This article was originally published in a journal published by Elsevier, and the attached copy is provided by Elsevier for the author’s benefit and for the benefit of the author’s institution, for non-commercial research and educational use including without limitation use in instruction at your institution, sending it to specific colleagues that you know, and providing a copy to your institution’s administrator.

All other uses, reproduction and distribution, including without limitation commercial reprints, selling or licensing copies or access, or posting on open internet sites, your personal or institution’s website or repository, are prohibited. For exceptions, permission may be sought for such use through Elsevier’s permissions site at:

http://www.elsevier.com/locate/permissionusematerial
Online reservoir adaptation by intrinsic plasticity for backpropagation–decorrelation and echo state learning

Jochen J. Steil*

Honda Research Institute Europe GmbH, Offenbach, Germany
University of Bielefeld, Neuroinformatics Group, P.O.-Box 10 01 31, 33501 Bielefeld, Germany

Abstract

We propose to use a biologically motivated learning rule based on neural intrinsic plasticity to optimize reservoirs of analog neurons. This rule is based on an information maximization principle, it is local in time and space and thus computationally efficient. We show experimentally that it can drive the neurons’ output activities to approximate exponential distributions. Thereby it implements sparse codes in the reservoir. Because of its incremental nature, the intrinsic plasticity learning is well suited for joint application with the online backpropagation–decorrelation or the least mean squares reservoir learning, whose performance can be strongly improved. We further show that classical echo state regression can also benefit from reservoirs, which are pre-trained on the given input signal with the implicit plasticity rule.

© 2007 Elsevier Ltd. All rights reserved.

Keywords: Online learning; Reservoir adaptation; Time series prediction; Intrinsic plasticity

1. Introduction

In recent years, different approaches to use fixed recurrent networks of spiking or analog neurons for computing have been introduced under the notions liquid state machine (LSM) (Maass, Natschläger, & Markram, 2002) or echo state network (ESN) (Jaeger, 2002; Jaeger & Haas, 2004), respectively. In an online learning setting, backpropagation–decorrelation (BPDC) learning was proposed (Steil, 2004) showing that a reservoir-like learning technique can be justified directly from an error minimization approach introduced originally in Atiya and Parlos (2000). The employed recurrent networks usually have sparse random connections and are commonly denoted (dynamic) reservoirs. As proposed also in Verstraeten, Schrauwen, D’Haene, and Stroobandt (2006) we will use the term “reservoir computing” to summarize the common background of these approaches. Reservoirs of spiking neurons can serve as a model for neural microcircuits (Natschläger & Maass, 2005), and have universal computational properties (Maass, Joshi, & Sontag, 2006). They have been applied to the task of generating complex movements (Joshi & Maass, 2005) and recognition of isolated words (Verstraeten, Schrauwen, Stroobandt, & Campenhout, 2005) and digits (Verstraeten, Schrauwen, & Van Campenhout, 2005). ESN networks have originally been applied to time-series prediction tasks (Jaeger & Haas, 2004), but recently also to model a brain–machine interface (Rao et al., 2005), or for speech recognition (Skowronski & Harris, 2006). BPDC learning has so far been applied to several benchmarks in time-series prediction (Steil, 2006).

The fundamental idea of reservoir computing is that inputs to the reservoir excite its nonlinear dynamics. This leads to a complex nonlinear transformation of the input signal into a high dimensional vector of reservoir neuron states. These state vectors can be regarded functionally as nonlinear temporal feature vectors encoding information about the time development of the inputs. Based on these features (or codes) a readout function can be learned effectively. This general approach resembles kernel methods and in fact has been shown to work effectively with even simpler temporal transformations as for instance a bank of linear filters with random coefficients (Fette & Eggert, 2005). In this paper, we focus on reservoirs of analog standard sigmoid neurons, that is on echo state and backpropagation–decorrelation networks.
Reservoir learning methods differ in the way the output function is learned. In its most popular version, ESN learning first collects state vectors of the reservoir for a reasonable long sequence of inputs. Then weights for a linear readout neuron are obtained by performing a standard linear regression with respect to the desired outputs, i.e. in an offline or batch technique. If feedback from the output into the network is desired, teacher forcing has to be used in this setting. On the contrary, BPDC learning, the related least means squares algorithm (LMS), or more advanced versions of LMS including regularization (Rao et al., 2005) incrementally adapt the weights connecting from the reservoir to the output. These algorithms simultaneously change the reservoir dynamics through feedback weights projecting back from the output into the reservoir. A more detailed account on BPDC, LMS, and its differences to ESN is given in Section 3 below.

