ESANN'2007 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 25-27 April 2007, d-side publi., ISBN 2-930307-07-2.

Intrinsic plasticity for reservoir learning algorithms

Marion Wardermann[1] and Jochen Steil[1]

1- University of Bielefeld - Faculty of Technology
Universitaetsstrasse 25, 33615 Bielefeld - Germany

Abstract. One of the most difficult problems in using dynamic reservoirs like echo state networks for signal processing is the choice of reservoir network parameters like connectivity or spectral radius of the weight matrix. In this article, we investigate the properties of an unsupervised intrinsic plasticity rule for signal specific adaptive shaping of the reservoir, which is local in space and time and aims at maximizing input–to–output information transmission for each neuron. We show that the rule consistently regulates the neurons' mean outputs and variances and is robust to learning parameter changes. Simulations reveals that this reservoir adaptation robustly enhances online learning of Backpropagation–Decorrelation recurrent learning for a tenth–order nonlinear NARMA benchmark problem.

1 Introduction

Reservoir Learning Algorithms (RLA) are a class of very fast learning algorithms for recurrent neural networks (see e.g. [1]). They use the recurrent network as a dynamical reservoir for temporal encoding of the input and adapt only a set of output weights by supervised learning to implement a readout function, which combines the activities of the nodes in the network to the desired outputs. Therefore the learning complexity of online RLAs can be reduced to linearity in the number of nodes, like in Backpropagation-Decorrelation (BPDC, [2]). Due to its ability to adapt very fast to changes in the reservoir behaviour only BPDC as sample RLA will be investigated in this paper.

A problem of RLAs is that the reservoir weights have to be carefully set. Firstly the network's states must depend only on a limited number of past time steps, i.e. they have to exhibit the "Echo State Property" (ESP, see e.g. [3]). Secondly the desired output function has to be in the functional space spanned by the vectors of activities of the neurons in the reservoir over time. The common way to increase the probability of success is simply increasing the functional space spanned by the states by enlarging the reservoir or scaling the weights.

Nevertheless, in large recurrent biological neural networks — like the brain — it seems to be possible to maintain the ESP while being able to generate generic outputs. Recently, a biologically motivated unsupervised learning rule has been developed ([4]), here called intrinsic plasticity (IP), which is able to locally steer a neuron's output distribution and thereby to some degree also more directly the network properties than by setting just global weight scaling. Its time needed during one time-step for adaptation grows linearly with the number of nodes, therefore it does not slow down BPDC. This paper derives theoretical

properties of IP learning and investigates in which way and with what parameters IP is capable of improving the reservoir such that an RLA's learning ability is increased. Two different trains of experiments will be presented: One will show the growth of the space spanned by the states of the neurons in the network and that the network still exhibits the ESP after adapting with IP. During the second train IP will be applied with an RLA on a classical benchmarking problem, the NARMA tenth order system.

2 Derivation of Intrinsic plasticity

IP adapts the neurons based on three principles: firstly information maximization, i.e. the output of the neuron should contain as much information on the input as possible. Secondly, as the energy available is limited, each neuron has to keep its average output at a certain level. The third idea is to adapt not the weights, but the intrinsic properties of the neurons, which is a common behavior in biology (see [5], [6]). Following these ideas Triesch ([4]) developed a learning rule for analog neurons which transfer their input x into the output y by using a generalized Fermi function $g_{a,b}(.)$ with two parameters controlling gain a and bias b, i.e. $y = g_{a,b}(x) = (1 + exp(-(ax + b)))^{-1}$. The exponential output function has got the maximum entropy for a given average, therefore a and b are adapted to minimize the Kullback–Leibler distance $D_{KL}(f_y; f_{exp})$ between the output distribution f_y and the desired exponential one f_{exp} with mean μ. The adaptation of a and b is performed by stochastic gradient descent updates for each time–step as

$$\begin{aligned} \Delta b &= \eta_{IP}(1 - (2 + \tfrac{1}{\mu})y + \tfrac{1}{\mu}y^2), \\ \Delta a &= \eta_{IP}(\tfrac{1}{a} + x - (2 + \tfrac{1}{\mu})xy + \tfrac{1}{\mu}xy^2) = (\tfrac{\eta}{a} + x\Delta b), \end{aligned} \tag{1}$$

where η_{IP} is a learning rate. For a detailed derivation see [4].

