Contents lists available at SciVerse ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Intrinsic plasticity via natural gradient descent with application to drift compensation

K. Neumann*, C. Strub, J.J. Steil

Research Institute for Cognition and Robotics (CoR-Lab), Bielefeld University, Germany

ARTICLE INFO

Available online 7 March 2013

Keywords:
Neural network
Learning
Intrinsic plasticity
Information geometry
Natural gradient
Metric tensor
Gradient descent
Drift

ABSTRACT

This paper investigates the learning dynamics of intrinsic plasticity (IP), which is a learning rule to tune a neuron's activation function such that its output distribution becomes approximately exponentially distributed. The information-geometric properties of intrinsic plasticity are analyzed and the improved natural gradient intrinsic plasticity (NIP) dynamics are evaluated for a variety of input distributions. Together with a further new modification of the IP rule, the high capability of NIP to cope with drift is demonstrated to have superior performance as compared to the standard gradient in experiments with synthetic and real world data.

1. Introduction

In 2004, Triesch introduced a biologically inspired model of intrinsic plasticity (IP) [1] for optimization of an artificial neuron's activation function based on stochastic gradient descent. The target of IP-learning is to approximate an exponential output distribution irrespective of the given input distribution. This maximizes the neuron's information transmission, related to the high entropy of the target distribution. Since its introduction, the IP-rule has been used to learn sensory representations [2] and enhance the encoding in reservoir networks [3,4], which are of main interest in this paper. A batch-version has been shown to improve extreme learning machines [5] and static reservoirs [6]. It was further shown that in synergies with synaptic plasticity IP can detect heavy tail input distributions [7]. Despite its success in these different domains, the IP learning dynamics have not yet been analyzed in detail and potentially suffer all known drawbacks of standard stochastic gradient. The parameter estimates can lead to small gradient norms in some regions of the parameter space, so-called plateaus, where convergence is slow.

One reason for this is that the parameterization and the corresponding output of a model are defined in different metric spaces. Most gradients defined on an error measure only utilize Euclidean metrics in parameter space. But, generally, there is no reason to assume that the Euclidean metric is the preferential distance measure between solutions. It is well known that the parameter space has a Riemannian metric structure in many cases, for instance in the weight space in neural networks [8]. The parameter space can be analyzed by means of information geometry–a theory which employs differential-geometric methods in statistics [9,10].

While the steepest direction in a parameter space with an Euclidean metric structure is given by the conventional gradient, the steepest direction in a parameter space with Riemannian metric structure is given by the so-called natural gradient. It is obtained by transforming the Euclidean metric in the output space by means of an often only locally defined metric tensor into the parameter space. The tensor needs to be well-suited to the Riemannian metric of the parameter space.

It has been shown that the natural gradient can be advantageous for different stochastic learning setups like blind-source separation or statistical estimation of probability density functions (see e.g. [8,11,12]). It has also been applied to improve the learning dynamics of multilayer networks [8,13]. For instance in [14] the authors distinguish between a transient and an asymptotic phase in the learning dynamics which both show significant gains in performance over standard gradient descent. The concept of natural gradient has further been extended to more general classes of multidimensional regression and classification problems in [15]. An alternative derivation of the natural gradient is given in [16], together with the natural equivalent of batch learning, linked to Levenberg–Marquardt optimization. Recently, the special case of learning for non-linear discriminant networks was improved by use of natural gradient in [17].

As opposed to standard neural learning, where the input weights are adapted, IP learning adapts parameters of the activation function. This paper defines and analyzes the corresponding

* Corresponding author. Tel.: +49 52110667108.
E-mail addresses: kneumann@cor-lab.uni-bielefeld.de (K. Neumann), cstrub@techfak.uni-bielefeld.de (C. Strub), jsteil@cor-lab.uni-bielefeld.de (J.J. Steil).

0925-2312/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.neucom.2012.12.047

Riemannian metric tensor for IP in detail which was first introduced in [18] and thereby introduces the natural gradient for IP. Like in other domains, where natural gradients were previously explored, experiments reveal that the Riemannian metric and the associated natural gradient are more suited to describe distance relations between output distributions for IP and provide superior learning dynamics.

First, IP learning is enhanced with the natural gradient and a further novel mechanism to adaptively transfer persistent changes of the activation function caused by IP to the weights. Furthermore, the paper shows that IP learning simultaneously provides unsupervised input drift compensation. Note that we do not consider the so-called concept drift, which means online changes in the desired output function [19], as opposed to compensation of online changes of the input signal. The latter is highly useful in real world applications where measurements are made long periods of time [20] or if inputs are systematically shifted and scaled through other processes like for instance a sudden change of illumination or a sudden displacement of a camera. The literature shows that a detection of the drift before the compensation is a promising approach [21,22]. In these cases a suitable compensation strategy needs to be chosen in order to cope with the drift successfully. One such strategy is to adjust the data accordingly, e.g. to recenter to compensate respective mean shifts or to rescale to compensate changes in variance. This requires external data analysis and appropriate measures but does not adapt the learned model to internalize the drift. Thereby the learned model does not actually encode for the current real world input, but rather for the input at learning time before the drift occurs.

This is opposed to an implicit approach to drift compensation that internalizes the drift into the model by continuous re-adaptation. In principle, continuous online learning of the weights through backpropagation can provide respective re-learning and that actually is a strong argument in favor of applying online-learning while already exploiting the learned model. But it requires that error feedback is continuously available to change the neural code that solved the learning task for the original data. In real applications, this may not be feasible. Consider for instance the application of a learned virtual sensor for which training data can be generated in the laboratory using a costly direct sensor. The goal is to replace this sensing in the real product, where supervised re-adaptation of the weights is consequently not possible by definition.

