a complex network?

To address this question we consider three reinforcement learning algorithms: a novel form of reward-modulated adaptive Hebbian learning (RAH), together with the previously described node perturbation (NP) (Werfel et al. 2005) and the associative reward penalty (ARP) algorithm (Mazzoni et al. 1991; Barto 1985, 1989; Barto and Jordan 1989). We investigate the performance of these algorithms on the monkey association paradigm (Asaad et al. 1998; Fusi et al. 2007) and different network architectures.

We find that both RAH and ARP can reproduce the monkey data, but not NP. In fact, adding a few layers slows down learning, much less in the case of RAH and ARP and much more in the case of NP. Nevertheless, the problem of learning with many layers is present in more general architectures (Fahlman and Lebiere 1991; Bak and Chialvo 2001; Hinton and Salakhutdinov 2006; Bengio and LeCun 2007) and though there are solutions for particular cases (Hinton and Salakhutdinov 2006; Bengio and LeCun 2007), no global solution exists. Our simulations imply that the less noise driven a rule is, the better its performance in a deep network structure.

2 Methods

2.1 Monkey experiment

In an experiment of an oculomotor paradigm (Asaad et al. 1998), monkeys were trained to associate visual stimuli (pictures) with delayed saccadic movements, left or right (Fig. 1a). In particular, two visual stimuli (A and B) were initially associated with left and right saccadic responses (L and R), respectively. From time to time the associations were reversed (from A→L and B→R to A→R and B→L, and vice versa) without any warning to the animal. Two other visual stimuli (C and D) were consistently associated with a fixed motor response throughout the experiment and they were randomly intermixed with the first two stimuli A and B. The reward in successful trials was given in the form of juice.

Whenever the associations of A and B were reversed, the monkey quickly forgot the old associations and slowly learned the new ones. After reversal, the animal almost immediately reverted to random responses followed by learning the new associations within 30 trials (Fig. 1b). It is worth noting that, within a block of trials, every mistake was bringing the system back to balanced configuration: the probability for a correct choice in the following presentation was reset to chance level 50%, independently of the previous success history, as shown in Fig. 1c. If we represent a correct answer with 1 and a mistake with 0, Fig. 1c shows the probability of finding a 1 after a sequence of trials of the form 111–10. Hence, we counted the sequences 111–101 and 111–100 between two reversals, and plotted the percentage of correct answers after an error (111–101), versus the number of consecutive correct trials (the number of 1s) before the error.

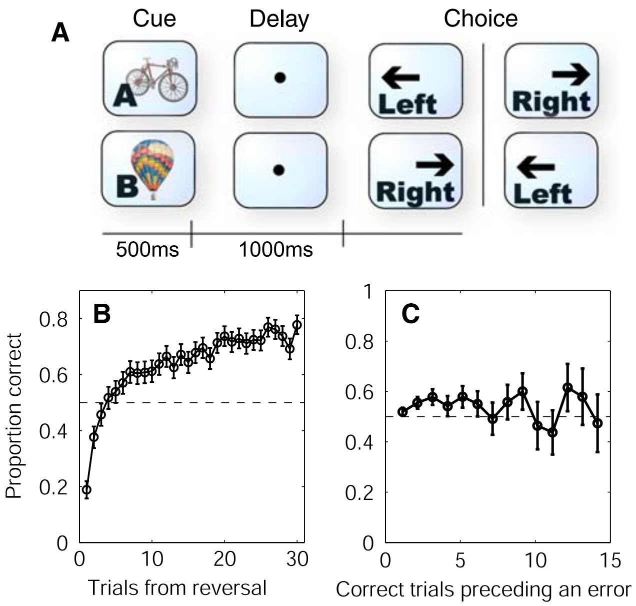

Fig. 1 Visuomotor association experiment from Asaad et al. (1998) and analysis from Fusi et al. (2007). **a** Task protocol: the monkey learns to associate four stimuli either with a left or right saccadic movement. The associations for two stimuli (shown in the figure) are reversed at unpredictable times and without explicit cues. For the other two stimuli (not shown) the associations are always the same. **b** The proportion of correct responses, averaged across all the blocks, is plotted against the number of trials from the time of reversal. Initially the monkey keeps responding according to the previously rewarded associations and makes the greatest number of mistakes. **c** After a single mistake, the monkey forgets quickly, whereupon performance rises to chance level (50%). The errors considered for the analysis can occur at any time within a block, and not necessarily immediately after reversal

The learning process is roughly independent for each of the two stimuli (picture A and B). In other words, whether the monkey has seen one object to be reversed does not greatly influence performance on the other, see Fusi et al. (2007). In what follows, we reproduce the curves in Fig. 1b and c as well as other psychophysical data. Under the assumption that visual stimuli representations are coded sparsely, we show that the network is able to learn the associations with little interference between the patterns.

2.2 The model

We use a multilayer feedforward network with two competing output neurons, each representing a population selective to the intended saccadic movements (“right” or “left”), see Fig. 2. The two populations compete through a group of inhibitory neurons. The synaptic weights of the network are analog (i.e., real valued), bounded and the neurons have two states: active and silent.

Since we focus on the dynamics of learning, the details of the neuron model is less crucial. In fact, the collective computational properties of many binary neurons are equivalent

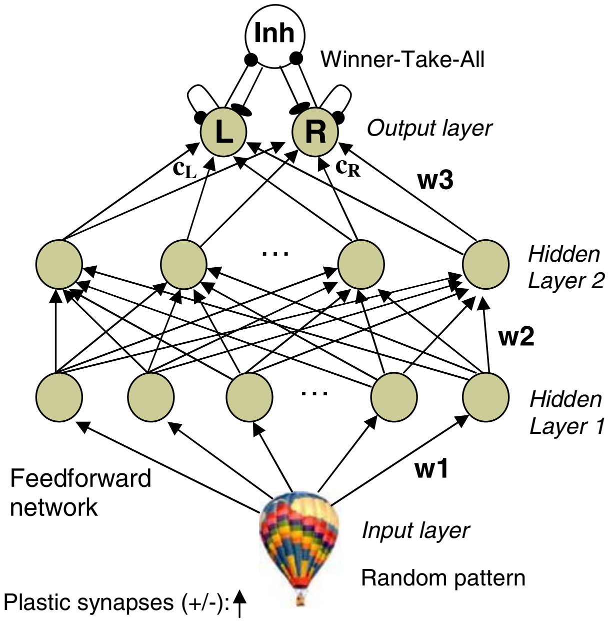

Fig. 2 Network architecture. Two hidden layer feedforward network with random input pattern, $w1$ denotes the synaptic connections from the input layer to hidden layer 1, $w2$ the connections from hidden layer 1 to hidden layer 2 and $w3$ the connections from hidden layer 2 to the output layer. The symbols c_R and c_L represent the feedforward synaptic inputs to the right and left selective populations, respectively, which compete through a group of inhibitory neurons Inh (winner-takes-all mechanism). The dynamics of the WTA mechanism are modeled via a sigmoidal function, see Sect. 2. Plastic synapses, which can be both excitatory and inhibitory, are plotted as arrows. Excitatory-only synaptic connections are depicted with a round-shaped ending and inhibitory only with an ellipsoid-shaped ending

to neurons with graded response (Hopfield 1984). Moreover, the dynamics of the readout network can safely be replaced by a stochastic winner-takes-all (WTA) mechanism, without the need to implement this explicitly in neuronal terms, as is done, e.g., in Fusi et al. (2007), Wang (2002), Wang (2005).