Note that the targeted exponential distribution is infinite, but a Fermi function is limited to $]0, 1[$, thus f_y has to differ from the desired exponential one f_{exp}. The optimal output distribution f_f minimizing the Kulback–Leibler distance to f_{exp} is the one having the same shape in [0,1] as f_{exp}. Thus we computed the actual average value μ_f and the actual standard deviation σ_f of f_f. The infinite exponential probability density function $p(y) = \lambda \exp(-\lambda y)$, $\lambda = 1/\mu$ must be normalized to result in a distribution in the limits]0:1[. Thus the finite distribution density function is

$$p_f(y) = \frac{p(y)}{\int_0^1 p(\psi)d\psi} = \frac{\lambda \exp(-\lambda y)}{1 - \exp -\lambda}. \tag{2}$$

Substituting equation 2 into $\mu_f = \int_0^1 y p_f(y)dy$ results in

$$\mu_f = -\frac{\lambda + 1 - \exp(\lambda)}{(\exp(\lambda) - 1)\lambda}. \tag{3}$$

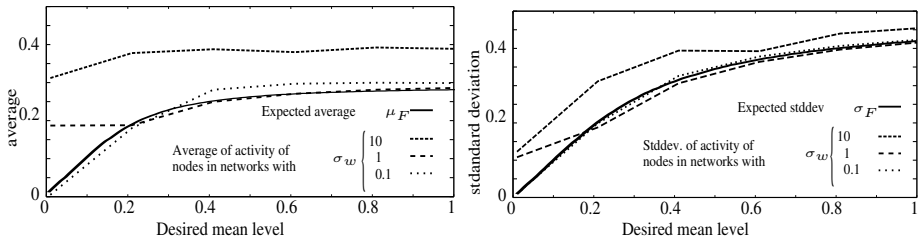


Fig. 1: Approximation quality of the output distributions. Shown are comparisons between ideally adapted output distribution (finite exponential distribution) and those measured in neurons in networks with 50 nodes.

Calculated in the same way, the standard deviation of the finite distribution has a value of

$$\sigma_f = \sqrt{\frac{-2e^{-2\lambda} + 4e^{-\lambda} + e^{-2\lambda}\lambda^2 - e^{-\lambda}\lambda^2 - e^{-2\lambda}\lambda - 2 + e^{-2\lambda}\lambda^3 + \lambda}{\lambda(e^{-\lambda} - 1)^2}} \tag{4}$$

The bold lines in figure 1 show μ_f and σ_f in relation to μ together with experimental data described in more detail below.

3 Impact of IP on an RNN

During the remainder of the article, we consider the recurrent network dynamics

$$\mathbf{x}(k+1) = \mathsf{W}_{res} g_{\mathbf{a},\mathbf{b}}(\mathbf{x}(k)) + \mathsf{W}_u \mathbf{u}(k), \tag{5}$$

where $x_i, i = 1, \ldots, N$ are the neural activations, $g_{\mathbf{a},\mathbf{b}}(.) \in \mathbb{N}^N \rightarrow \mathbb{N}^N$ is the generalized Fermi function with parameters $\mathbf{a} = a_1 \cdots a_N$ and $\mathbf{b} = b_1 \cdots b_N$ for each neuron, $\mathsf{W}_{res} \in \mathbb{N}^{N \times N}$ the weight matrix, $\mathsf{W}_u \in \mathbb{N}^{N \times N_I}$ the weight matrix connecting the N_I inputs with the network, and k is the time step. During each time step the inputs are the present and $N_I - 1$ past values of the given data set, the output is the activation of x_1. The network is fully connected and the weights are set by drawing from a gaussian random distribution with 0 as mean and, if not mentioned otherwise, 0.1 as standard deviation. Training one network on one dataset is called a "session".

The first question is to what degree IP manages to approximate an exponential output distribution in an RNN. Triesch ([4]) has already demonstrated that IP is able to find a transfer function as good as possible for a given input distribution in a non–recurrent single neuron. Note that the shape of this function is constricted, therefore the output distribution is not always exactly exponential. In an RNN, additionally the recurrency could pose a problem. Therefore we investigated the behaviour of the neurons in different RNN settings and input sizes.

