OPEN ACCESS Freely available online

PLOS ONE

Synergies between Intrinsic and Synaptic Plasticity Based on Information Theoretic Learning

Yuke Li, Chunguang Li*

Department of Information Science and Electronic Engineering, Zhejiang University, Hangzhou, People's Republic of China

Abstract

In experimental and theoretical neuroscience, synaptic plasticity has dominated the area of neural plasticity for a very long time. Recently, neuronal intrinsic plasticity (IP) has become a hot topic in this area. IP is sometimes thought to be an information-maximization mechanism. However, it is still unclear how IP affects the performance of artificial neural networks in supervised learning applications. From an information-theoretical perspective, the error-entropy minimization (MEE) algorithm has newly been proposed as an efficient training method. In this study, we propose a synergistic learning algorithm combining the MEE algorithm as the synaptic plasticity rule and an information-maximization algorithm as the intrinsic plasticity rule. We consider both feedforward and recurrent neural networks and study the interactions between intrinsic and synaptic plasticity. Simulations indicate that the intrinsic plasticity rule can improve the performance of artificial neural networks trained by the MEE algorithm.

Citation: Li Y, Li C (2013) Synergies between Intrinsic and Synaptic Plasticity Based on Information Theoretic Learning. PLoS ONE 8(5): e62894. doi:10.1371/journal.pone.0062894

Editor: Eleni Vasilaki, University of Sheffield, United Kingdom

Received October 3, 2012; **Accepted** March 26, 2013; **Published** May 9, 2013

Funding: This work was supported by the National Natural Science Foundation of China (Grant Nos. 61171153 and 61101045), the Foundation for the Author of National Excellent Doctoral Dissertation of PR China, the Scientific Research Foundation for the Returned Overseas Chinese Scholars, and Zhejiang Provincial Natural Science Foundation of China (Grant No. LR12F01001). The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Competing Interests: The authors have declared that no competing interests exist.

* E-mail: cgli@zju.edu.cn

Introduction

Artificial neural networks with nonlinear processing elements are designed to deal with the troublesome problem of nonlinear and nonstationary signal processing. In a supervised learning problem, we are provided with a training data set containing the input, $\mathbf{x}$, and the desired output (target), $\mathbf{d}$, and we aim at finding the input-output mapping that models the complicated relationship between $\mathbf{x}$ and $\mathbf{d}$. To solve such a problem, we can employ an artificial neural network trained by an appropriate learning algorithm to infer the mapping implied by the training data. Most current learning algorithms for artificial neural networks in applications rely on updating the connection weights w among neurons. This is often done with the aim of minimizing the mean square error (MSE) between the network output $\mathbf{y}$ and the desired output $\mathbf{d}$ over all input-target pairs, where the error is defined as $\mathbf{e}=\|\mathbf{d}-\mathbf{y}\|$. However, the MSE criterion takes into account only the first two moments of the error distribution, making it ill-suited to non-linear applications in which the errors are not normally distributed. The error entropy criterion (EEC) has been proposed on information-theoretic grounds by Principe et al. as an alternative cost function that takes into account the full distribution of errors [1]. This is the form of synaptic plasticity we consider in this article.

So far, experimental and theoretical studies on neural plasticity have mostly focused on synaptic plasticity, which is in accordance with Hebb's idea that memories are stored in the synaptic weights and learning is the process that changes those synaptic weights. Interestingly, recent experimental results have revealed that neurons are also capable of changing their intrinsic excitability to match the dramatic change of the level of received synaptic input [2–9]. This novel neural mechanism is referred to as *intrinsic plasticity* (IP). IP is hypothesized to maximize the information capacity while maintaining an individual neuron's homeostasis of its mean firing rate level [10–12]. To better understand the role IP might play in learning and memory, several IP rules [10,11,13,14] were proposed that bring the firing rate distribution into a desired one with a relatively low activity level as observed in visual cortical neurons [15]. Actually, upon neglecting the energy constraint, these IP rules [10,11,13,14] are closely related to the single-neuron case of Bell and Sejnowski's information-maximization algorithm [16]. When the input-output mutual information is maximized by this learning algorithm, a neuron uses all of its possible response levels equally and uses the steep parts of the activation function to respond to the high density parts of the input probability density function (PDF); therefore, this information-maximization algorithm enhances the discriminative ability of the neuron.

The two plasticity mechanisms, intrinsic plasticity and synaptic plasticity, have been studied mostly separately. We are wondering how these two plasticity mechanisms would cooperate in artificial neural networks to learn complex mappings. To this end, we combine Bell and Sejnowski's information-maximization algorithm [16] for intrinsic plasticity and the error-entropy minimization (MEE) algorithm [17] for synaptic plasticity, which we refer to as *synergistic information-theoretic learning*. We use the resulting synergistic procedure for training feedforward neural networks (FNN) and recurrent neural networks (RNN) and test them on two benchmark applications. For simplicity and clarity, we focus on the prediction problem in the presentation, but the learning algorithm can also be used for solving problems of regression, classification

and so on. Simulations indicate that Bell and Sejnowski's algorithm is appropriate for the proposed synergistic learning scheme and that the MEE algorithm combined with IP outperforms the MEE algorithm considered in isolation.

Materials and Methods

Information-maximization Algorithm as an Intrinsic Plasticity Rule

Studying the effects of intrinsic plasticity on various neural functions and dynamics relies on modelling intrinsic plasticity. In [10,11,13,14], several intrinsic plasticity rules were proposed, which bring the firing rate distribution into a desired one with a relatively low activity level as observed in visual cortical neurons [15]. In all these IP rules, the energy consumption of a biological neuron is considered as an important constraint. Keeping a low average output is critical for biological organisms due to energy expenditure, but it seems unnecessary for artificial neural networks such as the FNN and the RNN. In terms of choosing the IP rule in this situation, we prefer to emphasize the character of maximizing the information capacity. Neglecting the energy constraint in this study, we apply the information-maximization learning algorithm proposed by Bell and Sejnowski [16] as the intrinsic plasticity rule for individual neurons of artificial neural networks. This learning algorithm adjusts the slope and the bias of the activation function to maximize the mutual information between the input and the output of each neuron. As a result of this learning procedure, the activation function is adapted to match the input distribution, i.e., sensitive parts of the activation function respond to high density parts of the input probability density function. The sensitivity is characterized by the slope of the response curve and steep parts are more sensitive than flat parts. For the tanh activation function of the kth neuron $\varphi_k(\cdot)$,

$$\begin{aligned} y_k=\varphi_k(v_k;a_k,b_k) &= \tanh(a_k v_k+b_k) \\ &= \frac{\exp(a_k v_k+b_k)-\exp(-a_k v_k-b_k)}{\exp(a_k v_k+b_k)+\exp(-a_k v_k-b_k)}, \end{aligned} \tag{1}$$

where v_k is the input of the kth neuron, y_k is the output of the kth neuron, a_k represents the sensitivity of the activation function and b_k is the bias. The corresponding information-maximization learning algorithm can be obtained as follows

$$\begin{aligned} a_k &= a_k+\eta_{\mathrm{IP}}\left(\frac{1}{a_k}-2\mathrm{E}[v_k y_k]\right), \\ b_k &= b_k+\eta_{\mathrm{IP}}(-2\mathrm{E}[y_k]), \end{aligned} \tag{2}$$

where η_{IP} is the learning rate of intrinsic plasticity. For a training set including n_0 samples, the input-output pairs of the kth neuron, $[v_k(1),\ldots,v_k(n_0)]^{\mathrm{T}}$ and $[y_k(1),\ldots,y_k(n_0)]^{\mathrm{T}}$, are used to estimate the expected values $\mathrm{E}[v_k y_k]$ and $\mathrm{E}[y_k]$. This batch version of the information-maximization rule can be derived directly from the objective of entropy maximization (''online'' equivalent). Note that this learning rule neglects the recurrent interactions that may exist in the network, such that the entropy of the output of the kth unit is assumed to depend only on a_k and b_k, but not on any other $a_{k'}$ and $b_{k'}$ for $k\neq k'$.

We apply this information-maximization rule as the intrinsic plasticity rule for artificial neural networks in this paper. Note that the original information-maximization rule in [16] is an online weight update rule for ICA. For simplicity, in all of the following presentations and simulations, the tanh function is chosen as the activation function and the corresponding intrinsic plasticity rule is applied unless stated otherwise.

MEE Algorithm as a Synaptic Plasticity Rule

As mentioned above, the MSE criterion considers only the first two moments of the error distribution, thus it is ill-suited to non-linear applications in which the errors are not normally distributed. Recently, the error entropy (EEC) criterion based on ideas from information theory has been proposed as an alternative cost function for learning [1]. EEC aims at removing as much uncertainty as possible from the error signal, and this can be accomplished by calculating the entropy of the error and minimizing it with respect to the connection weights. In the ideal case, all the uncertainty in the error is removed and the error probability density function is a delta function. This method is called Minimization of the Error Entropy (MEE) [17].

In applications, Renyi's quadratic entropy, H_2, is often applied instead of Shannon's entropy [1]. We can easily use Renyi's quadratic entropy to derive a learning rule. For a probability density function $p(X)$ where X is a continuous random variable, the formula for Renyi's quadratic entropy is given by

$$H_2(p)=-\log\int p^2(X)\mathrm{d}X. \tag{3}$$

One can then define a quadratic potential $V_2(p)$, such that

$$H_2(p)=-\log V_2(p), \tag{4}$$

where

$$V_2(p)=\int p^2(X)\mathrm{d}X. \tag{5}$$

Thus, the minimization of Renyi's quadratic entropy in Eq. (3) is equivalent to the maximization of the information potential in Eq. (5). Importantly, Eq. (5) may be interpreted as an expectation of the function $p(X)$ under itself, that is, $V_2=E[p(X)]_{X\tilde{p}(X)}$. This means that, provided one can estimate $p(x)$ for any sample x, one may subsequently estimate V_2 through simple Monte-Carlo averaging of $p(x)$ over many independent samples from $p(x)$ (i.e. the data set). Here, we use Gaussian kernel density estimation with bandwidth σ, $\hat{p}(x)=\frac{1}{n_0}\sum_{i=1}^{n_0}G_\sigma(x-x(i))$, where $x(i),i=1,2,\ldots,n_0$ are n_0 samples from the true underlying distribution $p(X)$, G denotes the Gaussian kernel function, and σ represents the kernel size for probability density function estimation [18]. Assuming Gaussian kernels and substituting this in the quadratic entropy expression Eq. (5), we get the following estimator for V_2 [1],

$$\begin{aligned} \hat{V}_{2,\sigma}(p(x)) &= \int\hat{p}^2(x)\mathrm{d}x \\ &= \int\left(\frac{1}{n_0}\sum_{i=1}^{n_0}G_\sigma(x-x(i))\right)^2\mathrm{d}x \\ &= \frac{1}{{n_0}^2}\sum_{i=1}^{n_0}\sum_{j=1}^{n_0}G_{\sigma\sqrt{2}}(x(j)-x(i)), \end{aligned} \tag{6}$$

where $\{x(1),x(2),\ldots,x(n_0)\}$ is a set of data samples. With the steepest ascent approach, the training algorithm for weight

updating to maximize the quadratic information potential of the error e, $V_2(p(e))$, becomes

$$\Delta w = \eta \frac{\partial \hat{V}_{2,\sigma}(p(e))}{\partial w}, \tag{7}$$

where Δw denotes the change of the weight w and η is the learning rate. The gradient of the quadratic information potential with respect to the connection weight is

$$\begin{aligned}\frac{\partial \hat{V}_{2,\sigma}(p(e))}{\partial w} = \frac{1}{2n_0{}^2\sigma^2}\sum_{i=1}^{n_0}\sum_{j=1}^{n_0}(e(i)-e(j)) \\ \cdot G_{\sigma\sqrt{2}}(e(j)-e(i))\cdot \quad (\tfrac{\partial e(j)}{\partial w}-\tfrac{\partial e(i)}{\partial w}).\end{aligned} \tag{8}$$

In training with entropy-based criteria, one important point to note is that since entropy does not change with the mean of the distribution the algorithm will converge to a set of optimal weights that may not yield zero-mean error. This problem can be easily solved by adding a bias to the final output to yield zero mean error over training data set after the training procedure ends [19].

We now introduce our synergistic information-theoretic learning algorithm, which is the simple combination of the IP rule of Eq. (2) and the synaptic plasticity rule of Eq. (7).

Synergistic Information-theoretic Learning

From the perspective of information maximization, there are potential advantages of intrinsic plasticity in training artificial neural networks. For traditional weight update learning algorithms (synaptic plasticity rules), the activation functions of neurons are fixed during the training procedure. However, an invariant nonlinear activation function might be unsuitable for the input distribution. In an extreme case, the output of the neuron may be constantly found at saturation, and therefore carry very little information about the input. For real-world applications, the desired output distribution of a single neuron is far from these distributions with very low information. As we stated in the previous section, the information-maximization algorithm (the intrinsic plasticity rule) can adjust the shape of the activation function to match the input distribution and consequently increase the mutual information between the input and the output. Therefore, we hypothesize that the intrinsic plasticity rule might be beneficial to learning in artificial neural networks.

