

LETTER

Active Learning in Recurrent Neural Networks Facilitated by a Hebb-like Learning Rule with Memory

Frank Emmert-Streib

Institut für Theoretische Physik, Universität Bremen, Otto-Hahn-Allee, 28334 Bremen, Germany
E-mail: fes@stowers-institute.org

(Submitted on July 11, 2005)

Abstract - We demonstrate in this article that a Hebb-like learning rule with memory paves the way for active learning in the context of recurrent neural networks. We compare active with passive learning and a Hebb-like learning rule with and without memory for the problem of timing to be learned by the neural network. Moreover, we study the influence of the topology of the recurrent neural network. Our results from numerical simulations reveal that active learning decreases the learning time significantly only for the Hebb-like learning rule with memory whereas the learning rule without memory remains unaffected. This result can be observed in all investigated network topologies, indicating the robustness of this effect.

Keywords - Active learning, Hebb-like learning rule, Recurrent neural network

1. Introduction

The brain of animals and humans is one of the most fascinating complex adaptive system in nature. Despite its relatively simple basic units, the neurons, its cooperative behavior of the interconnected neurons and their functional implications are only poorly understood. The problem one faces investigating this system is not only its complexity, because, e.g., the human brain consists of about 10^{12} neurons [4], but also its characteristic functional structure embedded in an environment, which is known as action-perception-cycle. The difficulty with the action-perception-cycle, which was already known to von Uexküll in 1928 [15], is that a formulation of the problem must include a coupled description of the brain and the environment, because the actions of an animal are transformed by the environment into perceptions, which are transformed by the brain into actions and so on. In the terminology of control theory this is called closed-loop control [16] and the action of an animal can be seen as feedback represented as perception. From this it is clear that neither the perceptions nor the actions in the system are randomly generated. Interestingly, traditional approaches aiming to train artificial neural networks to learn mappings from input to output patterns select the input pattern to be learned randomly from the set of available patterns [14, 12]. Recently, this point has been addressed by theoretical investigations [3] and is called *active learning*. The basic idea of active learning is to *choose* the next pattern to be learned according to, e.g., an objective function. It has been shown that active learning can reduce the learning time needed to train artificial feedforward neural networks [10].

In this article we investigate the effect of active learning in the context of biologically motivated learning rules and recurrent neural networks. We study the problem of timing in a recurrent network architecture. Timing means that the network has to learn to map an input pattern to an output pattern in *exactly* T_c time steps. Moreover, we study the influence of the network topology on the learning behavior. We use the algorithm of Watts and Strogatz [17], because this algorithm allows to generate network topologies in dependence on a parameter $p_{rw} \in [0, 1]$. The resulting topology is regular ($p_{rw} = 0$), random ($p_{rw} = 1$) or something in-between ($p_{rw} \in (0, 1)$) including so called small-world networks. Recently, a similar problem has been studied by Bak and Chialvo [2]. However, they allowed the mappings to be learned within $\leq T_c$ time steps. This is less restrictive and, hence, easier to learn, because the number of solutions is increased by the mappings found in the network that need less than T_c time steps. Moreover, they used only a random network topology for the neural network. No attempt was made to study the influence of the network topology itself on the learning dynamics. For this reason we use for our studies the

learning rule of Bak and Chialvo [2, 1] and compare the results with a learning rule proposed by the author [5, 6, 7]. From a neurobiological point of view, the learning rule of Bak and Chialvo [2, 1] combines experimental findings of Frey and Morris [9] about *synaptic tagging* with a global reinforcement signal that can be interpreted as a dopamine signal, e.g., as in the experiments of Otmakhova and Lisman [13]. The learning rule introduced by the author [5, 6] extends the learning rule of Bak and Chialvo by the experimental results of Fitzsimonds et al. [8] about heterosynaptic plasticity, which can be qualitatively explained additionally by our learning rule. Both learning rules are local in the sense that the information, which is used for the synaptic modification, is only provided by the neurons that enclose the synapse and, hence, can be interpreted as extensions to the classical Hebbian learning rule [11] due to the fact that both learning rules use additionally a reinforcement signal as feedback signal of the performance of the network. We call such learning rules Hebb-like to indicate that they extend the classical Hebbian learning rule [11], but are still biologically plausible.

This paper is organized as follows: In Section 2 we define our model. In Section 3 we present our results from numerical simulations. We compare the learning behavior of our learning rule [5, 6, 7] with the learning rule of Bak and Chialvo [2, 1] in dependence on two different pattern-selection-mechanisms and the network topology. The paper ends in Section 4 with conclusions and a discussion of the results.