Obviously, the potential of reservoir computing depends on the properties and the quality of the encoding of the input signal in the reservoir. Due to its nonlinear nature there are to our knowledge no systematic information theoretic measures to grade the efficiency of the generated reservoir codes. Further, as the reservoirs usually are designed to have stable unforced dynamics (Jaeger, 2002; Steil, 2005), computations are actually based on the transients induced by the inputs. Thus the quality of the encoding is dependent on the amount of temporal memory realized in these transients. In Jaeger (2002), Jaeger and Haas (2004) it was claimed that a sufficiently rich dynamics in the reservoir is essential for efficient operation and that echo state reservoirs should be sparsely connected. Additionally, it was argued that the spectral radius of the weight matrix should be scaled such that the reservoir eigenvalues lie well inside the unit circle. This viewpoint has been criticized already in Verstraeten et al. (2006), where evidence was given that scaling to larger spectral radii might actually improve performance. In Steil (2006), a different approach to scale the reservoir weight matrix has been proposed which essentially tries to move the reservoir as close as possible to the border of the region where stability cannot be guaranteed any more. The rationale behind this is to induce longer transients in the network and indeed experiments in time-series prediction given in Steil (2006) show that the performance of the BPDC algorithm is best in this region. This approach of making the reservoir dynamics as complex as possible while maintaining stability is in line with further results on standard discrete recurrent networks implementing finite memory machines (Hammer & Tiffo, 2003), which show that the amount of memory in the system increases for larger weights. It has recently also been argued that for a simplified liquid state machine model it is favorable to operate close to the stability border (Bertschinger & Natschl¨ager, 2004; Natschl¨ager, Bertschinger, & Legionstein, 2005). Neither of these considerations, however, give hints on how to optimize a reservoir in a signal specific way.

In this contribution, we show that a biologically motivated and computationally efficient online learning rule to adjust threshold and gain of sigmoid reservoir neurons can improve the encoding in the reservoir. This “intrinsic plasticity” (IP) rule has first been introduced in Triesch (2005a) in the context of sparse coding in feedforward networks. It is local in time and space and tries to optimize information transmission of the neuron with respect to its input distribution. This distribution in turn is dependent on the input signal. Therefore, the IP rule for the first time yields a method to learn an input specific optimized reservoir encoding. It turns out that the respectively shaped reservoirs are scaled even far beyond the border where analytical proofs can validate stability, i.e. have eigenvalues far outside the unit circle. We show in simulation experiments that this kind of online reservoir adaptation interacts very well with the BPDC and LMS online learning rules and can also improve the performance of the offline echo state regression.

2. Recurrent reservoir dynamics

In the following, we consider the recurrent reservoir dynamics

$$\mathbf{x}(k + 1) = \mathbf{W}_{\text{res}} \mathbf{y}(k) + \mathbf{W}_{\text{imp}} \mathbf{u}(k),$$

(1)

where \(x_i, i = 1, \ldots, N\) are the neural activations, \(\mathbf{W}_{\text{res}} \in \mathbb{R}^{N \times N}\) is the reservoir weight matrix, \(\mathbf{W}_{\text{imp}} \in \mathbb{R}^{N \times R}\) the input weight matrix, \(k\) the discrete time step, and \(\mathbf{u}(k) = (u_1(k), \ldots, u_N(k))^T\) the \(R\)-dimensional vector of inputs. Throughout the paper we assume that \(\mathbf{y} = \mathbf{f}(\mathbf{x})\) is the vector of neuron activities obtained by applying parameterized Fermi functions component wise to the vector \(\mathbf{x}\) as

$$y_i = f_i(x_i, a_i, b_i) = \frac{1}{1 + \exp(-a_i x_i - b_i)}.$$  

(2)

Note that an individual bias for neuron \(i\) can be implemented by setting \(b_i\) to a nonzero value. As we include recurrent connections from output neurons into the reservoir, we conventionally do not distinguish between output and reservoir neurons. We define for a one-dimensional target signal \(d_t(k)\) the activation \(x_1(k)\) of neuron 1 as readout. Then Eq. (1) can be rewritten in explicit matrix–vector form as

$$\begin{pmatrix} x_1(k + 1) \\ x_2(k + 1) \\ \vdots \\ x_N(k + 1) \end{pmatrix} = \begin{pmatrix} W_{\text{res}}^{\text{out}} \mathbf{y}(k) \\ \mathbf{w}_{\text{res}}^{\text{res}} \\ \mathbf{w}_{\text{imp}}^{\text{res}} \\ \mathbf{u}(k) \end{pmatrix},$$

(3)

$$\begin{pmatrix} y_1(k) \\ y_2(k) \\ \vdots \\ y_N(k) \end{pmatrix}$$

where we have further subdivided the input-to-reservoir matrix \(\mathbf{W}_{\text{imp}}\) and the reservoir-to-reservoir matrix \(\mathbf{W}_{\text{res}}\) to separate the
functionally different regions of the weight matrix. We denote by \( W_{\text{in}} \) the connections from \( \ast \) to \( \square \) using \( \text{inp} \) for input, out for output, and \( \text{res} \) for inner reservoir neurons. The first row of the weight matrix in (3) contains the trainable connections projecting from input/reservoir to the output neuron, see also Fig. 1 for illustration. Unless otherwise stated, we assume full connectivity everywhere except in the inner reservoir where \( W_{\text{res}} \) has 10% connectivity only. All weights are initialized in \([-\alpha, +\alpha]\) with \( \alpha = 0.05 \). The exact value of \( \alpha \) for initialization is not very important, because we will adapt the reservoir later.

