Local learning algorithm for optical neural networks

Yong Qiao and Demetri Psaltis

An anti-Hebbian local learning algorithm for two-layer optical neural networks is introduced. With this learning rule, the weight update for a certain connection depends only on the input and output of that connection and a global, scalar error signal. Therefore the backpropagation of error signals through the network, as required by the commonly used back error propagation algorithm, is avoided. It still guarantees, however, that the synaptic weights are updated in the error descent direction. With the apparent advantage of simpler optical implementation this learning rule is also shown by simulations to be computationally effective.

Key words: Optical neural networks, anti-Hebbian local learning.

The most widely used algorithm for training multilayer neural networks is the backward error propagation (BEP) algorithm,[1] which is a steepest-descent algorithm that minimizes the error at the output of the network with respect to the weight values of the connections between the neurons. Methods for the optical implementation of BEP have been proposed[2,3] but the practical realization of these ideas is complicated by the fact that BEP is a nonlocal learning rule. Consider the two-layer network shown in Fig. 1. If we change any one of the weights in the first layer (decrease or increase) the effect of this change on the output of the network would depend on the value and sign of the weights of the second layer. BEP is a steepest-descent method and, therefore, attempts to decrease the output error as quickly as possible. To accomplish this, we require knowledge of the second-layer weights in order to change the weights in the first layer. From an implementation point of view, this complicates matters since we must communicate information in both directions and have bidirectional neurons with a different functionality in the forward and backward directions.[2,3] Here we describe an algorithm for training two-layer optical neural networks, in which the weight updates are calculated from the signal at the input of each connection, the signal at the other end of the same connection, and a global scalar error signal. The advantage of this algorithm is that it can be implemented with signals that are locally available, which simplifies the optical system.

We can get an intuitive feel for how our algorithm works by considering the network of Fig. 1 with only a single output neuron. Suppose that the output of that neuron takes the value $+1$ or -1. The weights of the second layer are trained by using the same procedure as in BEP. For the weights of the first layer, if the output is wrong for a particular input, we can correct it by adjusting the weights of the first layer to produce the negative of the current response of the hidden layer.[4] Therefore we can treat the first layer as a single-layer net with a known desired output, and thus it can be trained with one of the existing algorithms for training single-layer nets. Below we show that it is possible to select the training algorithm for the first layer so as to guarantee that the output error will decrease at each iteration.

Let the numbers of neurons for the input, first, and second layers of the two-layer network shown in Fig. 1 be N_0, N_1, and N_2, respectively. The inputs to the neurons of the nth layer are

$$s_j^{(n)} = \sum_{i=1}^{N_{n-1}} w_{ji}^{(n)} o_i^{(n-1)}, \tag{1}$$

where $w_{ji}^{(n)}$ is the weight of the interconnection between the jth neuron in the nth layer and the ith neuron in the previous layer, and $o_i^{(n)}$ is the output of the ith neuron in the nth layer. The signal at the input layer is denoted by $o_i^{(0)}$. The first and second

The authors are with the Department of Electrical Engineering, California Institute of Technology, Pasadena, California 91125.

Received 30 September, 1991
0003-6935/92/173285-04$05.00/0.

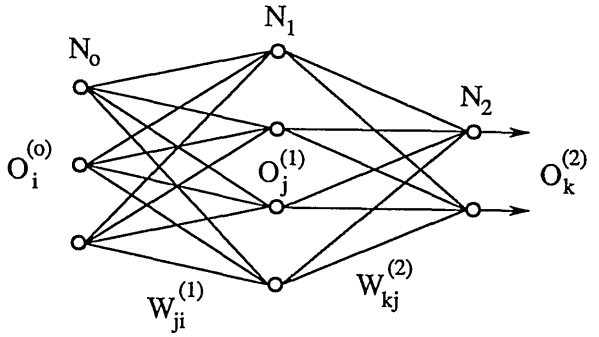

Fig. 1. Schematic diagram of a feed-forward two-layer neural network.

layers of neurons perform a soft thresholding operation on their inputs to produce the outputs

$$o_j^{(n)} = f[s_j^{(n)}], \tag{2}$$

where the function f is chosen as $f(x) = \tanh(x)$ for the analysis.