The MEE algorithm requires several samples to accurately estimate the information potential. We therefore perform batch (epochwise) learning iterations, whereby the weights w are updated according to Eq. (8), on the basis of the output-target pairs collected from all samples in the training set [20]. This allows for a correct estimation of the quadratic potential, and therefore more stable learning. Note that the exact form of the gradient in Eq. (8) depends on the network architecture, and is derived below for both feedforward and recurrent networks. Following this weight update, which we call the "synaptic stage" of a learning iteration, we update the parameters a and b of the activation function φ of each neuron to implement intrinsic plasticity according to Eq. (2). This we call the "intrinsic stage". Note that the expected values $\mathrm{E}[\cdot]$ in Eq. (2) are again estimated from the input-output pairs collected from all samples in the training set. The synaptic and intrinsic stages together define one learning iteration (epoch), which we repeat until the stopping criterion is satisfied. In the following simulations, we stop the learning process after a certain number of learning iterations.

One may think that the effects of synaptic plasticity and intrinsic plasticity merely superpose in the learning process. In fact, we argue that they interact, which is why we call this combination "synergistic learning". Indeed, the weight updating procedure affects the input of the neuron, and further influences the IP learning; the IP learning procedure affects the output of the neuron, and further influences the weight updating.

Stability of Synergistic Learning

It has been noted that, in reservoir networks, due to the incremental nature of intrinsic plasticity (the value of parameter a increases during learning), too large a value for a can cause unstable learning behavior (oscillations in the learning curve) and thus the performance might deteriorate as learning goes on [21]. This phenomenon is due to the cancellation effect, whereby high gains can be compensated by small synaptic weights. Nevertheless, with a proper IP learning rate, unstable behavior takes place only when the IP rule is applied for a very long training time; thus, IP learning can be kept stable before the stopping criterion of the cost function is satisfied [21].

Construction of the FNN

In order to study the performance of the proposed synergistic learning algorithm, we first choose a general class of feedforward neural networks (FNN) as an example. As illustrated in Fig. 1, this neural network is composed of an input layer, a single hidden layer and an output layer. The activation function of each neuron is $\varphi(\cdot)=\tanh(\cdot)$. The network input consisting of P external elements can be described by the $P\times 1$ vector, $\mathbf{u}=[u_1,u_2,\ldots,u_P]^{\mathrm{T}}$. In the hidden layer, there are M neurons (processing elements). Each neuron in the hidden layer receives the weighted sum of the network input $\mathbf{u}$. The output of these neurons, $\mathbf{y}^{\mathrm{h}}$, is described as $\mathbf{y}^{\mathrm{h}}=[y_1^{\mathrm{h}},y_2^{\mathrm{h}},\ldots,y_M^{\mathrm{h}}]^{\mathrm{T}}$, and is calculated by

$$\begin{aligned}\mathbf{v}^{\mathrm{h}} &= (\mathbf{W}^{\mathrm{h}})^{\mathrm{T}}\mathbf{u}, \\ \mathbf{y}^{\mathrm{h}} &= \mathbf{j}(\mathbf{v}^{\mathrm{h}}),\end{aligned} \tag{9}$$

where $\mathbf{v}^{\mathrm{h}}=[v_1^{\mathrm{h}},v_2^{\mathrm{h}},\ldots,v_M^{\mathrm{h}}]^{\mathrm{T}}$, and $\mathbf{W}^{\mathrm{h}}$ represents the $P\times M$ synaptic weight matrix connecting the input layer to the hidden one. An element $w_{k,l}^{\mathrm{h}}$ of this matrix represents the weight connection from the lth input node to the kth hidden neuron. For the output layer, we consider only one neuron, which receives the weighted sum of the output of hidden neurons, $\mathbf{y}^{\mathrm{h}}$, and produces the network output, y^{o}. The calculation is described as

$$\begin{aligned}v^{\mathrm{o}} &= (\mathbf{W}^{\mathrm{o}})^{\mathrm{T}}\mathbf{y}^{\mathrm{h}}, \\ y^{\mathrm{o}} &= \varphi(v^{\mathrm{o}}),\end{aligned} \tag{10}$$

where $\mathbf{W}^{\mathrm{o}}$ represents the $M\times 1$ synaptic weight matrix linking the hidden layer to the output unit. An element w_i^{o} of this matrix represents the connection weight from the ith hidden neuron to the final output.

Synergistic Algorithm for the FNN

The difference between the desired output, d, and the network output, y^o, is defined as the error of the FNN, $e=d-y^o$. For the output layer of a single-output FNN, the derivative of the error e with respect to the weight w_i^{o} in the $M\times 1$ matrix $\mathbf{W}^{\mathrm{o}}$ can be calculated as

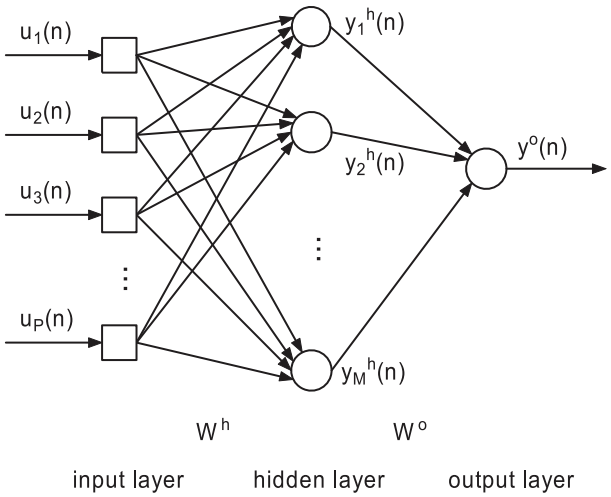

Figure 1. Structure of the feedforward neural networks.
doi:10.1371/journal.pone.0062894.g001

$$\frac{\partial e}{\partial w_i^{\mathrm{o}}} = -\varphi'(v^{\mathrm{o}})y_i^{\mathrm{h}}, \tag{11}$$

where

$$\varphi'(v) = a(1-\tanh^2(av+b)) = a(1-y^2).$$

In a multi-layer FNN, a backpropagation algorithm is usually used to train the weight matrix from the input layer to the hidden layer, $\mathbf{W}^{\mathrm{h}}$. If the EEC cost function is used, the training algorithm is the MEE algorithm [22]. The derivative of the error e with respect to the weight $w_{k,l}^{\mathrm{h}}$ in the matrix $\mathbf{W}^{\mathrm{h}}$ can be calculated as

$$\frac{\partial e}{\partial w_{k,l}^{\mathrm{h}}} = -\varphi'(v^{\mathrm{o}})w_k^{\mathrm{o}}\varphi'(v_k^{\mathrm{h}})u_l. \tag{12}$$

By Eq. (11) and Eq. (12), the weight update rule in Eq. (7) and Eq. (8) can be calculated.

On the basis of the algorithm description in [23], the proposed synergistic learning algorithm for the FNN is summarized as follows:

Step 1. Initialization. Choose a random set of small values for the $P\times M$ hidden layer weight matrix $\mathbf{W}^{\mathrm{h}}$ and the $M\times 1$ output layer weight matrix $\mathbf{W}^{\mathrm{o}}$. Set $a=1$ and $b=0$ for each neuron. Let $\mathbf{u}(n)=[u_1(n),u_2(n),\ldots,u_P(n)]^{\mathrm{T}}$ be the input signal and let $d(n)$ be the corresponding desired network output. The number of samples in the training set is n_0, thus $1\le n\le n_0$.

Step 2. Repetition. The epochwise training procedure begins with $n=1$. Repeat the following calculations with the input vector $\mathbf{u}(n)$ and the target output $d(n)$ for $1\le n\le n_0$

$$\begin{aligned}
\mathbf{v}^{\mathrm{h}}(n) &= (\mathbf{W}^{\mathrm{h}})^{\mathrm{T}}\mathbf{u}(n),\\
\mathbf{y}^{\mathrm{h}}(n) &= \varphi^{\mathrm{h}}(\mathbf{v}^{\mathrm{h}}(n)),\\
v^{\mathrm{o}}(n) &= (\mathbf{W}^{\mathrm{o}})^{\mathrm{T}}\mathbf{y}^{\mathrm{h}}(n),\\
y^{\mathrm{o}}(n) &= \varphi^{\mathrm{o}}(v^{\mathrm{o}}(n)),\\
e(n) &= d(n)-y^{\mathrm{o}}(n),
\end{aligned} \tag{13}$$

where

$$\begin{aligned}
\mathbf{v}^{\mathrm{h}}(n) &= [v_1^{\mathrm{h}}(n),v_2^{\mathrm{h}}(n),\ldots,v_M^{\mathrm{h}}(n)]^{\mathrm{T}},\\
\varphi^{\mathrm{h}}(\mathbf{v}^{\mathrm{h}}(n)) &= [\varphi_1^{\mathrm{h}}(v_1^{\mathrm{h}}(n);a_1,b_1),\varphi_2^{\mathrm{h}}(v_2^{\mathrm{h}}(n);a_2,b_2),\ldots,\\
&\quad \varphi_M^{\mathrm{h}}(v_M^{\mathrm{h}}(n);a_M,b_M)]^{\mathrm{T}},\\
\mathbf{y}^{\mathrm{h}}(n) &= [y_1^{\mathrm{h}}(n),y_2^{\mathrm{h}}(n),\ldots,y_M^{\mathrm{h}}(n)]^{\mathrm{T}}.
\end{aligned}$$

Step 3. Weight Matrix Update. Update the weight matrices $\mathbf{W}^{\mathrm{o}}$ and $\mathbf{W}^{\mathrm{h}}$ by the weight update algorithm. Calculation results of $v_k(n)$ and $y_k(n)$ in Eq. (13) are used to compute the derivatives of the error with respect to the weight in Eq. (11) and Eq. (12); with the results of the derivatives and the errors $e(n)$ in Eq. (13), Eq. (8) can be computed and finally the weight matrices can be updated by Eq. (7).

Step 4. Activation Function Update. Update the parameters a_k and b_k of the activation function φ_k of the neuron k using the intrinsic plasticity rule described in Eq. (2) with all values of $v_k(n)$ and $y_k(n)$. By the batch version of the IP rule, the parameters a_k and b_k are only updated once during an epoch.

Step 5. Return or Stop. If the stopping criterion is satisfied, the training procedure is stopped; otherwise, set $n=1$ and return to Step 2.

Construction of the RNN

In this section, we continue to study the proposed synergistic learning algorithm in a general class of recurrent neural networks [23,24]. As illustrated in Fig. 2, the neural network contains N neurons. The input vector $\mathbf{u}$ is comprised of the external signal vector of P elements $[u_1,u_2,\ldots,u_P]^{\mathrm{T}}$, and the feedback vector $\mathbf{r}=[r_1,r_2,\ldots,r_N]^{\mathrm{T}}$. The feedback signal r_k after a delay of one time unit is the output of the kth neuron y_k, $r_k(n)=y_k(n-1)$, thus the feedback vector at the time point n can be rewritten as $\mathbf{r}(n)=[y_1(n-1),y_2(n-1),\ldots,y_N(n-1)]^{\mathrm{T}}$. Then the input vector at the time point n is given by

$$\mathbf{u}(n)=[u_1(n),u_2(n),\ldots,u_P(n),y_1(n-1),y_2(n-1),\ldots,y_N(n-1)]^{\mathrm{T}}.$$

The $(P+1+N)\times N$ synaptic weight matrix of the recurrent network is represented by $\mathbf{W}$. An element $w_{k,l}$ of this matrix represents the connection weight from the lth input node to the kth neuron. With the input vector $\mathbf{u}$ and the activation function φ, the $N\times 1$ output vector $\mathbf{y}=[y_1,y_2,\ldots,y_N]$ is calculated as

$$\begin{aligned}
\mathbf{v} &= \mathbf{W}^{\mathrm{T}}\mathbf{u},\\
\mathbf{y} &= \varphi(\mathbf{v}),
\end{aligned} \tag{14}$$

where $\mathbf{v}=[v_1,v_2,\ldots,v_N]$ and y_1 is the single output of the network.

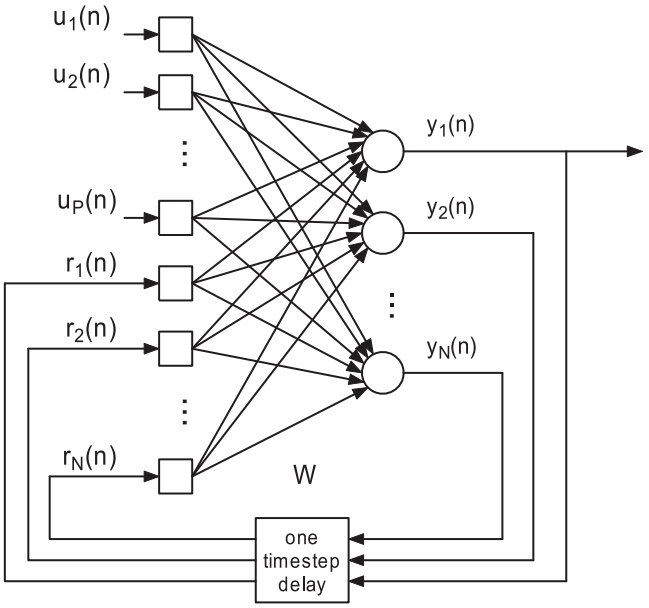

Figure 2. Structure of the recurrent neural networks.
doi:10.1371/journal.pone.0062894.g002

Synergistic Algorithm for the RNN

Following the approach of [25], a recursive learning algorithm can be derived for the recurrent neural network. Referring to [26], the gradients of the outputs of the neurons $\partial y_j(n)/\partial w_{k,l}(n)$ can be computed recursively as follows

$$\frac{\partial y_j(n)}{\partial w_{k,l}(n)} \approx \varphi'(v_k)[\sum_{\alpha=1}^{N} \frac{\partial y_\alpha(n-1)}{\partial w_{k,l}(n-1)} w_{j,\alpha+P+1}(n) + \delta_{kj} u_l(n)], \tag{15}$$

where $\delta_{kj}=1$ for $k=j$, otherwise, $\delta_{kj}=0$. The initial state is $\partial y_j(0)/\partial w_{k,l}(0)=0$. With the relationship $e=d-y_1$, where d is the desired output and y_1 is the true output of the RNN, the partial derivative of the error with respect to the weight becomes

$$\frac{\partial e}{\partial w_{k,l}} = -\frac{\partial y_1}{\partial w_{k,l}}. \tag{16}$$

By using Eq. (15) and Eq. (16), the weight update rule in Eq. (7) and Eq. (8) can be calculated.