3. Intrinsic plasticity reservoir optimization

Intrinsic plasticity (IP) is a well known principle found everywhere in the brain. It denotes the capability of biological neurons to change their excitability according to the stimulus distribution the neuron is exposed to (see e.g., the recent reviews (Daoaud & Debanne, 2003; Destexhe & Marder, 2004; Zhang & Linden, 2003)). While only a few formal models are available to explain intrinsic plasticity mechanisms, there are in vitro measurements of neurons in the temporal visual cortex firing with exponential output rates (Badeley et al., 1997). In Stemmler and Koch (1999), it has been reasoned that the underlying principle can be that exponential output distributions maximize information transmission under the constraints imposed by maintaining a constant mean firing rate. This is reasonable to control the metabolic costs of firing and ensure a certain degree of sparsity in the resulting code accordingly. This idea has been pursued in Triesch (2005b) and been transferred to analog formal neurons with continuous nonlinear activation function. A learning rule was given to adjust parameters of the nonlinearity, based on estimating and adjusting moments of the neurons output distribution. It has been shown that a sparse code for the bars problem can be

We first investigate the generic properties of intrinsic plasticity learning and its effects on coding of temporal signals in the reservoir without application of output learning. We expect that the IP adaptation results in an approximate exponential output distribution of the neuron and therefore sparsity in the temporal encoding. Figs. 2 and 3 show that such behavior can experimentally be observed at the single neuron level and at the level of average activity in the network. Fig. 2 displays a randomly chosen neuron’s output distribution after application of the IP rule to a network of 200 neurons. We compare the behavior for input from the Mackey–Glass...
The output distribution of a single randomly chosen neuron under IP learning with Mackey–Glass or random input to the reservoir approaches exponential characteristics. The neuron’s output activity is displayed as a histogram in bins of 0.01 and displayed as the percentage of time steps having the given activity within the epoch of 4000 time steps. For the sake of comparison, IP learning is not applied in the first epoch (top figures).

Development of the spatio-temporal reservoir output distribution of all neurons in the network under IP learning with Mackey–Glass or random input to the reservoir. All neurons’ outputs activities are histogramed in bins of 0.01 and displayed as the percentage of time steps having the given activity within the epoch of 4000 time steps.

Spatial distribution of activity in a 1000 neuron network. Histograms are in bins of 0.01 and displayed as the percentage of neurons with the given activity.

time series and random input in [0, 1]. The networks are iterated in epochs of 4300 steps while the IP learning rule is applied in each step with target average activity \( \mu = 0.2 \) and learning rate \( \zeta = 0.001 \). We use identical inputs and zero initial states for each epoch, record the network outputs \( y(k) \) after a relaxation phase of 300 steps, and plot the histogram of activities of a randomly chosen neuron in Fig. 2 in 100 bins of width 0.01. It can clearly be recognized that the single neuron qualitatively approximates an exponential output distribution taking large values only at a few time steps in the epoch. In Fig. 3 we plot the spatio-temporal histogram of all network activities collected over the same 4000 time steps, which shows that also at the network level an exponential-like spatio-temporal distribution pattern is reached. Fig. 4 illustrates that also the spatial distribution of activity in a single time step is qualitatively similar to an exponential. To have enough data for histogramming, we apply for this case IP training to a 1000 neuron network driven by the same random input as for the previous experiment and show the spatial activity at the end of 200 epochs of IP learning as well the average spatial histogram taken over the last 100 steps of learning in the final epoch. The

1 Identical random input is used for the tenth order system benchmark described in the Appendix and in Section 4.
similarity to an exponential distribution is apparent if only of qualitative nature. We elaborate on some theoretical problems and perspectives on this issue in the discussion section.

As expected from the nature of the IP learning rule, we achieve after 100 epochs of IP learning approximately the same output distribution for the different input distributions, which are driven by the different input signals. Also other inputs like periodic functions or the Santa Fe laser data lead after IP learning to qualitatively the same output pattern (and therefore are not shown). As a result, the IP rule imposes a strong regularization on the overall network activity while enforcing a certain degree of sparseness in the spatio-temporal encoding.

3.2. Gains, eigenvalues, and stability

The IP rule guarantees that the targeted distribution average $\mu$ is approximately reached quickly, typically within the first epochs of learning. This level is then maintained further on. With respect to Eq. (4), the IP rule then aims at maximizing the output distribution entropy to match the desired exponential output distribution, which leads to the output patterns shown in Figs. 2 and 3. We always observe that this results in larger gains $a_i$, such that the neurons are increasingly active in a very small specialized region. Fig. 5 shows a typical development of the gains. It can be explained by rewriting the adaptation rule for the gains as $\Delta a(k) = \zeta a^{-1}(k) + \Delta b(k)$. For small $\Delta b$ the solution of this difference equation is approximately $a(k) = c \sqrt{k}$, which qualitatively fits the plot in Fig. 5.

