Proceedings of 1993 International Joint Conference on Neural Networks

IMPROVEMENT OF SUPERVISED LEARNING BY LINEAR MAPPING

Chin-Chi Teng and Benjamin W. Wah
Coordinated Science Laboratory
University of Illinois at Urbana-Champaign
1308 West Main Street, Urbana, Illinois 61801
{teng, wah}@manip.crhc.uiuc.edu

Abstract

In this paper, we propose a new supervised learning method to improve the learning speed for feed-forward neural networks. Our method is different from traditional supervised learning methods (such as back-propagation) that find a one-to-one mapping between the given input pattern matrix and the desired output pattern matrix. Instead, it finds one of the one-to-many mappings between the input matrix and an intermediate output matrix, and transforms the intermediate output matrix to the desired output matrix in one step using linear mapping techniques. Learning is faster with our method because there exist many intermediate output matrices, and learning can stop whenever one such matrix is found. Our extensive experimental results show that our learning algorithm converges to within a reasonable range of error after a few training epochs, making it suitable for dynamic real-time applications in which the network may need to be re-trained periodically.

1. Introduction

In this paper we propose a new supervised learning algorithm for artificial neural networks (ANNs). The applications we have considered can be modeled as a mapping of a k-by-m *input pattern matrix* P (with k patterns, each with m values) to a k-by-n *desired output pattern matrix* D *(with k patterns, each with n values). The ANN to be learned performs a mapping from* P *to* D.

Existing supervised learning algorithms focus on finding a one-to-one mapping between the input matrix and the corresponding output matrix to within a prescribed error tolerance. They generally compare the actual output matrix and the desired output matrix, and according to the error, decide how to adapt the weights of the connections (as done in the back-propagation algorithm), or how to adapt the network configuration (as done in the cascade-correlation algorithm) [2]. Learning time is long due to the convergence time needed for finding one-to-one mappings.

Researchs supported by Joint Services Electronics Program Contract JSEP N00014-90-J-1270.

Learning time can be significantly reduced if the output matrix can be flexible; that is, learning is faster if there is a large pool of desired output matrices, and learning can stop whenever one of them is found. The new supervised learning method we study in this paper exploits this property. It first transforms the original problem of finding a one-to-one mapping from P to D into one that finds a one-to-one mapping from P to one of a large set of possible output matrices (called I_{real}). It then transforms I_{real} to D by finding the least-squared-error solution of the set of linear equations $I_{real}W = D$.

This paper is organized as follows. We present linear mapping techniques for training single-layer ANNs in Section 2. In Sections 3, we propose a new supervised learning method and discuss its characteristics. Section 4 shows our experimental results and compares them with that of the back-propagation, Quickprop, and cascade correlation learning algorithms. Finally, conclusions are drawn in Section 5.

2. Applying Linear Mapping to Train Single-Layer ANNs

The operations performed in a layer of an ANN can be considered as a matrix-vector multiplication followed by a nonlinear transformation. For instance, given a single-layer feed-forward ANN with m input units and n output units, we can represent the weights between the input and output layers as a weight matrix $W_{m\times n}$, where $w_{i,j}$ is the weight of the connection between input unit i and output unit j. The actual output pattern matrix D_{real} can then be calculated by matrix multiplication as follows.

$$((D_{real})_{k\times m})_{i,j} = f((P \cdot W)_{i,j}) \quad i,j = 1,2,...,k, \quad (1)$$

where $f(x)$ is the activation function of an output unit.

Given a set of training patterns represented as matrices P and D, the training of a single-layer ANN with activation function $f(x)$ has been shown to be equivalent to solving a set of linear equations $P \cdot W = D$ in order to obtain weight matrix $W_{m\times n}$ [6].

If the activation function is a sigmoid function, then we can derive a *modified desired output matrix* D_0 in such a way that $sigmoid((D_0)_{i,j}) \approx (D)_{i,j}$. If D is

binary, then D_0 can be defined as follows [4].

$$(D_0)_{i,j} = \begin{cases} \rho & \text{if } (D)_{i,j} = 1 \\ -\rho & \text{if } (D)_{i,j} = 0 \end{cases} \quad (2)$$

where ρ is a large positive number. As a result, the learning problem becomes the problem of solving a set of linear equations

$$P \cdot W = D_0. \quad (3)$$

It is clear that if we can obtain W such that $P \cdot W = D_0$, then the outputs after applying the sigmoid function to D_0 are very close to those of D.

When training single-layer ANNs, the linear mapping technique described above is faster than general supervised learning methods because it is not iterative. However, this linear mapping technique is not universal and cannot be applied to multi-layer ANNs. In this paper, we propose a hybrid learning method that combines the linear mapping method described above and traditional iterative learning methods for training multi-layer ANNs.

3. Hybrid Supervised Learning Method

Figure 1 illustrates our hybrid supervised learning method. An ANN with input matrix P and desired output matrix D can be trained by a traditional supervised learning algorithm. During training, there is a *monitor* that checks periodically whether the column space of an *intermediate output matrix* I_{real} contains that of the desired output matrix, where $(I_{real})_{i,j}$ is the output of the j-th neuron that is connected to output layer when the i-th training pattern is applied.