Drift compensation through IP learning in the presence of mean and variance changes offers a different and novel approach, which internalizes the drift in the network model so that the input data does not need to be analyzed. It simultaneously sustains the neural encoding, which was learned using the error feedback for the original data, so that there is also no need for continuous error feedback and retraining. To the best of our knowledge, there is currently no other approach with these two features. IP achieves this by exploiting that the considered networks restrict error driven weight adaptation to the outputs of the network, whereas optimization of the input encoding in the hidden layer is decoupled from the output weight adaptation and provided by the IP learning. The drift compensation is therefore in some sense a desired side effect of the local and unsupervised optimization of the encoding of each single neuron in the network. It turns out that only the combination of the proposed modification of the IP rule and the usage of the natural gradient provides an IP learning dynamics that it is well suited for compensating such drifts.

The paper is organized as follows: Sections 2 and 3 review the IP learning rule and describe how the natural gradient is defined for IP. Section 4 describes a technique to tackle numerical issues of the IP learning rule. Experiments that analyze the differential-geometric properties of IP are provided in Section 5 in order to complement the theory of natural gradient. It is also shown how the learning dynamics change when following the natural gradient. Section 6 demonstrates the effects of natural gradient IP learning for compensating drifts in the input, including a real world learning task from robotics. Finally, Section 7 concludes the paper.

2. Intrinsic plasticity

Intrinsic Plasticity (IP) was developed by Triesch in 2004 [1] as a model of homeostatic plasticity for analog neurons with parameterized Fermi functions $y_\theta(x) = (1+\exp(-ax-b))^{-1}$ as activation and parameters $\theta = (a,b)^T$. The goal is to optimize the information transmission of a single neuron strictly locally by adaptation of slope a and bias b such that the neuron's output y becomes approximately exponentially distributed with a fixed mean μ. This is done with respect to the input sample distribution $f_x(x)$, where x is the synaptic sum arriving at the neuron. Typically μ is chosen to be in the interval [0.1,0.3]. IP-learning can be derived by using the insight from statistics $f_y(y) = f_x(x)\cdot(\partial y/\partial x)^{-1}$ and the equation $\partial y/\partial x = ay(1-y)$ obtained by analyzing the Fermi function. Minimization of the difference $L(f_y, f_{\exp}) = L(\theta)$ between the output f_y and an exponential distribution $f_{\exp}$, quantized by the Kullback–Leibler-divergence [23] (KLD), delivers the following:

$$L(\theta) = \mathbb{E}[l(y,\theta)] = \int_\Omega f_y(y)\ln\left(\frac{f_y(y)}{f_{\exp}(y)}\right)\mathrm{d}y \tag{1}$$

$$= \int_\Omega f_y(y)\ln\left(\frac{f_x(x)}{\left(\frac{\partial y}{\partial x}\right)\frac{1}{\mu}e^{-(1/\mu)y}}\right)\mathrm{d}y \tag{2}$$

$$= \underbrace{\int_\Omega f_y(y)\ln(\mu f_x(x))\,\mathrm{d}y}_{C} - \int_\Omega f_y(y)\ln\left(\frac{\partial y/\partial x}{e^{(1/\mu)y}}\right)\mathrm{d}y \tag{3}$$

$$= C + \underbrace{\int_\Omega -\ln\left(\frac{ay(1-y)}{e^{(1/\mu)y}}\right)}_{l(y,\theta)} f_y(y)\,\mathrm{d}y, \tag{4}$$

where C remains as a constant of the potential and can be neglected without loss of information. Since IP was introduced in the form of a stochastic gradient decent rule, the respective online loss function can be identified with the integrand $l(y,\theta)$ (see Eq. (4)). Here the KLD is interpreted as expected loss $\mathbb{E}[l(y,\theta)]$ for the input samples x distributed according to $f_x(x)$. A separation of Eq. (4) into the entropy $H_x[y]$ and the expectation value of the output distribution $\mathbb{E}_x[y]$ is possible [24], which directly shows that a minimization of $L(\theta)$ for a fixed mean $\mathbb{E}_x[y]$ is equivalent to entropy maximization of the output distribution. Additional information about the KLD and its distance to exponential distributions can be found in [25]. The typical approach is to use the stochastic gradient of this potential in order to find a minimum of the expected loss function. The following online update equations for slope and bias – scaled by the step width η_{IP} – are obtained:

$$\Delta a = \frac{\eta_{\mathrm{IP}}}{a} + x\Delta b, \quad \Delta b = \eta_{\mathrm{IP}}\left(1-\left(2+\frac{1}{\mu}\right)y + \frac{1}{\mu}y^2\right). \tag{5}$$

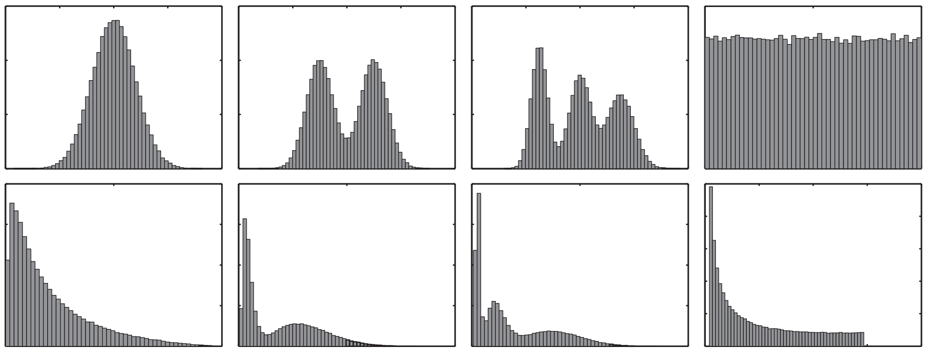

Fig. 1. Four input distributions $f_x(x)$ (1st row) and the learned exponential-like output distributions $f_y(y)$ for $\mu = 0.2$ (2nd row).

Fig. 1 shows how four different input distributions (first row in the figure) are transformed into exponential-like distributions (second row in the figure) after training with IP. The figure clearly reveals that the best possible fit after IP learning is highly dependent on the input distribution. This is due to the fact that only two parameters in the Fermi function are adapted. These distributions will be used as input for the experiments in the following sections.