The desired response that corresponds to the input $o_i^{(0)}$ is the vector $\mathbf{t}$ whose elements are binary, $t_k \in \{1, -1\}$ for $k = 1, 2, \ldots, N_2$. The output error of the network is measured by the logarithmic energy function[5,6]:

$$E = \sum_{k=1}^{N_2} \left[(1 + t_k) \ln \frac{1 + t_k}{1 + o_k^{(2)}} + (1 - t_k) \ln \frac{1 - t_k}{1 - o_k^{(2)}} \right]. \tag{3}$$

Equation (3) has its global minimum at $E = 0$ and reaches this minimum if and only if the network output is the same as the desired response. We chose this form of error function instead of the more commonly used quadratic error function because we found in simulations that for the problems we investigated this energy function gave a better performance. The algorithm we describe here also works with the quadratic error function with a slight, straightforward modification of the weight update rule we derive below.

The BEP rule changes the weights by means of the gradient descent, i.e.,

$$\Delta w_{kj}^{(2)} \propto -\frac{\partial E}{\partial w_{kj}^{(2)}} = 2\delta_k o_j^{(1)}, \tag{4}$$

$$\Delta w_{ji}^{(1)} \propto -\frac{\partial E}{\partial w_{ji}^{(1)}} = 2[1 - [o_j^{(1)}]^2] o_i^{(0)} \sum_{k=1}^{N_2} \delta_k w_{kj}^{(2)}, \tag{5}$$

where $\delta_k = t_k - o_k^{(2)}$ is the output error signal. The nonlocal nature of the BEP algorithm is due to the $\sum_{k=1}^{N_2} \delta_k w_{kj}^{(2)}$ factor in proportion (5), which contains the values of the weights of the second layer.

Let $\gamma_k = \delta_k s_k^{(2)}$. Then γ_k is positive if and only if the sign of the kth output unit matches the sign of t_k. For example let us assume that $t_k = -1$ and $o_k^{(2)} > 0$. Then $\delta_k < 0$. The input signal to the kth output unit $s_k^{(2)}$ has the same sign as the output of that unit and is, therefore, also positive. This gives us $\gamma_k < 0$ for this case. Let us now define the quantity $\gamma = \sum_{k=1}^{N_2} \gamma_k$, which will be positive if the sign of most of the output units matches the target sign and will be negative if the reverse is true. We can construct a learning rule for the first layer of weights by using γ as a performance metric. Notice that γ is a scalar quantity that can be calculated from signals that are available at the output layer of the network. The basic idea is to modify the first layer of weights so that we reinforce the production of the current hidden layer response if γ is positive or reinforce the production of the negative of the current hidden layer response if γ is negative. We know that the simple Hebbian learning rule

$$\Delta w_{ji}^{(1)} \propto o_j^{(1)} o_i^{(0)} \tag{6}$$

reinforces the reproduction of the current response when presented with the same input. Therefore, if we want to reinforce the negative of the current response when the output is incorrect, then we can simply adopt an anti-Hebbian rule by multiplying the right-hand side of proportion (6) by γ. This idea leads to the following anti-Hebbian local learning (ALL) algorithm for the first layer:

$$\Delta w_{ji}^{(1)} \propto \frac{\gamma o_j^{(1)} o_i^{(0)}}{\{1 - [o_j^{(1)}]^2\}}. \tag{7}$$

The denominator in proportion (7) is an additional term that is needed to guarantee that this learning rule always decreases the overall error at the output. By using proportions (5) and (7), and by assuming that the weights of interconnections between any input neuron and all hidden neurons are updated simultaneously (which is true in most practical situations), we obtain

$$\begin{aligned} \Delta E &= \sum_{j=1}^{N_1} \frac{\partial E}{\partial w_{ji}^{(1)}} \Delta w_{ji}^{(1)} \\ &\propto -\gamma [o_i^{(0)}]^2 \sum_{k=1}^{N_2} \delta_k \sum_{j=1}^{N_1} w_{kj}^{(2)} o_j^{(1)} \\ &= -\gamma [o_i^{(0)}]^2 \sum_{k=1}^{N_2} \delta S_{k\mathrm{sbx}}^{(2)} \\ &= -\gamma^2 [o_i^{(0)}]^2 \\ &\le 0, \end{aligned} \tag{8}$$

which proves our claim. Thus we have proved that even though the ALL algorithm is not a steepest-descent rule it is still a descent rule.