We represent the two stimuli A and B as a pair of fixed random binary patterns created with coding level $f = 0.01$, i.e., with probability f of each component of a pattern being active and probability $1 - f$ of being inactive. This results in a probability of 10^{-4} of having the same neuron activated by two patterns. This condition is imposed to model the experimental results showing independence in learning the patterns. A stimulus, either A or B, randomly chosen, is presented to the network and the total synaptic input c_L and c_R to the two competing output neurons "left" and "right" is calculated. The probability of "left" winning over "right" follows a sigmoidal function of the difference between the synaptic inputs c_L and c_R, with the parameter σ controlling the steepness of the sigmoid:

$$Q_L = \frac{1}{1 + \mathrm{e}^{-(c_L - c_R)/\sigma}}. \tag{1}$$

The network chooses randomly one of the two saccadic movements with equal probability when the inputs activated by the visual stimuli to the two output neurons are perfectly balanced. Any imbalance in the input would bias the decision. As the difference increases in favor of, e.g., the left neuron, the probability for the model to choose left increases, and eventually the model's behavior becomes close to deterministic. Reward is given if the answer is correct and the weights are updated according to the learning rule. After a random number of trials (between 30 and 60) the association is reversed.

2.3 Learning rules

2.3.1 Reward-modulated adaptive Hebbian learning

Motivation for the rule. We choose a Hebbian-type learning rule that allows for bounding the synaptic weights, to maintain a degree of biological realism. Many reinforcement learning algorithms (Sutton and Barto 1998) described in the REINFORCE framework (Williams 1992; Dayan 1990) assume noise on the neuronal level in order to explore the space of the network states. Here, we suggest that this exploration could be mediated by the randomness induced by the decision-making (probabilistic WTA) network, the stochasticity in choosing the pattern or the induction of synaptic modifications which is inherently stochastic (see also Amit and Fusi 1994).

Rule formulation. The (total) synaptic input h_i^μ of a neuron is calculated according the following equation:

$$h_i^\mu = \frac{1}{N+1}\left(\sum_{j=1}^{N} w_{ij} y_j^\mu - b_i\right) \tag{2}$$

where y_j^μ is the activity of the presynaptic neuron with j taking values from 1 to N, w_{ij} is the synaptic weight from the presynaptic neuron j to the postsynaptic neuron i, b_i is the bias to neuron i, and μ is an index over the patterns presented in the network. The activity of the neuron i is given by:

$$y_i^\mu = \mathcal{H}(h_i^\mu), \tag{3}$$

where $\mathcal{H}$ is the Heaviside (step) function.

The synaptic changes are given by the equation:

$$\Delta w_{ij}^0 = \begin{cases} q^+ \left(y_i^\mu - P_{ij}^{11}\right) y_j^\mu & \text{if successful trial} \\ -q^- \left(y_i^\mu - P_{ij}^{11}\right) y_j^\mu & \text{if unsuccessful trial,} \end{cases} \tag{4}$$

where q^+ is the learning rate in rewarded trials, q^- is the learning rate in failures, and P_{ij}^{11} is the probability of the postsynaptic neuron being active given that the presynaptic neuron is active, i.e.,

$$P_{ij}^{11} = \left\langle y_i^{\mu} = 1 | y_j^{\mu} = 1 \right\rangle_{\mu}. \qquad (5)$$

This probability reflects the averaged postsynaptic activity across all pattern presentations. It plays the role of an long-term potentiation (LTP)/long-term depression (LTD) threshold as, depending on its value, it determines the amount of LTP or LTD to be induced. To calculate the P^{11} threshold we use a running mean, which could be locally implemented in the synapse:

$$\Delta \widetilde{P_{ij}^{11}} = \rho \left(y_i^{\mu} - \widetilde{P_{ij}^{11}} \right) y_j^{\mu}, \qquad (6)$$

where ρ defines the update rate of the running mean and $\widetilde{P_{ij}^{11}}$ is our estimate of P_{ij}^{11}. Since $\widetilde{P_{ij}^{11}}$ is a conditional probability, the update $\Delta \widetilde{P_{ij}^{11}}$ is zero in the case that the presynaptic neuron is inactive. We assume that in between these updates $\widetilde{P_{ij}^{11}}$ does not decay in time, and hence implicitly assume a long memory time constant for $\widetilde{P_{ij}^{11}}$ compared with the rhythm of pattern presentation, as is also assumed for w_{ij}.

For biological realism and consistency with previous theory (Senn and Fusi 2005a) we require that the synapses are soft-bounded, i.e., that the synaptic changes (Eq. 4) are modified through the effect of the synaptic boundaries $w_{\min}$ and $w_{\max}$:

$$\Delta \widetilde{w}_{ij} = \begin{cases} \Delta w_{ij}^0 (w_{\max} - w_{ij}) & \text{if} \quad \Delta w_{ij}^0 > 0 \\ \Delta w_{ij}^0 (w_{ij} - w_{\min}) & \text{if} \quad \Delta w_{ij}^0 < 0, \end{cases} \qquad (7)$$

where $w_{\max}$ and $w_{\min}$ are, respectively, the maximum and minimum values that a synapse can take. This equation restricts synaptic weights between $[w_{\min}, w_{\max}]$, as long as the learning rate is sufficiently small.

In addition, the algorithm requires the implementation of a stop-learning criterion (similar to Senn and Fusi 2005a) which restrict the updates in rewarded cases to those synapses whose postsynaptic neuron has a total synaptic input $-\delta_0 < h_i^{\mu} < \delta_0$, δ_0 being a small positive number:

$$\Delta w_{ij} = \begin{cases} \Delta \widetilde{w}_{ij} & \text{if} \quad \text{unsuccessful trial} \\ \Delta \widetilde{w}_{ij} & \text{if} \quad \text{successful trial and } -\delta_0 < h_i^{\mu} < \delta_0 \\ 0 & \text{elsewhere.} \end{cases} \qquad (8)$$

The importance of implementing a stop-learning criterion will be further discussed in Sect. 3.3.

For the bias, we assume an additional input unit always active (with $y_j^{\mu} = 1$) and treat its synapse just like another weight that changes according to the Eqs. 4–8. The only source of stochasticity is the probabilistic winner-takes-all and the randomness in choosing the input pattern. It is emphasized that the RAH rule is an ad hoc rule and cannot be proven to improve task performance in general. The rule with a fixed value for P_{ij}^{11} corresponds to the one used in Fusi et al. (2007) (cf. also Soltani et al. 2006b).

