Supervised Learning in Neural Networks: Feedback-Network-Free Implementation and Biological Plausibility

Feng Lin, *Fellow, IEEE*

***Abstract*— The well-known backpropagation learning algorithm is probably the most popular learning algorithm in artificial neural networks. It has been widely used in various applications of deep learning. The backpropagation algorithm requires a separate feedback network to back propagate errors. This feedback network must have the same topology and connection strengths (weights) as the feed-forward network. In this article, we propose a new learning algorithm that is mathematically equivalent to the backpropagation algorithm but does not require a feedback network. The elimination of the feedback network makes the implementation of the new algorithm much simpler. The elimination of the feedback network also significantly increases biological plausibility for biological neural networks to learn using the new algorithm by means of some retrograde regulatory mechanisms that may exist in neurons. This new algorithm also eliminates the need for two-phase adaptation (feed-forward phase and feedback phase). Hence, neurons can adapt asynchronously and concurrently in a way analogous to that of biological neurons.**

***Index Terms*— Backpropagation, biological plausibility, deep learning, neural networks.**

I. Introduction

SINCE the introduction of (artificial) neural networks more than half-century ago, the field has made great progress, in terms of both theories and applications [1]–[4]. More recently, due to the availability of a large amount of data and much more powerful and fast computers, deep learning using various neural networks has significantly improved the state of the art in speech recognition, object recognition, and detection [5]–[13], and in many other domains such as smart grids [14], robot manipulators [15], and liquid crystals [16]. It leads to breakthroughs in processing images, video, audio, and other fields. For example, accurate speech recognition and face recognition not possible just ten years ago are now widely available.

Among various learning algorithms[1] used in neural networks, the backpropagation algorithm is probably the most well-known and widely used [17]–[19]. It allows errors to be backpropagated via a feedback network so that the strengths/weights of each synapse/connection can be learned/adapted to reduce the errors.

Researchers have investigated whether an analogous adaptation mechanism might occur in biological neural systems [20] since the introduction of the backpropagation algorithm. The consensus among neuroscientists is that the backpropagation is not likely to occur in biological neural systems [21]. The main reason for this consensus is that the backpropagation algorithm requires a dedicated feedback network to backpropagate errors (see, for example, [17]). Such a separate feedback network does not exist as a biological neural system. It is unreasonable to require that a biological neural system has a one-to-one correspondence between synapses in the feed-forward network and feedback network. It is even more unreasonable to require that the corresponding connections in the two networks maintain identical strengths as they adapt, considering the fact that a connection between two neurons is composed of many (even hundreds) of synapses in most biological neural systems.

On the other hand, several retrograde regulatory mechanisms in biological neural systems have been discovered by neuroscientists, such as reuptake of neurotransmitter and neuromodulators [22]–[24], modulation of adaptation by means of synaptic feedback via diffusion of molecules such as nitric oxide and carbon monoxide [25]–[27], retrograde axonal transport [28]–[30], action of second messengers [31], [32], and side-effects of synapse-related RNA transcription [33], [34].

Obviously, there is a discrepancy between the backpropagation algorithm and what can actually happen in biological neural systems. This discrepancy needs to be investigated and explained. To this end, we need to investigate whether the feedback network is absolutely necessary for adaptation to take place, although it is required by the backpropagation algorithm. In other words, is it possible to backpropagate errors without using a feedback network? Instead of using a feedback network, can errors be backpropagated by some retrograde mechanism similar to those discovered by neuroscientists? The new algorithm derived in this article gives affirmative answers to the above questions. In fact, the relevant error feedback is implicit in the strengths of axonic connections and their rates of change. We prove formally that the error feedback for any neuron is proportional to the derivative of the sum of the squares of the strengths/weights of the axonic

Manuscript received 4 June 2020; revised 15 August 2020, 16 November 2020, and 28 February 2021; accepted 19 May 2021. Date of publication 28 June 2021; date of current version 1 December 2022.

The author is with the Department of Electrical and Computer Engineering, Wayne State University, Detroit, MI 48202 USA (e-mail: flin@wayne.edu).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TNNLS.2021.3089134.

Digital Object Identifier 10.1109/TNNLS.2021.3089134

[1]The word "algorithm" is used in this article in a generalized sense to mean a model or a mathematical description for generating error feedback and updating synapse/connection strengths/weights.

synapse/connections. Hence, there is no need to construct a separate feedback network using the new algorithm.

The adaptation using the backpropagation algorithm is divided into two phases. The first phase is the feed-forward phase, where the outputs of a neural network for given inputs are calculated. The second phase is the feedback phase, where the errors propagate backward through the feedback network and the weights are adapted. Hence, the backpropagation algorithm requires that neurons and synapses alternate between two distinct feed-forward and feedback phases. However, biological neurons do not have distinct feed-forward and feedback phases and their responses and adaptations occur concurrently. Since no separate feedback network to backpropagate the error is required by the new algorithm, it eliminates the need for two distinct phases. In other words, the new algorithm allows asynchronous and concurrent processing by the neurons. Therefore, the new algorithm explicitly answers another skepticism by neuroscientists.

Furthermore, the adaptation parameter is canceled in all but output neurons using the new algorithm. Hence, the adaptation parameter only needs to be set at the output neurons. Therefore, there is no need to broadcast this parameter to all neurons. In biological neural systems, it is not easy to broadcast the adaptation parameter.

If there is a need to implement neurons on silicon, the new algorithm provides a much simpler implementation than that has been previously considered [35], [36]. In particular, the absence of the feedback network means that adding learning/adaptation to silicon neurons does not involve additional wiring layout complexity among neurons. An adaptive neuron can be designed as a standard unit without considering network topology or adaptation parameters. These adaptive neurons can then be connected in any way required by a particular application. Obviously, this increases the potential for designing neural networks with dynamically reconfigurable topologies.

The new algorithm is not limited to neural networks. With some modifications, it can be applied to other complex man-made systems. To do so, a system is decomposed into interconnected subsystems and adaptation occurs in the interactions [37]. This approach of adaptive interaction has been successfully applied to self-tuning of proportional—integral—differential (PID) controllers [38].

Preliminary results on the new algorithm were published in a conference paper [39] in 1996, before the era of deep learning. We add proofs, new results, perspectives, and references to reflect the current interests in neural networks and deep learning. In particular, omitted proofs are added to the article, and Sections II, III, and V are new.

The article is organized as follows. In Section II, we introduce basic notations of neural networks and recall the conventional backpropagation algorithm. In Section III, we derive our algorithm and compare it with the backpropagation algorithm. We also show the feedback-network-free implementation in MATLAB/Simulink and check the effectiveness of the implementation by simulations. In Section IV, we provide a model for biological neural networks. In Section V, we discuss the biological plausibility of our algorithm and compare it with the existing results in the literature. Conclusions are given in Section VI.

II. Neural Networks and Backpropagation

Let us briefly introduce notations of a neural network consisting of L layers as shown in Fig. 1.

Layer k consists of N_k neurons, $k = 1, 2, \ldots, L$. The outputs (firing rates) of the neurons in Layer k are denoted by $r_i^k, i = 1, 2, \ldots, N_k$. The inputs to the neural network are $r_i^0, i = 1, 2, \ldots, N_0$ and the output from the neural network are $r_i^L, i = 1, 2, \ldots, N_L$, where L is the last (output) layer. The weight between the ith neuron in Layer $k-1$ and the jth neuron in Layer k is denoted by w_{ij}^k. The outputs of neurons in Layer k can be calculated as follows:

$$\begin{aligned} p_j^k &= \sum_{i=1}^{N_{k-1}} w_{ij}^k r_i^{k-1} \\ r_j^k &= \sigma\left(p_j^k\right) \end{aligned} \tag{1}$$

where p_j^k are the membrane potentials of the neurons and $\sigma(x) = 1/(1+e^{-x})$ is the sigmoidal function.

The goal is to learn/adapt the weights w_{ij}^k so that the following least square error is minimized:

$$E = \frac{1}{2}\sum_{n=1}^{N_L}\left(r_n^L - \bar{r}_n^L\right)^2 \tag{2}$$

where $\bar{r}_n^L$ is the desired/target output of the nth neuron in the last (output) layer.

Next, let us briefly recall the conventional backpropagation algorithm [17]–[19] (abbreviated as C-B algorithm in the rest of the article). To reduce the error, we first calculate the derivatives

$$\frac{dE}{dw_{ij}^k}.$$

The adaptation is then achieved by letting

$$\dot{w}_{ij}^k = -\alpha \frac{dE}{dw_{ij}^k}$$

where $\alpha > 0$ is the adaptation parameter that can be adjusted during the adaptation.