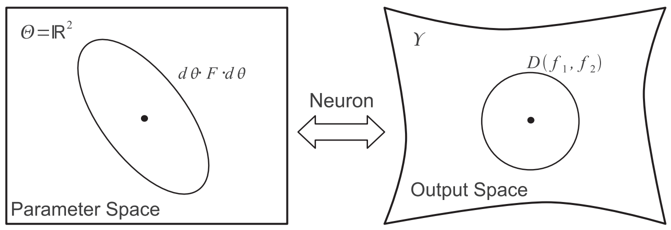

Fig. 2. Differentiable relation (Neuron) and metrics F and D between parameter space $\Theta = \mathbb{R}^2$ and manifold of possible output distributions Υ.

3. The natural gradient for intrinsic plasticity

Given an input distribution $f_x(x)$, an analog neuron establishes a differentiable mapping between the parameter space $\Theta = \mathbb{R}^2$ and the manifold of possible output distributions Υ. The KLD comparing a given distribution to the exponential distribution with fixed mean μ in Eq. (4) can be used to derive a canonical distance measure on the output distribution space resulting in a Riemannian metric F on the parameter space Θ. The metric determining the distance between two output distributions $y_1(x) = y(x,\theta_1)$ and $y_2(x) = y(x,\theta_2)$ in Υ defined by the parameter settings θ_1 and $\theta_2 = \theta_1 + \mathrm{d}\theta$ in Θ for an infinitesimal change of parameters $\mathrm{d}\theta$ is given by $D(y_1,y_2)$. This distance measure is transformed such that it induces the Riemannian metric tensor $F(\theta)$ – a 2×2 positive definite matrix given by the Fisher information [26] – as a pull-back onto the parameter space:

$$D(y_1,y_2) = \mathbb{E}_x[(l(y_1,\theta_1) - l(y_2,\theta_2))^2] \tag{6}$$

$$D(y_1,y_2) = \mathbb{E}_x[(l(y_1,\theta_1) - l(y_1,\theta_1) - \nabla l(y_1,\theta_1)\,\mathrm{d}\theta)^2] \tag{7}$$

$$D(y_1,y_2) = \mathbb{E}_x[(\nabla l(y_1,\theta_1)\,\mathrm{d}\theta)^2] \tag{8}$$

$$D(y_1,y_2) = \mathrm{d}\theta \cdot \mathbb{E}_x[\nabla l(y_1,\theta_1) \cdot (\nabla l(y_1,\theta_1))^T] \cdot \mathrm{d}\theta = \mathrm{d}\theta \cdot F(\theta) \cdot \mathrm{d}\theta. \tag{9}$$

This idea guarantees that the distance between two parameter vectors θ_1 and θ_2 – as measured by the length of the geodesic with respect to the metric tensor $F(\theta)$ in Eq. (9) – is equal to the previously defined distance measure $D(y_1,y_2)$ in Eq. (6) on the corresponding output distributions y_1 and y_2 in Υ. The relation between parameters and output distributions established by a nonlinear transfer function of a neuron and its corresponding distance measures is schematically illustrated in Fig. 2.

As already mentioned before the parameter spaces spanned by neural networks have a Riemannian character [9,26]. The steepest descent direction of a potential with Riemannian structure is given by the natural gradient defined by the metric tensor. The following update equation is obtained when using the natural gradient for IP:

$$\theta_{t+1} = \theta_t - \eta(F(\theta) + \varepsilon I)^{-1}\nabla l(y,\theta) = \theta_t - \eta \nabla_{\mathrm{NIP}} l(y,\theta), \tag{10}$$

where I is the 2×2-identity matrix and $\varepsilon \geq 0$ is a positive scalar. We call $\nabla_{\mathrm{NIP}} := (F(\theta) + \varepsilon I)^{-1}\nabla$ the natural gradient operator for IP. Typically ε can be set to zero to obtain a plain natural gradient formulation. But in the more general definition equation (10), ε introduces a blending between standard and natural gradient. Note that this blending influences the step width of the numerically applied gradient descent and stabilizes the inversion of the metric tensor F.

The main problem with this formulation is that the expected gradient with respect to the input is needed, but not available in an online framework. However, it was shown in [13] that is possible to estimate the metric tensor online by defining a proportional control law:

$$\dot{\hat{F}}(\theta) = \lambda(F(x,\theta) - \hat{F}(\theta)), \tag{11}$$

where $\hat{F}$ is the estimate of the Fisher matrix and $F(x,\theta)$ the Fisher information for one input element x. The problem then reduces to finding a good λ, which must be small since the loss function is continuous and a good initial value $\hat{F}_0(\theta)$. The NIP learning then becomes online capable and computationally feasible.

4. Working-point transformation for intrinsic plasticity

A closer inspection of Eq. (5) (left) reveals that the standard IP rule can suffer from numerical instabilities in particular for large input amplitudes which lead to small slopes a. In this regime of small slopes the discretization becomes problematic due to the singularity induced by the $1/a$-term, illustrated in Fig. 5(A). In combination with the results of experiments in Section 6.1 this reveals that IP has a “working point” at $a=1$. It is therefore favorable to keep the parameter a close to this “working point”, which can be achieved by a novel modification of IP learning. It proposes to scale the neuron's input weights with the slope in order to transform the working point appropriately:

$$\Delta \vec{w} = \eta_{\mathrm{ws}} \cdot (-\vec{w} + a \cdot \vec{w}) \tag{12}$$

with $\eta_{\mathrm{ws}} < \eta_{\mathrm{IP}}$ (η_{ws} is set to 10^{-5} in the experiments). With this additional adaptation rule, the slope tends to converge back to unity, as the weights converge to the former slope values. Hence the semantics of the IP learning rule remain the same while transferring the learned pertinent slope information from the slope parameter a to the synaptic weights.

The collection of learning rules given by Eqs. (10)–(12) will be used in the following experiments and called natural IP (NIP).

5. The impact of the natural gradient on IP

This section presents experimental results for a single-neuron model with parameterized Fermi function where the proposed learning rules are used. The experiments are performed with different inputs: the first row in Fig. 1 shows the four different input distributions that are used for investigation. A Gaussian (1-G), a bipartite (2-G), a tripartite (3-G) Gaussian and a uniform (U) distribution. $N_{\mathrm{tr}} = 100$ samples are independently drawn from each distribution and used for training. A step width of $\eta = 10^{-3}$ and a numerical stabilization of $\varepsilon = 10^{-1}$ is used. For online estimation of the metric tensor a decay rate of $\lambda = 0.01$ is used.