Simulation parameters for reproducing the monkey data: Stimuli are represented as random binary patterns. We employ sparse coding, i.e., the random patterns were created with code level $f = 0.01$. Parameters were chosen as follows: probabilistic WTA $\sigma = 0.02$, learning rate in absence of reward $q^- = 0.02$, learning rate in case of reward $q^+ = 0.005$, and time constant of the running mean $\tau = 10^{-3}$. The parameter δ_0 was set to half of the mean value of the absolute synaptic current for the input layer, i.e., 0.5×10^{-4}. Synaptic weights are uniformly distributed in $[-1, 1]$. Synaptic boundaries are set to $w_{\max} = 1$ and $w_{\min} = -1$.

2.3.2 Node perturbation

NP (Werfel et al. 2005) is a non-Hebbian algorithm based on the idea that random search can lead to sophisticated learning. Unlike Hebbian learning, this rule does not exploit presynaptic and postsynaptic correlations. The synaptic modifications are governed by the random fluctuations in the postsynaptic neuron independently of the postsynaptic activity per se. If reward is given to the system, the synaptic modifications follow the direction of the random fluctuations; if not, they follow the opposite direction.

We implemented NP is its pure form, where the synaptic changes are given by the equation:

$$\Delta w_{ij} = \begin{cases} q^+ \; \xi \; y_j^{\mu} & \text{if} \quad \text{successful trial} \\ -q^- \; \xi \; y_j^{\mu} & \text{if} \quad \text{unsuccessful trial} \end{cases} \qquad (9)$$

with q^+ being the learning rate in a successful trial, q^- the learning rate in an unsuccessful trial, y_j the presynaptic neuronal activity, and ξ Gaussian-distributed noise inserted at the postsynaptic neuron. To be consistent with the other algorithms, neuronal activities are binary, and are calculated from Eqs. 2–3.

Simulation parameters for reproducing the monkey data: Setup was similar to RAH but, in order to improve the performance of NP, we allowed a deterministic WTA and noise decay. Noise ξ is Gaussian with $\sigma = 1$ and decays proportionally to $1 - \langle R \rangle$, where $\langle R \rangle$ is the mean network performance with values between 0 and 1 ($R = 1$ in a successful trial and $R = 0$ in an unsuccessful trial). The learning rate in both successful and unsuccessful trials is $q^+ = q^- = 0.01$. Initial synaptic weights are uniformly distributed in $[-1, 1]$. In NP we relieved the synapses from their bounds to allow for an improvement of the signal-to-noise ratio by an unconstraint growth of the weights.

2.3.3 Associative reward penalty

ARP (Mazzoni et al. 1991; Barto 1985, 1989; Barto and Jordan 1989) is a Hebbian-type algorithm, well established in machine learning.

We can rewrite the ARP equation (Mazzoni et al. 1991) as follows:

$$\Delta w_{ij}^{0} = \begin{cases} q^{+}(y_i^{\mu} - p_i^{\mu})y_j^{\mu} & \text{if trial successful} \\ -q^{-}(y_i^{\mu} - (1 - p_i^{\mu}))y_j^{\mu} & \text{if unsuccessful,} \end{cases} \qquad (10)$$

where y_i^{μ} is the binary output of the ith neuron, $p_i^{\mu} = \langle y_i^{\mu} \rangle_{y_i^{\mu}=0,1}$ its probability of firing given the applied pattern μ, w_{ij} the synaptic weight from neuron j to neuron i, q^{+} the learning rate in rewarded trials, and q^{-} the learning rate in nonreward trials. The probability p_i^{μ} is assumed to be a sigmoidal function of h_i^{μ} (Eq. 2),

$$p_i = \frac{1}{1 + \mathrm{e}^{-h_i^{\mu}/\sigma}}. \qquad (11)$$

Typically $\sigma = 1$, but in the following we will treat σ as a parameter of the learning rule. In case of reward, the LTP/LTD threshold is the term p_i, corresponding to our P_{ij}^{11}. In case of punishment, the LTP/LTD threshold is $1 - p_i$.

We restrict the synaptic weights according to the following equation (soft bounds):

$$\Delta w_{ij} = \begin{cases} \Delta w_{ij}^{0}(w_{\max} - w_{ij}) & \text{if } \Delta w_{ij}^{0} > 0 \\ \Delta w_{ij}^{0}(w_{ij} - w_{\min}) & \text{if } \Delta w_{ij}^{0} < 0. \end{cases} \qquad (12)$$

This algorithm is closely related to our novel rule. Under certain constraints, they become identical, as we show in Sect. 3.1.3. Their main differences are the stochastic nature of the neuron in ARP and the way in which the LTP/LTD threshold is calculated. In RAH this threshold depends on the average activity of the postsynaptic neuron, while in ARP it depends on the postsynaptic current at the particular time step.

Simulation parameters for reproducing the monkey data: Setup was similar to RAH with $q^{+} = 0.025$, $q^{-} = 0.3$, and $\sigma = 2 \times 10^{-4}$. Synaptic weights are uniformly distributed in $[-1, 1]$. Synapses are soft bounded as in RAH.

3 Results

3.1 Reproducing the experimental data with a multilayer network

We apply the RAH algorithm on the network described in Sect. 2, and reproduce with very good accuracy the experimental data from Asaad et al. (1998), Fusi et al. (2007). We plot four performance-related curves and compare them to the outcome of the simulation. The fitting is done by tuning two parameters: the learning rate q^{-} in nonrewarded trials and the learning rate q^{+} in rewarded trials. These parameters are chosen such that there is a good match between the experimental results and simulations shown in Fig. 3a and c. The matching for the rest of the curves follows without any additional parameter tuning. Further, we present simulations using NP (Werfel et al. 2005) and ARP (Mazzoni et al. 1991; Barto 1985, 1989; Barto and Jordan 1989) and analyze the results.