The proposed synergistic learning algorithm for the RNN is summarized as follows:

Step 1. Initialization. Choose a random set of small values for the $(P+1+N)\times N$ weight matrix $\mathbf{W}$ and the $N\times 1$ feedback vector $\mathbf{r}(n)=[r_1(n),r_2(n),\ldots,r_N(n)]^{\mathrm{T}}$. Set $a=1$ and $b=0$ for all neurons. Obtain the $P\times 1$ external input vector and the desired signal $d(n)$ with $1\le n\le n_0$.

Step 2. Repetition. The epochwise training procedure begins with $n=1$. Input the external input vector, the feedback vector and the desired signal, and perform the following calculations

$$\begin{aligned} \mathbf{v}(n) &= \mathbf{W}^{\mathrm{T}}\mathbf{u}(n), \\ \mathbf{y}(n) &= \varphi(\mathbf{v}(n)), \\ e(n) &= d(n)-y_1(n), \end{aligned} \tag{17}$$

where

$$\begin{aligned} \mathbf{v}(n) &= [v_1(n),v_2(n),\ldots,v_N(n)]^{\mathrm{T}}, \\ \varphi(\mathbf{v}(n)) &= [\varphi_1(v_1(n);a_1,b_1),\varphi_2(v_2(n);a_2,b_2),\ldots, \\ &\quad \varphi_N(v_N(n);a_N,b_N)]^{\mathrm{T}}, \\ \mathbf{y}(n) &= [y_1(n),y_2(n),\ldots,y_N(n)]^{\mathrm{T}}. \end{aligned}$$

Let $n=n+1$, and set

$$\mathbf{r}(n)=\mathbf{y}(n-1).$$

.

Repeat the calculation until $n=n_0$.

Step 3. Weight Matrix Update. Update the weight matrix $\mathbf{W}$ by the MEE learning algorithm. Calculation results of Eq. (17) of the current epoch are used to compute the derivatives of the error with respect to the weight in Eq. (15) and Eq. (16); with the results of the derivatives and the errors $e(n)$ in Eq. (17), Eq. (8) can be computed and finally the weight matrix can be updated by Eq. (7).

Step 4. Activation Function Update. Update the parameters a_k and b_k of the activation function φ_k of the neuron k using the intrinsic plasticity rule described in Eq. (2). With the batch version of the IP rule, the parameters a_k and b_k are only updated once during an epoch.

Step 5. Return or Stop. As one epoch ends, if the stopping criterion is satisfied, the training procedure is stopped; otherwise, set $n=1$ and $\mathbf{r}(1)=\mathbf{y}(n_0)$, and return to Step 2.

Results

The FNN and RNN are widely applicable to a set of problems such as regression and classification. As a typical example, we test the proposed synergistic learning algorithm on the single-step prediction of time series. For comparison, we also perform simulations for the MEE algorithm alone. The time series is denoted as $s(i),i=1,\cdots,n_0$. In the following simulations, two data sets of different time series are used. The first data set is the well-known Mackey-Glass chaotic time series, which often serves as a benchmark in testing prediction algorithms in the literature. The Mackey-Glass system (for $\tau=17$) is described by the following differential equation

$$\frac{ds(t)}{dt} = -0.1s(t) + \frac{0.2s(t-17)}{1+s(t-17)^{10}}, \tag{18}$$

which is a chaotic system modelling irregular behaviors in biological systems [27]. In our simulations, we use the Runge-Kutta method with time-step 0.1 to integrate Eq. (18), and then we draw samples at $T=1$s interval to obtain the discrete time series. We use 300 samples for training and 10000 new samples generated from a different initial condition for testing. We use "MG" to denote this data set. The other data set is a speech signal obtained from an audio report in the program of "Scientific

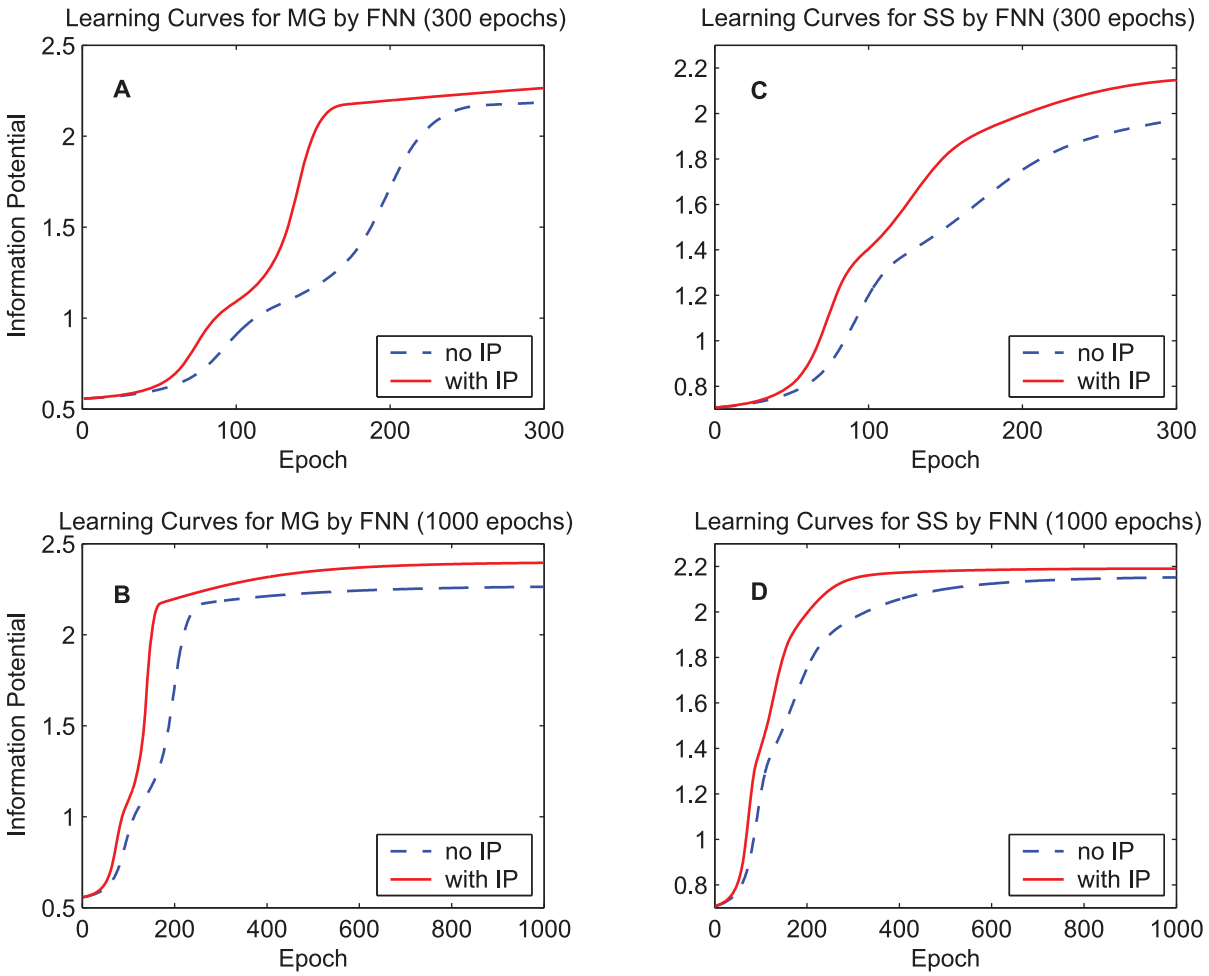

Figure 3. Learning curves of the quadratic information potential by the FNN. The dashed lines denote the learning curves of the MEE algorithm, and the solid lines denote the learning curves of the synergistic algorithm. (A) 300-epoch learning curves for the training data set "MG". (B) 1000-epoch learning curves of "MG". (C) 300-epoch learning curves for the training data set "SS". (D) 1000-epoch learning curves of "SS".
doi:10.1371/journal.pone.0062894.g003

American 60 Seconds Science". We name this data set "SS". We use 300 samples from this speech signal for training and 10000 different samples for testing. The values of these two data sets are all normalized in the range $[-1, 1]$.

Results of the FNN

The P external time-delay signals serve as input of the FNN. At the nth time point, the input vector $\mathbf{u}(n)$ is described by the $P\times 1$ vector,

$$\mathbf{u}(n)=[s(n-1),s(n-2),\ldots,s(n-P)]^{\mathrm{T}},$$

where $(P+1)\le n\le(P+n_0)$. In the output layer, there is only one neuron. The actual prediction made by the feedforward neural network at time n is the output of the FNN $y^{\mathrm{o}}(n)$, and the desired prediction is $s(n)$.

In the following simulations, the elements of the initial weight matrices $\mathbf{W}^{\mathrm{h}}$ and $\mathbf{W}^{\mathrm{o}}$ are randomly selected as small values uniformly distributed in [0, 0.05]. All numerical results in this section are averaged over 10 independent runs. The learning curves of these 10 runs are quite similar and the standard deviations of the learning results across the 10 independent runs are very small, and are therefore not shown here. As for many other learning algorithms, the convergence of the MEE-BP algorithm is slow when a small learning rate is adopted. Certainly we can increase the learning rate to accelerate the training process, but in this situation the learning curve tends to oscillate slightly at the latter stage of the training process. In this paper, we use a damping learning rate to make the learning process fast at the beginning and to prevent oscillations in the long run. The initial learning rate is set to $\eta=0.015$ and the learning rate decreases exponentially from one epoch to the next, $\eta=0.996\eta$. As we have mentioned in the previous section, a damping IP learning rate is also used to prevent unstable learning behavior during a long training process. The initial learning rate of IP in Eq. (2) is $\eta_{\mathrm{IP}}=0.002$, and the IP learning rate also decreases exponentially, $\eta_{\mathrm{IP}}=0.998\eta_{\mathrm{IP}}$.

The first simulation compares the learning curves of the MEE algorithm and the synergistic algorithm. In synergistic learning, the activation functions of neurons in both the hidden layer and the output layer are adjusted by IP. In this simulation, structural parameters of the FNN are set to $P=6$ and $M=5$. A Gaussian kernel is used to estimate entropy in all simulations with kernel size $\sigma=0.1$. The initial values of all activation functions are set to $a=1$ and $b=0$. Figure 3 shows the learning curves of quadratic information potentials of the training error and Fig. 4 shows the