5.1. Information geometry

The following experiment visualizes how the geometry of the potential L changes by use of the metric tensor F at the attractor θ^*. The 1-G distribution is used as input to a single Fermi neuron model for illustration.

Fig. 3 (left) shows schematically the Euclidean and the Fisher metric at a given parameter configuration $\theta = (2,-0.5)^T$. Note that the gradient induced by the Fisher metric is not orthogonal to the equipotential lines anymore. The direction of the steepest descent therefore changes and points in a more direct way to the attractor than the standard gradient for IP which is using the plain Euclidean metric visualized as black circle in the figure. Fig. 3 (center) shows the potential $L(\theta)$ with a clearly visible plateau in b-direction, where the change in the KLD is small. The dashed line is the unit circle with a radius of η in the geometry defined by the metric tensor $F(\theta^*)$, which is well suited to the potential: the unit circle is stretched in b-direction. Fig. 3 (right) visualizes the distortion of the potential after transformation with $F(\theta^*)$. The induced landscape becomes "Euclidean-like" after transformation and loses the plateau—the transformed potential is isotropic.

5.2. Information geodesy

The following experiments focus on a more global analysis of the natural gradient descent. A gradient descent from a given starting point θ to the attractor θ^* is performed while the relative geodesic length (RGL) of the path is recorded. The RGL gives the length of the geodesic γ from starting point θ to the attractor θ^* with respect to the shortest way in the parameter space:

$$\mathrm{RGL}(\theta) = \int_{\gamma} \mathrm{d}s / \|\theta - \theta^*\|, \tag{13}$$

where $\mathrm{d}s = \sqrt{\mathrm{d}a^2 + \mathrm{d}b^2}$ is the infinitesimal arc length in parameter space.

Fig. 4 (left) shows the potential field $L(\theta)$ of the Gaussian input distribution (1-G) while the right hand side of the figure shows the potential field $L(\theta)$ of the Uniform input distribution (U). It also shows four starting points θ_{1-4} for the learning of each input distribution. The black lines show gradient descents performed by IP, while the yellow lines are the geodesics from the NIP learning. Both approaches have the same fixed-point, but the geodesics of the NIP learning imply a more direct path to the attractor in parameter space. Thus the natural gradient method is better suited to the Potential than the conventional IP gradient. Table 1 displays the results of an experiment where the RGL is measured for $N = 100$ different starting points drawn from a Gaussian distribution centered around the attractor with covariance matrix $\Sigma = I$. It shows the average RGL and its standard deviation.

Since the minimum value for the RGL is one (which corresponds to a straight line from the initial point to the attractor in parameter space), the values for the RGL in Table 1 show that the geodesic lines of NIP are almost straight for all tested input distributions (visualized in Fig. 4). In addition, the low standard deviation demonstrates that the curvature of the geodesic is more independent from the initial point in the potential, compared with using the Euclidean metric.

6. Drift compensation with IP

Drift compensation is a practically highly relevant issue for the application of machine learning algorithms, because in real plants sensors and actuators are typically subject to wear and other nonstationary effects e.g. induced by temperature changes. As discussed in the Introduction, drift can be externally compensated by re-adjusting the data which requires an additional mechanism, or internalized into the learned model for continuous adaptation. IP provides a novel approach to the latter, because it internalizes the drift in an unsupervised and local way, while not relying

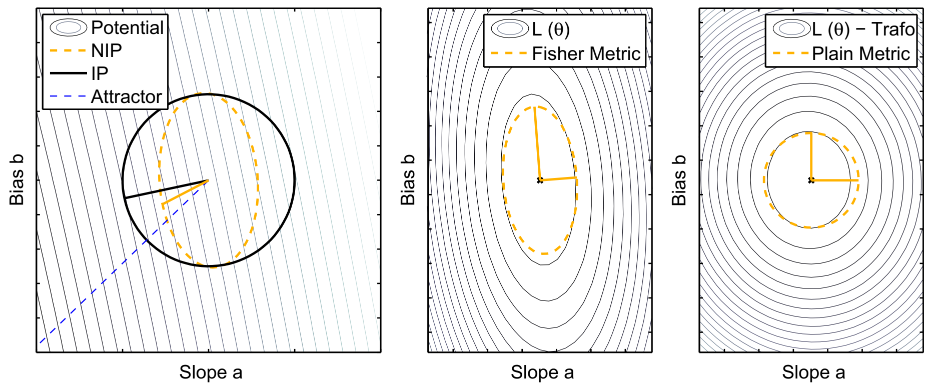

Fig. 3. Fisher metric at point θ for the 1-G distribution (left). The geometry change of the attractor basin using the natural gradient (center, right). IP potential and Fisher metric at the attractor (center). NIP potential and plain Euclidean metric (right).

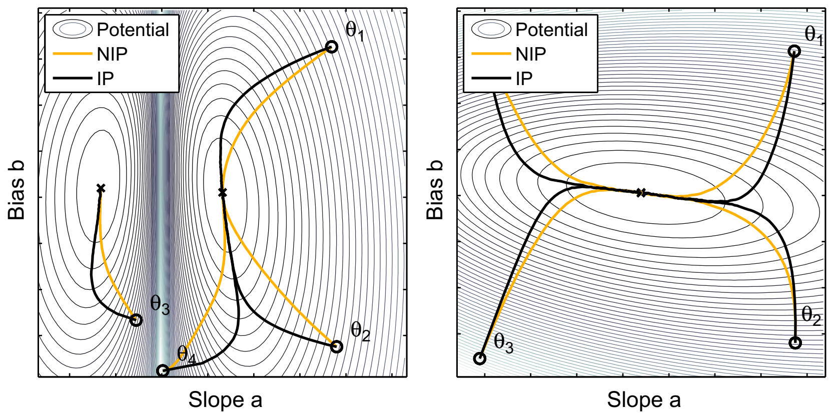

Fig. 4. Geodesics in the IP potential. Geodesic lines for the Gaussian (1-G) distribution (left). Geodesic lines for the Uniform (U) distribution (right). (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this article.)