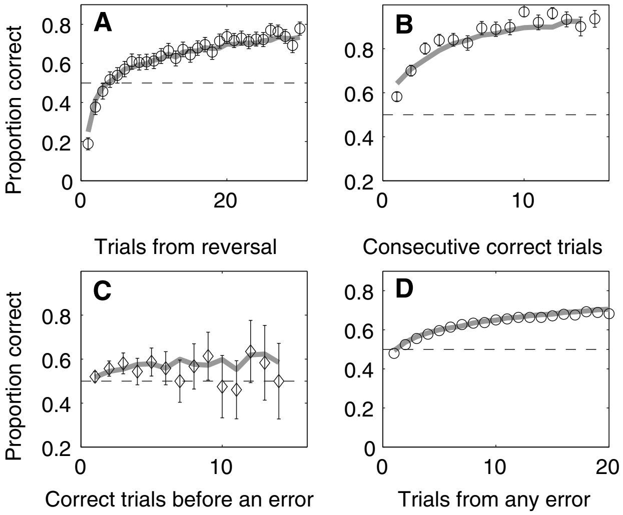

Fig. 3 Reproducing experimental results (*circles*) with a two hidden layers network 100 × 30 × 30 × 2 (*solid line*) and the RAH algorithm. **a** Performance versus number of trials after the reversal of associations. The curve starts at a proportion of 0.2 correct because this represents the counter probability of 0.8 for correctness after successful learning, immediately before the reversal. **b** Probability of having a correct trial after a sequence of N consecutively correct trials. **c** Performance in the next trial after an error in any pattern versus the length of consecutively correct trials before the error. Every single mistake resets the monkey's performance to almost chance level, independently of the previous success history. **d** Development of the average performance in the following trials after any error. The error bars for the data points are negligible

3.1.1 Reward-modulated adaptive Hebbian learning

The model learns the left–right assignment task within 30 trials, similar to the monkey behavior (Fig. 3a). Figure 3b shows the probability of a successful trial versus the number of past consecutively correct trials. Figure 3c shows the probability of having a successful trial after a mistake that follows a number of consecutively correct trials. We observe that, after a single mistake, the response of the monkey is almost random (reset), regardless of the recent history. To reproduce this effect, we set the learning rate in absence of reward four times higher than in rewarded trials. Figure 3d shows learning after any error onwards. Statistics are performed on 500 different sequences of trial length between 300 and 600, with ten reversals in each sequence.

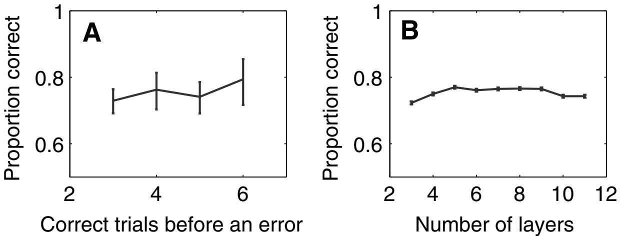

Fig. 4 Results of RAH algorithm on a multilayer network. **a** The mistake in one pattern does not affect the classification of the other pattern in the case where learning is almost achieved. To produce the graph, we extract from the end of a trial block a number of consecutive correct answers, mostly of pattern A, followed by a single error in B, followed by a trial where A is again presented. **b** The performance of the network is retained as we increase the number of layers, without retuning the parameters. Learning can be generalized in a network of many layers

In Fig. 4a, we show in simulations the effect of an error in the classification of pattern B on classifying pattern A, versus the previous history of subsequent correct trials of pattern A (and vice versa). The data are extracted from the end of the trial block, i.e., when performance is over 70%. The plot shows that the mistake in one pattern does not affect the classification of the other pattern, after learning is almost achieved. We attribute this behavior to our choice of sparse-coded input patterns, which possibly leads to different pathways carrying out the information from the input to the output layer for each pattern.

Further, we investigate how the performance changes as the number of layer increases. It is worth noting that adding layers does not require further parameter tuning; the performance remains the same and the monkey data always fit well. Since the monkey learns the task within 30 trials, we plot the performance of the 30th trial versus the number of network layers (Fig. 4b) and demonstrate that it is almost a straight line.

In summary, the network allows us to reproduce the experimental results with very good accuracy, as long as the synaptic changes in case of punishment are four times stronger than in the case of reward and sparse coding is used to create the random patterns.

3.1.2 Node perturbation

NP, implemented in its original formulation (see Sect. 2), is at least two orders of magnitude slower than the monkey performance (Fig. 5) with all parameter sets we tried. We attribute this slow performance to the fact that the algorithm does not exploit the correlations between presynaptic and postsynaptic activity, and therefore searches a larger parameter space for appropriate solutions. In addition, as we demonstrate in Sect. 3.2, the number of layers significantly slows down the performance of NP, at least for our architectural choices. The slow performance by itself excludes NP as a candidate for reproducing the monkey data, and as a consequence we reproduce neither other behavioral data nor the performance with many layers (see also Fig. 8).

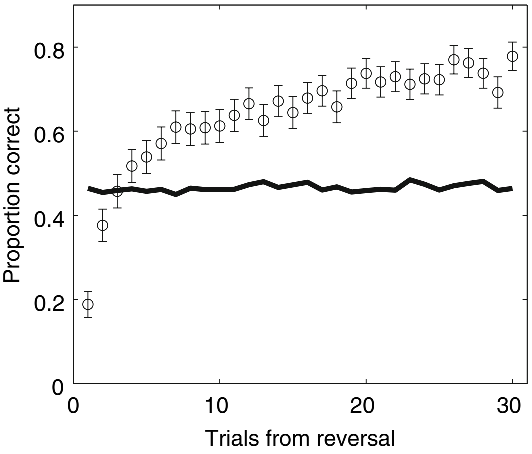

Fig. 5 Reproducing the experimental data with NP. The learning in simulations is two orders of magnitude slower than the experimental data. The network learns equally from successful and unsuccessful trials. Learning mainly from unsuccessful trials did not improve performance

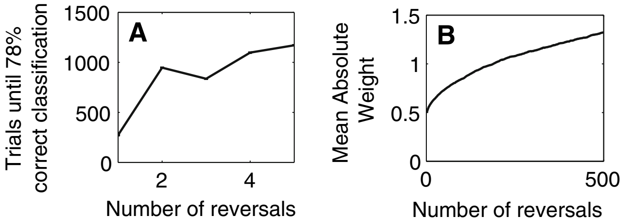

Fig. 6 Learning and forgetting associations with NP. **a** Learning slows down as the number of reversals is increased. **b** Growth of weights versus number of association reversals for NP. It was not possible to add soft bounds, at least with this particular implementation of NP, because the synaptic weights tend to grow outside the initial distribution. Unbounded weights are related to difficulties in forgetting. Statistics for 500 initial conditions

Furthermore, we note that the algorithm has difficulties in forgetting the previously learned associations (Fig. 6a). The time it takes to learn the task increases with the number of the prior reversals. This slowing down arises due to the continuous growth of the synapses with each reversal (Fig. 6b), which prevented us from applying soft bounds. In fact, applying the same synaptic weight bounds to NP as we did for the RAH rule reduced the performance of NP even more. The tendency for strong weight growth is inherent to NP. It can be explained by the fixed stochasticity, which prevents NP from yielding 100% correct classification. To further improve the reliability of the classification the weights are increased during ongoing learning, which itself increases the signal-to-noise ratio and therefore the performance. The Hebbian-type algorithms (RAH and ARP), which allow for zero or low noise levels, on the other hand, can cope with