2. The model

The model we investigate in the following consists of N binary neurons $x_i \in \{0, 1\}$, $i \in \{1, \dots, N\}$. As topology of the neural network we use an architecture that can be generated by an algorithm of Watts and Strogatz [17] with $N = 200$ neurons and $k = 10$ synapses for each neuron. The algorithm proceeds in the following way. First, arrange all neurons on a ring and connect each neuron with its $k/2$ nearest neighbors, as depicted in the left figure 1. Second, start with an arbitrary neuron i and rewire its connection to its nearest neighbor on, e.g., the left side with probability p_{rw} to any other neuron j in the network. If neuron i and j are already connected reject this selection and change nothing. Then choose the next neuron in the ring in a, e.g., clockwise direction and repeat this procedure. Third, after all next neighbor connections have been checked repeat this procedure for the second and all higher next neighbors successively. This algorithm guarantees that each connection occurring in the network is chosen exactly once to test for a rewiring with probability p_{rw} . The rewiring probability p_{rw} controls the disorder of the resulting topology. For $p_{rw} = 0$ the regular topology is conserved, whereas $p_{rw} = 1.0$ leads to a random network. Intermediate values $0 < p_{rw} < 1$ give a topological structure that is between regular and random. See Figure 1 for a visualization. The network dynamics in the neural network is given by a winner-take-all mechanism.

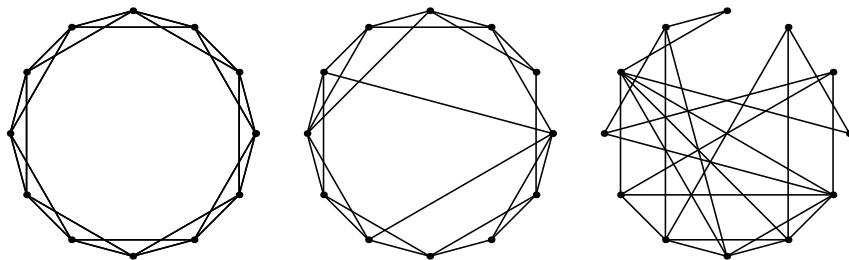


Figure 1. Visualization of network topologies obtained by the algorithm of Watts and Strogatz [17] for different values of the rewiring parameter p_{rw} . Left: $p_{rw} = 0.0$ (regular network). Middle: $p_{rw} = 0.2$ (small-world network). Right: $p_{rw} = 1.0$ (random network).

The inner field of the neurons is calculated by

$$h_j = \sum_i^{\text{all}} w_{ji} x_i \quad (1)$$

Here “all” indicates that the summation is carried out over all connected neurons. From the obtained inner fields h_j we select the one with the highest value

$$i_{\max} = \operatorname{argmax}_i(h_i) \quad (2)$$

and set the corresponding neuron activity to one. The activity of the remaining neurons is set to zero.

$$x_i = \begin{cases} 1, & i = i_{\max} \\ 0, & i \neq i_{\max} \end{cases} \quad (3)$$

For this the network dynamics is called winner-take-all mechanism, because only the neuron with the highest inner field becomes activated.

The problem we want to learn with the neural network is timing. That means, we want to learn a mapping from input neurons to output neurons in *exactly* $T_c = 4$ time steps. As input (output) neurons we define $x_{5(m-1)+1}$ ($x_{100+(m-1)5}$) in the neural network for a pattern $m \in M$. The mapping consists in a connection from input neuron $x_{5(m-1)+1}$ via inter neurons to output neuron $x_{100+(m-1)5}$ in *exactly* $T_c = 4$ time steps. Arriving sooner or later at the predefined output neuron is assumed as wrong network output. For this we call the problem to be learned timing. It is clear that the problem of timing can only be studied in a recurrent network topology, because a multilayer feedforward network reaches always after #layer time steps the output layer and the only problem left is to map to the desired output neurons. In a recurrent network topology one has the additional difficulty to reach the predefined output neuron after exactly T_c time steps. A similar problem has been studied by Bak and Chialvo [2]. However, they used a random topology of the neural network and learned a mapping within $\leq T_c$ time steps, which is less restrictive and, hence, easier to learn. In Figure 2 we depict schematically the overall organization of the learning procedure. The black neurons in the recurrent neural network (RNN) indicate the input (left side) and output (right side) neurons that are arranged on opposite sides of the network. In this paper we want to investigate