This change of gains also leads to a redistribution of eigenvalues of the reservoir weight matrix, which is fixed in testing mode and therefore can be compared to a fixed standard reservoir matrix. It is common practice in reservoir learning to scale the weight matrix $W$ such that it has a spectral radius $\rho(W)$ smaller than one (assuming uniform gain parameters $a_i = 1, i = 1 \ldots N$). This usually ensures global exponential stability of the unique equilibrium of the autonomous network running without inputs and thereby avoids the occurrence of undesired autonomous dynamics in the network. However, changing the gains can potentially change the stability behavior of the network. To compute the effective weight matrix eigenvalues, we multiply all input weights with the gain factor and compute the eigenvalues $\lambda_i(WD)$, where $D = \text{diag}(a_i)$ is a diagonal matrix consisting of the gains $a_i$ of the transfer functions $f_i$. Fig. 6 shows that already after a few epochs of learning the eigenvalues move quickly away from the center of the unit disk. Note that as soon as the eigenvalues move outside the unit disk, the recurrent network leaves the small gain stability region, where global exponential stability of the unforced network can be guaranteed by standard feedback system analysis (Steil, 2006). The corresponding sufficient small gain stability criterion approximates the nonlinear functions $f_i(x_i)$ by linear functions $a_i x_i$ employing the maximum of the derivative, which is the gain $a_i$. Very recently it has also been shown in Wang and Xu (2006) that for an even more general network model the criterion $\rho(WD) < 1$ evaluated at the equilibrium point is also necessary for global exponential stability of the equilibrium, where $\rho$ is the spectral radius and $D = \text{diag}(a_i)$ the diagonal matrix of the gains. But this criterion only states that in case of $\rho(WD) \geq 1$ there exist initial conditions which converge to a different equilibrium $x'$ or lead to autonomous persisting network dynamics like (quasiperiodic) oscillations.

Summarizing, the conventional stability analysis by computing properties of the Jacobian $WD$ or directly the weight matrix eigenvalues would call for a criterion limiting the size of the eigenvalues, which for IP learning means to limit the gains and stop IP adaptation early. But in practice, despite violating the analytic (typically only sufficient) criteria for global exponential stability, we observe that the IP trained networks behave very stably in the sense that their trajectories have no tendency to diverge or to develop self-sustained non-trivial dynamics. When removing the input, IP trained networks quickly
converge to a stable equilibrium. We have observed unstable behavior only when IP learning is applied for very long training times (>1000 epochs) resulting in very large gains and long after the eigenvalues leave the unit circle. Thus, in practice, stopping IP is necessary only long after a stopping criterion for output learning typically applies. Stability in the practical sense is maintained by IP learning rather dynamically. It keeps the mean output close to the desired µr, which prevents too much activity and divergence in the overall network by means of the very definition of the method.

4. Backpropagation–decorrelation and intrinsic plasticity

Now we investigate whether the online reservoir learning methods BPDC and LMS can profit from the sparse encoding pattern, which results from IP as shown in the previous section. While the influence of adaptation of a single output neuron on the input distributions for the reservoir neurons is hard to quantify, intuitively we expect that the changes should be small and can be balanced by ongoing IP learning. This gives us reason to hope that online learning of output weights even in the presence of output-to-reservoir feedback connections will hardly interfere with the online IP process. On the other hand, it can be expected that the corresponding sparse reservoir code facilitates learning for the output neuron.

4.1. The online backpropagation–decorrelation rule

In Steil (2004, 2006) the backpropagation–decorrelation (BPDC) learning rule has been introduced as a new online approach to reservoir computing. It follows the basic principles of reservoir computing: the employment of a non-adaptive recurrent reservoir of inner neurons to encode input signals, a back propagation of current errors to train a readout layer, and the usage of the temporal memory in the network reservoir dynamics. However, compared to a standard echo state network the BPDC algorithm computes the errors in a non-standard way based on an earlier introduced error minimization formalism and applies the learning rule online. It can cope with full feedback from the output to the reservoir while learning and therefore also adapts the temporal dynamics in the reservoir online. This process can be shown to follow a “decorrelation of activations principle” (Schiller & Steil, 2005; Steil, 2004).

Finally, it includes more explicitly a linear feedforward path enabling us to utilize a standard linear combination of inputs as part of the output represented by the weight vector w_out as shown in Fig. 1.

BPDC learning is a supervised learning technique, has linear complexity $O(N)$ in the number of neurons $N$, and in Steil (2004, 2006) it has already been shown that BPDC performs well on a number of standard benchmarks of time series prediction. It can be operated in a regime where stability of the overall feedback system is continuously monitored and can be enforced online at virtually no extra computational costs (Steil, 2005, 2006).