Table 1
Relative average length of the geodesics $\mathbb{E}[\text{RGL}]$ and their standard derivation $\sqrt{\mathbb{E}[(\text{RGL}-\mathbb{E}[\text{RGL}])^2]}$ for IP and NIP learning.

Task	$\mathbb{E}[\text{RGL}]$ (IP)	$\mathbb{E}[\text{RGL}]$ (NIP)
1-G	1.3493 ± 0.5730	1.0748 ± 0.0526
2-G	1.0473 ± 0.0300	1.0234 ± 0.0342
3-G	1.1209 ± 0.0753	1.0505 ± 0.0506
U	1.0219 ± 0.0206	1.0056 ± 0.0099

on continuous error feedback learning. It optimizes the neural encoding by shifting the mean through the bias and scaling the variance through the slope parameter in the activation function. Drift compensation can thus be considered an inherent side effect of the IP learning approach, which has not been analyzed in the IP literature yet. Obviously, the learning dynamics of IP changes the effectiveness of drift compensation and the following sections will show that the interplay of input drift and the standard IP learning dynamics leads to typical plateaus, which can be avoided when using NIP.

6.1. Analysis of IP learning in the presence of input drifts

The following experiment with a synthetic input signal provides an initial analysis of IP learning with and without the natural gradient in the presence of drifts. It demonstrates that standard IP always implies some drift compensation, but is not perfect on the other hand and can be improved through the proposed modifications. Experiments are performed using an input signal comprising a product of oscillations $x(t) = \sin(0.2t) \cdot \sin(0.053t) \cdot \sin(0.092t)$. In the beginning, IP learning is applied for 5×10^4 steps in order to let the parameters converge with a learning rate of $\eta_{\text{IP}} = 10^{-3}$. After learning, two manipulations of the input signal are carried out to analyze the impact of the natural gradient and weight scaling on the IP learning dynamics in the presence of drifts: (i) gradual scaling of the signal changing the variance and (ii) a gradual shift of the signal changing the mean. In the first experiment, a gradual linear scaling of the input signal up to a factor of 100 or $\frac{1}{100}$ respectively starting from 1 in 10^6 steps is applied. In the second experiment, a gradual linear shifting to 50 or -50 respectively starting from 0 in 5×10^5 steps is applied. Whereas these scaling and shifts are taken to the extremes, they are meant to show the full behavior of the learning algorithm and to allow to clearly display and discuss the effects of the NIP and weight scaling modifications to the original IP rule.

The plots in Fig. 5 show the IP (black) and NIP (yellow) learning dynamics for the described scalings and shifts. The blue-dashed lines show the computed ground truth target slope and bias, which are necessary to perfectly compensate the input signal manipulation. The left column summarizes the results for the scaling while the right column outlines the effects for the shifting of the input signal. The first row in Fig. 5(A, B) displays the logarithm of the slope ratio for IP (black) and NIP (yellow), which is defined as follows:

$$\text{“Slope Ratio” (IP)}: \frac{a(t)}{a_{\text{train}}}, \quad \text{“Slope Ratio” (NIP)}: \frac{a(t) \cdot s(t)}{a_{\text{train}} \cdot s_{\text{train}}}$$

where $a(t)$ is the recorded slope at time t and a_{train} is the learned slope for the non-scaled and non-shifted input. The term $s(t)$ simply denotes the scaling factor of the weights at time step t with respect to the weight with unity norm, in order to make the results comparable. The second row (C, D) shows the shifting of the bias $b(t)$ with respect to the bias b_{train} for the non-manipulated input signal divided by the actual slope $a(t)$. Hence, the plots show the effective mean shift by IP (black) and the adapted NIP (yellow) learning rule.

$$\text{“Bias Shift” (IP)}: \frac{b(t) - b_{\text{train}}}{a(t)}, \quad \text{“Bias Shift” (NIP)}: \frac{b(t) - b_{\text{train}}}{a(t) \cdot s(t)}$$

The KLD is computed to measure the success of the respective adaptation by IP or NIP (see E and F).

Variance shift reveals working point: Fig. 5(A) shows that the IP learning rule is not able to counteract the signal manipulation for an increasingly small scaling within the given time constraints. While compensation is good for small variations, IP learning degenerates when the scaling decreases further. In the case of a linearly increasing scaling, the slopes get very small until numerical instabilities occur due to discretization. The plot also shows that the adapted learning rule from Eq. (12) suffices to achieve the target slope ratio and resolve the numerical instabilities.

Basically, the shift of the working point by the Δw learning rule is responsible for the good matching of the target—see the NIP line (yellow) in Fig. 5(A). Fig. 5(C) shows that IP as well as NIP react suitably and hardly adapt the bias when scaling the input signal. The oscillations of the “Bias Shift” in the plot are mainly induced by the division with the slope—small variations in the bias get magnified for very small slopes. This oscillation is an effect of discretization, although not really a problem because the actual bias changes itself are very small. The KLD for the IP and NIP cases are shown in Fig. 5(E) and confirm the increased performance—the KLD is small for the new learning rule.