The inputs were constructed by randomly drawing from gaussian distributions with means of 0, 1 and 10 and standard deviations of 0.1, 1 and 10, which is a total of 9 different time series. For studying the impact of different types of network behaviour, the weights were initialized once in a region of provable stability, once beyond the edge of provable stability, and once far into the chaotic regime. Using the Contraction Mapping Theorem ([7]), the region of provable stability — where the ESP exists — is there, where the maximal eigenvalue of the weight matrix scaled by the maximal derivatives of the transfer functions is smaller than 1. As the fermi function has a maximum gain of $\frac{1}{4}$, this means that the network will be stable for weight matrices with a maximal eigenvalue smaller or equal to 4. The weight matrices were set to exhibit a spectral radius of 1, 10 and 100, i.e. the standard deviation of the weights were 0.1, 1 and 10 (see figure 1). η_{IP} was set to 0.001. The desired mean levels investigated were 0.1, 0.2, 0.4, 0.8 and 1. Alltogether thus 135 training sessions were conducted. Each training session consisted out of 100,000 training steps, where the network was adapted with $\eta_{IP} = 0.001$, and 1000 test steps without adaptation, during which the average and standard deviation of the nodes are calculated.

Mean and standard deviation are close to the expected ones for networks not in the chaotic regime, as expected.

Chaotic behaviour of a network — which means losing the ESP — would also make learning for an RLA impossible. Non–chaotic behaviour of the network can only be guaranteed for a spectral radius of the weight matrix being smaller or equal than 4. But if applying the gains altered under the IP learning rule on the weight matrix instead of the activation function and calculating the resulting spectral radius, this can grow beyond the regions of provable stability.

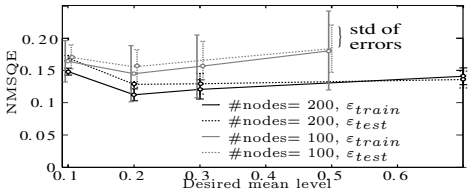
Even if, though there is no proof that IP maintains the ESP, we observe it always. For example, for each of the weight matrices mentioned above and input following the equation $y = \sin(0.2t) + \sin(0.311t)$, the network has been trained over 100000 steps by IP with $\eta_{IP} = 0.001$ and $\mu = 0.3$. Running the network twice for 10000 steps with different state initializations but the same inputs and network parameters, after 1000 discarded steps, the difference between these two runs is always smaller than 10^{-27}. The difference is here defined as the normalized mean square error (NMSQE). It follows that the ESP is still maintained after adaptation by IP.

4 Application of IP and RLA on the Tenth–Order NARMA problem

Now that the two prerequesites for the possibility of improved learning with an RLA have been shown, the improvement in RLA learning will be demonstrated here by a popular benchmarking problem. The benchmark used is a tenth order NARMA system ([8]). This is a dynamical system driven with random inputs u and computed by $y(k+1) = 0.3y(k)+0.05y(k)[\sum_{i=0}^{9} y(k-i)]+1.5u(k-9)u(k)+0.1$, where y denotes the output of the network. The task is predicting $y(k+1)$, while $u(k)$ and $u(k-9)$ are presented as input. The tenth–order problem seems

η_{IP}	η_{RLA}	ε_{train}	ε_{test}
0	0.01	0.200331	0.224519
0	0.1	0.13723	0.162927
0.0001	0.01	0.1033	0.1103
0.0001	0.1	0.1010	0.1132
0.0001	0.3	0.1103	0.1651
0.001	0.01	0.1243	0.1129
0.001	0.1	0.1114	0.1192
0.001	0.3	0.1129	0.1435
0.01	0.01	0.1484	0.1298
0.01	0.1	0.1262	0.1305

(a)



(b)

Fig. 2: Errors for different parameter settings:
2(a): Optimizing η_{RLA} and η_{IP}, parameters held fixed are μ at 0.3 and the number of nodes at 100
2(b): Optimizing μ and node number with $\eta_{RLA} = 0.01$ and $\eta_{IP} = 0.0001$

to be well–suited for learning with IP as the input's short–time statistics do not change over time and the input distribution is furthermore gaussian, which is the optimal shape for producing exponential output distributions by a generalized fermi–function.