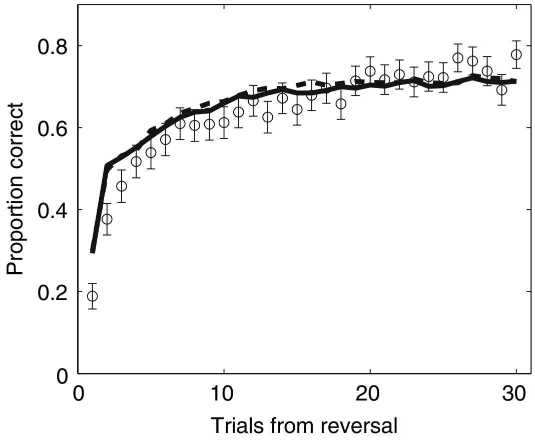

Fig. 7 Reproducing the experimental data with ARP. Successful reproduction of the performance can only be achieved with a steep activation function. *Solid line* step activation function, *dashed line* sigmoidal activation function with $\sigma = 2 \times 10^{-4}$ and $q^- = 12\,q^+$

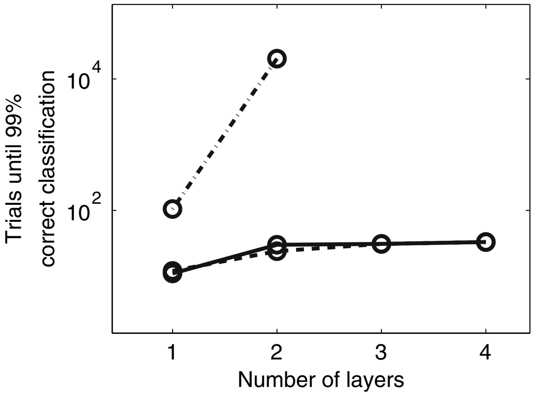

Fig. 8 Learning a simple problem with a multilayer network. For NP, which uses an external noise to explore the solution space, performance drops as the number of layer is increased. For the Hebbian-type learning rates, performance quickly saturates. Under all conditions, RAH and ARP perform much faster than NP for this task. *Solid line* RAH, *dashed line* ARP with little stochasticity, *dash–dotted line* NP. Parameters optimized for each layer using GAs. Weights are set to arbitrary initial conditions

bounded synapses, and as a result the network easily forgets the previous associations while quickly learning the new ones (see Figs. 3a, 7 and section below). For a discussion of learning and forgetting with bounded synapses see Senn and Fusi (2005b), Fusi and Senn (2006).

3.1.3 Associative reward penalty

ARP is fast enough to reproduce all the learning curves of Fig. 3 with reasonable accuracy, and to maintain its performance with many layers (see also Fig. 8). However, it requires tuning of an additional parameter, the slope of the neuronal transfer function (expressed by its inverse, σ, in Eq. 11). Further, parameter tuning brings the algorithm close to the naturally emerging characteristics of RAH. The tuning results in 12 times more synaptic changes (i.e., learning rates) in case of punishment than reward and a sharp activation function. With these parameters, the algorithm behaves approximately as if we had replaced the sigmoidal activation function with a step function (Fig. 7). Under these conditions, we were able to both soft-bound the synapses and reproduce the monkey data under the assumption of sparse coding and normalized inputs, i.e., low synaptic currents.

Formally, in the limit of a step function ($\sigma = 0$), positive postsynaptic currents h_i^μ lead to a probability of firing $p_i^\mu = 1$ and negative postsynaptic current to $p_i^\mu = 0$. however, since these two cases are characterized by $y_i^\mu = 1$ and $y_i^\mu = 0$, respectively, we get that $y_i = p_i$. For this setting the original rule (Eq. 10) becomes:

$$\Delta w_{ij} = \begin{cases} 0 & \text{if successful trial} \\ -2q^-(y_i - 0.5)y_j & \text{if unsuccessful trial.} \end{cases} \quad (13)$$

In this deterministic limit, the algorithm learns from punishment only. It also becomes very similar to the RAH algorithm, when we approximate the LTP/LTD threshold by $P_{ij}^{11} = 0.5$, and when we assume a tight stop-learning criterion ($\delta_0 = 0$), i.e., no synaptic changes in case of reward. Such a strategy is reasonable if the chance for a reward is already high so that learning can be based on the rare and informative nonrewarded events. Learning from mistakes is not a new idea; an algorithm that makes synaptic changes only in the case of punishment was previously suggested by Narendra and Thathacher (1989) and Bak and Chialvo (2001), Chialvo and Bak (1999).

3.2 Performance of the learning algorithms in networks of many layers

3.2.1 Linearly separable problem

We further investigate the performance of the three learning rules on the simple, linearly separable task of learning two random associations as the number of layers is increased. This task is in fact a simplified version of the monkey problem without association reversals and allows to compare the three rules without being concerned about the possible influence of the reversals on the results.

For this comparison, we use the genetic algorithms (GAs) toolbox of Matlab to find one optimal set of parameters per layer for each algorithm. The fitting function is the summation of the error in the first 200 trials, repeated over 1,000 instances of the network. For consistency with our previous architectural choices, we again created the patterns with coding level $f = 0.01$. The weights are set to arbitrary initial conditions. In Fig. 8 we observe that the performance of NP significantly deteriorates with the number of layers, but the performance of RAH and ARP quickly saturates. This result

Table 1 Parameters found by the genetic algorithm for the simple association task

Layer	q^+	q^-	
RAH			
1	3.98×10^{-4}	0.995	
2	9.02×10^{-5}	0.2255	
3	1.625×10^{-4}	0.1625	
4	9.9×10^{-5}	0.0990	
Layer	q	σ	
NP			
1	0.2281	4.5100	
2	0.1375	0.6250	
Layer	q^+	q^-	σ
ARP			
1	0.9632	0.9276	4×10^{-5}
2	0.9262	0.0690	2×10^{-5}
3	0.3789	0.0379	1×10^{-5}
4	0.3749	0.0258	2×10^{-5}

Coding level of the input patterns $f = 0.01$

is consistent with the previously observed inability of NP to reproduce the monkey results with a three-layer network.

If we use a baseline subtraction, i.e., modify the learning equations such that we include a term $R - \langle R \rangle$ (where $R = 1$ in rewarded cases and $R = 0$ in nonrewarded cases), learning speeds up. Though the learning times increase less dramatically for NP, we can still observe the deterioration of its performance as the number of layers is increased, also for a coding level $f = 0.1$ (with one layer it takes 41 trials to learn the task and with two layers 791 trials).

These results indicate that, for simple associative problems and learning rules that heavily depend on noise to explore the solution space, increasing the number of layers may eventually prevent learning. The noise should be continuously decreased, as layers are added, to preserve the information from input to output layers, at least for a network with random initial synaptic efficacies. This would progressively lead to deterioration and eventually prohibit the network learning (Table 1).

3.2.2 Complex problem

We have seen that RAH copes well with the linearly separable problem as the number of layers is increased. To see if our conclusions hold for a more complex problem, we further investigate its performance on a nonlinearly separable task, starting again from random initial synaptic efficacies.