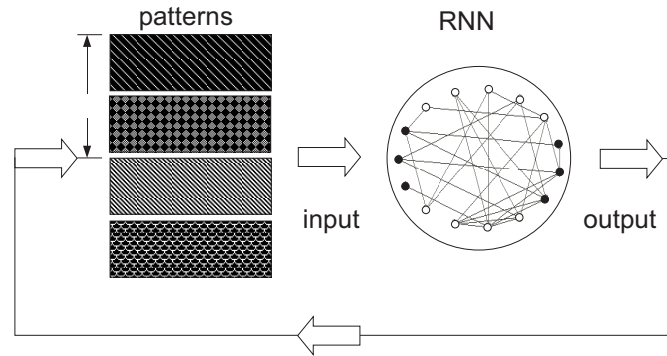


Figure 2. Visualization of the overall organization of the learning procedure. The black neurons in the recurrent neural network (RNN) indicate the input (left side) and output (right side) neurons of the neural network.

the influence of active learning on the convergence behavior of a recurrent neural network. That means, we want to study, if the way patterns are presented effects learning. For this reason it is useful to study both, active and passive learning and compare the results. During passive learning (PL) the pattern to be learned by the neural network is uniformly drawn from the complete set of patterns. This is the conventional way used to select patterns during training an artificial neural network. For the pattern selection-mechanism of active learning (AL) we give one example, because there are infinite many possibilities to define an active learning strategy for a given unstructured set of patterns. The reason therefore is that there are infinite many ways to structure the pattern set. We suggest a simple mechanism, which works in the following way: First, we enumerate all patterns. Then we select the first pattern m_1 and use it as input pattern for the neural network. We present this pattern m_1 so long to the network until the network has learned to map it correctly to the desired output neurons. If the network has learned the mapping we select additionally pattern m_2 . Now, we present pattern m_1 and pattern m_2 alternately until both mappings are learned correctly. The reason, why we present both patterns alternately and not just m_2 , because m_1 has been already learned, is the following. Suppose, pattern m_2 is used as input pattern and the output of the network is wrong. This will induce a synaptic update. After the update it is possible that the previously learned mapping for input pattern m_1 is destroyed due to this synaptic update. Learning both patterns alternately counteracts the tendency of the network to *completely* unlearn already learned mappings. If the network has learned both patterns m_1 and m_2 we select the next pattern m_3 and proceed presenting all three patterns cyclically. This procedure is carried out until all M patterns are learned. Schematically, this stepwise pattern-selection mechanism is depicted in Figure 3.

We want to compare two different learning rules. The first learning rule was introduced by Bak and Chialvo [2, 1] and the second by the author [5, 6, 7]. In the following we use the abbreviation LR1 and LR2 for the learning rule introduced by Bak and Chialvo and by the author, respectively. Both learning rules are biologically inspired and have in common that each neuron is connected with a reinforcement signal r indicating, if the input pattern was correctly mapped to an output pattern $r = 1$ or not $r = -1$. Moreover, both learning rules use only local information to adjust the synaptic weights and are, hence, Hebb-like learning rules. The learning rule of Bak and Chialvo [2, 1] reduces the weight of active synapses

$$w_{ij} \rightarrow w'_{ij} = w_{ij} - \delta, \quad (4)$$

with δ randomly drawn from $[0, 1]$ only, if the output of the network was wrong indicated by the reinforcement signal $r = -1$. Otherwise no learning takes place. The synapses (neurons) are called active, if they were involved in the signal processing.

The learning rule of the author was introduced in [5, 6, 7]. We present here a simplified version, which is a special case of the original learning rule, with one degree of freedom per neuron less. Similar to [2, 1] only active synapses w_{ij} can be updated, if the network output was wrong, $r = -1$. However, now a synapse is updated with a certain probability

$$P_{update} = p_{d_{ij}}^r. \quad (5)$$

If a synapse is determined to be updated, then the synaptic weight is depressed by

$$w_{ij} \rightarrow w'_{ij} = w_{ij} - \delta, \quad (6)$$

with δ randomly drawn from $[0, 1]$. The stochastic update condition Eq.(5) is based on neuron counters c_i assigned to each neuron in the network, whose dynamics is given by

$$c_i \rightarrow c'_i = \begin{cases} \Theta, & c_i - r > \Theta \\ c_i - r, & \Theta \geq c_i - r \geq 0 \\ 0, & 0 > c_i - r. \end{cases} \quad (7)$$

Here $\Theta \in \mathbb{N}$ is the memory length of the neuron counters and r a reinforcement signal. Eq.(7) concerns only the active neurons in the network. The other neuron counters remain unchanged. The probability $p_{d_{ij}}^r$ of the stochastic update condition Eq.(5) is obtained by the following procedure: First, calculate the approximated synapse counters d_{ij} of the active synapses with the neuron counters Eq.(7) by,

$$d_{ij} = c_i + c_j \quad (8)$$

Second, from $c_i \in \mathbb{N}$ for all $i \in \{1, \dots, N\}$ follows $d_{ij} \in \mathbb{N}$. We use this property to map the values of the approximated synapse counter to a probability $p_{d_{ij}}^r$ by defining a discrete probability distribution P_k^r over all possible values of d_{ij} .