In some cases it may happen that during training the output of a certain hidden neuron becomes so close to $+1$ or -1 that the denominator $\{1 - [o_j^{(1)}]^2\}$ in proportion (7) is close to zero. This causes numerical instability. One way to avoid this is to find $o_{\max}^{(1)}$, which is the hidden neuron output that has the maximum magnitude, and normalize the right-hand side of proportion (7) by the factor $\{1 - [o_{\max}^{(1)}]^2\}$.

Computer simulations of the ALL algorithm were performed for the problem of recognizing handwritten zip-code digits provided by U.S. Postal Service. For comparison, the BEP algorithm was also used to solve the same problem. The handwritten zip codes were first segmented into single digits, and then each digit was reduced to fit a 10×10 binary pixel grid. A network of 100 input neurons (to match the 10×10 pixel grid), 5 hidden neurons, and 3 output neurons were selected and trained to provide classification for three classes of handwritten digits, 3, 6, and 8. Each output neuron responds to one class only. 600 digit patterns, with 200 patterns from each class, were selected. These 600 patterns were partitioned into 300 training samples, 150 validation samples, and 150 test samples. The validation samples were used after each learning iteration (i.e., presentation of the whole training set) to calculate the classification error of the network. The network training stopped when the classification error of the network on the validation set stopped decreasing with further iteration. After the network was trained, the test samples were persented to the network to find its generalization error. For the ALL algorithm, the first layer was trained in 1 of 40 iterations only. By doing this, we relied more on the steepest-descent training in the second layer and it improved learning convergence for this particular classification problem and this particular network. For given training, validation, and test sets, the network was trained four times with different initial conditions for both the ALL and the BEP algorithms. The same step size was used for the two algorithms for the purpose of comparison. The same simulations were repeated with different training, validation, and test sets obtained from different partitioning of the 600 digit patterns (the numbers of the training, validation, and test samples were still 300, 150, and 150, respectively). Therefore, there were total of eight runs for each algorithm.

For the ALL algorithm, the network was able to converge (meaning all the training patterns were classified correctly) in seven cases. In only one case the network fell into a local minimum and gave a training error of 1%. For the BEP algorithm, the network was able to converge in all eight cases. The average generalization errors for the ALL and the BEP algorithms were 9% and 8%, respectively. As for the average convergence rate of the two algorithms, it took 581 iterations for the BEP algorithm to converge, and 3,665 iterations for the ALL algorithm to converge. However, since the amount of computation involved in each learning iteration is different for the two algorithms, we should also compare the convergence rate in terms of number of computational steps. It turns out that, for this size of network, the number of computational steps in each iteration for BEP is ~ 2.4 times of that for ALL. In this sense, ALL is only ~ 2.6 times slower than BEP when implemented by a serial digital computer. In the optical implementation that is fully parallel the appropriate speed comparison between BEP and ALL should be based on the number of learning iterations since the time required to complete each iteration is roughly the same for both algorithms. Therefore, for the problem that we studied in our experiment, a parallel optical implementation of ALL would be ~ 6 times slower than BEP. However, the optical system that implements ALL is much simpler than the BEP system, and this makes it much more likely that an ALL system can be built in practice. We should point out that the relative convergence rates of the two algorithms quoted above apply only to the problem we have tried and the relative performance will be generally problem dependent.