It is well known that one can store up to $2N$ uncorrelated patterns in a simple perceptron with N input neurons (Bethge et al. 1994). To construct a nonlinearly separable problem, we store 30 random patterns of dimension $N = 10$ using a multilayer network.

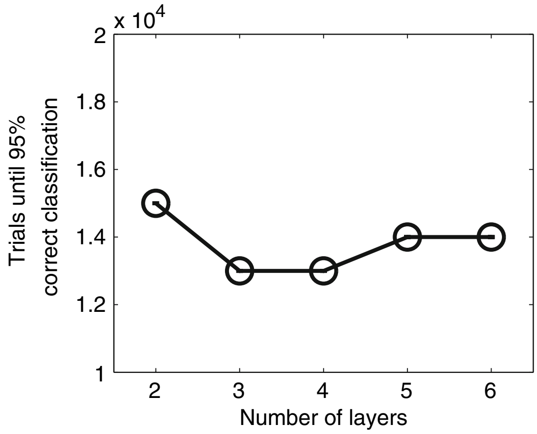

Fig. 9 Learning a complex problem with a multilayer network and the RAH algorithm. Performance is maintained for at least six layers. Number of inputs was 10, and number of neurons per hidden layer was 50. Learning rates are uniformly distributed between 0 and 0.01; 95% confidence intervals for the mean are negligible

To address this more complex problem with a network of many layers, we assign to each synapse a learning rate drawn from a uniform distribution with minimum 0 and maximum 0.01. Due to this setup, we are not obliged to fine-tune a learning parameter depending on the number of layers but we can keep the same parameters throughout our simulations. Such an approach would allow a dynamic growth of the network in general cases, without being concerned about how the synapses would adapt their learning rate, and is inspired from the theoretical work on learning in multiple timescales (Fusi et al. 2005). In this setting, we observe that the global performance does not drop as the number of layers increases up to six layers (Fig. 9). There is a systematic increase in the performance from two layers to three layers, which we attribute to the fact that in multilayer perceptrons with Heaviside activation functions, three layers (two hidden layers) are required for full generality (Sontag 1992). For a network structure with more than six layers, retuning the maximum learning rate is required. Nevertheless, we consider this result reasonably good for solving a complex problem with such a deep network structure.

3.3 Importance of the stop-learning criterion for the RAH algorithm

In what follows, we show the importance of implementing a stop-learning criterion for the present network architecture when the RAH algorithm is applied. For this purpose, we use the XOR problem as the simplest nontrivial problem, and demonstrate that optimal performance requires either learning from mistakes or a stop-learning criterion. Further, we show that our results are also valid for a complex problem, namely the modified National Institute of Standards and Technology (MNIST) database.

3.3.1 *Solving a problem with and without a stop-learning criterion*

We solve the XOR problem using a network with a three-neuron input layer (two inputs and one bias unit), one hidden layer of 17 neurons, and a one-neuron output layer. The number of neurons in the hidden layer is chosen to optimize speed. Fewer neurons result in slower learning, while adding more neurons results in equally fast learning, but performance always reaches 100%. The task can be solved with as few as two hidden neurons, but within the spirit of this work we try to solve the problem with a large network. The XOR example here is used to demonstrate the necessity of a stop-learning criterion; without it, the algorithm fails to solve more complex problems except for in the extreme condition of having synaptic changes only in the case of punishment.

The learning curves presented in Fig. 10 are produced by averaging the reward over 100 runs. In all simulations we observe that, the smaller the synaptic changes in case of reward, the better the performance (Fig. 10a). Best results are produced if no synaptic synaptic modifications take place in rewarded trials (Fig. 10a, black curve). The learning rate is set to 0.03 and the synapses are soft bounded.

However such a configuration may raise questions about the biological plausibility of the network, as synaptic changes are also recorded in case of reward (Asaad et al. 1998). Further, learning by mistakes leads to learning the patterns only marginally, with no tolerance to noise. Weight changes stop when the current is above or below the firing threshold, and thus the distance from the threshold can be arbitrary small. Thus, a small amount of noise could lead to the wrong response. The binary perceptron theory (Senn and Fusi 2005b) allows for a margin in learning, which we also adopt here. Weights will be modified upon correct classification while the synaptic current is lower than the firing threshold plus a predefined margin δ_0, if firing is the desirable behavior of the neuron. Weights will be also modified while the synaptic current is higher than the firing threshold minus the margin δ_0, if silence is the desired behavior (see Eq. 8). This margin, induced by a stop-learning criterion, provides robustness to noise; for example, if the current is reduced by some amount due to sources of noise, as long as the quantity is smaller than the margin, the neuronal response will still be correct.

In Fig. 10b we show how these two issues can be addressed by implementing an explicit stop-learning criterion (with $\delta_0 = 0.01$, i.e., mean value of the absolute synaptic current). In this case, performance becomes again optimal, regardless of our choice of learning rate in rewarded cases. Within the framework of the stop-learning criterion, learning from mistakes corresponds to an implicit, extreme stop-learning criterion with $\delta_0 = 0$.

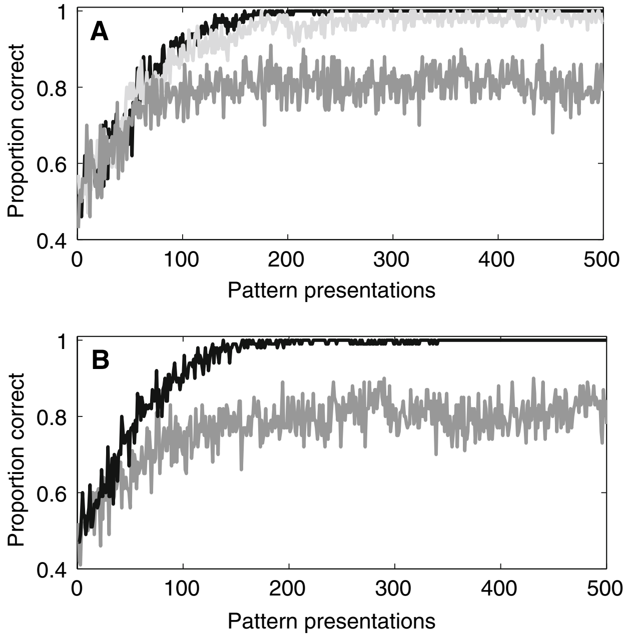

Fig. 10 Solving the XOR Problem without and with a stop-learning criterion. **a** Without a stop-learning criterion, learning from mistakes only is optimal. However, it is not robust to noise since the network only learns the patterns marginally. *Black curve* learning from mistakes only. *Light gray curve* learning when synaptic changes in absence of reward are 20 times stronger than in case of reward ($q^- = 20q^+$). *Dark gray curve* learning when synaptic changes in absence of reward are two times stronger than in case of reward ($q^- = 2q^+$). **b** Implementing an explicit stop-learning criterion restores speed while allowing for robustness. *Black curve* same parameters as for the *dark gray curve* in **a** ($q^- = 2q^+$) but with a stop-learning criterion. Performance is nearly optimal. *Dark gray curve* same parameter values as *dark gray curve* in **a** ($q^- = 2q^+$, no stop-learning criterion). Learning curves are computed by averaging the reward over 100 repetitions. Learning rate in absence of reward $q^- = 0.03$, $\rho = 10^{-3}$

3.3.2 *Learning a complex problem from mistakes*

Since XOR is a simple problem of only two classes, we would like to show that our conclusions are still valid for a complex, real-world problem. For this purpose, we chose the MNIST database, a collection of handwritten digits often used as a benchmark for algorithmic performance (LeCun et al. 2001).