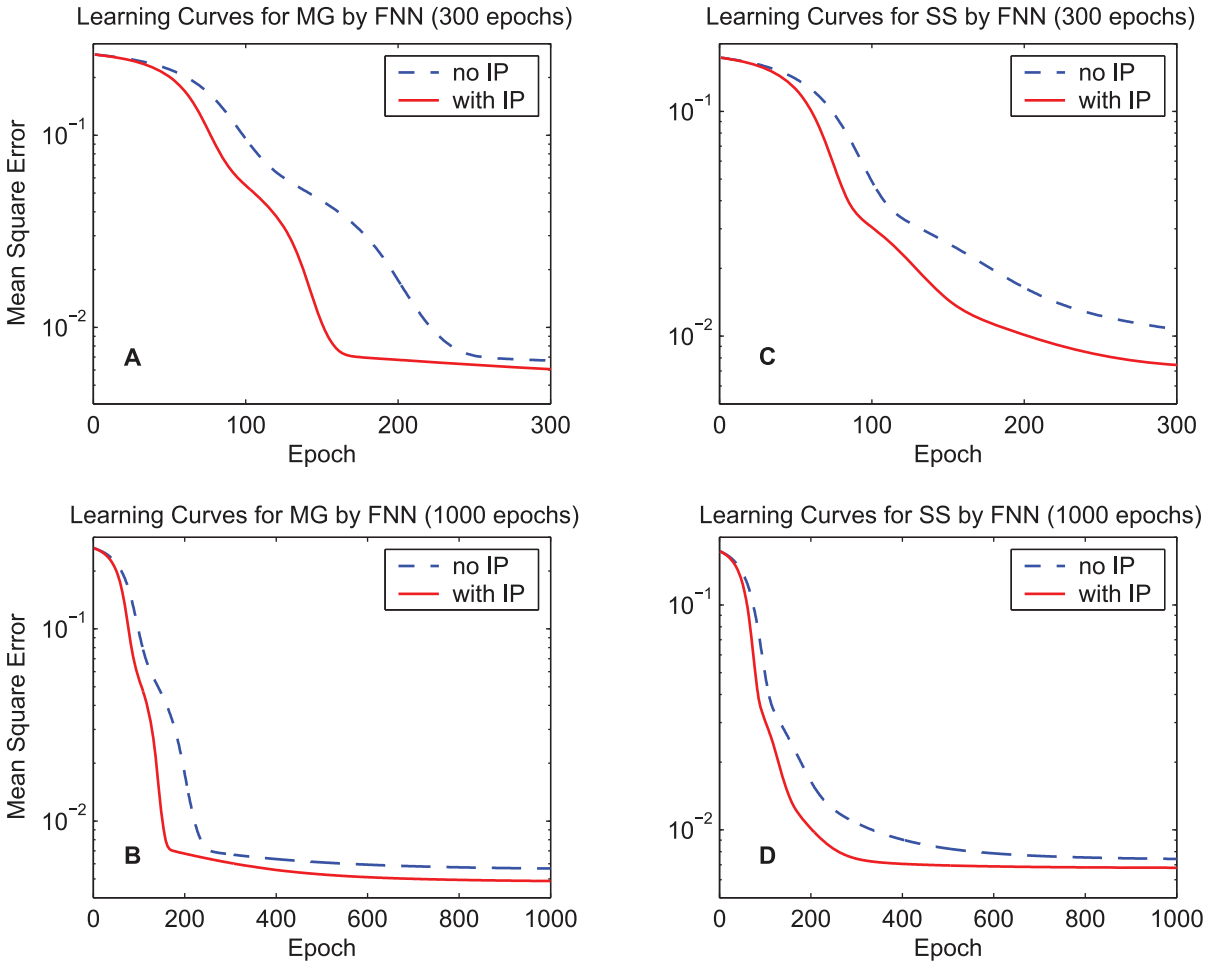

Figure 4. Learning curves of the mean square error by the FNN. The dashed lines denote the learning curves of the MEE algorithm, and the solid lines denote the learning curves of the synergistic algorithm. (A) 300-epoch learning curves for the training data set "MG". (B) 1000-epoch learning curves of "MG". (C) 300-epoch learning curves for the training data set "SS". (D) 1000-epoch learning curves of "SS".
doi:10.1371/journal.pone.0062894.g004

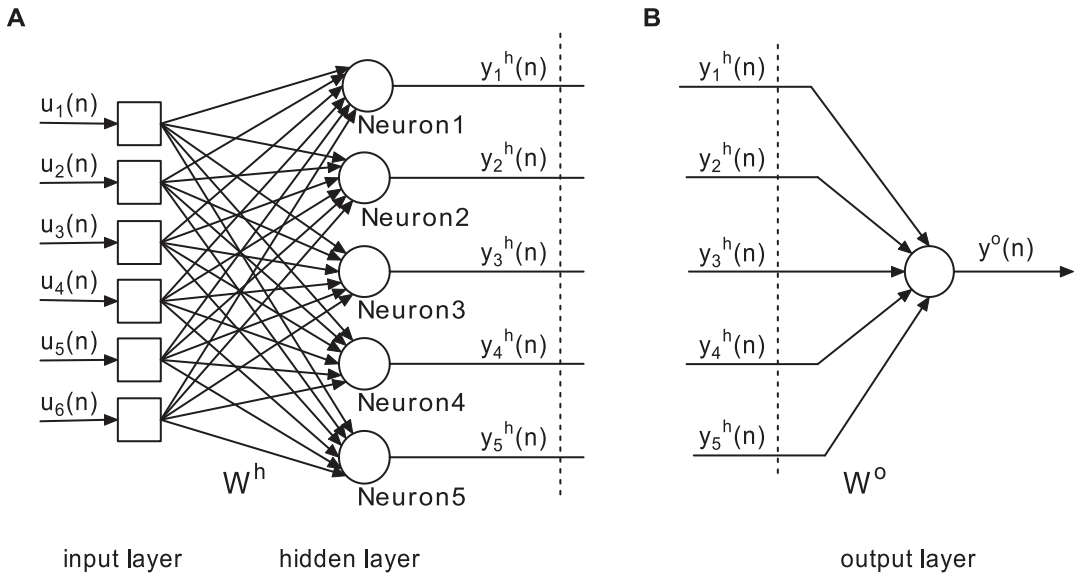

Figure 5. Decomposition of the FNN. (A) The input layer and the hidden layer of the FNN. (B) The output layer of the FNN.
doi:10.1371/journal.pone.0062894.g005

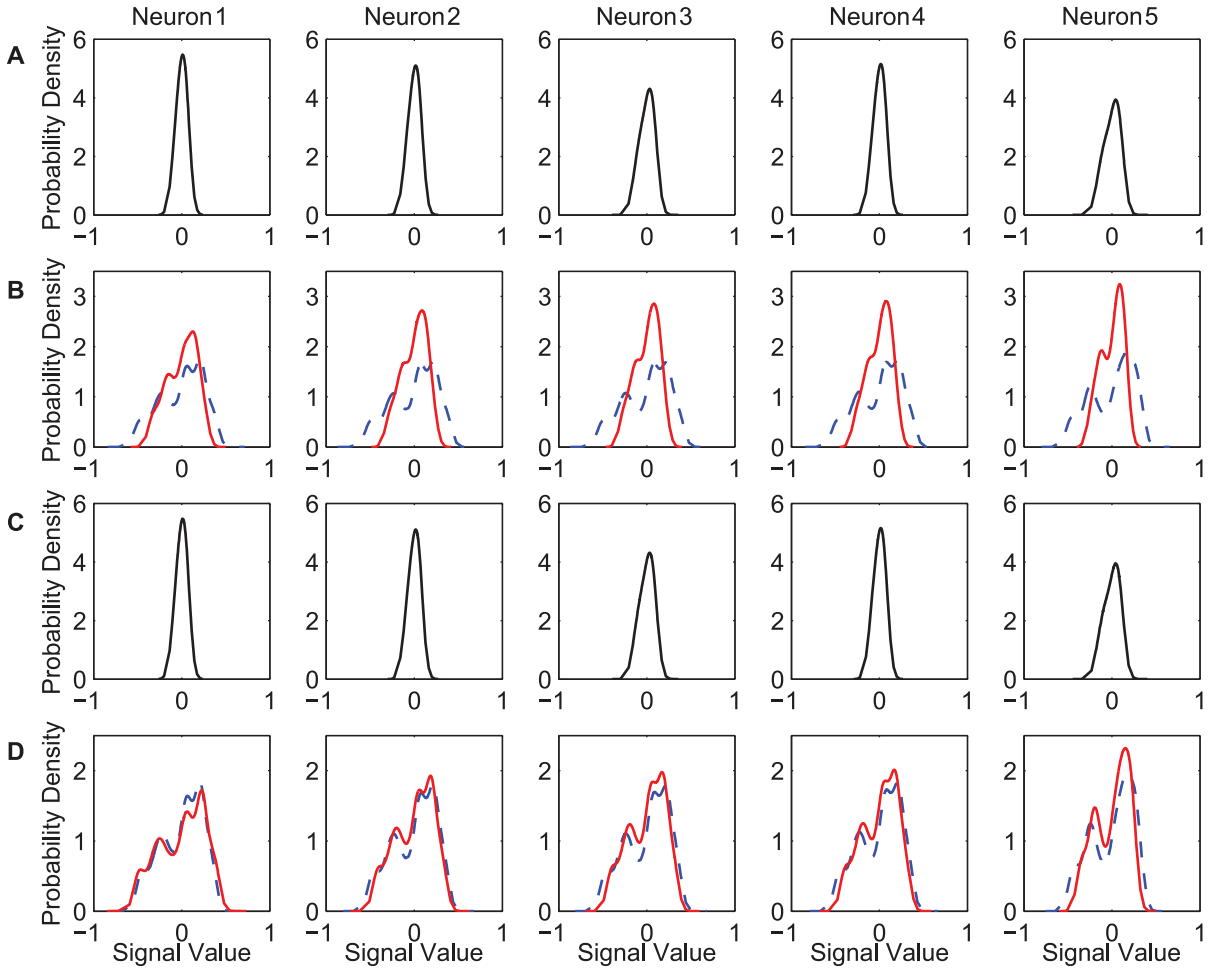

Figure 6. Input and output distributions for neurons in the hidden layer of the FNN. Input and output distributions for the five hidden neurons with the training data set "MG" are displayed. (A) Initial input distributions for the five hidden neurons. (B) Input distributions after 1000-epoch training for the two algorithms. (C) Initial output distributions for the five hidden neurons. (D) Output distributions after 1000-epoch training for the two algorithms. In (B) and (D), the dash lines denote the distributions obtained by the MEE algorithm, and the solid lines denote the distributions obtained by the synergistic algorithm.
doi:10.1371/journal.pone.0062894.g006

learning curves of the training MSE. When calculating the MSE during the training procedure, the bias of the output is adjusted so as to cancel the mean error over the training set. For each learning curve, we display 300 epochs to compare the training speed and also display 1000 epochs to measure the final performance. As the learning curves of the information potential and the MSE show, the synergistic algorithm outperforms the MEE algorithm with regard to the convergence speed. After a long run, i.e., 1000 epochs, the synergistic algorithm can still maintain good performance. As a classical performance criterion, the mean square errors of the training set and the testing set after the 1000-epoch training process are summarized in Table 1 for "MG" and Table 2 for "SS". In the last row of each table, the improvement percentage of the performance is the difference between the MSE of the MEE algorithm and the MSE of the synergistic algorithm, divided by the MSE of the MEE algorithm. According to the training and testing results of the MSE, the synergistic learning algorithm performs better than the MEE algorithm considered in isolation. The quadratic information potentials are also summarized in these tables. The improvement of the quadratic information potential is not as significant as that of the MSE.

In order to analyze the synergies between IP and synaptic plasticity in detail, input, output, and error distributions of neurons for the training set "MG" are presented. All these distributions are obtained by kernel density estimation in a single run, and results of two independent runs are similar. In order to explain the results clearly, we decompose the FNN into two parts, which are shown in Fig. 5.

Figure 6 shows the input and output distributions of the neurons in the hidden layer. We can refer to Figure 5(A) while analyzing the results shown in Fig. 6. Figure 6(A) shows the initial input distributions for five hidden neurons. As the elements of the initial weight matrices are randomly selected small values, the initial input of each hidden neuron is concentrated on a relatively small range. Figure 6(B) shows the input distributions after 1000 epochs, which are expanded into a relatively wide range in contrast to the initial input distributions. During the training process, the network input vectors $\mathbf{u}$ of each epoch for the two algorithms are totally identical, therefore the change of the input distributions of five hidden neurons is due to the updating of the synaptic weights $\mathbf{W}^{h}$. After training, the input distributions for the synergistic algorithm are more similar to the initial input distributions than those for the MEE algorithm, in other words, the change of the input

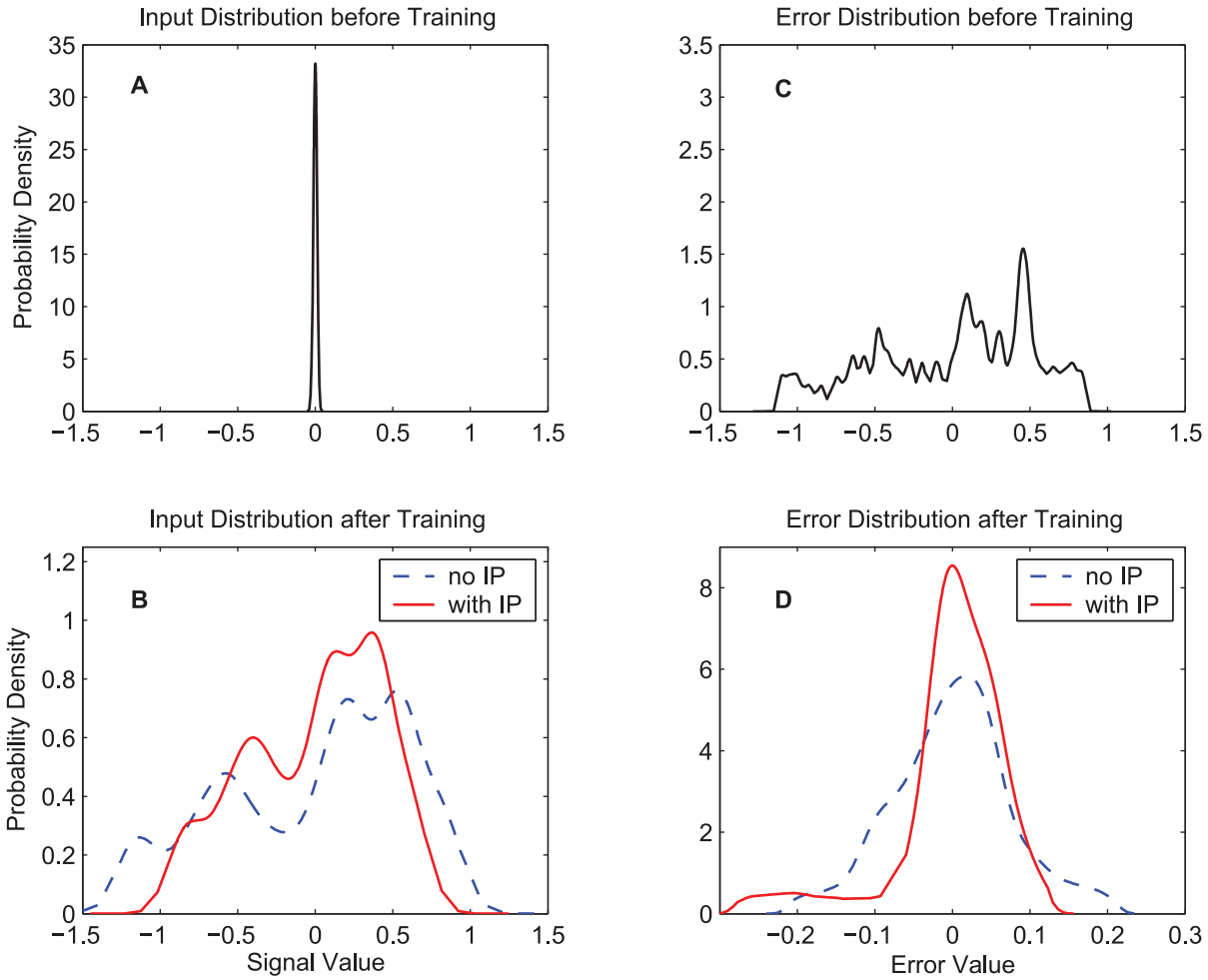

Figure 7. Input distributions for the output neuron and error distributions of the FNN. Input distributions for the single output neuron and error distributions with the training data set "MG" are presented. (A) Initial input distribution. (B) Input distributions after 1000-epoch training for the two algorithms. (C) Initial error distribution. (D) Error distributions after 1000-epoch training for the two algorithms. In (B) and (D), the dash lines denote the distributions obtained by the MEE algorithm, and the solid lines denote the distributions obtained by the synergistic algorithm.
doi:10.1371/journal.pone.0062894.g007

distributions for the synergistic algorithm is relatively small. We infer that it is relatively easy for the weight update rule to form such input distributions from the initial input distributions, thereby accelerating the training process. Figure 6(C) shows the initial output distributions of the neurons in the hidden layer. The initial output is also concentrated on a small range. Figure 6(D) shows the output distributions after 1000 epochs. After training, the output distributions of the two different algorithms are similar for each hidden neuron.