One possible implementation of the ALL algorithm is shown in Fig. 2. This architecture is quite similar to the architecture described in Refs. 2 and 3 for the implementation of BEP, with a few key differences. The input images are recorded on an electrically addressed spatial light modulator (EASLM1), and hologram #1 interconnects the pixels at the input plane to the pixels at the intermediate or hidden layer plane. The nonlinear response of the neurons at the hidden layer is simulated by an optically addressed spatial light modulator (OASLM). The second layer is similar, with hologram #2 interconnecting pixels from the OASLM to the output plane where a two-dimensional CCD detector (CCD1) is placed to detect the light. At the input stage there is a second EASLM (EASLM2) on which the reference signals that are necessary for the adaptation of hologram #1 are recorded. Similarly at the hidden layer there is an EASLM3 to record the reference for hologram #2, which is simply the error signal δ_k [see proportion (4)]. This error signal is produced by subtracting the network output from the desired target signal, a point operation that can be accomplished either optically or electronically. The reference for hologram #1 is more difficult to derive [see proportion (7)]. It involves the global error signal γ and the response of the hidden layer ($\mathbf{o}^{(1)}$). γ can be calculated with point operations from signals that are already available at the output of the system, but we need a second detector (CCD2) at the output to record ($\mathbf{o}^{(1)}$) as it is imaged through hologram #2. Once the reference

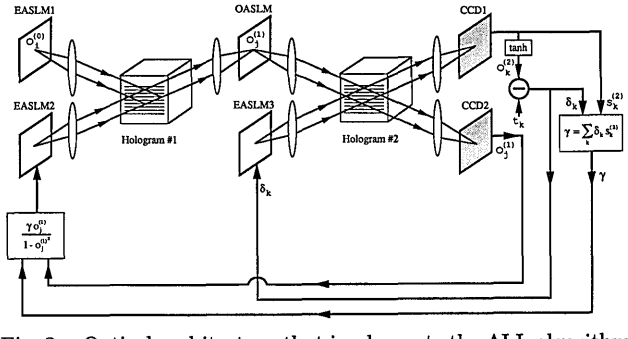

Fig. 2. Optical architecture that implements the ALL algorithm.

signals are calculated and recorded on EASLM2 and EASLM3, hologram #1 (hologram #2) is exposed to the interference between the signal recorded on EASLM2 (EASLM3) and $\mathbf{o}^{(0)}$ ($\mathbf{o}^{(1)}$). This requires that a latching device, such as the microchannel spatial light modulator, be used as the OASLM. An exposure schedule (which is described in Ref. 3) must be used to ensure that each holographic exposure contributes equally to the overall hologram. The key difference between the ALL architecture and the BEP architecture described in Refs. 2 and 3 is that the light always travels in the same direction throughout the system (left to right in Fig. 2). This simplifies the construction and the alignment of the system and, most importantly, does not require a device at the hidden layer (OASLM) that operates in both operations and has a different response function in each direction. Thus the ALL architecture is much more likely to be constructed in the foreseeable future.

This work was supported by the Defense Advanced Research Projects Agency and the U.S. Air Force Office of Scientific Research.

References

1. D. E. Rumelhart, G. E. Hinton, and R. J. Williams, in *Parallel Distributed Processing,* D. E. Rumelhart and J. L. McClelland, eds. (MIT Press, Cambridge, Mass., 1986), Vol. 1, pp. 318–362.
2. K. Wagner and D. Psaltis, "Multilayer optical learning networks," Appl. Opt. **26,** 5061–5076 (1987).
3. D. Psaltis, D. Brady, and K. Wagner, "Adaptive optical networks using photorefractive crystals," Appl. Opt. **27,** 1752–1759 (1988).
4. T. Grossman, R. Meir, and E. Domany, "Learning by choice of internal representations," Complex Syst. **2,** 555–575 (1988).
5. J. J. Hopfield, "Learning algorithms and probability distributions in feed-forward and feed-back networks," Proc. Natl. Acad. Sci. USA **84,** 8429–8433 (1987).
6. S. A. Solla, E. Levin, and M. Fleisher, "Accelerated learning in layered neural networks," Complex Syst. **2,** 625–640 (1988).