We use a network with 784 input neurons (image size 28 × 28), 25 hidden neurons, and 4 output neurons, since the data we use for the training are limited to the digits 0–3. The training dataset consists of 2,023 training samples and the test dataset of 434 samples. In order to speed up learning, we implement at the output layer a WTA mechanism as a maximum (MAX) operator. Thus, the output which receives the highest synaptic input is set to 1 and all the others to 0. Optimal performance is, again, achieved when either no

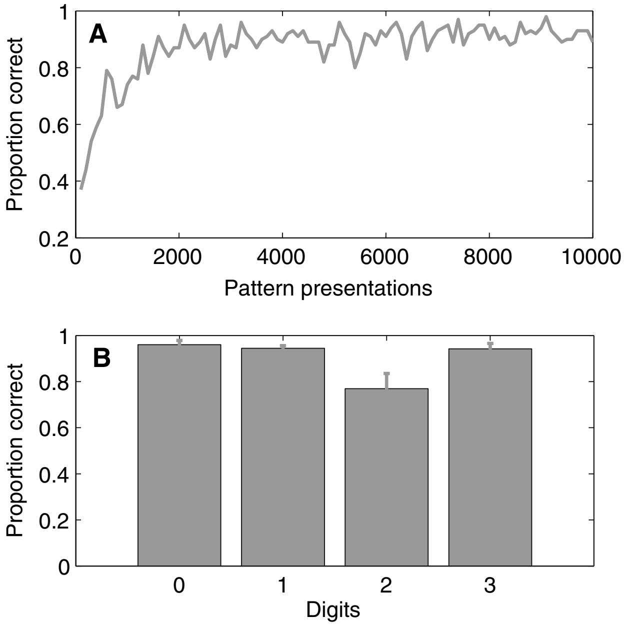

Fig. 11 Learning by mistakes for a reduced MNIST dataset of 2,023 training samples and 434 test samples. **a** Learning curve is computed by averaging the reward over 100 repetitions. Network architecture: one hidden layer of 50 neurons. Learning rate $q^- = 0.1, q^+ = 0$, $\rho = 10^{-3}$. Unbounded synapses. **b** Performance for previously unseen (test) data. Worst performance is for digit 2, which can be confused with 0 and 3

synaptic changes are made in case of rewarded trials, or a stop-learning criterion is implemented.

Learning is achieved in about 3,000 presentations for unbounded synapses (Fig. 11a) and 30,000 for bounded synapses (not shown here), since we have to choose a smaller learning rate in the presence of synaptic bounds. Given that the nature of the problem does not involve learning and forgetting, network performance is faster with unbounded synapses and a large learning rate.

The performance on previously unseen (test) data is reported in Fig. 11b. No special preprocessing techniques or fine-tuning was used. When P_{ij}^{11} in the RAH learning rule is replaced by a fixed value, e.g., 0.5, the learning performance saturates at low levels.

4 Discussion

The experiment of Asaad et al. (1998) provides us with interesting information on how the monkey behaves in a rapidly changing environment. The monkey is not able to follow an optimal game strategy, but instead takes 30 trials to learn the task. Every time the associations are reversed, it completely forgets the old associations and learns the new ones from the beginning. This behavior can be attributed to the difficulty of the task; the two noninverting associations confuse the monkey, as they create uncertainty about the nature of the mistakes.

Fusi et al. (2007) have proposed a simple associative model for reproducing this behavior, tested on a single-layer network. The model however implies that learning could take place through a multilayered network. Thus, we tested the model, implementing it in a multilayer network using one novel (RAH) and two well-known (NP and ARP) algorithms. We showed that, for the considered network architecture, only the Hebbian-type algorithms (RAH and ARP) allow us to reproduce the monkey behavior within reasonable accuracy. In contrast, the performance of NP significantly decreases as the number of association reversals increases.

Our simulations require that synaptic modifications in case of an error must be larger than in case of a successful trial to reproduce the experimental data. This imbalance in the synaptic changes provides us with an explanation for the strong reset observed after a mistake; strong synaptic changes in unsuccessful trials cause the synapses to rapidly forget their past memories and learn the new associations. The absence of pattern interference can be simultaneously achieved, as long as sparse coding in the input layer is used.

The imbalance between the synaptic changes in rewarded and nonrewarded trials may also cause undesired forgetfulness of past skills with each mistake. The sparseness in the representation of the patterns can also limit this problem. If sparseness is maintained through the structure, for different stimuli there are different learning pathways with little overlapping. Nevertheless, the problem of memory preservation while learning new features (stability–plasticity dilemma) is common to all algorithms but appears more prominent when synapses are bounded. There is no global solution to this issue, but various suggestions have been made such as smart stochastic selection of synaptic updates (Fusi and Senn 2006), the cascade model (Fusi et al. 2005), or modularity (Calabretta and Parisi 2005; Calabretta et al. 1998).

In general, the Hebbian-type algorithms (RAH and ARP) seem to generalize relatively well in large associative networks of input patterns with sparse representation. We investigated their behavior on a simple linearly separable problem, without association reversals, and saw that both rules retain their performance when adding layers, in contrast to the NP. In addition, we saw that this conclusion still holds for RAH (and ARP) for a more complex associative problem, since in the limit of low neuronal stochasticity, the two rules become identical. A similar type of reward-based Hebbian synaptic plasticity which depends on the pre- and postsynaptic mean firing rates was used to learn the inputs to a recurrently connected decision network (Fusi et al. 2007; Soltani et al. 2006a,b), but in these cases the effective potentiation and depression rates were constant (corresponding to a fixed value for P_{ij}^{11}).

Despite the promising simulation results, a formal proof that reward-modulated Hebbian learning follows the expected reward gradient is lacking. In contrast, NP is gradient ascending (Werfel et al. 2005), and this property can also be proven for implementations of NP in spiking neurons and arbitrary network architectures (Fiete and Seung 2006). Hence, although for the selected problems and architectures RAH surpasses NP in learning speed and performance, NP will always raise the expected reward for sufficiently small learning rates, unless the dynamics is already in a local reward maximum. While with increasing number of layers within the network the local gradient property seems to loose its relevance, it remains an open question why synaptic modifications in Hebbian or anti-Hebbian directions seem to gain importance.