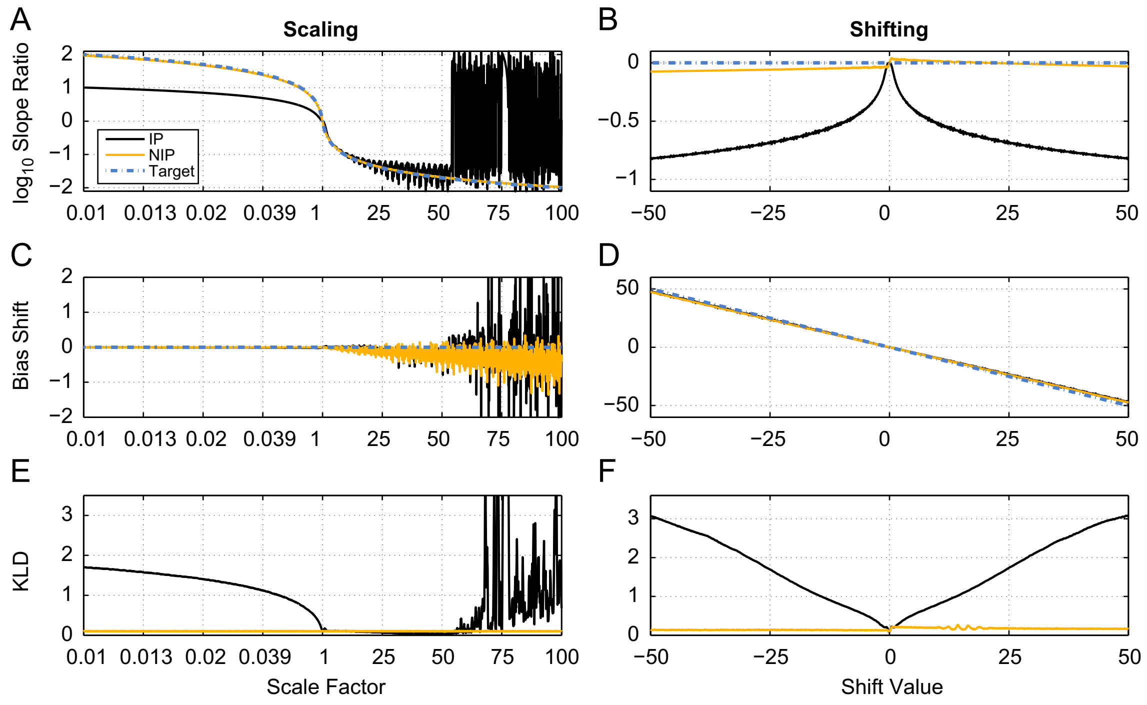

Fig. 5. IP and NIP in a single neuron. The logarithm of the slope ratio for each scaling and shifting value, respectively (A, B). The bias shift divided by the slope for the according scaling and shifting value (C, D). The KL-divergence for scaling and shifting (E, F). (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this article.)

Mean shift leads to plateaus: Fig. 5(B) illustrates the occurrence of a systematic overestimation of the input variance by IP when the signal is shifted. The reason for the overestimation is that decreasing the slope leads to an increased effect of the bias shift. In fact, IP drives the neuron into a parameter regime where the gradient nearly vanishes which leads to very slow convergence, e.g. a typical plateau that prevents efficient learning. This sub-optimal behavior is rectified by using NIP which gives a much better estimation of the gradient direction in parameter space. Therefore the slope ratio with NIP hardly changes during a pure shift of the input signal, although a small underestimation can still be seen. Fig. 5(D) shows that IP as well as NIP achieve a good compensation for the shift by use of the bias. However, the KLD for NIP stays close to the optimal value for shifting of the input signal—in contrast to the results for IP adaptation (see Fig. 5(F)).

6.2. A real world example: learning to point with the humanoid robot iCub

In this section, a real world task involving the humanoid robot iCub demonstrates that drift compensation is possible by sustaining the neural encoding with the proposed learning scheme. Such robots are typically designed to solve service tasks in environments where a high flexibility is required. To cope with various kinds of drifts in the input, e.g. with changing illumination or a displacement of a sensor is a prerequisite for such systems. We use a hand-eye coordination task that is inspired by [27] and is discussed in depth in a further contribution in this volume [28]. The humanoid robot iCub learns to point towards an object based on the raw visual input from his head cameras. We investigate, if NIP learning can cope with the input shift that is associated with a small displacement of the head, a typical problem if there is wear in the mechanical mechanism. As before, we explore quite extreme and even exaggerated displacements to challenge the NIP algorithm.

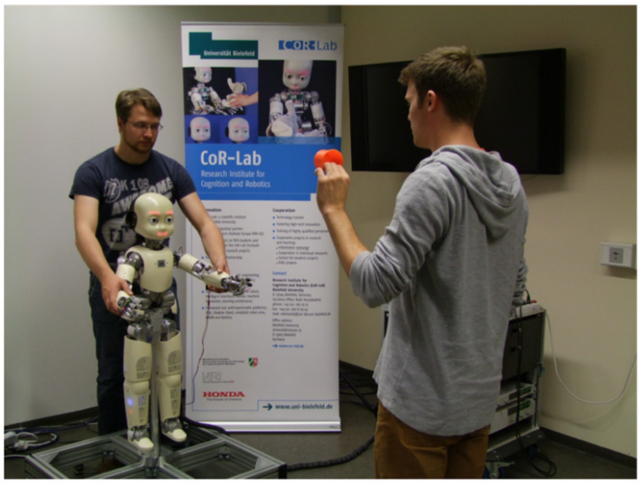

Fig. 6. Kinesthetic teaching of the humanoid robot iCub. One tutor guides iCub's left arm and another holds a red cup to point at in his left hand. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this article.)

The experimental setting is illustrated in Fig. 6. A human tutor physically guides the iCub's arm by means of kinesthetic teaching using a recently established force control on the robot. The tutor can thereby actively move all joints of the arm to place the end-effector at the desired position. To create training data, a second person in an approximate distance of 2 m to the robot moves an object in the visual field of iCub while the human tutor is guiding its arm to point at the object by means of a laser pointer attached to the robot's hand. The 2D-pixel coordinates of the object in both eye-cameras are extracted by a tracking system and recorded together with the joint angles of the arm, for further details see [28].