$$P_k^r \propto k^{-\tau}, \tau \in \mathbb{R}^+, k \in \{1, \dots, 2\Theta + 3\} \quad (9)$$

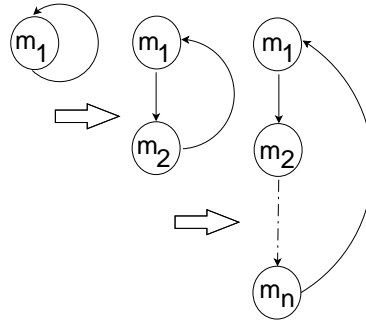


Figure 3. Pattern-selection mechanism used to train the neural network. After the network has learned all patterns of a subset one additional pattern is added to this set and the network is trained with this new subset of patterns cyclically.

with the mapping $k = 2\Theta + 3 - d_{ij}$. We call P_k^r the rank ordering distribution. The basic idea of our learning rule is to update an active synapse with a high probability, if the value of d_{ij} is high, because the approximated synapse counter reflects the averaged performance of the synapse in the network. This allows each synapse to make a probabilistic decision, if a synaptic update shall take place or not. For the following simulations we use a neuron memory of length $\Theta = 3$ and $\tau = 2.0$ as exponent of the rank ordering distribution.

3. Results

In this section we present the results for the model defined in Section 2. We evaluate the performance of the network with the first-passage time, when the network error reaches the first time zero. We use the first-passage time for the evaluation, because the learning process of the neural network is a stochastic process due to the fact that, e.g., the synaptic weights are initialized randomly and the synaptic modification δ is randomly drawn from $[0, 1]$. Hence, the prediction error

$$E_{ipe}(t) = \frac{\# \text{ patterns learned up to time step } t}{M} \quad (10)$$

of the network is a random variable. This is the individual prediction error (ipe) of one network at time step t . 'Individual' indicates that this measure is up to now not averaged over an ensemble of networks. The first-passage time for E_{ipe} is the time, when the individual prediction error of the network reaches the first time zero, $E_{ipe}(t_{FPT}) = 0$. The first-passage times of an ensemble of networks determine the first-passage probability distribution p^{FPT} of the learning process. From the distribution p^{FPT} one obtains the distribution function

$$P^{E=0}(t) = \sum_{t'=0}^t p_{t'}^{FPT} \quad (11)$$

and the mean first-passage time

$$T_{mean} = \sum_{t'=0}^{\infty} t' p_{t'}^{FPT} \quad (12)$$

The distribution function $P^{E=0}(t)$ is restricted between 0 and 1 and indicates the percentage of the networks that learned the mapping of all patterns up to time step t . This allows a convenient comparison between different distribution functions for different conditions. The mean first-passage time is suitable to compress the information of the first-passage distribution in one scalar value to visualize the results from many different simulations.

Figure 4 shows exemplary the distribution p^{FPT} of the first-passage times. The shape of p^{FPT} is characteristic and reflects by a long tail that some networks need much more time to learn the mapping than others.

3.1 Passive learning

We start our investigations by comparing the learning rules for different network topologies and passive learning. The rewiring parameter p_{rw} was varied from 0.1 to 1.0 in steps of 0.1.

M=3 patterns: Figure 5 shows the distribution function $P^{E=0}$ for learning rule LR1 (dotted lines) and LR2 (full lines) in dependence on the rewiring parameter p_{rw} . It is clear to recognize that the convergence behavior for learning rule LR2 is always significantly better. In general, the less p_{rw} becomes the longer it takes to converge. This is due to the fact that for $p_{rw} = 0$ the network is regularly connected without any shortcuts between further remote neurons, hence, there is no path that connects the input with the output neurons within $T_c = 4$ time steps at all. If one increases p_{rw} there exist more and more such shortcuts and the problem can be learned more and more easily. For $p_{rw} > 0$ the problem consists not only in finding connecting paths between input and output neurons, but also in preserving paths for already correctly learned mappings. This interplay between path exploration and path conservation makes the problem hard especially for low values of the rewiring parameter, when only a small number of paths exists within the network.