5 Conclusion

We modeled the learning of visuomotor associations by monkeys (Asaad et al. 1998) using a multilayer neural network. Three reinforcement learning algorithms have been used, one novel and two from the literature. The simulation results showed that:

(1) Learning in the case of punishment must be faster than in the case of reward in order to recreate outcomes comparable to the monkey data. This is achieved by either a very low learning rate for reward as compared with punishment, or by the stop-learning criterion in case of reward.

(2) In a multilayer feedforward network, the learning speed of monkeys can be reached by the reward-modulated Hebbian-type synaptic plasticity with a stop-learning criterion (i.e., by our RAH algorithm, or by ARP with an appropriately steep activation function).

(3) The high stochasticity used by reinforcement learning rules to quickly learn in the case of a single hidden layer may prevent the learning of simple associations within a reasonable speed when adding more hidden layers. When the level of stochasticity is low, or if stochasticity is absent, adding a few other hidden layers does not affect the speed of learning.

Acknowledgments We are grateful to Peter Dayan for discussions on memory-preservation mechanisms and comments on the manuscript. We also thank Denis Sheynikhovich and Gediminas Luksys for various discussions and many constructive comments, and Eilif Mueller for corrections on the text. This work was supported by the Swiss National Science Foundation, grant no. 3152A0-105966 to W.S.

References

Amit D, Fusi S (1994) Learning in neural networks with material synapses. Neural Comput 6:957–982

Asaad WF, Rainer G, Miller EK (1998) Neural activity in the primate prefrontal cortex during associative learning. Neuron 21: 1399–1407

Bak P, Chialvo DR (2001) Adaptive learning by extreme dynamics and negative feedback. Phys Rev E 63(3):031912

Barto AG (1985) Learning by statistical cooperation of self-interested neuron-like computing elements. Hum Neurobiol 4: 229–256

Barto AG (1989) From chemotaxis to cooperativity: abstract exercises in neuronal learning strategies. In: Durbin R, Miall C, Mitchison G (eds) The computing neuron. Addison-Wesley, Reading

Barto AG, Jordan MI (1989) Gradient following without backpropagation in layered networks. In: Proceedings of the IEEE first annual conference on neural networks, vol 2, San Diego, pp 629–636

Bengio Y, LeCun Y (2007) Scaling learning algorithms towards AI. In: Bottou L, Chapelle O, DeCoste D, Weston J (eds) Large-scale kernel machines. MIT Press, Cambridge

Bethge A, Kuhn R, Horner H (1994) Storage capacity of a two-layer perceptron with fixed preprocessing in the first layer. J Phys A Math Gen 27:1929–1937

Calabretta R, Parisi D (2005) Evolutionary connectionism and mind/brain modularity. In: Modularity. Understanding the development and evolution of complex natural systems. The MIT Press, Cambridge, pp 309–330

Calabretta R, Nolfi S, Parisi D, Wagner GP (1998) A case study of the evolution of modularity: towards a bridge between evolutionary biology, artificial life, neuro- and cognitive-science. In: Proceedings of the sixth international conference on artificial life. The MIT Press, Cambridge, 275–284

Chialvo DR, Bak P (1999) Learning from mistakes. Neuroscience 90(4):1137–1148

Dayan P (1990) Reinforcement comparison. In: Touretzky DS, Elman JL, Sejnowski TJ, Hinton GE (eds) Proceedings of the 1990 connectionist models summer school. Morgan Kaufmann, San Mateo, pp 45–51

Dayan P, Willshaw DJ (1991) Optimising synaptic learning rules in linear associative memories. Biol Cybern 65:253–265

Fahlman SE, Lebiere C (1991) The cascade-correlation learning architecture. In: Touretzky DS (ed) Advances in neural information processing systems, vol 2. Morgan Kaufmann, San Mateo

Fiete IR, Seung HS (2006) Gradient learning in spiking neural networks by dynamic perturbation of conductances. Phys Rev Lett 97(4):048104

Fusi S, Senn W (2006) Eluding oblivion with smart stochastic selection of synaptic updates. Chaos, vol 16, 026112, pp 1–11

Fusi S, Drew PJ, Abbott LF (2005) Cascade models of synaptically stored memories. Neuron 45:599–611

Fusi S, Asaad WF, Miller EK, Wang X-J (2007) A neural circuit model of flexible sensori-motor mapping: learning and forgetting on multiple timescales. Neuron 54:319–333

Hinton GE, Salakhutdinov RR (2006) Reducing the dimensionality of data with neural networks. Science 313(5786):504–507

Hopfield JJ (1984) Neurons with graded response have collective computational properties like those of two-state neurons. Proc Natl Acad Sci 81:3088–3092

LeCun Y, Bottou L, Bengio Y, Haffner P (2001) Gradient-based learning applied to document recognition. In: Intelligent signal processing, pp 306–351

Mazzoni P, Andersen RA, Jordan MI (1991) A more biologically plausible learning rule for neural networks. Proc Natl Acad Sci 88:4433–443

Narendra K, Thathacher M (1989) Learning automata: an introduction. Prentice-Hall, Englewood Cliffs

Senn W, Fusi S (2005a) Convergence of stochastic learning in perceptrons with binary synapses. Phys Rev E 71(6):061907

Senn W, Fusi S (2005b) Learning only when necessary: better memories of correlated patterns in networks with bounded synapses. Neural Comput 17:2106–2138

Soltani A, Lee D, Wang X-J (2006a) A biophysically-based neural model of matching law behavior: melioration by stochastic synapses. J Neurosci 26:3731–3744

Soltani A, Lee D, Wang X-J (2006b) Neural mechanism for stochastic behavior during a competitive game. Neural Netw 19(8): 1075–1090

Sontag ED (1992) Feedback stabilization using two-hidden-layer nets. IEEE Trans Neural Netw 3:981–990

Sutton RS, Barto AG (1998) Reinforcement learning: an introduction. MIT Press, Cambridge

Wang X-J (2002) Probabilistic decision making by slow reverberation in cortical circuits. Neuron 36:955–968

Wang X-J (2005) A microcircuit model of prefrontal functions: Ying and Yang of reverberatory neurodynamics in cognition. In: Risberg J, Grafman J, Boller F (eds) The prefrontal lobes: development, function and pathology, Cambridge University Press, London

Werfel J, Xie X, Seung SH (2005) Learning curves for stochastic gradient descent in linear feedforward networks. Neural Comput 17:2699–2718

Williams RJ (1992) Simple statistical gradient-following algorithms for connectionist reinforcement learning. Mach Learn 8:229–256

Willshaw DJ, Dayan P (1990) Optimal plasticity in matrix memories: what goes up must come down. Neural Comput 2:85–93