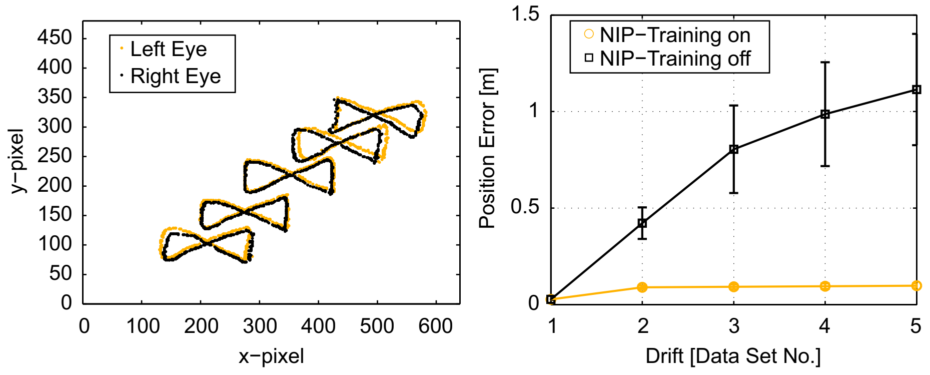

Fig. 7. Drift compensation by sustaining the neural encoding with NIP for the humanoid robot iCub.

The task is to learn the mapping from the $2 \times 2D$ pixel coordinates received by both cameras (with a resolution of 640×480 pixels) onto the end effector configuration of the robot's left arm, i.e. the respective joint angle configuration. $N=5$ data sets are recorded where the tutor in front of iCub painted eight-like figures. The data sets comprise 458–506 samples where $N_{\text{tr}}=300$ where used for training and the remaining samples were used for testing. For each data set, a different head configuration for the pan and the tilt-angle of the neck was chosen from $\theta_{\text{pan}} \in [-20,20]$ and $\theta_{\text{tilt}} \in [-20,20]$ divided equally into 5 steps. This represents a relatively strong displacement of the head, which leads to a shift in the input data as visualized in Fig. 7 (left). The pointing task remains invariant. Note that in this case it is infeasible to cope with the input shift by continuous online supervised error learning, because no error feedback is available when exploiting the learned model to realize the actual pointing on the robot.

To test the proposed form of intrinsic plasticity an extreme learning machine (ELM) [29], which is basically a feed-forward neural network with one hidden layer, is applied in the experiments.

Such networks comprise three different layers: $\mathbf{x} \in \mathbb{R}^I$ denotes the input, $\mathbf{h} \in \mathbb{R}^R$ the hidden, and $\mathbf{y} \in \mathbb{R}^O$ the output neurons. The input is connected to the hidden layer through the input matrix $W^{\mathbf{inp}} \in \mathbb{R}^{R\times I}$ which remain fixed after random initialization. The read-out is given by the matrix $W^{\mathbf{out}} \in \mathbb{R}^{O\times R}$ subject to supervised learning, which will be done by ridge regression (RR). The calculation for the ith output neuron for input $\mathbf{x}^k$ is thus given by

$$y_i(\mathbf{x}^k) = \sum_j W_{ij}^{\mathbf{out}} f\left(a_j \sum_n W_{jn}^{\mathbf{inp}} x_n^k + b_j\right) \tag{14}$$

where a_j, b_j are slope and bias parameterizing the component-wise applied Fermi function $f(x) = 1/(1+e^{-x})$, and $j = 1 \ldots R$ is the index of the hidden layer neuron. IP adapts the slopes a_j and biases b_j of the hidden layer neurons in an unsupervised fashion. The hidden layer of the networks used in the experiments consists of $R=100$ neurons. The network's weights and biases are initialized randomly from a uniform distribution in the interval $[-1,1]$, while the slopes are initially set to one. The regression parameter is $\alpha = 10^{-3}$ in the following experiments. The interplay between IP and ELMs has been analyzed in rigorous detail in [6]. An highly efficient batch version of IP suited for ELMs was proposed in [5].

Fig. 7 (right) compares the ELM network's performance with and without NIP on the shifted data without re-adapting the output weights. The network is pre-trained on the first data set by means of NIP for 1000 epochs and then trained by RR as e.g. in [5]. Then the network is tested on the shifted data sets 2–5, either directly or after additional 1000 NIP epochs on each new data set to compensate the shift. All results are averaged over 10 network initializations. The test error significantly increases for data sets 2–5, if no NIP training is applied. This is expected, since the data varies with changing head configuration and the network cannot arbitrarily generalize. However, the NIP training can compensate the shift and keep the error low, despite relatively large displacements and without re-training the output weights.

7. Conclusion

This paper makes two interconnected contributions to intrinsic plasticity learning. IP has previously been introduced as a biologically plausible and computationally very effective means to optimize the encoding of inputs in neural networks of various types.

First, the well known natural gradient is introduced and analyzed for intrinsic plasticity. The significant impact of the natural gradient for this learning dynamics is shown and an additional modification of IP learning introduced, which targets to further optimize IP by keeping the parameters close to a suitable working point. These modifications improve IP learning over the previous learning scheme and can be applied to any of the known applications of IP learning.

Second, the implicit capability of IP learning to cope with drifts in the input is analyzed for the first time and identified as a very special mechanism and novel approach to drift compensation. It internalizes the effects of drift into the learned model by adaptation of the activation function parameters without the need to change the input data. This adaptation is achieved without the usage of online error feedback. This is a novel and highly useful approach and can be applied when using ELM feed forward type networks. However, it turns out that the drift compensation effect works well only in connection with the proposed modifications and improvements of the IP learning dynamics. Further work shall be directed towards a closer comparison of possible drift compensation schemes and towards the identification of problem domains where this scheme works well. In particular the interplay of IP learning speed and drift speed deserves attention in order to reliably compensate input changes. Nevertheless, the current contribution provides a first account on drift compensation with IP learning and yields encouraging results on synthetic and first real world data obtained through improved natural gradient learning.

References

[1] J. Triesch, Synergies between intrinsic and synaptic plasticity in individual model neurons, in: Proceedings of the NIPS, vol. 17, 2005, pp. 1417–1424.
[2] N.J. Butko, J. Triesch, Learning sensory representations with intrinsic plasticity, Neurocomputing 70 (7) (2007) 1130–1138.
[3] J.J. Steil, Online reservoir adaptation by intrinsic plasticity for backpropagation-decorrelation and echo state learning, Neural Networks 20 (3) (2007) 353–364.