For $k = L$, that is, the last layer of the neural network

$$\begin{aligned} \frac{dE}{dw_{ij}^L} &= \frac{dE}{dr_j^L}\frac{dr_j^L}{dp_j^L}\frac{dp_j^L}{dw_{ij}^L} \\ &= \left(r_j^L - \bar{r}_j^L\right)\sigma'\left(p_j^L\right) r_i^{L-1} \end{aligned}$$

where

$$\sigma'(x) = \left(\frac{1}{1+e^{-x}}\right)' = \sigma(x)(1-\sigma(x)).$$

Hence

$$\frac{dE}{dw_{ij}^L} = \left(r_j^L - \bar{r}_j^L\right) r_j^L \left(1 - r_j^L\right) r_i^{L-1}.$$

Define the error signal as

$$\phi_j^L = \left(r_j^L - \bar{r}_j^L\right) r_j^L \left(1 - r_j^L\right).$$

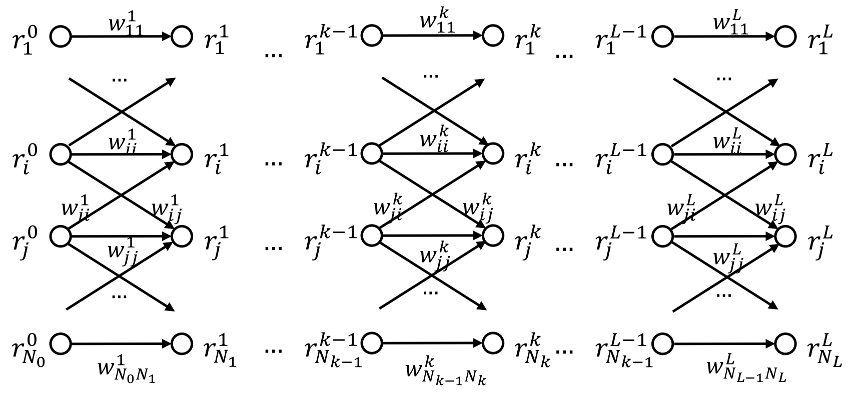

Fig. 1. Neural network with L layers. Layer k consists of N_k neurons, $k = 1, 2, \ldots, L$. The inputs to the neural network are $r_i^0, i = 1, 2, \ldots, N_0$ and the output from the neural network are $r_i^L, i = 1, 2, \ldots, N_L$, where L is the last (output) layer.

Then

$$\frac{dE}{dw_{ij}^L} = \phi_j^L r_i^{L-1}.$$

For other layers $k = L-1, \ldots, 2, 1$

$$\begin{aligned}\frac{dE}{dw_{ij}^k} &= \frac{dE}{dr_j^k}\frac{dr_j^k}{dp_j^k}\frac{dp_j^k}{dw_{ij}^k}\\ &= \frac{dE}{dr_j^k} r_j^k\left(1-r_j^k\right) r_i^{k-1}.\end{aligned}$$

Define the error signal as

$$\phi_j^k = \frac{dE}{dr_j^k} r_j^k\left(1-r_j^k\right).$$

Then

$$\frac{dE}{dw_{ij}^k} = \phi_j^k r_i^{k-1}.$$

The error signal ϕ_j^k can be calculated recursively as

$$\phi_j^k = r_j^k\left(1-r_j^k\right)\sum_{n=1}^{N_{k+1}} \phi_n^{k+1} w_{jn}^{k+1}.$$

Therefore, the C-B algorithm can be summarized as follows:

$$\dot{w}_{ij}^k = -\alpha\frac{dE}{dw_{ij}^k} = -\alpha\phi_j^k r_i^{k-1} \tag{3}$$

where

$$\begin{aligned}\phi_j^L &= r_j^L\left(1-r_j^L\right)\left(r_j^L-\bar{r}_j^L\right), \quad L \text{ is the last layer}\\ \phi_j^k &= r_j^k\left(1-r_j^k\right)\sum_{n=1}^{N_{k+1}} \phi_n^{k+1} w_{jn}^{k+1}, \quad k = L-1, \ldots, 1.\end{aligned} \tag{4}$$

Note that a feedback network is needed to backpropagate the error signal from ϕ_n^{k+1} to ϕ_j^k. The weights of the feedback network must be kept the same as the feed-forward network because w_{ij}^k appears in both (1) and (4).

III. Algorithm and Implementation Without Feedback Network

For efficient implementation and biological plausibility, let us derive the following equivalent learning algorithm (abbreviated as B-L algorithm in the rest of the article) first proposed in [39]. By (3)

$$\dot{w}_{jn}^{k+1} = -\alpha r_j^k \phi_n^{k+1} \quad\Rightarrow\quad \phi_n^{k+1} = \frac{\dot{w}_{jn}^{k+1}}{-\alpha r_j^k}.$$

Hence, for $k = L-1, \ldots, 1$

$$\begin{aligned}\dot{w}_{ij}^k &= -\alpha r_i^{k-1}\phi_j^k\\ &= -\alpha r_i^{k-1} r_j^k\left(1-r_j^k\right)\sum_{n=1}^{N_{k+1}} \phi_n^{k+1} w_{jn}^{k+1}\\ &\quad [\text{by } (4)]\\ &= -\alpha r_i^{k-1} r_j^k\left(1-r_j^k\right)\sum_{n=1}^{N_{k+1}} \frac{\dot{w}_{jn}^{k+1}}{-\alpha r_j^k} w_{jn}^{k+1}\\ &= r_i^{k-1}\left(1-r_j^k\right)\sum_{n=1}^{N_{k+1}} \dot{w}_{jn}^{k+1} w_{jn}^{k+1}.\end{aligned}$$

Therefore, the B-L algorithm can be summarized as follows:

$$\begin{aligned}\dot{w}_{ij}^L &= -\alpha r_i^{L-1} r_j^L\left(1-r_j^L\right)\left(r_j^L-\bar{r}_j^L\right), \quad L \text{ is the last layer}\\ \dot{w}_{ij}^k &= r_i^{k-1}\left(1-r_j^k\right)\sum_{n=1}^{N_{k+1}} \dot{w}_{jn}^{k+1} w_{jn}^{k+1}, \quad k = L-1, \ldots, 1.\end{aligned} \tag{5}$$

The above B-L algorithm has the following properties.

1) Since the B-L algorithm is derived mathematically from the C-B algorithm, two algorithms are mathematically equivalent. In other words, the B-L algorithm can be used wherever the C-B algorithm can be used. The performance of the B-L algorithm will be as good/bad as the C-B algorithm.
2) The implementation of the B-L algorithm does not require a feedback network whose topology and weights must be kept the same as the feed-forward network. A feedback-network-free implementation is given below.

3) Because of the removal of this requirement, adaptation according to the B-L algorithm is much more plausible in biological neural systems as to be further discussed in Section V. This is because it is unlikely that the requirement of feedback network can be met in biological neural systems.
4) Comparing (3) with (5), we observe that the relevant error feedback is implicit in the weights (w_{ij}^k) and their rates of change ($\dot{w}_{ij}^k$). More precisely, the appropriate error feedback for any neuron is proportional to the derivative of the sum of the squares of the strengths of the axonic connections, because $(\sum_{n=1}^{N_{k+1}}(w_{jn}^{k+1})^2)' = 2\sum_{n=1}^{N_{k+1}}\dot{w}_{jn}^{k+1}w_{jn}^{k+1}$.
5) Since the B-L algorithm does not require a separate feedback network, it eliminates the need for having two separate phases: 1) a feed-forward phase when the feed-forward network updates its variables and 2) a feedback phase when the feedback network updates its variables. In other words, adaptation can be performed in a phaseless fashion by processing information asynchronously and concurrently.
6) The adaptation parameter α appears only at the last layer. That means α can be easily adjusted during the adaptation.
7) As to be shown next, all layers of the neural network have the same structure, except the last layer, which is slightly different. If all layers have the same number of neurons, then all layers, except the last layer, are identical. Hence, only two types of implementation blocks need to be built: one for the last layer and one for the other layers. These blocks can then be easily connected together in Simulink and other implementation platforms.
8) If there is a need to implement neurons on silicon, the B-L algorithm provides a much simpler implementation. The elimination of the feedback network significantly reduces wiring layout complexity among neurons, as there is no need to implement a separate feedback network and then connect neurons in the feedback network with the corresponding neurons in the feed-forward network.
9) Using the B-L algorithm, an adaptive neuron can be designed as an identical and standard unit without considering network topology or adaptation parameter α. These adaptive neurons can then be interconnected arbitrarily. This is very useful because neural networks often have different topologies for different applications. The B-L algorithm also provides the potential for designing neural networks with dynamically reconfigurable topologies.
10) Since all feedback and connections are local, implementations using the B-L algorithm are more fault-tolerant in the sense that failures of some neurons will not cause the entire neural network to become nonfunctional.