M=5 patterns: In Figure 6 the corresponding results for $M = 5$ patterns are shown. Here the effects mentioned above are increased by increasing the number of patterns to be learned. This leads to an almost complete break

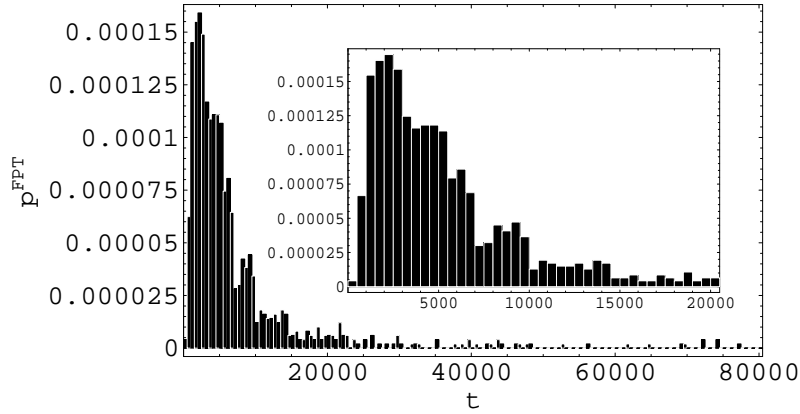


Figure 4. Distribution p^{FPT} of the first-passage times for learning rule LR2, rewiring parameter $p_{\text{rw}} = 1.0$ and $M = 3$ patterns, generated by a passive learning pattern-selection mechanism. The histogram, with bin width 500, was generated by simulations over an ensemble of size $N = 1000$. The inner figure is a magnification of the first 20000 time steps.

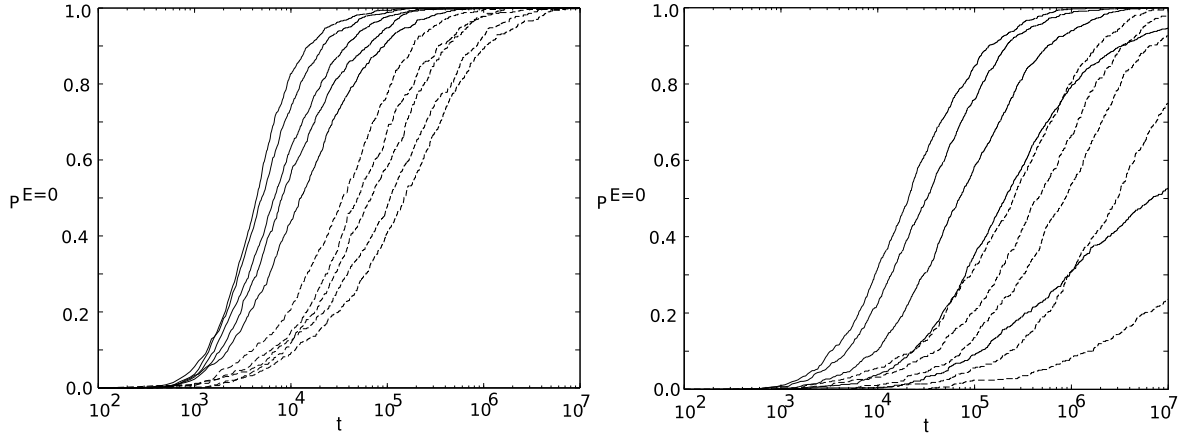


Figure 5. Distribution function $P^{E=0}$ for $M = 3$ and learning rule LR1 (dash-dot line) and LR2 (full line), which were obtained for PL. The curves are parameterized from above to below from $p_{\text{rw}} = 1.0$ to $p_{\text{rw}} = 0.6$ (left figure) and from $p_{\text{rw}} = 0.5$ to $p_{\text{rw}} = 0.1$ (right figure).

down for learning rule LR1, which is now only able to learn the problem for $p_{\text{rw}} = \{0.9, 1.0\}$ for a few networks, see Figure 7(a) for more details concerning quantitative values of the number of converged networks. Learning rule LR2 works again significantly better.

Figure 7(a) shows the percentage of converged networks in dependence on p_{rw} after a simulation time of $T = 10^7$ time steps. Converged means here that the prediction error of a network reached zero. The dash-dot lines correspond to LR1 and the full lines to LR2. The upper pair of curves corresponds to $M = 3$ and the lower pair of curves to $M = 5$. One can clearly see that the number of converged networks trained with LR1 is much lower, especially for $M = 5$ in which case almost no network learned all patterns correctly.

3.2 Active learning

The percentage of converged networks for active learning for $M = 3$ and $M = 5$ patterns are summarized in Figure 7(b).

One recognizes by comparison with Figure 7(a) that the overall results are confirmed. Learning rule LR2

converges always significantly better than LR1. A thorough comparison between active and passive learning for LR2 by comparing Figures 7(a) and 7(b) reveals that the convergence is clearly improved. This effect is especially good to see for intermediate values of the rewiring parameter p_{rw} and $M = 5$. The obtained learning times for active learning will be given in the next subsection, where we compare directly the results for active and passive learning.