BPDC roots in a constraint optimization formalism introduced first in Atiya and Parlos (2000) which has been denoted as virtual teacher forcing in Schiller and Steil (2005). The basic idea is to start with the standard quadratic error for the teacher signal $d_t(k)$

$$E = \sum_k (x_t(k) - d_t(k))^2$$

and to compute a desired change in the state variables $\Delta x_t(k) = -\frac{\partial E}{\partial x_t(k)}$ proportional to the negative gradient of $E$ with respect to the state — the desired virtual teacher. Then the network equations are used as constraints to compute a $\Delta w(k)$ to realize this $\Delta x_t(k)$, which in turn would decrease the error function $E$. A formalism to derive the corresponding $\Delta w(k)$ was given in Steil (2004, 2006) and leads to the following BPDC rule. For a weight $w_{ij}$ connecting the reservoir neuron $j$, $i = 1 \ldots N$ to the output neuron $x_1$ (including the recurrent connection $w_{11}$), we have

$$\Delta w_{ij}(k) = \tilde{\eta}(k)f_j(x_t(k - 1))\gamma_t(k), \quad (7)$$

and for all weights $w_{ir}$ connecting inputs $u_r(k)$, $r = 1 \ldots R$ to the output neuron we have

$$\Delta w_{ir}(k) = \tilde{\eta}(k)u_r(k - 1)\gamma_t(k). \quad (8)$$

Here

$$\tilde{\eta}(k) = \frac{1}{\|f(x(k - 1))\|^2 + \|u(k - 1)\|^2 + \epsilon} \quad (9)$$

can be interpreted as a time dependent learning rate. It scales the BPDC learning rate $\eta$ with a factor dependent on the overall network activities, the input activities, and a regularization constant $\epsilon = 0.002$. The error term $\gamma_t(k)$ is defined as

$$\gamma_t(k) = -e_t(k) + w_{11}f'(x_t(k - 1))e_t(k - 1), \quad (10)$$

where $e_t(k)$ is the error at time $k : e_t(k) = x_t(k) - d_t(k)$ with respect to the teaching signal $d_t(k)$. In Steil (2004, 2006) the BPDC formalism was given in a slightly different version, because inputs were assumed to be latched to certain neurons such that the $u(k - 1)$ did not appear explicitly in (8) and (9). For the sake of more clarity we made the dependence on the input clearly visible here.

It is easy to see that the BPDC learning constitutes a perceptron-like error correction rule with time-dependent learning rate $\tilde{\eta}(k)$, input predicate $y_j(k) = f_j(x_j(k))$, and modified error $\gamma_t(k)$. If the self recurrent weight $w_{11}$ is not present, then the error is the standard error of the delta rule. If we further omit the time-dependent scaling factor in (9), the BPDC rule reduces to the standard least mean squares (LMS) error correction.

4.2. BPDC and LMS learning with IP for forward prediction

In the following, we give simulation results for time series prediction tasks on the Mackey–Glass attractor, a tenth order nonlinear system driven by random input, and the Santa Fe laser data, which are described in more detail in the Appendix. The common task is a one step ahead prediction of the time series based on a history of previous inputs. For Mackey–Glass and Santa Fe we use a time window of previous values as input, i.e.
Fig. 7. One step forward prediction results for the tenth order system: NMSQE vs. Epochs of 1000 training/test steps in linear scale. With IP learning, BPDC and LMS perform almost identically. Without IP learning, LMS learns faster in the first epochs, but BPDC can be superior in the long run.

\[ u_1(k) = d(k), u_2(k) = d(k - 1), \ldots, u_R(k) = d(k - (R - 1)) \]

for the target value \( d(k + 1) \). In the case of the tenth order system, the input values are drawn from a uniform distribution, i.e. \( u_1(k) \in [0, 1] \) and \( u_2(k) = u_1(k - 9) \). Testing feeds the continuation of the true time series (including the full history window) for the Mackey–Glass and Santa Fe data, while for the tenth order system respective random inputs are drawn for the test steps recorded, and then reused in all epochs.

We train the reservoir network in training/test epochs. For each epoch, 1000 points of the respective trajectory are presented to the network for training and the next 1000 points for testing. For each epoch the network states are reinitialized to zero. A single training step comprises the forward iteration of the network, error computation with respect to the target signal, the BPDC or LMS adaptation of the output weights, and finally application of the intrinsic plasticity rule for the non-output neurons. When testing, all weights are fixed and neither output nor IP adaptation takes place. Training error computation is started after discarding the first 100 points to let transients decay (this initial phase is often called washout in the Echo State literature), while the test errors are computed over the full 1000 test points. In Figs. 6–9, we plot the respective normalized mean square errors (NMSQE) against epochs of learning.