An implementation of the B-L algorithm in Simulink is shown in Fig. 2. The main purpose of the implementation and simulations is to show that the B-L algorithm can be implemented without a feedback network. No attempt is made to claim the superior performance of the B-L algorithm, as its performance is the same as that of the C-B algorithm.

In Fig. 2, the neural network with fixed layers (at the top of the figure) are used to generate the desired/target outputs. In other words, it implements

$$\bar{p}_j^k = \sum_{m=1}^{N_{k-1}} \bar{w}_{mj}^k \bar{r}_m^{k-1}$$
$$\bar{r}_j^k = \sigma\left(\bar{p}_j^k\right)$$

where the weights $\bar{w}_{ij}^k$ are selected randomly. The neural network with adaptive layers (at the bottom of the figure) is the neural network where we apply the B-L algorithm. In other words, it implements

$$\dot{w}_{ij}^L = -\alpha r_i^{L-1} r_j^L\left(1 - r_j^L\right)\left(r_j^L - \bar{r}_j^L\right), \quad L \text{ is the last layer}$$
$$\dot{w}_{ij}^k = r_i^{k-1}\left(1 - r_j^k\right)\sum_{n=1}^{N_{k+1}} \dot{w}_{jn}^{k+1} w_{jn}^{k+1}, \quad k = L-1, \ldots, 1.$$

The adaptive neural network learns their weights w_{ij}^k so that the error

$$E = \frac{1}{2}\sum_{n=1}^{N_L}\left(r_n^L - \bar{r}_n^L\right)^2$$

is reduced. In a sense, the fixed neural network generates a training set for the adaptive neural network to learn. To do so, the same input signals are applied to both the fixed neural network and the adaptive neural network, that is, $r_n^0 = \bar{r}_n^0$, $n = 1, 2, \ldots, N_0$. We simulate the neural networks using different input signals (square, sawtooth, and sinusoidal) and find that they all work well.

Note that 1) Adaptive Layer 1 is identical to Adaptive Layer 2, and is slightly different than Adaptive Layer 3 (the output layer); 2) there is not feedback network to backpropagate the errors; and 3) the adaptation parameter α only appears in the output layer.

We run several simulations to test the effectiveness of the B-L algorithm. A typical simulation is as follows. Let

$$L = 3$$
$$N_1 = N_2 = N_3 = 3.$$

We select $\bar{w}_{ij}^k$ with $k = 1, 2, 3$ and $i, j = 1, 2, 3$ randomly as

$$\begin{bmatrix} \bar{w}_{11}^1 & \bar{w}_{12}^1 & \bar{w}_{13}^1 \\ \bar{w}_{21}^1 & \bar{w}_{22}^1 & \bar{w}_{23}^1 \\ \bar{w}_{31}^1 & \bar{w}_{32}^1 & \bar{w}_{33}^1 \end{bmatrix} = \begin{bmatrix} -3.8880 & -2.5831 & -3.6803 \\ 2.8025 & -0.9609 & 4.4205 \\ -1.1026 & -4.0355 & 4.5613 \end{bmatrix}$$
$$\begin{bmatrix} \bar{w}_{11}^2 & \bar{w}_{12}^2 & \bar{w}_{13}^2 \\ \bar{w}_{21}^2 & \bar{w}_{22}^2 & \bar{w}_{23}^2 \\ \bar{w}_{31}^2 & \bar{w}_{32}^2 & \bar{w}_{33}^2 \end{bmatrix} = \begin{bmatrix} -0.8273 & 4.4479 & -1.6228 \\ -4.5035 & -0.0914 & 4.0005 \\ 4.0272 & -0.1075 & -1.3075 \end{bmatrix}$$
$$\begin{bmatrix} \bar{w}_{11}^3 & \bar{w}_{12}^3 & \bar{w}_{13}^3 \\ \bar{w}_{21}^3 & \bar{w}_{22}^3 & \bar{w}_{23}^3 \\ \bar{w}_{31}^3 & \bar{w}_{32}^3 & \bar{w}_{33}^3 \end{bmatrix} = \begin{bmatrix} 1.2206 & -0.9819 & -3.7668 \\ -1.4905 & -4.2403 & -3.1609 \\ 0.1325 & -2.6008 & -2.6005 \end{bmatrix}.$$

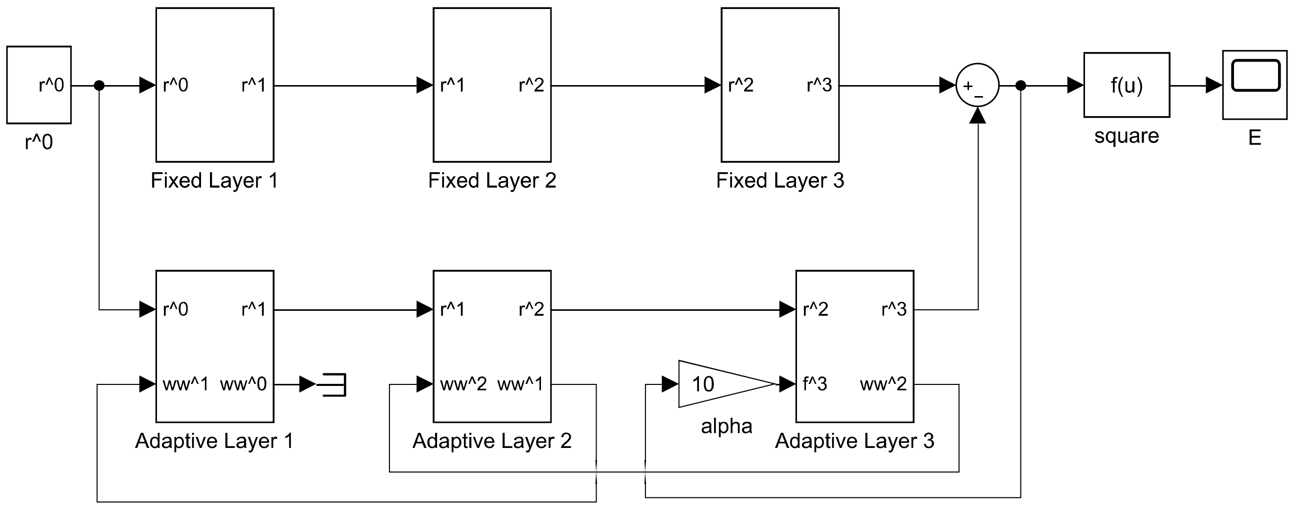

Fig. 2. Simulink implementation of the B-L algorithm without feedback network. The neural network at the top generates a training set for the neural network at the bottom to learn.

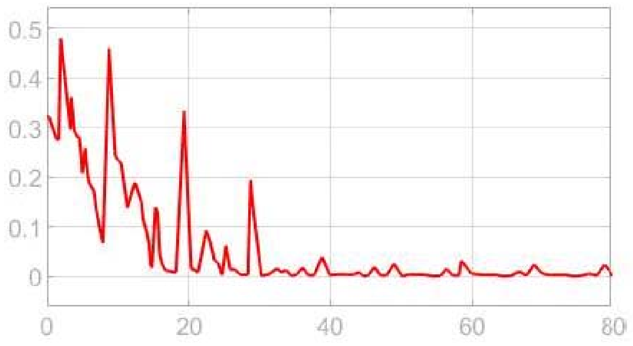

Fig. 3. Error $E = (1/2)\sum_{n=1}^{N_L}(r_n^L - \bar{r}_n^L)^2$ reduces during the simulation using the B-L algorithm for supervised learning. The horizontal axis shows the simulation time (seconds).

The initial weights (before learning) for w_{ij}^k with $k = 1, 2, 3$ and $i, j = 1, 2, 3$ are also selected randomly as

$$\begin{bmatrix} w_{11}^1 & w_{12}^1 & w_{13}^1 \\ w_{21}^1 & w_{22}^1 & w_{23}^1 \\ w_{31}^1 & w_{32}^1 & w_{33}^1 \end{bmatrix} = \begin{bmatrix} 2.3172 & 1.4775 & -0.4908 \\ 0.4701 & -2.0368 & 2.4469 \\ -3.1104 & 1.8678 & -3.1649 \end{bmatrix}$$

$$\begin{bmatrix} w_{11}^2 & w_{12}^2 & w_{13}^2 \\ w_{21}^2 & w_{22}^2 & w_{23}^2 \\ w_{31}^2 & w_{32}^2 & w_{33}^2 \end{bmatrix} = \begin{bmatrix} 0.7521 & -4.4022 & -2.6522 \\ -1.4684 & 3.2119 & -4.8460 \\ -4.5698 & -3.3101 & 1.4912 \end{bmatrix}$$

$$\begin{bmatrix} w_{11}^3 & w_{12}^3 & w_{13}^3 \\ w_{21}^3 & w_{22}^3 & w_{23}^3 \\ w_{31}^3 & w_{32}^3 & w_{33}^3 \end{bmatrix} = \begin{bmatrix} -0.4115 & 4.6309 & 0.4681 \\ 0.2114 & -2.6841 & -0.1110 \\ 1.2406 & 1.7914 & -1.0448 \end{bmatrix}.$$

A typical run is shown in Figs. 3 and 4. Fig. 3 shows that the error reduces as expected. Fig. 4 shows the convergence of weights in the adaptive layers.