[4] B. Schrauwen, M. Wardermann, D. Verstraeten, J.J. Steil, D. Stroobandt, Improving reservoirs using intrinsic plasticity, Neurocomputing 71 (7) (2008) 1159–1171.
[5] K. Neumann, J.J. Steil, Optimizing extreme learning machines via ridge regression and batch intrinsic plasticity, Neurocomputing, http://dx.doi.org/10.1016/j.neucom.2012.01.041, in press.
[6] Klaus Neumann, Christian Emmerich, Jochen J. Steil, Regularization by intrinsic plasticity and its synergies with recurrence for random projection methods, J. Intelligent Learn. Syst. Appl. 4 (3) (2012) 230–246.
[7] J. Triesch, Synergies between intrinsic and synaptic plasticity mechanisms, Neural Comput. 19 (4) (2007) 885–909.
[8] S.-I. Amari, Natural gradient works efficiently in learning, Neural Comput. 10 (1998) 251–276.
[9] S.-I. Amari, Differential–geometrical Methods in Statistics, Lecture Notes in Statistics, vol. 28, Springer-Verlag, New-York, 1985.
[10] M.K. Murray, J.W. Rice, Differential Geometry and Statistics, Chapman and Hall/CRC, 1993.
[11] S.-I. Amari, S.C. Douglas, Why natural gradient? Acoust. Speech Signal Process. 2 (1998) 1213–1216.
[12] C.C. Douglas, J. Hu, J. Ray, D.T. Thorne, R.S. Tuminaro, Natural-gradient adaptation, in: Unsupervised Adaptive Filtering, Vol. I: Blind Source Separation, vol. 1, 2000, pp. 13–61.
[13] M. Rattray, D. Saad, S.-I. Amari, Natural gradient descent for on-line learning, Phys. Rev. Lett. 81 (1998) 5461–5464.
[14] M. Rattray, D. Saad, Analysis of natural gradient descent for multilayer neural networks, Phys. Rev. E 59 (1999) 4523–4532.
[15] H. Park, S.-I. Amari, K. Fukumizu, Adaptive natural gradient learning algorithms for various stochastic models, Neural Networks 13 (7) (2000) 755–764.
[16] T. Heskes, On natural learning and pruning in multilayered perceptrons, Neural Comput. 12 (4) (2000) 881–901.
[17] A. Gonzalez, J.R. Dorronsoro, Natural learning in nlda networks, Neural Networks 20 (5) (2007) 610–620.
[18] K. Neumann, J.J. Steil, Intrinsic plasticity via natural gradient descent, in: Proceedings of the ESANN, 2012, pp. 555–560.
[19] M. Holmberg, T. Artursson, Drift compensation, standards and calibration methods, in: Handbook of Machine Olfaction: Electronic Nose Technology, 2002, pp. 325–346.
[20] J. Gama, P. Medas, G. Castillo, P. Rodrigues, Learning with drift detection, in: Proceedings of the SBIA, 2004, pp. 286–295.
[21] R. Klinkenberg, T. Joachims, Detecting concept drift with support vector machines, in: Proceedings of the Seventeenth International Conference on Machine Learning (ICML), Morgan Kaufmann, 2000, 487–494.
[22] A. Dries, U. Rückert, Adaptive concept drift detection, Stat. Anal. Data Mining 2 (5–6) (2009) 311–327.
[23] S. Kullback, A. Leibler, On information and sufficiency, Ann. Math. Stat. 22 (1) (1951) 79–86.
[24] J. Triesch, A gradient rule for the plasticity of a neurons intrinsic excitability, in: Proceedings of the ICANN, 2005, pp. 65–70.
[25] L. Kostal, P. Lansky, Classification of stationary neuronal activity according to its information rate, in: Unsupervised Adaptive Filtering, Vol. I: Blind Source Separation, vol. 17, 2006, pp. 193–210.
[26] C.R. Rao, Information and the accuracy attainable in the estimation of statistical parameters, Bull. Calcutta Math. Soc. 37 (1945) 81–91.
[27] A. Freire, A. Lemme, J.J. Steil, G. Barreto, Learning visuo-motor coordination for pointing without depth calculation, in: Proceedings of the ESANN, 2012, pp. 91–96.
[28] A. Lemme, A. Freire, G. Barreto, J.J. Steil, Kinesthetic teaching of visuomotor coordination for pointing by the humanoid robot icub, Neurocomputing, http://dx.doi.org/10.1016/j.neucom.2012.12.040, in press (Special issue ESANN 2012).
[29] G.-B. Huang, Q.-Y. Zhu, C.-K. Siew, Extreme learning machine: theory and applications, Neurocomputing 70 (1–3) (2006) 489–501.

Klaus Neumann received his B.Sc. and M.Sc. in applied computer science from Bielefeld University in 2007 and 2010, respectively. He is currently a Ph.D. student in the 'Cognitive Robotics and Learning'—Group at the Research Institute for Cognition and Robotics (CoR-Lab) located at Bielefeld University. His research interests include neural networks, machine learning and robotics.

Claudius Strub received a B.Sc. in cognitive informatics in 2010 and is currently completing his M.Sc. in intelligent systems at Bielefeld University, Germany. He is generally interested in neural computation and machine learning. In particular, unsupervised learning mechanisms for recurrent neural networks and self-organization of dynamical systems are of special interest to him.

Jochen Steil received the diploma in mathematics and Ph.D. in computer science from the University of Bielefeld, Germany, in 1993 and 1999, respectively, where he was working in the Neuroinformatics Group. In 1995/1996 he spent one year at the St. Petersburg Electrotechnical University, Russia, supported by a German Academic Exchange Foundation (DAAD) grant. After his Ph.D. degree with a dissertation on "Input–output stability of recurrent neural networks", he received the venia legendi in Neuroinformatics in 2006. In the same year he visited the Honda Research Institute Europe (Offenbach) as a principal scientist. After his return to Bielefeld University he became managing director of the newly founded Institute for Cognition and Robotics (CoR-Lab) in 2007, scientific board member of the Cognitive Interaction Technology Excellence Cluster (CITEC) and apl. Professor for Neuroinformatics at the Faculty of Technology in 2008. Jochen Steil's research interests are learning in recurrent networks, with a particular emphasis on applications in robotics.