For all cases shown we compare BPDC against LMS with and without IP learning. To make a fair comparison, we use a fixed learning rate for LMS, which approximates the BPDC scaled learning rate as given in (9) by substituting means for the inputs and the network activities. We set \( \eta_{\text{LMS}} \) for LMS to

\[ \eta_{\text{LMS}} = \eta_{\text{BPDC}} \frac{1}{N \mu^2 + \sum_r \bar{u}_r^2}. \]  

Here \( N \) is the number of neurons, \( \bar{u}_r \) is the mean of the \( r \)-th input, and \( N \mu^2 \) approximates the sum of squared network activities in the denominator of (9) assuming that the IP rule succeeds to drive the activities to the desired mean value. Because this is valid only after longer training, the LMS rate is larger than that for BPDC in the first epochs resulting
All results indicate that already at the beginning of learning after a few epochs the inclusion of intrinsic plasticity significantly improves the learning performance as compared to application of BPDC/LMS alone. This effect comes surprisingly fast as we choose the learning rate for IP as $\xi = 0.001$, much slower than the BPDC rate $\eta = 0.03$. It is also apparent that in the long term, i.e. for many epochs, training including the IP rule term drives the system to a much lower error level. This is consistent with the findings in the previous sections, which show that IP in the beginning quickly distributes activation to a wider spectrum of output values. This explains the early improvements in training with IP. In the long run, i.e. after many epochs of training, the optimization of the network’s spatio-temporal activity distribution towards an exponential characteristics typically enables us to achieve a smaller error level than without IP learning. Generally we observe that under the IP regularization the differences between LMS and BPDC often are insignificant indicating that both algorithms are capable of exploiting the advantages of the IP coding in the reservoir to the same degree.

The varying characteristics of the three time series also lead to differences in performance. As shown before in Section 3, the random input applied for the tenth order system imposes a quite regular and widespread activity distribution in the network (compare Fig. 2) in early epochs of learning. Fig. 7(a) and (c) consequently show a very regular error development, which hardly differs between training and test and is similar for LMS and BPDC. There is a slight long term decrease of errors accountable to the IP process. The development of errors without using IP in Fig. 7(b) and (d) shows more interesting features. Initially LMS performs much better than BPDC, both on training and test, however, if training for very long, BPDC outperforms LMS on the test data. The two cases shown differ only in the number of neurons used and show qualitatively similar behavior. The main difference is the speed of error decrease, which is implicitly caused by the number of neurons.

This number determines the effective learning rate to be larger for the smaller network because of the activity dependent scaling.

Fig. 8 shows results for the Mackey–Glass time series for two different settings of the history window with length one in (a), (b) and length five in (c), (d). As for the tenth order system, the train/test errors hardly differ if IP learning is used and BPDC and LMS perform very similar to each other. Without IP, errors stabilize at a much larger value than for IP learning and, consistent with the results for the tenth order, LMS initially performs much better than BPDC, while BPDC can catch up to or even outperform LMS in the long run (Fig. 8(b), (d)). The most interesting feature is the dependence of the results on the history length. A non-trivial input time window introduces a direct linear input–output path in the system and provides more explicit information to the system, which therefore does not have to be stored in the reservoir. Thus it can be expected that a longer input history window leads to better performance and without IP learning this indeed is supported by the simulations. Introducing IP learning changes this picture as can be seen from the Fig. 8(a) vs. (c). In the first approximately 100 epochs the network with more history is driven towards a lower error level, as expected, but with the long term fine tuning of the distribution performed by the IP learning, the dynamic effects in the reservoir seem to take over and the network without history in fact approaches a final even smaller level much faster. It seems that in the case of the larger history and the existence of the direct input–output path, which is not influenced by the IP learning, the network falls into a local minimum largely determined by this linear combination of the inputs. Then it is more difficult for the network to exploit the sparse code formed in the reservoir.

Figs. 9 and 10 show larger differences between BPDC and LMS for the more rapidly oscillating Laser data. The advantage of IP learning is once more clearly visible. While LMS with IP learning achieves better training error, it performs badly in testing, both with and without IP learning. Fig. 9 highlights a further very interesting behavior of IP learning, which we
have observed quite frequently when learning the Laser data. At about epoch 150, we have a typical overfitting situation, where the training error decreases, but the test error increases. However, as the IP learning proceeds, it succeeds in attracting back the test error to much smaller error levels. It seems that the long term shift of the activity distributions towards exponential output under maintenance of a small training error constitutes a kind of homeostatic code shift towards more sparsity, which is favorable for generalization.

4.3. Transients and recursive prediction

In time series prediction, extrapolation of the training data into the future by recursive prediction is commonly used to benchmark different methods. The goal is to implement a target attractor dynamics by feedback of outputs to inputs after training. This closes an outer loop not present at training time, which potentially destabilizes the recursively connected network. With the goal to implement an attractor, the recursively connected network needs to have persistent autonomous dynamics and must not converge to an equilibrium. Thus learning has to increase the length of transients, which can experimentally be measured by recursively connecting the network and then measuring the time until convergence. A similar approach to account for complexity of the reservoir dynamics has been pursued in Triesch (2006) in the context of a very simple spiking network.