In the simulation, the adaptation parameter is set at $\alpha = 10$. We do not make a systematic effort to search for the optimal value for α. Rather, we check several different values for α and then use the value that is reasonable.

IV. Biological Neural Network Model

Since biological neural networks are not strictly layered, in this section, we propose a neural network model, in which neurons can be interconnected arbitrarily. In particular, it can model both hierarchical neural networks and nonhierarchical neural networks. To best represent the configuration and interconnection of a neural network, we use the following notation (see Fig. 5).

n	is the label for a particular neuron;
s	is the label for a particular synapse;
D_n	is the set of dendritic synapses of neuron, n;
A_n	is the set of (feedback-generating) axonic synapses of neuron, n;
pre_s	is the presynaptic neuron corresponding to synapse, s;
post_s	is the postsynaptic neuron corresponding to synapse, s; and
ϵ_s	indicates whether synapse, s, is excitatory (+1) or inhibitory (−1).

Therefore, for each synapse s, the triplet $(\text{pre}_s, \text{post}_s, \epsilon_s)$ specifies, respectively, the presynaptic neuron, the postsynaptic neuron, and whether the synapse is excitatory or inhibitory.

We use subscripts to associate variables with a particular synapse or neuron:

w_s	is the strength of synapse, s;
p_n	is the membrane potential of neuron, n; and
r_n	is the firing rate (output) of neuron, n.

A neuron transmits a signal from dendrites to the axon. This signal flow involves the generation of postsynaptic potentials following activation of a synapse by neurotransmitters, spatial and temporal integration of postsynaptic potentials at the soma, triggering of action potentials along the axon hillock, propagation of action potentials along the axon, and the release of neurotransmitters from presynaptic terminals.

The firing rate r_n of a neuron can be defined as the reciprocal of the most recent interspike interval. The synapse strength w_s

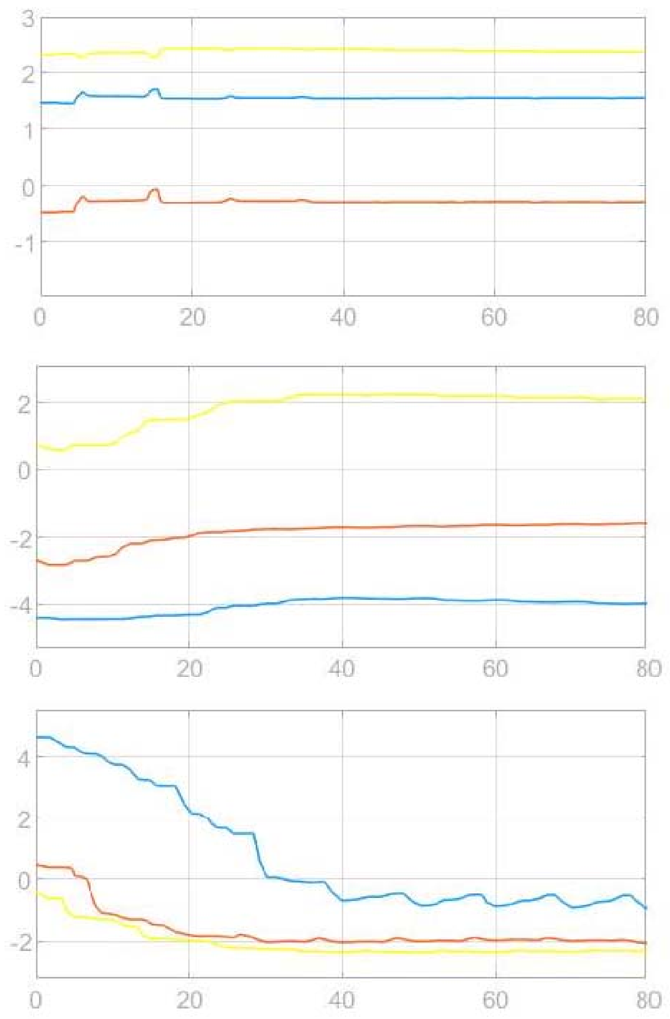

Fig. 4. Figures show the adaptation of three weights (out of nine weights) in each layer during a typical simulation run. The horizontal axis shows the simulation time (seconds).

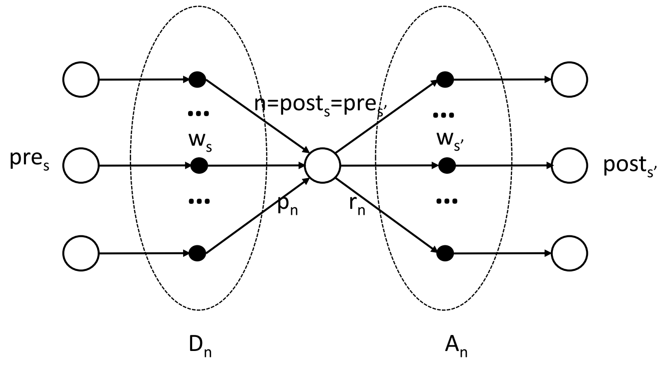

Fig. 5. Labels for neurons and synapses in a neural network. Circles denote neurons and dots denote synapses. Arrows denote information flows.

of a synapse is assumed to be proportional to the number of neurotransmitters released when a spike arrives at the synapse. It is further assumed that the short-time average of the postsynaptic potential is proportional to the product of the synapse strength w_s and the presynaptic firing rate r_{pre_s}. The membrane potential at the axon hillock p_n is thought to be an attenuated sum of the postsynaptic potentials

$$p_n = \sum_{s\in D_n} \epsilon_s \eta_s w_s r_{\text{pre}_s} - \theta_n \tag{6}$$

where

η_s is the decremental conduction coefficient for synapse [40], s; and
θ_n is the offset of the membrane potential of neuron [41], n.

The parameter η_s represents the attenuation of the postsynaptic potential that is primarily a consequence of the distance between the synapse and the soma [42]. The firing rate r_n of the neuron is assumed to be a sigmoidal function of its membrane potential

$$r_n = \rho_n \sigma(\lambda_n p_n) \tag{7}$$

where

ρ_n is the maximal firing rate of neuron, n; and
λ_n is the measure of the steepness of the sigmoidal characteristic of neuron, n.

For neurons to adapt, we assume that some sort of direct feedback is available to some neurons. These neurons can be viewed as output neurons.

As before, the goal is to minimize an error E. One such error is the least square error given in (2). For general biological neural networks, we do not put constraints on the forms of errors. To minimize a general error E, we would like to set

$$\dot{w}_s = -\alpha_s \frac{\gamma}{\eta_s} \frac{dE}{dw_s}$$

where

$\alpha_s > 0$ is the postsynaptic adaptation parameter for synapse, s; and
$\gamma > 0$ is the direct feedback coefficient for all neurons.

A. Hierarchical Neural Networks

Denote the set of all neurons in a neural network by $\mathcal{N} = \{n_i : i = 1, 2, \ldots, K\}$, where K is the number of neurons. In a hierarchical neural network, we can find a partial order $<$ on $\mathcal{N}$ such that if $n_1 < n_2$, then there exist no axonic synapses of n_2 with n_1, that is, $D_{n_1} \cap A_{n_2} = \emptyset$. This order ensures that the output of any neuron n is an explicit function of the outputs of some preceding neurons (neurons n' satisfying $n' < n$). Hence, the output of each neuron is always well-defined and unique.

The following learning algorithm, requiring no feedback network, provides a way to minimize the error.