The check for containment by the monitor is equivalent to finding the least-squared solution W^+ of $P \cdot W^+ = D_0$ (see Eq. (3)), which exists if and only if

$$\text{span}\{D_0\} \subseteq \text{span}\{P\}. \quad (4)$$

Hence, once the column space of I_{real} contains $\text{span}\{D_0\}$, then the linear mapping described in the previous section can be employed to map I_{real} to D in one step, and learning completes. At this point, W' replaces W in the original network.

Our hybrid learning method improves learning time since it has a relaxed learning objective. In traditional supervised learning of ANNs, the objective is to

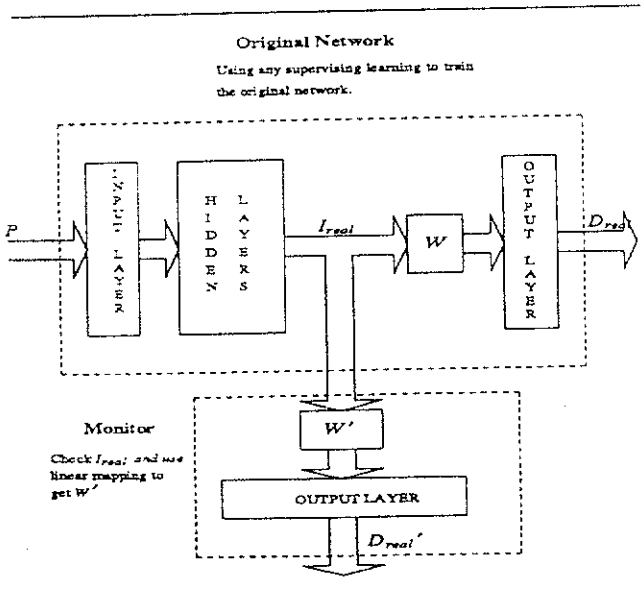


Figure 1. Hybrid Supervised Learning Method.

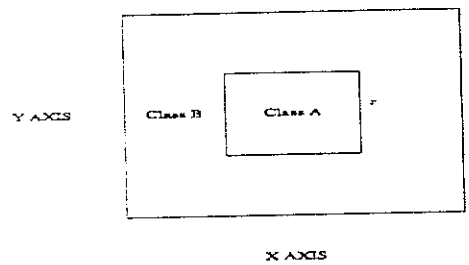
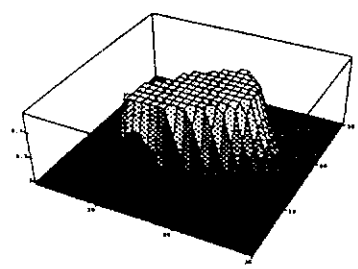
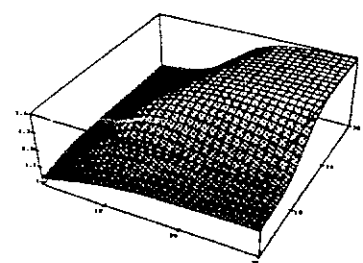


Figure 2. Problem to classify points in Classes A and B.



(a) Performance of proposed method.



(b) Performance of back-propagation.

Figure 3. Test results after 10 epochs of learning.

find a one-to-one mapping between the given input and output matrices. In contrast, our proposed method tries to get an intermediate output matrix I_{real} whose column space contains the column space of the desired output matrix. Since there are many intermediate matrices satisfying this criterion, supervised learning involves finding one of the one-to-many mappings and is, therefore, easier. The additional overhead incurred involves solving a set of linear equations and is very small as compared to the savings in learning time in the original learning algorithm.

4. Experimental Results

For uniformity in reporting results, we use the 40-20-40 criterion for the binary case; that is, we consider an output a logic ONE if it is larger than 0.6, and a logic ZERO if it is smaller than 0.4. We report both the number of epochs and the computing time needed for convergence. Our simulations were performed on a Sun Sparcstation 10/20.

Our hybrid learning method converges very quickly after a few training epochs. We illustrate this by using a simple classification problem shown in Figure 2. The training set has 200 patterns and is randomly selected. After 10 training epochs, we test two ANNs, one trained by back-propagation and another by our hybrid learning method. The results shown in Figure 3 illustrates that our learning method converges very quickly to within acceptable errors with a few training epochs, whereas the network trained by back-propagation does not. This feature is especially useful when ANNs need to be retrained in real time.

Table 1 shows the performance of our learning method when the original network is trained by the Quickprop learning algorithm [3]. Learning in our method stops when the monitor finds a linear mapping with acceptable error tolerance. We train the networks (one with monitor and one without) for three benchmark problems: XOR, SONAR [5], and WAVEFORM [1]. Each benchmark problem is trained thirty times by the two learning methods. Table 2 shows the performance when the back-propagation learning method is used instead. Our results confirms that the monitor allows learning to converge very quickly in terms of both the number of epochs and the actual CPU time.