Fig. 11 plots the length of transients against epochs learned for the Mackey–Glass prediction task. Length is measured from the point when, after training, the output is fed back to the network for recursive prediction. With perfect learning of the attractor, the network would autonomously produce the non-stationary chaotic dynamics. In this case, the recursively connected network does not converge to a stable state. In practice, the network at first follows the non-stationary attractor dynamics only for some time before it converges to a stable state. We take the length of this transient as an indicator for how close the network is to a state where autonomous non-trivial dynamics are present. Length here is measured by the number of steps after recursively connecting the network until a moving average of the activity of the output neuron does not change more than 0.002 in 20 steps. It is apparent from Fig. 11 that without IP the online learning is not at all capable of driving the network out of the range, where undesired convergence to stable states is present. Naturally, online algorithms are less suited for recursive prediction tasks, because they are always more sensitive to the most recent data presented. The necessarily correlated presentation of data adds inevitable local trends, which can disturb a successful recursive prediction. Nevertheless, Fig. 12 shows that even with BPDC/LMS reasonable results for recursive prediction can be obtained when using small learning rates and allowing for long training time.

5. Echo state and intrinsic plasticity

Though by its incremental nature the IP rule is ideally suited for interaction with the online BPDC-rule, it is as well possible to use it in the echo state context. We demonstrate this in the setting of recursive prediction of the Mackey–Glass attractor similar to the experiment described in detail in Jaeger and Haas (2004), see also Appendix. Basically the reservoir is fed with the input signal, the state vectors in the reservoir are recorded for 2000 steps (after ignoring the first 1000 steps to wash out the transients in the network), and a standard tool for linear regression is used to compute suitable output weights. Then for recursive prediction the output is connected to the input and the network runs autonomously in this configuration. While we have chosen most parameters of the training setting identical to Jaeger and Haas (2004), there is a major difference in that we employ only 100 neurons instead of the much larger reservoirs there, where 1000 neurons or pools of several networks of 1000 neurons were used. We restrict ourselves to smaller networks, because we encountered many unstable network configurations for larger networks when recursively connected, in particular when they are pre-trained with IP. This happens despite the network being stable without the recursive outer loop. Thus,
Fig. 12. Recursive prediction of the Mackey–Glass time series after 980 epochs of learning. Parameters: IP learning rate = 0.0001, BPDC learning rate = 0.01, input history window length = 10, recursive prediction starts after training at time step 2000.

Fig. 13. Comparison of results for linear regression output learning on reservoirs without IP adaptation (standard ESN) and after application of IP learning for 5, 10, 15, 20 epochs before fixing the reservoir weights and application of the output regression. Left: Time window for inputs of length 10. Right: Time window for inputs of length 15. All results are averages over 20 runs with different training sequences evaluated on 20 test sequences each.

we do not aim at reproducing or improving the extremely sharp results from Jaeger and Haas (2004). The results rather show that under limited resources IP can improve coding to gain performance for echo state learning as well. This underlines the claim that the improvements in performance by means of IP learning, which we observed in the previous section, are due to the improved spatio-temporal encoding in the reservoir.

Fig. 13 shows that the normalized mean square error for the recursive prediction of the Mackey–Glass attractor is significantly smaller when we first train the network for several epochs with IP rule and then apply echo state regression for the output weights. This indicates that indeed the encoding of the temporal dynamics of the input in the network is improved by the intrinsic plasticity.

6. Conclusion

To our knowledge, with the intrinsic plasticity learning we have proposed the first online adaptation rule for signal specific optimization of a reservoir. It is rooted in a biologically plausible mechanism for optimizing information transmission while maintaining a metabolically favorable mean firing rate. Transferred to the recurrent reservoir network, it drives the network to approximate exponential activity distributions with given mean activity rate, temporally at the neuron level, spatially at the network level, and also on the spatio-temporal level. These distributions imply a sparse coding principle, because on average only a few neurons can be active. In the experiments we observe a considerable regularization effect in the sense that the IP rule always succeeded to drive the spatio-temporal activity distribution to very similar results, when training long enough. That is, IP learning changes the dynamics in a signal specific way to achieve a regularized standard network mean activity pattern. Setting the targeted mean distribution average rate to small values, e.g. $\mu \in [0.1, 0.3]$, also leads to very stable network dynamics. Stability in this case is rather maintained dynamically by controlling the output distribution than by enforcing conservative stability theory constraints as proposed in Steil (2006) and discussed in Section 3.

The most striking and yet unexplained feature of the intrinsic plasticity learning is its empirically shown success in driving the network to the desired exponential-like distribution. In fact, adjusting only two parameters locally per neuron can maximally fix mean and standard deviation of the respective distribution. The corresponding maximization of the entropy,
however, may lead to local minima resulting in non-exponential distributions. We found empirically that the IP rule achieves to obtain the targeted means and variances (Wardermann & Steil, 2007), but this still does not explain why the learning seems to succeed in getting close to the global minimum of an exponential distribution. We speculate that the stochastic nature of the online update may be useful and a reasonable route to investigate this issue could be to test batch updates of the IP learning as well. Further mathematical investigations also seem to be necessary to clarify whether there is something like a global convergence to a unique final distribution in the network. Carefully conducted statistical tests on properties of the final distribution may help in this direction. Such investigations might then lead to modifications of the learning rule, for instance with the goal to stop or schedule IP learning, and to extensions of the idea towards other nonlinear transfer functions. A first attempt in this direction has been made in Verstraeten, Schrauwen, and Stroobandt (2007), where the Kullback–Leibler divergence with a Gaussian target distribution is minimized for neurons with tanh transfer functions.