Theorem 1: Consider a hierarchical neural network, whose neuron have firing rates satisfy the equilibrium (6) and (7). Consider the following adaptation law:

$$\dot{w}_s = \alpha_s \epsilon_s r_{\text{pre}_s} \rho_{\text{post}_s} \lambda_{\text{post}_s} \sigma'(\lambda_{\text{post}_s} p_{\text{post}_s}) \times \left(\frac{\varphi_{\text{post}_s}}{r_{\text{post}_s}} - \gamma \frac{\partial E}{\partial r_{\text{post}_s}} \right) \tag{8}$$

where[2]

$$\varphi_{\text{post}_s} = \sum_{s' \in A_{\text{post}_s}} \frac{\eta_{s'}}{\alpha_{s'}} w_{s'} \dot{w}_{s'}. \tag{9}$$

This adaptation law reduces the error E monotonically because the following equation is satisfied:

$$\dot{w}_s = -\alpha_s \frac{\gamma}{\eta_s} \frac{dE}{dw_s}. \tag{10}$$

Proof: To prove the theorem, it is sufficient to show that

$$\dot{w}_s = -\alpha_s \frac{\gamma}{\eta_s} \frac{dE}{dw_s}$$

satisfies the following equation which combines (8) and (9):

$$\dot{w}_s = \alpha_s \epsilon_s r_{\text{pre}_s} \rho_{\text{post}_s} \lambda_{\text{post}_s} \sigma'(\lambda_{\text{post}_s} p_{\text{post}_s}) \times \left(\frac{1}{r_{\text{post}_s}} \sum_{s' \in A_{\text{post}_s}} \frac{\eta_{s'}}{\alpha_{s'}} w_{s'} \dot{w}_{s'} - \gamma \frac{\partial E}{\partial r_{\text{post}_s}} \right). \tag{11}$$

Indeed, for any synapse s

$$\frac{dE}{dw_s} = \frac{dE}{dr_{\text{post}_s}} \frac{dr_{\text{post}_s}}{dp_{\text{post}_s}} \frac{dp_{\text{post}_s}}{dw_s}.$$

By (7)

$$\frac{dr_{\text{post}_s}}{dp_{\text{post}_s}} = \rho_{\text{post}_s} \lambda_{\text{post}_s} \sigma'(\lambda_{\text{post}_s} p_{\text{post}_s}). \tag{12}$$

By (6)

$$\frac{dp_{\text{post}_s}}{dw_s} = \epsilon_s \eta_s r_{\text{pre}_s}.$$

Hence,

$$\begin{aligned} \frac{dE}{dw_s} &= \frac{dE}{dr_{\text{post}_s}} \frac{dr_{\text{post}_s}}{dp_{\text{post}_s}} \frac{dp_{\text{post}_s}}{dw_s} \\ &= \frac{dE}{dr_{\text{post}_s}} \rho_{\text{post}_s} \lambda_{\text{post}_s} \sigma'(\lambda_{\text{post}_s} p_{\text{post}_s}) \epsilon_s \eta_s r_{\text{pre}_s}. \end{aligned} \tag{13}$$

On the other hand, for any neuron n

$$\begin{aligned} \frac{dE}{dr_n} &= \frac{\partial E}{\partial r_n} + \sum_{s' \in A_n} \frac{dE}{dr_{\text{post}_{s'}}} \frac{dr_{\text{post}_{s'}}}{dr_n} \\ &= \frac{\partial E}{\partial r_n} + \sum_{s' \in A_n} \frac{dE}{dr_{\text{post}_{s'}}} \frac{dr_{\text{post}_{s'}}}{dp_{\text{post}_{s'}}} \frac{dp_{\text{post}_{s'}}}{dr_n}. \end{aligned}$$

By (6) and because for $s' \in A_n$

$$\frac{dp_{\text{post}_{s'}}}{dr_n} = \epsilon_{s'} \eta_{s'} w_{s'}.$$

By the above equation and (12) with s' replacing s, we have

$$\begin{aligned} \frac{dE}{dr_n} &= \frac{\partial E}{\partial r_n} + \sum_{s' \in A_n} \frac{dE}{dr_{\text{post}_{s'}}} \\ &\quad \times \rho_{\text{post}_{s'}} \lambda_{\text{post}_{s'}} \sigma'(\lambda_{\text{post}_{s'}} p_{\text{post}_{s'}}) \epsilon_{s'} \eta_{s'} w_{s'} \\ &= \frac{\partial E}{\partial r_n} + \sum_{s' \in A_n} \frac{dE}{dr_{\text{post}_{s'}}} \\ &\quad \times \rho_{\text{post}_{s'}} \lambda_{\text{post}_{s'}} \sigma'(\lambda_{\text{post}_{s'}} p_{\text{post}_{s'}}) \epsilon_{s'} \eta_{s'} r_{\text{pre}_{s'}} \frac{w_{s'}}{r_{\text{pre}_{s'}}}. \end{aligned}$$

By (13) with s' replacing s

$$\frac{dE}{dr_n} = \frac{\partial E}{\partial r_n} + \sum_{s' \in A_n} \frac{dE}{dw_{s'}} \frac{w_{s'}}{r_{\text{pre}_{s'}}}.$$

Let $n = \text{post}_s$ in the above equation and substitute it into (13), we have

$$\frac{dE}{dw_s} = \rho_{\text{post}_s} \lambda_{\text{post}_s} \sigma'(\lambda_{\text{post}_s} p_{\text{post}_s}) \epsilon_s \eta_s r_{\text{pre}_s} \times \left(\frac{\partial E}{\partial r_{\text{post}_s}} + \sum_{s' \in A_{\text{post}_s}} \frac{dE}{dw_{s'}} \frac{w_{s'}}{r_{\text{pre}_{s'}}} \right). \tag{14}$$

By (10)

$$\begin{aligned} \frac{dE}{dw_s} &= \frac{-\dot{w}_s \eta_s}{\gamma \alpha_s} \\ \frac{dE}{dw_{s'}} &= \frac{-\dot{w}_{s'} \eta_{s'}}{\gamma \alpha_{s'}}. \end{aligned}$$

Substitute the above into (14), we have

$$\frac{-\dot{w}_s \eta_s}{\gamma \alpha_s} = \rho_{\text{post}_s} \lambda_{\text{post}_s} \sigma'(\lambda_{\text{post}_s} p_{\text{post}_s}) \epsilon_s \eta_s r_{\text{pre}_s} \times \left(\frac{\partial E}{\partial r_{\text{post}_s}} + \sum_{s' \in A_{\text{post}_s}} \frac{-\dot{w}_{s'} \eta_{s'}}{\gamma \alpha_{s'}} \frac{w_{s'}}{r_{\text{pre}_{s'}}} \right).$$

Hence,

$$\dot{w}_s = \alpha_s \rho_{\text{post}_s} \lambda_{\text{post}_s} \sigma'(\lambda_{\text{post}_s} p_{\text{post}_s}) \epsilon_s r_{\text{pre}_s} \times \left(-\gamma \frac{\partial E}{\partial r_{\text{post}_s}} + \sum_{s' \in A_{\text{post}_s}} \frac{\dot{w}_{s'} \eta_{s'}}{\alpha_{s'}} \frac{w_{s'}}{r_{\text{pre}_{s'}}} \right).$$

Since $s' \in A_{\text{post}_s} \Rightarrow r_{\text{pre}_{s'}} = r_{\text{post}_s}$,

$$\dot{w}_s = \alpha_s \epsilon_s r_{\text{pre}_s} \rho_{\text{post}_s} \lambda_{\text{post}_s} \sigma'(\lambda_{\text{post}_s} p_{\text{post}_s}) \times \left(\frac{1}{r_{\text{post}_s}} \sum_{s' \in A_{\text{post}_s}} \frac{\eta_{s'}}{\alpha_{s'}} w_{s'} \dot{w}_{s'} - \gamma \frac{\partial E}{\partial r_{\text{post}_s}} \right).$$

This is (11). For hierarchical neural networks, the solution to (11) exists and is unique (see Proposition 2). Therefore, the solution is given by (10), that is, (10) is satisfied. ■

For output neurons, the adaptation law of (8) becomes

$$\dot{w}_s = -\gamma \alpha_s \epsilon_s r_{\text{pre}_s} \rho_{\text{post}_s} \lambda_{\text{post}_s} \sigma'(\lambda_{\text{post}_s} p_{\text{post}_s}) \frac{\partial E}{\partial r_{\text{post}_s}}.$$

For hidden neurons, the adaptation law of (8) becomes

$$\dot{w}_s = \alpha_s \epsilon_s r_{\text{pre}_s} \rho_{\text{post}_s} \lambda_{\text{post}_s} \sigma'(\lambda_{\text{post}_s} p_{\text{post}_s}) \frac{\varphi_{\text{post}_s}}{r_{\text{post}_s}}.$$

Note that, if the adaptation parameter is the same for all synapses, then it only needs to be set at the output neurons as shown in the following corollary.

Corollary 1: If $\alpha_s = \alpha$ is the same for all synapses, then (8) and (9) reduce to

$$\dot{w}_s = \epsilon_s r_{\text{pre}_s} \rho_{\text{post}_s} \lambda_{\text{post}_s} \sigma'(\lambda_{\text{post}_s} p_{\text{post}_s}) \left(\frac{\varphi_{\text{post}_s}}{r_{\text{post}_s}} - \alpha\gamma \frac{\partial E}{\partial r_{\text{post}_s}} \right)$$

[2]We use $(\partial E/\partial r_n)$ to denote the direct contribution of r_n to E and (dE/dr_n) to denote the total contribution of r_n to E.

where

$$\varphi_{\text{post}_s} = \sum_{s' \in A_{\text{post}_s}} \eta_{s'} w_{s'} \dot{w}_{s'}.$$

Note that, if there is no direct feedback signal for a particular neuron post_s, that is, $(\partial E/\partial r_{\text{post}_s}) = 0$, then the adaptation parameter α is absent.