3.3 Comparison between active and passive learning

To quantify the dependence of the learning behavior on the pattern-selection mechanism we calculate the mean first-passage time T_{mean} from the simulation results obtained so far. Figure 8 shows these results for learn-

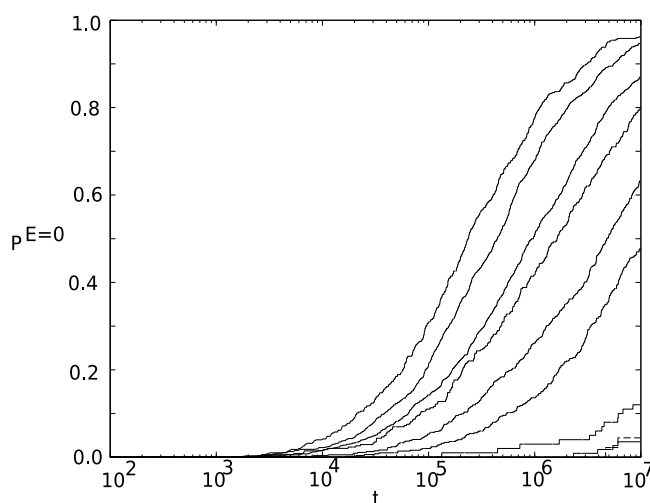


Figure 6. Distribution function $P^{E=0}$ for $M = 5$ and learning rule LR1 (dash-dot line) and LR2 (full line), which were obtained for PL. The curves for learning rule LR2 are parameterized from above to below from $p_{rw} = 1.0$ to $p_{rw} = 0.1$ and for learning rule LR1 from $p_{rw} = 1.0$ to $p_{rw} = 0.9$.

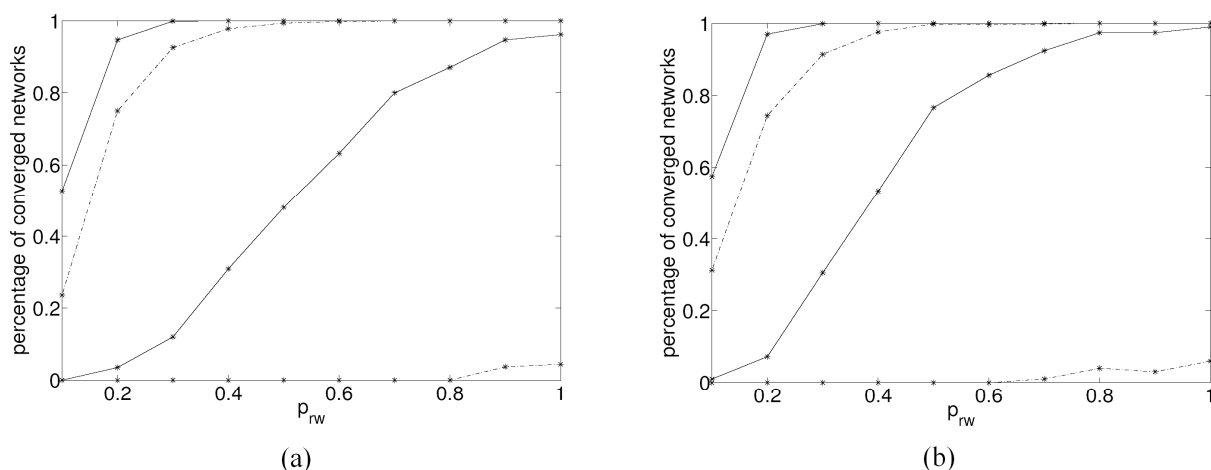


Figure 7. Percentage of converged networks for active learning in dependence on p_{rw} after a simulation time of $T = 10^7$ time steps. The dash-dot lines correspond to the learning rule of Bak and Chialvo (LR1) and the full lines to the learning rule suggested by the author (LR2). The upper pair of curves corresponds to $M = 3$, the lower pair of curves to $M = 5$. The size of the ensemble was 1000 for LR2 and $M = 3$ for all other simulations 500. (a) Passive learning. (b) Active learning.

ing rule LR1 and LR2 in dependence on the rewiring parameter p_{rw} and the patterns to be learned. Full lines correspond to simulations with active learning (AL) dash-dot lines with passive learning (PL). The two upper and the two lower curves correspond to learning rule LR2 and the two middle curves to learning rule LR1. The symbol “*” indicates $M = 3$ and ‘□’ $M = 5$ patterns to be learned by the network.