Figure 7 shows the input and training error distributions of the output neuron. Since the output of this neuron is closely related to the error, so we do not present the output distribution here. We can refer to Fig. 5(B) while analyzing the results shown in Fig. 7. Figure 7 (A) and (B) show the input distributions of the output neuron (distributions of v^o) before and after training, respectively. Before training, the input distribution is concentrated around zero. After 1000 epochs, the input distribution obtained by the synergistic learning algorithm is more concentrated than the distribution obtained by the MEE algorithm. Note that the input of the output neuron v^o is the weighted sum of the output of

Table 1. Performance comparison for the FNN using "MG".

Data set	Training set		Testing set	
Criterion	$\hat{V}_{2,\sigma}$	**MSE**	$\hat{V}_{2,\sigma}$	**MSE**
No IP	2.2632	0.0056670	2.2967	0.0051939
With IP	2.3950	0.0048786	2.4212	0.0044354
Improvement (MSE)	13.91%		14.60%	

doi:10.1371/journal.pone.0062894.t001

Table 2. Performance comparison for the FNN using "SS".

Data set	Training set		Testing set	
Criterion	$\hat{V}_{2,\sigma}$	**MSE**	$\hat{V}_{2,\sigma}$	**MSE**
No IP	2.1518	0.0074291	2.6066	0.0019096
With IP	2.1905	0.0067886	2.6703	0.0012658
Improvement (MSE)	8.62%		33.71%	

doi:10.1371/journal.pone.0062894.t002

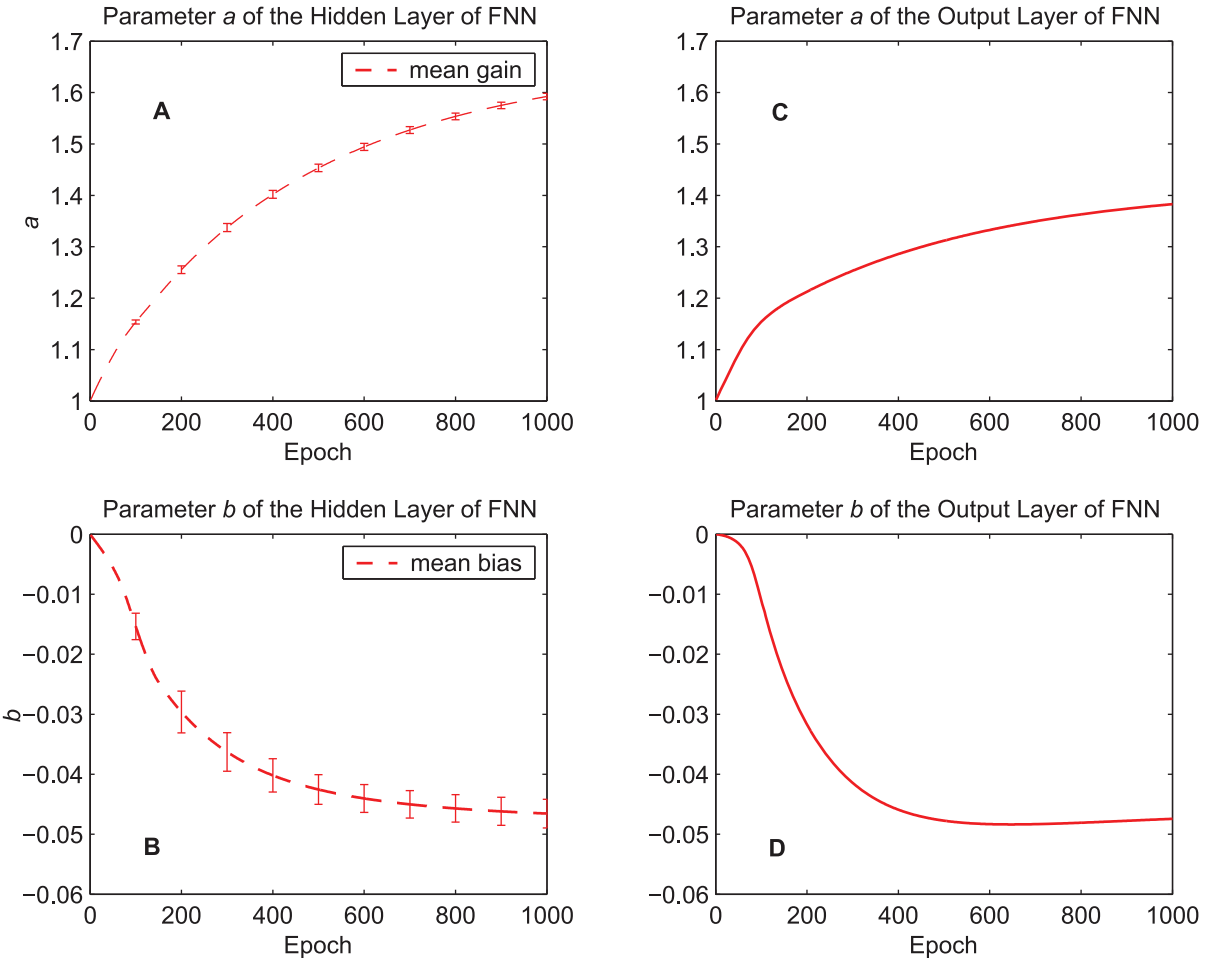

Figure 8. Evolution of the parameters of the activation functions in the FNN. The training data set "MG" is used. (A) Mean of the gain parameter a of the five hidden neurons. (B) Mean of the bias parameter b of the five hidden neurons. (C) The gain parameter a of the output neuron. (D) The bias parameter b of the output neuron.
doi:10.1371/journal.pone.0062894.g008

hidden neurons, $v^o = (\mathbf{W}^o)^T \mathbf{y}^h$. As we see from Fig. 6(D), the output distributions of each hidden neuron for the two algorithms are similar, which means that the statistical properties of the output of hidden neurons $\mathbf{y}^h$ are similar in the two algorithms. Thus, the main reason for the difference in the distributions of v^o after training is the updating of the synaptic weights $\mathbf{W}^o$. In the synergistic learning case, the distribution of v^o after training is relatively similar to the initial input distribution. This implies the weight updating process of the synergistic algorithm makes relatively small change of the synaptic weights $\mathbf{W}^o$, thus it is more easily achieved. Figure 7(C) and (D) show the error distributions of the two algorithms. After 1000 epochs, the FNN trained by the synergistic learning algorithm produces errors that are more concentrated around zero, which means there are higher number of small errors and fewer number of large errors, indicating better performance in terms of error. Although the increase of the quadratic information potential with IP is not substantial as shown in Table 1 and Table 2, learning with or without IP yields qualitatively different error distributions.

Thus, the analysis on the results in Fig. 6 and Fig. 7 provides an explanation for the fast convergence and good final performance of the synergistic learning shown in Fig. 3 and Fig. 4.

Figure 8 shows how the parameters of the activation functions in the FNN evolve over 1000 epochs. Figure 8 (A)(B) display the evolution of a and b averaged over the five hidden neurons, respectively. The changes of the parameter pairs for the five hidden neurons are very similar, thus we show the averaged results and the corresponding error bars (standard deviations across the five hidden neurons). Figure 8 (C)(D) display the evolution of the parameters of the output neuron. In the FNN trained by the algorithm with IP, the values of a get large constantly to steepen the activation functions while the values of b decrease.

The second simulation concerns the IP learning rate. Figure 9 compares the learning curves of the synergistic algorithm with various initial IP learning rates η_{IP} for the data set "MG". Four initial IP learning rates $\eta_{IP}=0.001$, $\eta_{IP}=0.002$, $\eta_{IP}=0.003$, and $\eta_{IP}=0$ (no IP) are used for comparison. The results in Fig. 9 indicate that with a relatively large IP learning rate the information potential increases and the MSE decreases faster, but the highest information potentials and the smallest mean-square-errors during training procedures with the three non-zero IP learning rates are similar. In terms of the convergence speed, a relatively large IP learning rate is preferable, however, if η_{IP} is set to be a much larger value (larger than 0.003), some ripples appear when the learning curve tends to converge, which is similar to that of the online reservoir adaptation by intrinsic plasticity in [21].

Figure 10 displays the performance of FNNs with different numbers of hidden neurons (from 3 to 15) using the training data

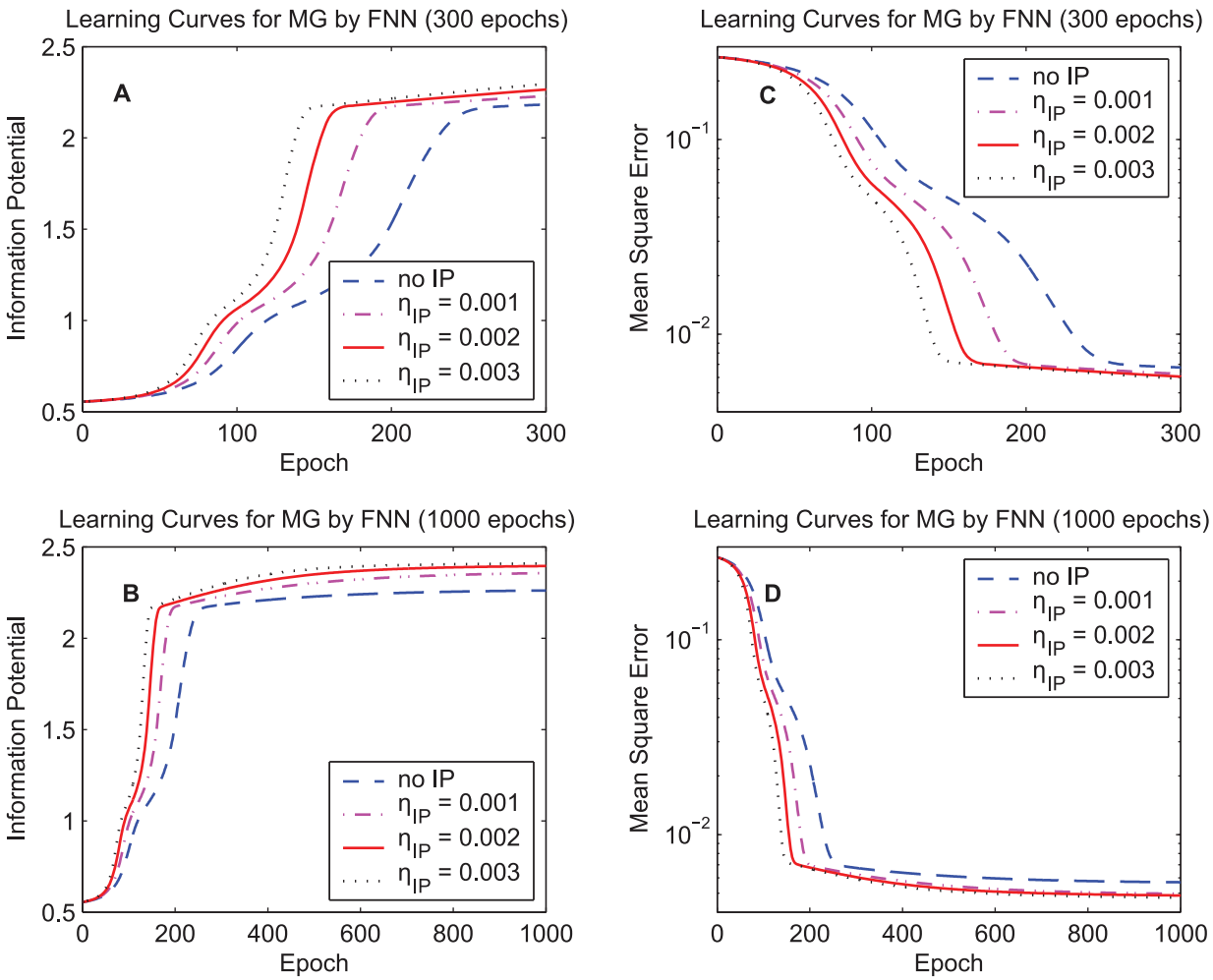

Figure 9. Learning curves of the FNN with different IP learning rates. The training data set "MG" is used. The initial IP learning rates $\eta_{IP}=0.001$, $\eta_{IP}=0.002$, $\eta_{IP}=0.003$, and $\eta_{IP}=0$ (no IP) are used for comparison. Learning curves of the quadratic information potential: (A) 300 epochs. (B) 1000 epochs. Learning curves of the mean square error: (C) 300 epochs. (D) 1000 epochs.
doi:10.1371/journal.pone.0062894.g009