B. Nonhierarchical Neural Networks

We now consider nonhierarchical neural networks. Unlike hierarchical neural networks, in a nonhierarchical neural network, there exists no partial order $<$ on $\mathcal{N}$ such that if $n_1 < n_2$, then $D_{n_1} \cap A_{n_2} = \emptyset$. Hence, the outputs of some neurons are implicit functions of the outputs of some other neurons. To ensure that the output of each neuron is well-defined and unique, we assume that the neural network converges to stable states, that is, the equilibrium (6) and (7) have at least one fixed point that is a stable attractor.

Let us label the neurons by $1, \ldots, K$ and the synapses by $1, \ldots, S$. We would like to show that the adaptation law given by (8) and (9) can be used for nonhierarchical networks as well. Since the network is nonhierarchical, (8) and (9) give only implicit functions of $\dot{w}_s$. To ensure the existence of these implicit functions, certain conditions must be satisfied, as stated in the following proposition.

Proposition 1: The synaptic adaptations $\{\dot{w}_s : s \in \{1, \ldots, S\}\}$ is implicitly defined by (8) and (9) if and only if the following Jacobian determinant is nonzero:

$$\det(\delta_{ss'} - g_{ss'}) \neq 0 \tag{15}$$

where $[\delta_{ss'} - g_{ss'}]$ is an $S \times S$ matrix with $s, s' \in \{1, \ldots, S\}$ and

$$\delta_{ss'} = \begin{cases} 1, & \text{if } s = s' \\ 0, & \text{otherwise.} \end{cases}$$

$$g_{ss'} = \begin{cases} \alpha_s \epsilon_s r_{\text{pre}_s} \rho_{\text{post}_s} \lambda_{\text{post}_s} & \\ \times\, \sigma'(\lambda_{\text{post}_s} p_{\text{post}_s}) \dfrac{\eta_{s'} w_{s'}}{r_{\text{post}_s} \alpha_{s'}}, & \text{if } s' \in A_{\text{post}_s} \\ 0, & \text{otherwise.} \end{cases}$$

Proof: For $s \in \{1, \ldots, S\}$, combine (8) and (9) as

$$\dot{w}_s = \alpha_s \epsilon_s r_{\text{pre}_s} \rho_{\text{post}_s} \lambda_{\text{post}_s} \sigma'(\lambda_{\text{post}_s} p_{\text{post}_s}) \times \left(\sum_{s' \in A_{\text{post}_s}} \frac{\eta_{s'} w_{s'}}{r_{\text{post}_s} \alpha_{s'}} \dot{w}_{s'} - \gamma \frac{\partial E}{\partial r_{\text{post}_s}} \right). \tag{16}$$

Write the S linear (in terms of $\dot{w}_s$) equations in the matrix form

$$[\delta_{ss'} - g_{ss'}][\dot{w}_s] = [b]$$

where $[\dot{w}_s]$ is an $S \times 1$ vector of $\dot{w}_s$ and $[b]$ is an $S \times 1$ vector representing the terms in (16) that are not related to $\dot{w}_s$. Then, clearly, $[\dot{w}_s]$ is uniquely defined if and only of $[\delta_{ss'} - g_{ss'}]$ is invertible, that is, Condition (15) is satisfied. ■

For hierarchical networks, we have the following proposition.

Proposition 2: Condition (15) is always satisfied in a hierarchical network.

Proof: For a hierarchical network, there is a partial order $<$ on the set of all neurons $\mathcal{N} = \{n_i : i = 1, 2, \ldots K\}$ such that if $n_1 < n_2$, then $D_{n_1} \cap A_{n_2} = \emptyset$.

In Proposition 1 and its proof, arrange (16) so that if $s' < s$, then $\text{post}_{s'} < \text{post}_s$. Clearly,

$$\text{post}_{s'} < \text{post}_s \Rightarrow D_{\text{post}_{s'}} \cap A_{\text{post}_s} = \emptyset \Rightarrow s' \notin A_{\text{post}_s}.$$

By the definition of $g_{ss'}$, the elements below the diagonal in $[\delta_{ss'} - g_{ss'}]$ are all zeros. Since all diagonal elements in $[\delta_{ss'} - g_{ss'}]$ are ones, $\det(\delta_{ss'} - g_{ss'}) = 1$, that is, Condition (15) is satisfied. ■

When Condition (15) is satisfied, the adaptation law given by (8) and (9) minimize the error as stated below.

Theorem 2: Consider a (nonhierarchical) neural network which has reached an equilibrium described by (6) and (7). Assume that Condition (15) is satisfied. If synapse strengths of the network adapt according to (8) and (9), then the error E will decrease monotonically with time. In fact, (10) is satisfied by all synapses.

Proof: The proof is the same as the proof of Theorem 1, except that the existence and uniqueness of the solution to (11) is guaranteed by Condition (15). ■

If Condition (15) is not satisfied for the entire set of synapses, we can find a (largest) subset of synapses that satisfies Condition (15) as stated in the following proposition.

Proposition 3: A subset of the synaptic adaptations $\{\dot{w}_s : s \in \{1, \ldots, M\}\}$ is implicitly defined as a function of the remaining synaptic adaptations $\{\dot{w}_s : s \in \{M+1, \ldots, S\}\}$ by (8) and (9) if and only if Condition (15) is satisfied for the subset $\{\dot{w}_s : s \in \{1, \ldots, M\}\}$.

Proof: The proof is the same as that of Proposition 1 with $\{\dot{w}_s : s \in \{1, \ldots, M\}\}$ replacing $\{\dot{w}_s : s \in \{1, \ldots, S\}\}$. ■

The adaptation law is then given in the following theorem.

Theorem 3: Consider a (nonhierarchical) neural network which has reached an equilibrium described by (6) and (7). Assume that Condition (15) is satisfied for the subset $\{\dot{w}_s : s \in \{1, \ldots, M\}\}$. If synapse strengths of the network adapt according to (8) and (9) for all $s \in \{1, \ldots, M\}$ and according to (10) for all $s \in \{M+1, \ldots, S\}$, then the error E will decrease monotonically with time. In fact, (10) is satisfied by all synapses.

Proof: For synapses $s \in \{M+1, \ldots, S\}$, (10) is satisfied by the definition. For synapses $s \in \{1, \ldots, M\}$, since Condition (15) is satisfied, $\dot{w}_s$ given by (10) is the unique solutions to (8) and (9). Hence, (10) is also satisfied. ■

V. Biological Plausibility

The problem of the biological plausibility of backpropagation has been investigated before. In this section, we first review some literature works on this topic. A good reference on this topic is a recent survey article [43], where a critical review of the state-of-the-art learning algorithms for spiking neural networks using single and multiple spikes is presented; and challenges and opportunities in the spiking neural network field are discussed. A recent special issue in artificial neural networks as models of neural information processing [44] also explores the use of artificial neural networks in the context of computational neuroscience from various perspectives.

In [45], equilibrium propagation is introduced that involves only one kind of neural computation, performed in both the first phase (when the prediction is made) and the second phase of training (after the target or prediction error is revealed). The algorithm computes the gradient of an objective function just like backpropagation but does not need a special computation or circuit for the second phase, where errors are implicitly propagated. Although their algorithm is significantly different from the B-L algorithm, their objective of removing feedback network and two-phase computations is similar to that of the B-L algorithm.

In [46], a new biologically plausible learning algorithm for recurrent networks, guided solely by delayed and phasic rewards at the end of each trial is proposed. This algorithm overcomes the shortcomings of other algorithms that are either biologically implausible and/or require a continuous, real-time error signal to guide learning. Note that the B-L algorithm also does not require a continuous, real-time error signal to guide learning.

Scellier and Bengio [47] showed how iterative inference can backpropagate error signals. Neurons move their activations toward configurations corresponding to lower energy and smaller prediction error. A new observation creates a perturbation at visible neurons that propagate into hidden layers, with these propagated perturbations corresponding to the backpropagated gradient. In comparison, the B-L algorithm directly backpropagate gradients via local feedback.

Mazzoni *et al.* [48] questioned the biological plausibility of backpropagation because "it still requires unrealistic conditions, such as symmetrical feedback pathways that are identical in every way, including the strength of the individual synaptic connections." They then propose to use a variant of the associative reward-penalty to the training of a multilayer neural network in a biologically relevant supervised learning task. As discussed in Section III, the B-L algorithm eliminates the need for such symmetrical feedback pathways.