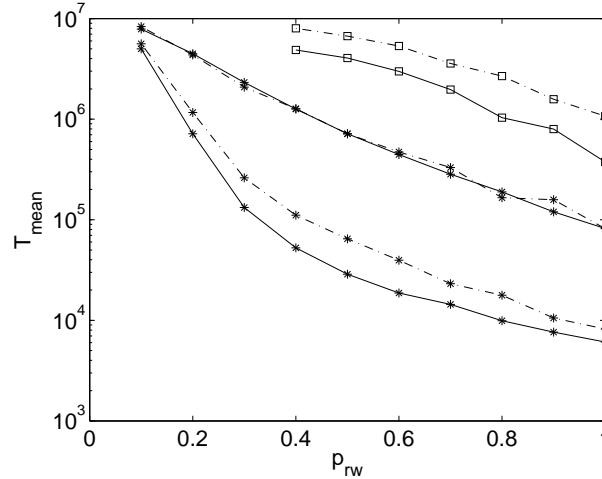


Figure 8. Mean first-passage time T_{mean} in dependence on the rewiring parameter p_{rw} . Full lines correspond to simulations with active learning dash-dot lines to simulations with passive learning. The two upper and the two lower curves are obtained by learning rule LR2 and the two middle curves by learning rule LR1. “*” indicates $M = 3$ and ‘□’ $M = 5$.

Figure 8 reveals two interesting results. First, LR2 is always significantly better than LR1, especially for $M = 5$, because LR1 can not learn this problem at all for the majority of networks. For this reason the curves are not present in Figure 8. Second, active learning (AL) has a positive effect on LR2 by reducing the mean first-passage time significantly compared to passive learning, whereas LR1 remains unaffected. Figure 9 shows the percentage of improvement (POI) for LR2. We define

$$POI(p_{rw}) = 1 - \frac{T_{mean}^{AL}(p_{rw})}{T_{mean}^{PL}(p_{rw})} \quad (13)$$

Active learning can reduce the mean first-passage learning time up to $\sim 65\%$ compared to passive learning. This is astonishing if one keeps in mind that we changed only the strategy for choosing patterns from a training set. The network topology, the learning rule and the network dynamics remained completely unchanged. To understand, why learning rule LR1 is not effected by different pattern-selection mechanisms, but LR2, we want to mention that learning the last pattern takes in average about 90% of the overall first-passage time. That means the active and passive pattern-selection strategy differ only in the presentation order of the patterns and not in the presentation statistics of the patterns, because the presentation statistics of active learning is exactly $p(m_i) = \frac{1}{M}$ for $i \in \{1, \dots, M\}$ for about 90% of the overall first-passage time. This corresponds to the statistics of passive learning. The reason, why learning rule LR1 is not sensitive to the presentation order is due to the lack of a memory with respect to the outcomes of past results. Learning rule LR2 has such a memory in form of the neuron counters, which influence the probability of a synaptic update and, hence, detect indirectly the presentation order of the patterns.

It seems to be plausible that an appropriately chosen pattern-selection strategy should have a positive influence on the convergence behavior of a neural network, because we do not choose our actions randomly as already discussed in the introduction. Moreover, it is not only plausible but also efficient to use the pattern-selection mechanism as additional source of information, which is clearly demonstrated by Figure 8 and 9.

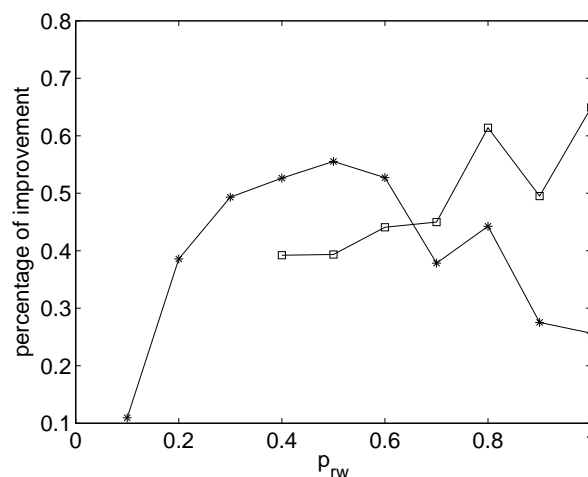


Figure 9. Percentage of improvement of the mean first-passage time due to active learning compared to passive learning. The two curves correspond to LR2, '*' indicates $M = 3$ and '□' $M = 5$.

4. Conclusions

In this article we compared two biologically inspired Hebb-like learning rules for neural networks and their dependence on the pattern-selection strategy. We learned the problem of timing in different recurrent network topologies of varying disorder generated by an algorithm of Watts and Strogatz [17]. Our results demonstrate that the stochastic Hebb-like learning rule introduced by the author [5, 6, 7] is not only always significantly better than the learning rule of Bak and Chialvo [2, 1], but facilitates active learning. The pattern-selection mechanism is a source of information and can significantly reduce the learning time needed to train a neural network as demonstrated by our results. This confirms previously obtained results, which showed that active learning can have a positive influence on the training time of feedforward neural networks if appropriately chosen [10].