The formulation of IP as an online learning rule, which is local in time and space, perfectly complements the BPDC and LMS online learning, which both have been shown to profit in performance from the intrinsic plasticity. The simulation results on three datasets with quite different characteristics indicate that the inclusion of IP learning is always favorable in forward prediction tasks. Comparing BPDC and LMS, the results show that BPDC behaves more stably and often outperforms LMS on the test set. However, also here it is not yet fully understood why and how exactly the used learning algorithms profit from the interaction with the IP learning. We believe that better ways to quantify the information transmission of the reservoir and the properties of the resulting codes will reveal improvements of the algorithms as well as help to a deeper understanding of the complex dynamical interplay of mechanisms inside the network. Despite these and the theoretical issues discussed above, we believe that the combination of BPDC and IP learning can serve as a powerful model for long term learning, which, for instance, should be present when motor behavior is continuously refined and reshaped according to behavioral practice. Therefore we investigate in ongoing work the potential of such networks for learning, recognition, and generation of robot trajectories.

From a different viewpoint, the analysis of the BPDC algorithm as an error correction rule, which is modulated by the overall activity level of the network, shows that both intrinsic plasticity and BPDC are biologically plausible processes. BPDC relies on error correction, which is a common principle in the brain and has been analyzed very well for instance in the cerebellum (Ramnani, 2006). Further, the time-dependent learning rate amplifies or dampens the learning step with respect to the global activity in the network, which is also a biologically plausible mechanism. Therefore, we speculate that a transfer of these principles to spiking networks is possible and that similar online learning rules can be developed in that domain as well.

Acknowledgments

The author would like to thank Dr. H. Wersing for comments on earlier versions of the manuscript. Three reviewers and the guest editors also helped to improve the current work with their most valuable comments. All responsibility for errors of course remains with the author.

Appendix. Experimental setting

For all experiments we chose the following common parameters for the BDPC and IP learning:
- BPDC learning rate $\eta = 0.03$,
- regularization constant $\varepsilon = 0.002$,
- IP learning rate $\zeta = 0.001$,
- IP mean activity level $\mu = 0.2$,
- initialization of weights uniform in $[-0.05, 0.05]$,
- number of nodes: 100
- reservoir sparseness: 10%
- output dimension $= 1$, $O = \{1\}$.

Errors are computed as normalized mean square errors (NMSQE):

$$\text{NMSQE} = \frac{1}{M} \sum_{m=1}^{M} \frac{(x_1(m) - d(m))^2}{\sigma^2},$$

where $\sigma^2$ is the variance of the target signal $d(m)$, and $M$ the number of training steps.

A.1. Laser data

For the experiments with the Santa Fe Laser data we use the first 1000 points of the time series for training, the next 1000 points for testing. In computing the NMSQE we discard the first 100 points of training/testing. One epoch refers to presenting once the training data. Network states are initialized at the beginning of each epoch, but not reinitialized before testing.

A.2. Mackey–Glass

As higher order benchmark we use the well known Mackey–Glass system with standard parameters

$$\dot{y}(t) = -0.1y(t) + \frac{0.2y(t - 17)}{1 + y(t - 17)^{10}},$$

where we integrate from $t \rightarrow t+1$ using 30 Runge–Kutta fourth order steps. One training/testing run for recursive prediction proceeds as follows: first we select an arbitrary sequence of length 3000 from the MG attractor, which is trained in epochs as for the Laser task with the IP rule. The first 1000 points are discarded for the error computations. For testing we select a different MG sequence generated with different initial conditions, iterate the network for 2000 points with the given sequence as input, and then connect the output back to the input. In this case a time history of length $R$ is used, this means that after $R$ steps the network runs fully on its own recursively predicted values. Errors after $k =$
21, 42, 63, 84, 105, 127, 146, 168 steps of recursive prediction are computed and averaged as NMSQE over 20 different testing sequences. The errors given in Fig. 13 are averages of these NMSQEs over 20 different such runs.

A.3. Tenth order system

The following problem in discrete time has also been considered in Atiya and Parlos (2000) and is frequently used as hard benchmark for forward predictions:

\[ y(k + 1) = 0.3 y(k) + 0.05 y(k) \sum_{i=0}^{9} y(k - i) + 1.5 \hat{u}(k - 9) \hat{u}(k) + 0.1, \]

where the random input \( \hat{u}(k) \) is uniformly drawn from \([0, 1] \). The task is to predict the next output \( y(k + 1) \). For the results given in Section 4 the input and target data are shifted by \(-0.5\) and scaled by 2, i.e. networks inputs are generated as \( u_1(k) = (\hat{u}(k) - 0.5) \ast 2, u_2(k) = u_1(k - 9) \).

References