Stork [49] discussed the biological plausibility of backpropagation, examine the local implementation of backpropagation, incorporating facts of neurobiology that are not controversial, and point to hypotheses that are not yet confirmed by neurobiology. The author then concludes that "there remain several unanswered questions about the possibility of biological implementations of backpropagation." The B-L algorithm answers some of those questions.

Balduzzi *et al.* [50] pointed out that weight updates under backpropagation depend on lengthy recursive computations and require separate output and error messages; and these features are not shared by biological neurons. To overcome this drawback, the authors propose a truncated error backpropagation algorithm called Kickback. They show that Kickback matches backpropagation's performance on real-world regression benchmarks. They emphasize that Kickback is not gradient descent on the output error. In comparison, we show that the B-L algorithm can achieve backpropagation exactly without the need for lengthy recursive computations.

Marblestone *et al.* [51] argued that implementations of credit assignment through multiple layers of neurons are compatible with the current knowledge of neural circuitry and that the brain's specialized systems can be interpreted as enabling efficient optimization for specific problem classes.

Durbin and Rumelhart [52] introduced a new type of computational unit that calculates a weighted product. Such a unit that can learn an arbitrary polynomial term is then fed into higher-level standard summing units. The authors then showed that there is a natural neurobiological interpretation for this type of combination of product and summing units in terms of a single neuron.

Bekolay *et al.* [53] proposed a learning rule for learning cognitively relevant transformations of neural representations in a biologically plausible manner. They showed that the unsupervised component of the rule increases the sparsity of connection weights at the cost of increased signal transmission error. They speculate that some parameters in the model are a result of the structure of the neuron and are fixed while some other parameters are related to the activity of neurons. They proposed to examine these possibilities in future work.

Bengio *et al.* [54] first discussed eight difficulties regarding the biological plausibility of backpropagation. They then explore more biologically plausible versions of deep learning, focusing mostly on unsupervised learning but developing a learning mechanism that could account for supervised, unsupervised, and reinforcement learning. They state that the gradients required for updating the hidden states can be estimated using an approximation that only requires propagating activations forward and backward.

The B-L algorithm overcomes some crucial difficulties in the biological plausibility of the C-B algorithm. Obviously, the B-L algorithm does not need a feedback network to backpropagate errors. It allows a phaseless adaptation. Let us look at the adaptation laws more closely. Consider the adaptation law in Theorem 1 for hidden neurons (for which, $(\partial E/\partial r_{\text{post}_s}) = 0$) and assume that adaptation parameter $\alpha_{s'}$ are same for all synapses, that is, $\alpha_{s'} = \alpha$. We have

$$\dot{w}_s = \epsilon_s r_{\text{pre}_s} \rho_{\text{post}_s} \lambda_{\text{post}_s} \sigma'(\lambda_{\text{post}_s} p_{\text{post}_s}) \frac{\varphi_{\text{post}_s}}{r_{\text{post}_s}}$$

where

$$\varphi_{\text{post}_s} = \sum_{s' \in A_{\text{post}_s}} \eta_{s'} w_{s'} \dot{w}_{s'}.$$

The above equation can be rewritten as follows:

$$\varphi_{\text{post}_s} = \frac{1}{2}\frac{d}{dt}\left(\sum_{s' \in A_{\text{post}_s}} \eta_{s'} (w_{s'})^2\right).$$

It is reasonable to assume that the decremental conduction coefficient $\eta_{s'}$ is same for all synapses $s' \in A_{\text{post}_s}$, that is, $\eta_{s'} = \eta$, for all $s' \in A_{\text{post}_s}$. Hence

$$\varphi_{\text{post}_s} = \frac{\eta}{2}\frac{d}{dt}\left(\sum_{s' \in A_{\text{post}_s}} (w_{s'})^2\right).$$

In other words, φ_{post_s} is proportional to the derivative (rate of change) of $\sum_{s' \in A_{\text{post}_s}} (w_{s'})^2$, which is the sum of the squares of strengths of axonic synapses. While it is known that synaptic strength $w_{s'}$ is related to the number of neurotransmitters generated, we hypothesize that the information related to the rate of change of $\sum_{s' \in A_{\text{post}_s}} (w_{s'})^2$ is also available in biological neurons. If that is the case, then the B-L algorithm is certainly biologically plausible, as all information needed is available locally in a biological neuron.

VI. Conclusion

In this article, we introduce a new learning algorithm. The main characteristics of the B-L algorithm are as follows.

1) It is mathematically equivalent to the C-B algorithm.
2) It does not need a feedback network to propagate the errors.
3) It eliminates the need for two phases in the adaptation and allows a phaseless adaptation by neurons.
4) It is applicable to both hierarchical and nonhierarchical neural networks.

Because of these characteristics, the adaptation based on the B-L algorithm is much more plausible in biological neurons. The B-L algorithm also makes the hardware implementation of neural networks on silicon much simpler because it is feedback-network-free.

Although the B-L algorithm is biologically plausible, we do not know the biological mechanisms that propagate the error signals from the axon to the dendrite of a neuron. In future works, we will investigate this and other problems.

Acknowledgment

The author would like to thank the anonymous reviewers for their excellent comments and suggestions that significantly improve the article.

References

[1] B. Widrow, D. E. Rumelhart, and M. A. Lehr, "Neural networks: Applications in industry, business and science," *Commun. ACM*, vol. 37, no. 3, pp. 93–106, 1994.

[2] A. K. Jain, J. Mao, and K. M. Mohiuddin, "Artificial neural networks: A tutorial," *Computer*, vol. 29, no. 3, pp. 31–44, Mar. 1996.

[3] D. P. Mandic and J. Chambers, *Recurrent Neural Networks for Prediction: Learning Algorithms, Architectures and Stability*. Hoboken, NJ, USA: Wiley, 2001.

[4] K. Gurney, *An Introduction to Neural Networks*. Boca Raton, FL, USA: CRC Press, 2014.

[5] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.

[6] L. Deng and D. Yu, "Deep learning: Methods and applications," *Found. Trends Signal Process.*, vol. 7, nos. 3–4, pp. 197–387, Jun. 2014.

[7] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Netw.*, vol. 61, pp. 85–117, Jan. 2015.

[8] W. Du *et al.*, "The networked evolutionary algorithm: A network science perspective," *Appl. Math. Comput.*, vol. 338, pp. 33–43, Dec. 2018.

[9] L. Liu, H. Zhang, X. Xu, Z. Zhang, and S. Yan, "Collocating clothes with generative adversarial networks cosupervised by categories and attributes: A multidiscriminator framework," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 9, pp. 3540–3554, Sep. 2020.

[10] R. Gao *et al.*, "Zero-VAE-GAN: Generating unseen features for generalized and transductive zero-shot learning," *IEEE Trans. Image Process.*, vol. 29, pp. 3665–3680, 2020.

[11] T. Dong and T. Huang, "Neural cryptography based on complex-valued neural network," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 11, pp. 4999–5004, Nov. 2020.

[12] H. Zhang, X. Wang, L. Liu, D. Zhou, and Z. Zhang, "WarpClothingOut: A stepwise framework for clothes translation from the human body to tiled images," *IEEE MultimediaMag.*, vol. 27, no. 4, pp. 58–68, Oct. 2020.

[13] H. Bao, A. Hu, W. Liu, and B. Bao, "Hidden bursting firings and bifurcation mechanisms in memristive neuron model with threshold electromagnetic induction," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 2, pp. 502–511, Feb. 2020.

[14] D. Srinivasan and G. K. Venayagamoorthy, "Guest editorial special issue on 'Neural networks and learning systems applications in smart Grid,'" *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 8, pp. 1601–1603, Aug. 2016.

[15] C. Yang, T. Teng, B. Xu, Z. Li, J. Na, and C.-Y. Su, "Global adaptive tracking control of robot manipulators using neural networks with finite-time learning convergence," *Int. J. Control, Autom. Syst.*, vol. 15, no. 4, pp. 1916–1924, Aug. 2017.

[16] H. Y. D. Sigaki, E. K. Lenzi, R. S. Zola, M. Perc, and H. V. Ribeiro, "Learning physical properties of liquid crystals with deep convolutional neural networks," *Sci. Rep.*, vol. 10, no. 1, pp. 1–10, Dec. 2020.

[17] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, Oct. 1986.

[18] R. Hecht-Nielsen, "Theory of the backpropagation neural network," in *Neural Networks for Perception*. Amsterdam, The Netherlands: Elsevier, 1992, pp. 65–93.

[19] Y. Chauvin and D. E. Rumelhart, *Backpropagation: Theory, Architectures, and Applications*. Hinsdale, NJ, USA: Psychology Press, 2013.