Training parameters for BPDC are the learning constant, η_{RLA}, and a regularization constant, which is set to 0.001. Before each training session the dataset is split up into 700 points for training and 500 for testing. Then the network is trained for several epochs, which begins each with initialization of the states randomly uniformly distributed in an interval of [0,1]. The first points of the training set are used for relaxation of the network without adaptation, the rest for adapting with IP. Finally, after each epoch the generalization ability of the network is tested with the datapoints from the test set, without adaptation. After each epoch of a training session, the error is calculated for the training as well as the test data set as NMSQE between the network output and the targeted one. We summarize performance for one training session by the minimal NMSQE with respect to all the epochs of one training session. Afterwards, we average the minimal errors over three sessions with different weight matrices but otherwise the same parameters. The result is denoted by ε_{train} or ε_{test}, respectively.

The best result for learning only with BPDC was obtained with $\eta_{RLA} = 0.1$, i.e. $\varepsilon_{train} = 0.13723$ and $\varepsilon_{test} = 0.162927$. As can be seen in figure 2 , adapting with IP leads to better results, the smallest ε_{train} is 0.101 for $\eta_{IP} = 0.0001$, $\eta_{RLA} = 0.1$, $\mu = 0.3$ and $N = 100$, the smallest ε_{test} 0.1103, but for $\eta_{RLA} = 0.01$. Another asset of IP is that the error when learning with IP is smaller than learning without from almost the first epoch on. Furthermore, in contrast to simple BPDC, the test error increase due to overfitting starts much later, if at all.

Often using additional learning techniques comes with the cost of additionally parameters which have to be fine–tuned. But setting parameters for IP is not difficult, as the results for all $\mu > 0.2$ are almost the same and η_{IP} just has to be set such that $100\eta_{IP} < \eta_{RLA}$. Furthermore, choosing parameters for BPDC

is easier, since it matters only little whether $\eta_{RLA} = 0.1$ or $\eta_{RLA} = 0.01$, and the network size is also not important.

The results show that using IP firstly improves learning with BPDC and secondly is more robust, because its performance depends neither on fine tuning the network size, the IP learning rate, the BPDC learning rate nor the desired mean level.

5 Conclusion

We have shown that IP learning is capable of driving the neurons in larger networks to the theoretically derived renormalized mean and variance of the desired exponential output distributions. We also showed at hand of the tenth order benchmark that the learning performance of RLAs can be improved considerably. Further experiments not documented here have shown that this is also the case for other standard problems and we have not experienced cases where IP learning decreases performance. The most encouraging result, however, is the enormous robustness against both varying weight and learning parameters. The reason is probably that IP controls reliably the size of the neurons' activity space (see section 3). We believe that this regulatory effect of IP learning largely facilitates to generate reservoirs with controlled performance fitting a given problem and should be included as standard approach at least in online-learning settings.

References

[1] David Verstraeten, Benjamin Schrauwen, Michiel D'Haene, and Dirk Stroobandt. The unified reservoir computing concept and its digital hardware implementations. In *Proceedings of the 2006 EPFL LATSIS Symposium*, pages 139–140, Lausanne, 3 2006.

[2] Jochen J. Steil. Backpropagation-decorrelation: Recurrent learning with o(n) complexity. In *Proc. IJCNN*, pages 843–848, 2004.

[3] Herbert Jäger. Reservoir riddles: Suggestions for echo state network research. In *Proceedings of International Joint Conference on Neural Networks, Montreal*, 2005.

[4] Jochen Triesch. A gradient rule for the plasticity of a neuron's intrinsic excitability. In *Proc. of the Int. Conf. on Artificial Neural Networks (ICANN)*, 2005.

[5] Wei Zhang and David J. Linden. The other side of the engram: Experience-driven changes in neuronal intrinsic excitability. *Nature Reviews Neuroscience*, 4:885–900, 2003.

[6] Alain Destexhe and Eve Marder. Plasticity in single neuron and circuit computations. *Nature*, 431:789–795, 2004.

[7] B. Cessac. Increase in complexity in random neural networks. *Journal Physique I France*, 5:409–432, 1995.

[8] A. Atiya and A.G. Parlos. New results on recurrent network training: unifying the algorithmsand accelerating convergence. *IEEE Trans. Neural Networks*, 11(3):697–709, 2000.