Due to the fact that animals do not select their actions randomly during learning animals perform a kind of active learning. If the selection mechanism had no positive influence on their learning behavior it is improbable that animals had developed a non-random selection strategy during evolution. Hence, it seems to be plausible that also in simulations an appropriately chosen active learning strategy should have a positive effect on the learning time of neural networks. More precisely, a learning rule for neural networks can only be biologically plausible, if it favors active over passive learning strategies. We tested several active learning strategies for the problem of timing, which are different from the one depicted in Figure 2. All these active learning strategies gave qualitatively the same results as the one used for our simulations presented in the results section. This indicates a robustness of our results. From this we conclude that the learning rule of Bak and Chialvo lacks a sensitivity against different pattern-selection strategies, whereas our stochastic learning rule can be positively effected. The reason therefore can be seen in the neuron counters, which provide our learning rule with a memory. The learning rule of Bak and Chialvo is memoryless. We hope that our results can help to stimulate further work in this exciting field to gain more insights in the complex working mechanisms and properties of the brain.

Acknowledgements: We would like to thank Tom Bielefeld, Rolf D. Henkel, Jens Otterpohl, Klaus Pawelzik, Roland Rothenstein, Peter Ryder, Heinz Georg Schuster and Helmut Schwegler for fruitful discussions.

References

- [1] P. Bak and D.R. Chialvo, "Adaptive learning by extremal dynamics and negative feedback", *Phys. Rev. E* **63**, 031912 (2001).
- [2] D.R. Chialvo and P. Bak, "Learning from Mistakes", *Neuroscience* **90**, 1137-1148 (1999).

- [3] D. A. Cohn, Z. Ghahramani and M. I. Jordan, "Active Learning with Statistical Models", *Journal of Artificial Intelligence Research*, **4**, 129-145 (1996).
- [4] P.S. Churchland and T.J. Sejnowski, "The Computational Brain", MIT Press (1992).
- [5] F. Emmert-Streib, "Aktive Computation in offenen Systemen. Lerndynamiken in biologischen Systemen: Vom Netzwerk zum Organismus". Ph.D. Thesis, Universität Bremen, Mensch & Buch Verlag (2003).
- [6] F. Emmert-Streib, submitted.
- [7] F. Emmert-Streib, "Self-organized annealing in laterally inhibited neural networks shows power law decay", *Neural Information Processing - Letters and Reviews* **7**(3), 29-38 (2005).
- [8] R.M. Fitzsimonds, H-j.Song and M-m. Poo, "Propagation of activity-dependent synaptic depression in small neural networks" *Nature* **388**, 439-448 (1997).
- [9] U. Frey and R.G.M. Morris, "Synaptic tagging and long-term potentiation", *Nature* **385**, 533-536 (1997).
- [10] M. Hasenjäger and H. Ritter, "Active Learning in Neural Networks", *Physica-Verlag Studies in Fuzziness and Soft Computing Series, New learning paradigms in soft computing*, Ed. L. C. Jain and J. Kacprzyk, 137-169 (2002).
- [11] D.O. Hebb, "The Organization of Behavior", Wiley, New York (1949).
- [12] J. Hertz, A. Krogh and R. G. Palmer, "Introduction to the Theory of Neural Computation", Perseus Books Group (1991).
- [13] N.A. Otmakhova and J.E. Lisman, "D1/D5 dopamine receptors inhibit depotentiation at CA1 synapses via cAMP-dependent mechanism", *J. Neuroscience* **18**, 1270-1279 (1998).
- [14] D.E. Rumelhart, G.E. Hinton and R.J. Williams, "Learning representations by back-propagating errors", *Nature* **323**, 533-536 (1986)
- [15] J. von Uexküll, "Theoretische Biologie", Springer Verlag (1928).
- [16] N. Wiener, "Cybernetics or Control and Communication in the Animal and the Machine", The MIT Press, Cambridge (Mass.), Wiley and Sons, New York (1948).
- [17] D.J. Watts and S.H. Strogatz, "Collective dynamics of 'small-world' networks", *Nature* **393**, 440-442 (1998).

Frank Emmert-Streib obtained his Diploma in Theoretical Physics in 1998 from the University of Siegen (Germany) and his Ph.D. in Theoretical Physics from the University of Bremen (Germany) in 2003. He is currently a postdoctoral research associate in Bioinformatics at the Stowers Institute for Medical Research, Bioinformatics, 1000 E. 50th Street, Kansas City, MO 64110, USA. His research interests include Computational and Systems Biology, Bioinformatics, Computational Statistics and Machine Learning. (Home page: <http://www.bio-complexity.com>)