[20] D. Zipser and D. E. Rumelhart, "The neurobiological significance of the new learning models," in *Computational Neuroscience*. Cambridge, MA, USA: MIT Press, 1993, pp. 192–200.

[21] F. Crick, "The recent excitement about neural networks," *Nature*, vol. 337, no. 6203, pp. 129–132, Jan. 1989.

[22] M. Meireles *et al.*, "Flavonoids as dopaminergic neuromodulators," *Mol. Nutrition Food Res.*, vol. 60, no. 3, pp. 495–501, Mar. 2016.

[23] P. De Camilli and R. Jahn, "Pathways to regulated exocytosis in neurons," *Annu. Rev. Physiol.*, vol. 52, no. 1, pp. 625–645, Oct. 1990.

[24] F. Torri-Tarelli, A. Villa, F. Valtorta, P. De Camilli, P. Greengard, and B. Ceccarelli, "Redistribution of synaptophysin and synapsin i during alpha-latrotoxin-induced release of neurotransmitter at the neuromuscular junction," *J. Cell Biol.*, vol. 110, no. 2, pp. 449–459, Feb. 1990.

[25] V. Belzer and M. Hanani, "Nitric oxide as a messenger between neurons and satellite glial cells in dorsal root ganglia," *Glia*, vol. 67, no. 7, pp. 1296–1307, Jul. 2019.

[26] M. Barinaga, "Is nitric oxide the retrograde messenger?" *Science*, vol. 254, no. 5036, pp. 1296–1298, 1991.

[27] E. Schuman and D. Madison, "A requirement for the intercellular messenger nitric oxide in long-term potentiation," *Science*, vol. 254, no. 5037, pp. 1503–1506, Dec. 1991.

[28] F. Green *et al.*, "Axonal transport of AAV9 in nonhuman primate brain," *Gene Therapy*, vol. 23, no. 6, pp. 520–526, Jun. 2016.

[29] W. S. Trimble, M. Linial, and R. H. Scheller, "Cellular and molecular biology of the presynaptic nerve terminal," *Annu. Rev. Neurosci.*, vol. 14, no. 1, pp. 93–122, Mar. 1991.

[30] R. B. Vallee and G. S. Bloom, "Mechanisms of fast and slow axonal transport," *Annu. Rev. Neurosci.*, vol. 14, no. 1, pp. 59–92, Mar. 1991.

[31] D. Piomelli *et al.*, "Lipoxygenase metabolites of arachidonic acid as second messengers for presynaptic inhibition of aplysia sensory cells," *Nature*, vol. 328, no. 6125, p. 38, 1987.

[32] J. H. Schwartz and S. M. Greenberg, "Molecular mechanisms for memory: Second-messenger induced modifications of protein kinases in nerve cells," *Annu. Rev. Neurosci.*, vol. 10, no. 1, pp. 459–476, Mar. 1987.

[33] L. Constantin, "Circular RNAs and neuronal development," in *Circular RNAs*. Singapore: Springer, 2018, pp. 205–213.

[34] G. Schratt, "microRNAs at the synapse," *Nature Rev. Neurosci.*, vol. 10, no. 12, p. 842, 2009.

[35] M. E. Zaghloul, J. L. Meador, and R. W. Newcomb, *Silicon Implementation of Pulse Coded Neural Networks*, vol. 266. New York, NY, USA: Springer, 2012.

[36] A. Basu, S. Shuo, H. Zhou, M. H. Lim, and G.-B. Huang, "Silicon spiking neurons for hardware implementation of extreme learning machines," *Neurocomputing*, vol. 102, pp. 125–134, Feb. 2013.

[37] R. D. Brandt and F. Lin, "Adaptive interaction and its application to neural networks," *Inf. Sci.*, vol. 121, nos. 3–4, pp. 201–215, Dec. 1999.

[38] F. Lin, R. D. Brandt, and G. Saikalis, “Self-tuning of PID controllers by adaptive interaction,” in *Proc. Amer. Control Conf. (ACC)*, vol. 5, Jun. 2000, pp. 3676–3681.

[39] R. D. Brandt and F. Lin, “Supervised learning in neural networks without feedback network,” in *Proc. IEEE Int. Symp. Intell. Control*, Sep. 1996, pp. 86–90.

[40] R. L. De Nó and G. Condouris, “Decremental conduction in peripheral nerve. integration of stimuli in the neuron,” *Proc. Nat. Acad. Sci. USA*, vol. 45, no. 4, p. 592, 1959.

[41] K. Montaigne and W. F. Pickard, “Offset of the vacuolar potential of Characean cells in response to electromagnetic radiation over the range 250 Hz-250 kHz,” *Bioelectromagn., J. Bioelectromagn. Soc., Soc. Phys. Regulation Biol. Med., Eur. Bioelectromagn. Assoc.*, vol. 5, no. 1, pp. 31–38, 1984.

[42] W. Rall, “Core conductor theory and cable properties of neurons,” in *The Nervous System*, vol. 1, E. kandel, Ed. Baltimore, MD, USA: Waverly Press, 1977.

[43] A. Taherkhani, A. Belatreche, Y. Li, G. Cosma, L. P. Maguire, and T. M. McGinnity, “A review of learning in biologically plausible spiking neural networks,” *Neural Netw.*, vol. 122, pp. 253–272, Feb. 2020.

[44] M. van Gerven and S. Bohte, “Editorial: Artificial neural networks as models of neural information processing,” *Frontiers Comput. Neurosci.*, vol. 11, p. 114, Dec. 2017.

[45] B. Scellier and Y. Bengio, “Equilibrium propagation: Bridging the gap between energy-based models and backpropagation,” *Frontiers Comput. Neurosci.*, vol. 11, p. 24, May 2017.

[46] T. Miconi, “Biologically plausible learning in recurrent neural networks reproduces neural dynamics observed during cognitive tasks,” *eLife*, vol. 6, Feb. 2017, Art. no. e20899.

[47] B. Scellier and Y. Bengio, “Towards a biologically plausible backprop,” 2016, *arXiv:1602.05179*. [Online]. Available: https://arxiv.org/abs/1602.05179

[48] P. Mazzoni, R. A. Andersen, and M. I. Jordan, “A more biologically plausible learning rule for neural networks,” *Proc. Nat. Acad. Sci. USA*, vol. 88, no. 10, pp. 4433–4437, 1991.

[49] D. G. Stork, “Is backpropagation biologically plausible,” in *Proc. Int. Joint Conf. Neural Netw.*, Washington, DC, USA, vol. 2, Jun. 1989, pp. 241–246.

[50] D. Balduzzi, H. Vanchinathan, and J. M. Buhmann, “Kickback cuts backprop’s red-tape: Biologically plausible credit assignment in neural networks,” in *Proc. AAAI*, 2015, pp. 485–491.

[51] A. H. Marblestone, G. Wayne, and K. P. Kording, “Toward an integration of deep learning and neuroscience,” *Frontiers Comput. Neurosci.*, vol. 10, p. 94, Sep. 2016.

[52] R. Durbin and D. E. Rumelhart, “Product units: A computationally powerful and biologically plausible extension to backpropagation networks,” *Neural Comput.*, vol. 1, no. 1, pp. 133–142, Mar. 1989.

[53] T. Bekolay, C. Kolbeck, and C. Eliasmith, “Simultaneous unsupervised and supervised learning of cognitive functions in biologically plausible spiking neural networks,” in *Proc. Annu. Meeting Cogn. Sci. Soc.*, vol. 35, 2013, pp. 1–7.

[54] Y. Bengio, D.-H. Lee, J. Bornschein, T. Mesnard, and Z. Lin, “Towards biologically plausible deep learning,” 2015, *arXiv:1502.04156*. [Online]. Available: http://arxiv.org/abs/1502.04156

Feng Lin (Fellow, IEEE) received the B.Eng. degree in electrical engineering from Shanghai Jiao Tong University, Shanghai, China, in 1982, and the M.A.Sc. and Ph.D. degrees in electrical engineering from the University of Toronto, Toronto, ON, Canada, in 1984 and 1988, respectively.

He was a Post-Doctoral Fellow with Harvard University, Cambridge, MA, USA, from 1987 to 1988. Since 1988, he has been with the Department of Electrical and Computer Engineering, Wayne State University, Detroit, MI, USA, where he is currently a Professor. He has authored a book entitled “*Robust Control Design: An Optimal Control Approach*.” His current research interests include discrete event systems, hybrid systems, robust control, artificial intelligence, and their applications in alternative energy, biomedical systems, and automotive control.

Dr. Lin was a recipient of the George Axelby outstanding paper award from the IEEE Control Systems Society. He was an Associate Editor of IEEE TRANSACTIONS ON AUTOMATIC CONTROL.