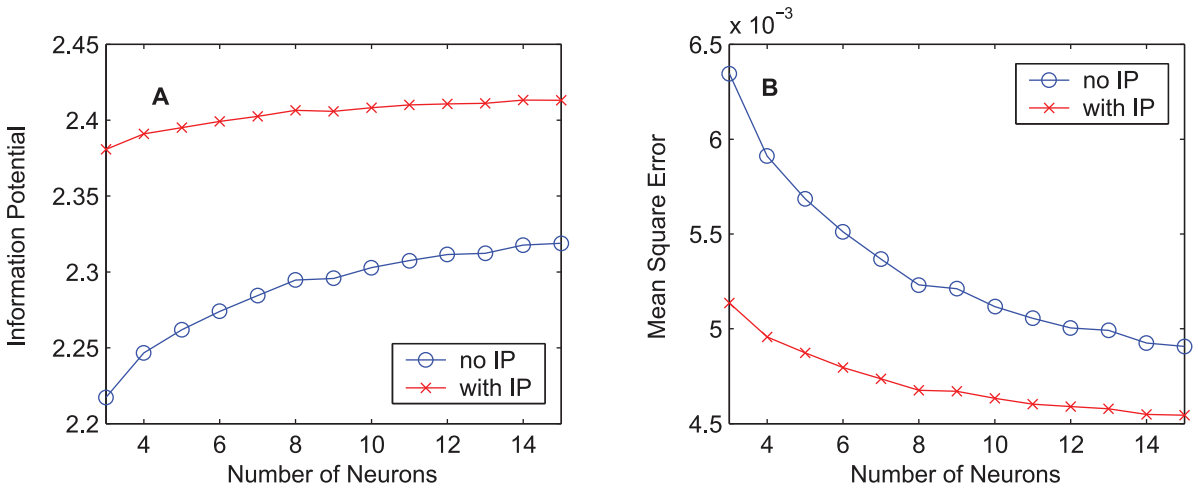

Figure 10. Relation between the training result and the number of hidden neurons of the FNN. Training results after 1000-epoch training for the case of the training data set "MG" are presented. The circle markers denote the results obtained by the MEE algorithm, and the cross markers denote the results obtained by the synergistic algorithm. (A) Results of the quadratic information potential. (B) Results of the mean square error.
doi:10.1371/journal.pone.0062894.g010

Table 3. Performance comparison for the RNN using "MG".

Data set	Training set		Testing set	
Criterion	$\hat{V}_{2,\sigma}$	MSE	$\hat{V}_{2,\sigma}$	MSE
No IP	2.5666	0.0021446	2.5876	0.0019388
With IP	2.6653	0.0013974	2.6994	0.0009594
Improvement (MSE)	34.84%		50.51%	

doi:10.1371/journal.pone.0062894.t003

Table 4. Performance comparison for the RNN using "SS".

Data set	Training set		Testing set	
Criterion	$\hat{V}_{2,\sigma}$	MSE	$\hat{V}_{2,\sigma}$	MSE
No IP	2.2362	0.0060375	2.6811	0.0011350
With IP	2.2765	0.0055386	2.7057	0.0009260
Improvement (MSE)	8.26%		18.41%	

doi:10.1371/journal.pone.0062894.t004

set "MG". The values of the estimated quadratic information potential and the mean square error after 1000 epochs are shown. With the assistance of IP, the performance of the FNN containing various numbers of hidden neurons is always better. In terms of the quadratic information potential, the result obtained by a FNN containing 3 hidden neurons with IP is better than that obtained by a FNN containing 15 hidden neurons without IP. As for the MSE, the result obtained by a FNN containing 3 hidden neurons with IP is comparable to that obtained by a FNN containing 10 hidden neurons without IP. The performance improvement caused by adding IP is more significant than that caused by increasing the number of hidden neurons. In addition, increasing the number of hidden neurons brings a heavier computational burden. In the FNN, adding one hidden neuron brings extra $(P+N)$ connection weights; for example, if we increase the number of hidden neurons from 5 to 10 in the above simulations, we have to update extra $(6+1)\times 5=35$ weights. However, we only need to update $(5+1)=6$ parameter pairs of (a,b) with simple calculations if we add IP to these five hidden neurons and the one output neuron. According to these results, the FNN trained by the synergistic learning algorithm can work well with fewer neurons and thus reduce the computational cost.

Results of the RNN

As in the case of the FNN, we discuss how the recurrent neural network handles the problem of single-step prediction using the same data sets. The input of the network consists of P time-delay signals and N feedback signals,

$$\mathbf{u}(n)=[s(n-1),s(n-2),\ldots,s(n-P),\\ y_1(n-1),y_2(n-2),\ldots,y_N(n-1)]^{\mathrm{T}}.$$

The prediction made by the recurrent neural network at time n is the output of the first neuron $y_1(n)$, and the desired output is also $s(n)$.

The values of elements in the initial weight matrix are also randomly selected as small values uniformly distributed in [0, 0.05]. Results in this section are also averaged over 10 independent runs. The MEE learning rate is set to $\eta=0.01$. The initial IP learning rate is $\eta_{\mathrm{IP}}=0.01$ and it decreases exponentially, $\eta_{\mathrm{IP}}=0.995\eta_{\mathrm{IP}}$.

The first simulation compares the learning curves of the synergistic algorithm and the MEE algorithm. Structural parameters of the RNN are set to $P=4$ and $N=2$. A Gaussian kernel with kernel size $\sigma=0.1$ is used to estimate entropy. The initial values of all activation functions are set to $a=1$ and $b=0$. Figure 11 shows the learning curves concerning the quadratic information potentials of the training error, and Fig. 12 shows the learning curves of the training MSE. After the training procedure, the mean square errors and the quadratic information potentials of the training set and the testing set are summarized in Table 3 for "MG" and Table 4 for "SS". These results manifest that the synergistic algorithm also outperforms the MEE algorithm for the RNN.

Figure 13 shows the input and output distributions of the neurons in the RNN. Before training, the input distribution of the first neuron (the output neuron of the RNN, denoted by "Neuron 1″ in the figure) is concentrated on a small range, as shown in Fig. 13(A). After training, in contrast to the situation without IP, the change of the input distribution from the initial state is relatively small in the situation with IP, as shown in Fig. 13(B). The training error distributions before and after training are shown in Fig. 13(C) and (D), respectively. The situation of the first neuron in the RNN is similar to that of the output neuron in the FNN, but the difference between the distributions with and without IP for the second neuron (denoted by "Neuron 2″ in the figure) seems interesting. Without IP, the input and output distributions of the second neuron after learning are restricted in a relatively small range. However, with IP, the input distribution after learning is expanded to a wider range; correspondingly, the output of the second neuron is also expanded from the initial distribution. Since the output signal returns to constitute the input of the RNN, if the output of the second neuron is restricted in a very small range, the input signal, $r_2(n)$, is ineffective. With a wide data range, $r_2(n)$ can provide more information (larger entropy) for the input of the RNN.

Figure 14 shows the evolution of the parameters of the activation functions in the RNN. The values of a get large constantly to steepen the activation functions while the values of b of the two neurons are adjusted to match the position of the input distribution.

The second simulation concerns the initial IP learning rate. Figure 15 compares the learning curves of the synergistic algorithm with various initial IP learning rate η_{IP} for the training data set "MG". The initial IP learning rates $\eta_{\mathrm{IP}}=0.005$, $\eta_{\mathrm{IP}}=0.01$, $\eta_{\mathrm{IP}}=0.015$, and $\eta_{\mathrm{IP}}=0$ (no IP) are used. In terms of both the convergence speed and the final result, a relatively large IP learning rate is better; however, oscillation behavior appears when η_{IP} gets much larger. This phenomenon of the intrinsic plasticity rule may be ubiquitous in different kinds of neural networks.

Figure 16 displays the performance of the RNNs with different numbers of neurons using the training data set "MG". Without IP, the performance improvement is trivial with the increasing of the number of neurons. The MEE learning algorithm for the RNN seems insensitive to the number of neurons. In this situation, using two neurons seems effective enough since adding neurons increases the computational cost but neither raises the quadratic information potential nor lowers the MSE substantially. For the RNNs, the performance improvement caused by adding IP is far more significant than that caused by increasing the number of hidden neurons.

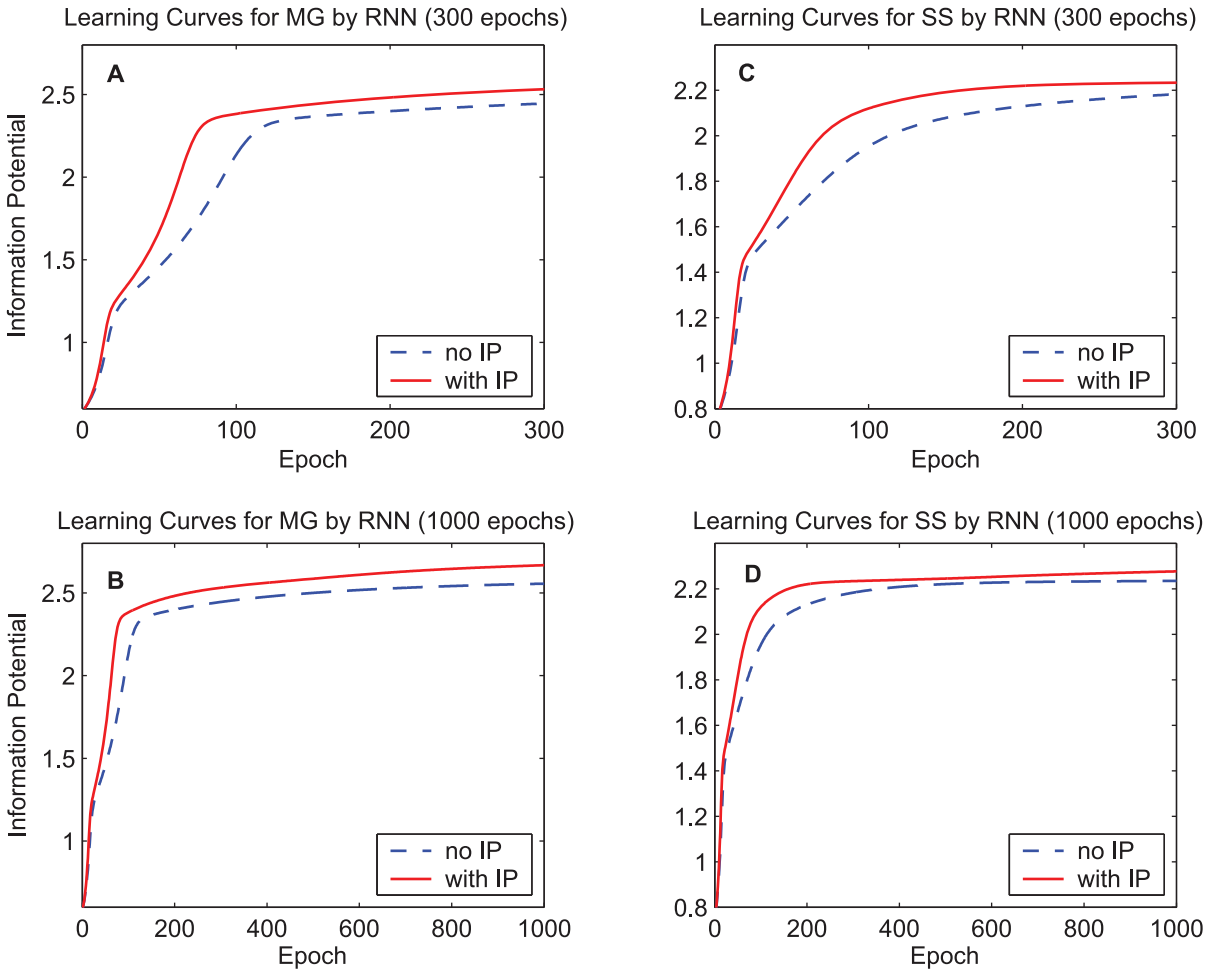

Figure 11. Learning curves of the quadratic information potential by the RNN. The dashed lines denote the learning curves of the MEE algorithm, and the solid lines denote the learning curves of the synergistic algorithm. (A) 300-epoch learning curves for the training data set "MG". (B) 1000-epoch learning curves of "MG". (C) 300-epoch learning curves for the training data set "SS". (D) 1000-epoch learning curves of "SS".
doi:10.1371/journal.pone.0062894.g011

In some of the above simulations, we do not show the results for the data set "SS", since they are similar to those for the data set "MG". All activation functions used in the above-mentioned neural networks are tanh functions, thus the intrinsic plasticity rule for the tanh function is applied to the synergistic learning algorithm. In the case of using logistic functions, similar results can be obtained.

Discussion

Combining the MEE algorithm as the synaptic plasticity rule and the information-maximization algorithm as the intrinsic plasticity rule, we proposed a synergistic information-theoretic learning algorithm for training artificial neural networks. Whereas the information-maximization algorithm can increase the mutual information of a single neuron, it can not optimize the cost function such as EEC and MSE. Nevertheless, simulations have shown that this information-maximization-based IP rule benefits the artificial neural networks in both the convergence speed and the final learning result. As the IP rule adjusts the activation function of a single neuron to match its input distribution so that all output levels tend to appear equivalently, the input can be encoded much more efficiently and the discriminative ability of the neuron is enhanced. We believe that the discriminative ability of a neuron plays a nontrivial role in the performance of artificial neural networks. In terms of the FNN, the synergistic learning algorithm with IP only in the hidden layer or only in the output neuron still outperforms the MEE algorithm without IP, but is inferior to the learning algorithm with IP in both layers (we do not present these results in the paper).

Compared with the original algorithm, the synergistic learning algorithm can be performed with a relatively small increase in computational cost due to the local nature of the IP mechanism and the simplicity of the information-maximization algorithm. In addition, we have used the efficient batch version of the information-maximization algorithm. In applications, a long training process is unnecessary since the improvement is minor at the end part of the training. For example, with a 300-epoch training, the IP rule is quite effective to improve the performance. In a long run, the synergistic learning maintains good performance.

Advanced search methods for nonlinear optimization such as conjugate gradient algorithms and the Levenberg-Marquardt algorithm can be used to further speed up the learning process. In order to focus on the synergies between IP and synaptic plasticity and preclude influences of other advanced search methods on learning, we used the simple gradient descent (GD) method.

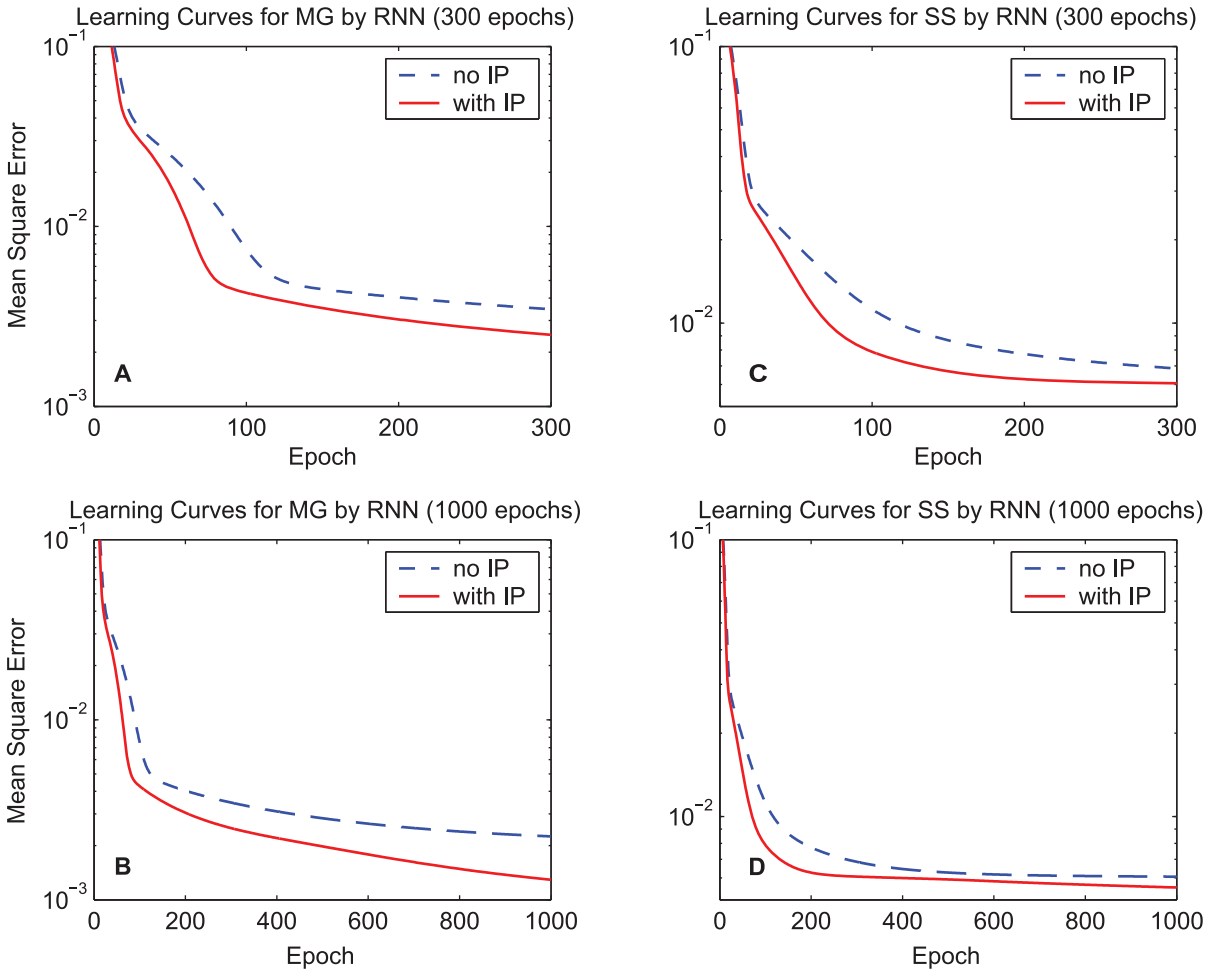

Figure 12. Learning curves of the mean square error by the RNN. The dashed lines denote the learning curves of the MEE algorithm, and the solid lines denote the learning curves of the synergistic algorithm. (A) 300-epoch learning curves for the training data set "MG". (B) 1000-epoch learning curves of "MG". (C) 300-epoch learning curves for the training data set "SS". (D) 1000-epoch learning curves of "SS".
doi:10.1371/journal.pone.0062894.g012

In biology, Bell and Sejnowski's information-maximization algorithm can match the statistics of naturally occurring visual contrasts to the response amplitudes of the blowfly's large monopolar cell (LMC). The contrast-response function of the LMCs in the blowfly's compound eye approximates to the cumulative probability distribution of contrast levels in natural scenes, thus the inputs are encoded so that all response levels are used with equal frequency, resulting in a uniform output distribution [28]. We may regard this experimental result as the biological justification of the proposed synergistic learning rule.

As related work, several studies have combined synaptic learning algorithms with Triesch's IP rule [10,11,29] or other revised versions for training artificial neural networks. In [30], an unsupervised scheme including the IP rule for pretraining extreme learning machines is introduced. In [21,31], an online adaptation rule with IP for the reservoir networks is presented. To the best of our knowledge, all these previous studies on the effects of IP on neural network learning have used the MSE criterion rather than the EEC criterion [21,30,31]. Besides, the energy consumption of a biological neuron is considered as an important constraint for the IP rules used in these previous studies. In this study, we neglect this energy constraint and regard Bell and Sejnowski's information-maximization algorithm for a single neuron's activation function as the intrinsic plasticity rule. In a recent study related to ours, Lazar et al. presented a self-organizing recurrent network (SORN) combining intrinsic plasticity and synaptic plasticity that learns spatio-temporal patterns in its input while maintaining its dynamics in a healthy regime suitable for learning, in which the IP rule regulates a neuron's firing threshold to maintain a low average activity level and the synaptic rule is a simple model of STDP [29]. This work implies that as we try to understand neural plasticity and how it shapes the brain's representation and processing, it is insufficient to study individual mechanisms in isolation and studying their interactions is necessary [29]. In this study, we have shown how the information-maximization IP rule improves the performance of FNNs and RNNs trained with the EEC criterion and we draw the conclusion that the interactions of different plasticity mechanisms can benefit artificial neural networks in supervised learning applications. Here we have focused on providing an upgraded information-theoretic learning method for applications and we have not specifically attempted to emphasize on the biological relevance.

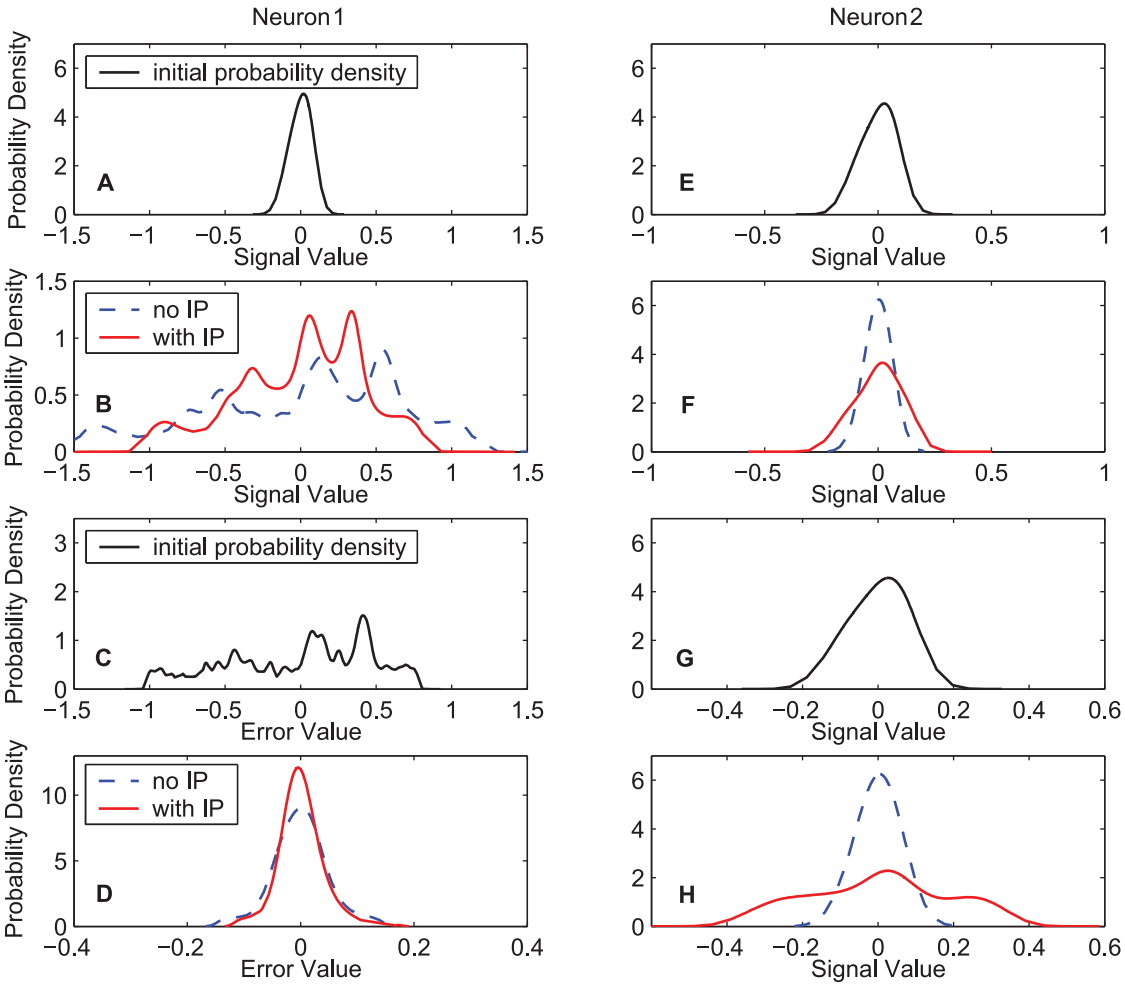

Figure 13. Input, output and error distributions for neurons of the RNN. The training data set "MG" is used. Neuron 1 (output neuron): (A) Initial input distribution. (B) Input distributions after 1000-epoch training for the two algorithms. (C) Initial error distribution. (D) Error distributions after 1000-epoch training for the two algorithms. Neuron 2: (E) Initial input distribution. (F) Input distributions after 1000-epoch training for the two algorithms. (G) Initial output distribution. (H) Output distributions after 1000-epoch training for the two algorithms. In (B), (D), (F), and (H), the dash lines denote the distributions obtained by the MEE algorithm, and the solid lines denote the distributions obtained by the synergistic algorithm.
doi:10.1371/journal.pone.0062894.g013

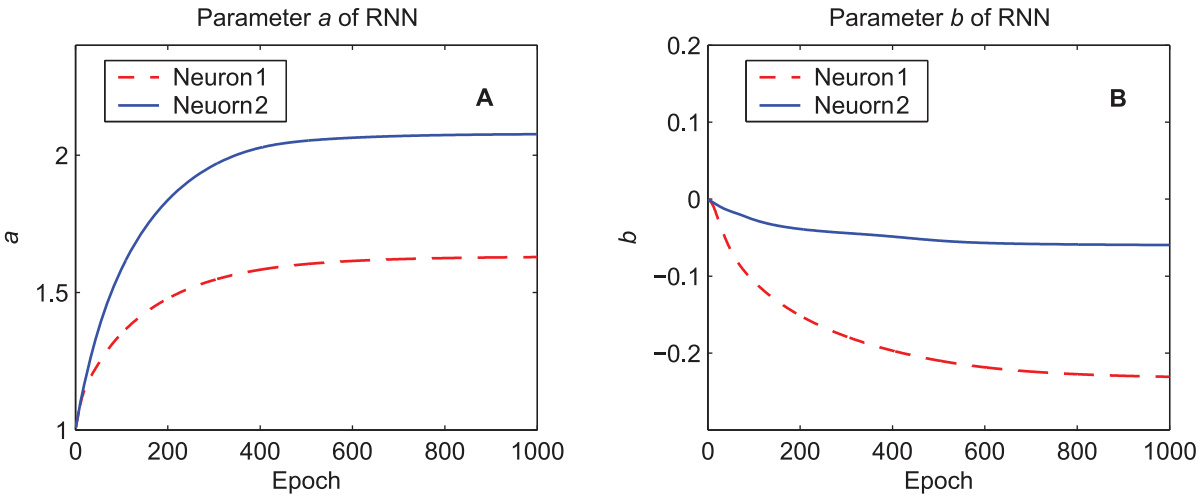

Figure 14. Evolution of the parameters of the activation functions in the RNN. The training data set "MG" is used. (A) The gain parameter a. (B) The bias parameter b.
doi:10.1371/journal.pone.0062894.g014

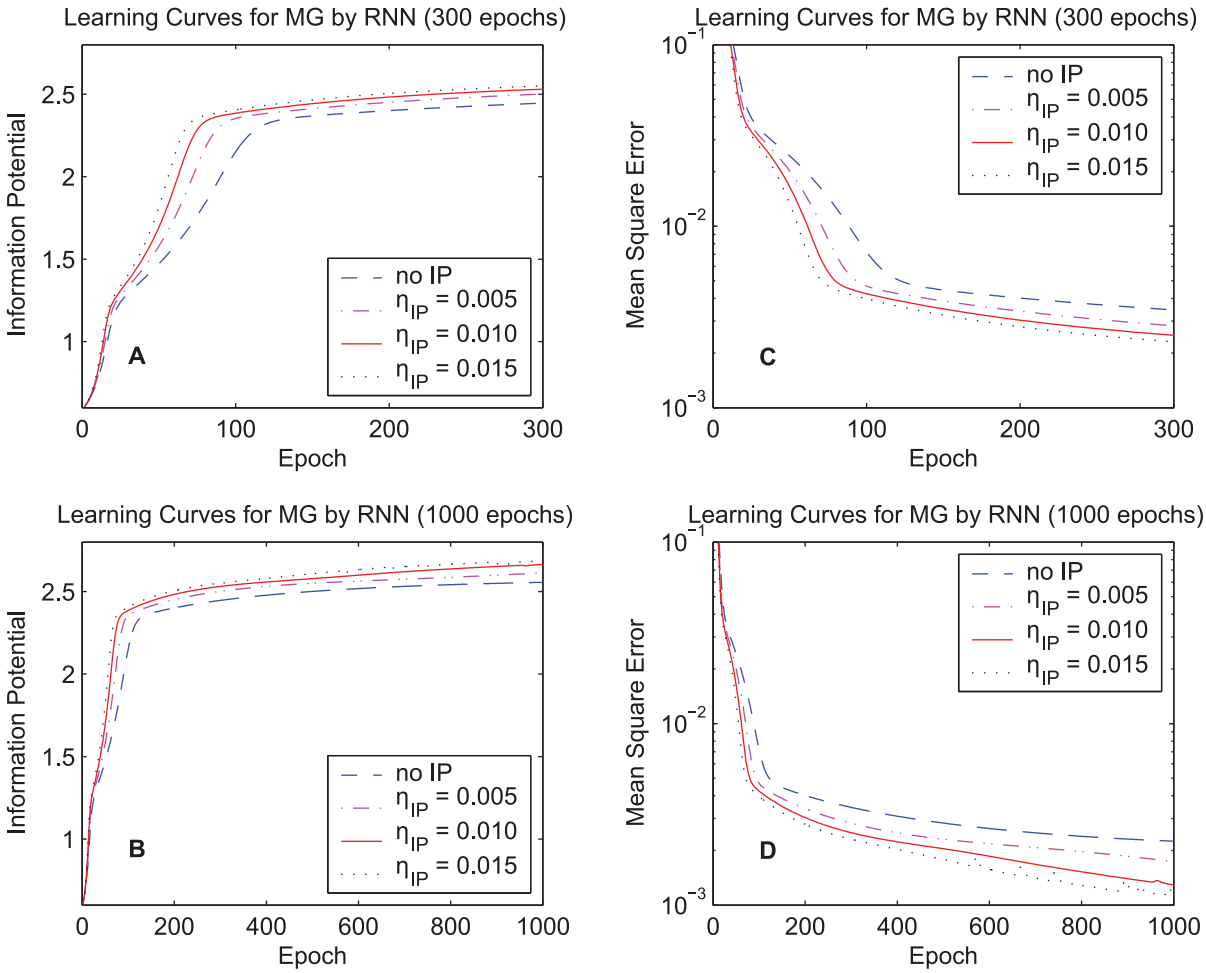

Figure 15. Learning curves by the RNN with different IP learning rates. The training data set "MG" is used. The initial IP learning rates $\eta_{IP}=0.005$, $\eta_{IP}=0.01$, $\eta_{IP}=0.015$, and $\eta_{IP}=0$ (no IP) are used for comparison. Learning curves of the quadratic information potential: (A) 300 epochs. (B) 1000 epochs. Learning curves of the mean square error: (C) 300 epochs. (D) 1000 epochs.
doi:10.1371/journal.pone.0062894.g015

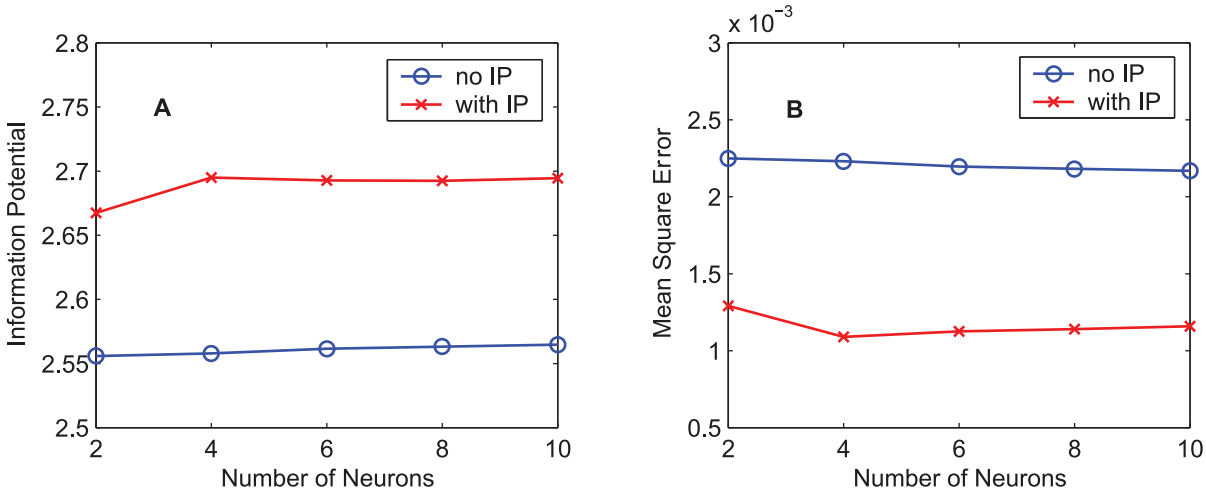

Figure 16. Relation between the training result and the number of neurons of the RNN. Training results after 1000-epoch training for the case of the training data set "MG" are presented. The circle markers denote the results obtained by the MEE algorithm, and the cross markers denote the results obtained by the synergistic algorithm. (A) Results of the quadratic information potential. (B) Results of the mean square error.
doi:10.1371/journal.pone.0062894.g016

Acknowledgments

We gratefully acknowledge the anonymous reviewers for providing valuable comments and suggestions, which greatly improved our paper.

Author Contributions

Conceived and designed the experiments: CL. Performed the experiments: YL. Analyzed the data: YL CL. Wrote the paper: YL CL.

References

1. Principe JC (2010) Information theoretic learning: Renyi's entropy and kernel perspectives. New York, Dordrecht, Heidelberg, London: Springer.
2. Marder E, Abbott LF, Turrigiano GG, Liu Z, Golowasch J (1996) Memory from the dynamics of intrinsic membrane currents. Proc Natl Acad Sci USA 93: 13481–13486.
3. Desai NS, Rutherford LC, Turrigiano GG (1999) Plasticity in the intrinsic excitability of cortical pyramidal neurons. Nat Neurosci 2: 515–520.
4. Zhang W, Linden DJ (2003) The other side of the engram: Experience-driven changes in neuronal intrinsic excitability. Nat Rev Neurosci 4: 885–900.
5. Daoudal G, Debanne D (2003) Long-term plasticity of intrinsic excitability: Learning rules and mechanisms. Learn Mem 10: 456–465.
6. Zhang M, Hung F, Zhu Y, Xie Z, Wang JH (2004) Calcium signal-dependent plasticity of neuronal excitability developed postnatally. J Neurobiol 61: 277–287.
7. Cudmore RH, Turrigiano GG (2004) Long-term potentiation of intrinsic excitability in LV visual cortical neurons. J Neurophysiol 92: 341–348.
8. Mozzachiodi R, Byrne JH (2009) More than synaptic plasticity: role of nonsynaptic plasticity in learning and memory. Trends Neurosci 33: 17–26.
9. Watt AJ, Desai NS (2010) Homeostatic plasticity and STDP: keeping a neuron's cool in a fluctuating world. Front Synaptic Neurosci 2(5).
10. Triesch J (2005) A gradient rule for the plasticity of a neuron's intrinsic excitability. Artificial Neural Networks: Biological Inspirations – ICANN 2005, Springer Berlin Heidelberg, 3696: 65–70.
11. Triesch J (2007) Synergies between intrinsic and synaptic plasticity mechanisms. Neural Comput 19(4): 885–909.
12. Stemmler M, Koch C (1999) How voltage-dependent conductances can adapt to maximize the information encoded by neuronal firing rate. Nat Neurosci 2: 521–527.
13. Li C (2011) A model of neuronal intrinsic plasticity. IEEE Trans Auton Ment Dev 3(4): 277–284.
14. Li C, Li Y (2013) A spike-based model of neuronal intrinsic plasticity. IEEE Trans Auton Ment Dev 5(1): 62–73.
15. Baddeley R, Abbott LF, Booth MC, Sengpiel F, Freeman T (1997) Responses of neurons in primary and inferior temporal visual cortices to natural scenes. Proc R Soc Lond B Biol Sci 264: 1775–1783.
16. Bell AJ, Sejnowski TJ (1995) An information-maximization approach to blind separation and blind deconvolution. Neural Comput 7: 1129–1159.
17. Erdogmus D, Principe JC (2002) An error-entropy minimization algorithm for supervised training of nonlinear adaptive systems. IEEE Trans Signal Process 50: 1780–1786.
18. Parzen E (1962) On the estimation of a probability density function and the mode. The annals of mathematical statistics 33: 1065–1076.
19. Erdogmus D, Principe JC (2000) Comparison of entropy and mean square error criteria in adaptive system training using higher order statistics. Proc ICA, Helsinki, Finland.
20. Williams RJ, Zipser D (1995) Gradient-based learning algorithms for recurrent networks and their computational complexity. In: Chauvin Y, Rumelhart DE. Backpropagation: Theory, architectures and applications. Hillsdale, New Jersey: Psychology Press. 433–486.
21. Steil JJ (2007) Online reservoir adaptation by intrinsic plasticity for back-propagation-decorrelation and echo state learning. Neural Netw 20: 353–364.
22. Principe JC, Euliano NR, LefebvreWC (1999) Neural and adaptive systems: fundamentals through simulations with CD-ROM. NewYork: JohnWiley & Sons, Inc.
23. Haykin S, Li L (1995) Nonlinear adaptive prediction of nonstationary signals. IEEE Trans Signal Process 43: 526–535.
24. Mandic DP, Chambers JA (1999) Toward an optimal PRNN-based nonlinear predictor. IEEE Trans Neural Netw 10: 1435–1442.
25. Williams RJ, Zipser D (1989) A learning algorithm for continually running fully recurrent neural networks. Neural Comput 1(2): 270–280.
26. Baltersee J, Chambers JA (1998) Nonlinear adaptive prediction of speech with a pipelined recurrent neural network. IEEE Trans Signal Process 46: 2207–2216.
27. Glass L, Mackey MC (1998) From clock to chaos. New York: Princeton University Press.
28. Laughlin SB (1981) A simple coding procedure enhances a neuron's information capacity. Z Naturforsch 36: 910–912.
29. Lazar A, Pipa G, Triesch J (2009) SORN: a self-organizing recurrent neural network. Front Comput Neurosci 3: 23.
30. Neumann K, Steil JJ (2011) Batch intrinsic plasticity for extreme learning machines. Artificial Neural Networks and Machine Learning – ICANN 2011, Springer Berlin Heidelberg, 6791: 339–346.
31. Schrauwen B, Wardermann M, Verstraeten D, Steil JJ, Stroobandt D (2008) Improving reservoirs using intrinsic plasticity. Neurocomputing 71: 1159–1171.