Finally, we compare the performance of our learning method with that of the cascade-correlation learning algorithm [2]. There are two training phases in the cascade-correlation algorithm: *TRAIN_INPUT* phase for adding new hidden units, and *TRAIN_OUTPUT* phase for training the weights in the output layer. These two phases are executed alternatively. In adding a monitor to the cascade-correlation algorithm, we notice a) that I_{real} cannot be acquired in the first phase, as the new hidden unit has not been decided upon, and b) that I_{real} is frozen in the *TRAIN_OUTPUT* phase. Consequently, we only need to use the monitor to find a linear mapping in the first epoch of each *TRAIN_OUTPUT* phase.

Table 3 compares the performance of cascade correlation with and without a monitor. We see that the addition of a monitor does not improve the learning time significantly, and in some cases, results in increased learning time. This happens because the number of hidden units is too small to find an appropriate I_{real} (as is in the WAVE problem). However, during the experiments, we found that our algorithm can sometimes find ANNs with smaller number of hidden units when they converge. The reduction in the number of hidden units is important since a trained ANN may have to be applied many times in the future.

5. Conclusions

In this paper we propose a new learning algorithm that uses a monitor to admonish outputs from neurons in hidden layers that are connected to the output layer. Learning stops whenever the monitor finds a linear mapping from these outputs to the desired outputs of the output layer. Our learning method has the following advantages.

(1) It transforms supervised learning from a one-to-one mapping to a one-to-many mapping. That is equivalent to relaxing the objective of learning and making it easier. The overhead incurred is very small as it involves solving a set of linear equations.

(2) The error in our learning method converges very quickly after a few training epochs. This feature is very useful when ANNs have to be retrained in real-time applications.

(3) Our proposed learning method is flexible and can adapt to many existing supervised learning algorithms.

(4) Our proposed method does not significantly improve the performance of the cascade correlation algorithm. However, our method can sometimes find ANNs with smaller number of hidden units.

References

[1] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and Regression Trees*, pp. 49-55, Wadsworth International, Belmont, CA, 1984.

[2] S. E. Fahlman and Christian Lebiere, "The Cascade-Correlation Learning Architecture," pp.

Table 1. Learning performance of Quickprop with and without a monitor.

Problem	Performance measures		without monitor	with monitor
XOR	Epochs	avg	44.93	1.17
		max	304	3
		min	21	1
	Time (sec.)	avg	0.031	8.e-4
		max	0.18	2.e-3
		min	0.01	7.e-4
SONAR	Epochs	avg	366.3	74.27
		max	818	564
		min	131	1
	Time (sec.)	avg	47.89	19.46
		max	106.4	147.9
		min	17.05	0.26
WAVE	Epochs	avg	221.5	66.20
		max	442	409
		min	105	8
	Time (sec.)	avg	28.87	17.69
		max	59.07	107.9
		min	28.87	2.29

Table 2. Learning performance of back-propagation with and without a monitor.

Problem	Performance measures		without monitor	with monitor
XOR	Epochs	avg	182.3	1
		max	508	1
		min	73	1
	Time (sec.)	avg	0.15	8.e-4
		max	0.9	8.e-4
		min	0.03	8.e-4
SONAR	Epochs	avg	245.9	32.13
		max	750	218
		min	94	1
	Time (sec.)	avg	42.73	11.30
		max	129.4	75.73
		min	16.26	0.513
WAVE	Epochs	avg	116.5	47.9
		max	177	135
		min	95	9
	Time (sec.)	avg	22.68	18.70
		max	34.07	52.29
		min	18.61	3.67

Table 3. Learning performance of cascade-correlation with and without a monitor.

Problem	Performance measures		without monitor	with monitor
XOR	Epochs	avg	54.14	43.00
		max	67	50
		min	47	37
	Time (sec.)	avg	0.024	0.01
		max	0.03	0.02
		min	0.01	1.e-3
SONAR	Epochs	avg	389.6	277.7
		max	554	491
		min	298	218
	Time (sec.)	avg	6.16	5.94
		max	9.97	10.81
		min	4.36	4.47
WAVE	Epochs	avg	491.7	532.5
		max	618	783
		min	406	319
	Time (sec.)	avg	11.88	14.61
		max	14.60	20.33
		min	9.55	9.20

524-532 in *Advances in Neural Information Processing Systems* 2, ed. D. S. Touretzky, Morgan Kaufmann, San Mateo, 1990.

[3] S. E. Fahlmann, *An Empirical Study of Learning Speed in Back-Propagation Networks*, Pittsburgh, PA, 1988.

[4] S. D. D. Goggin, K. E. Gustafson, and K. M. Johonson, "An Asymptotic Singular Value Decomposition Analysis of Nonlinear Multilayer Neural Networks," *Int'l Joint Conf. on Neural Networks*, Seattle, WA, July 1991.

[5] R. P. Gorman and T. J. Sejnowski, "Analysis of Hidden Units in a Layered Network Trained to Classify Sonar Targets," *Neural Networks*, vol. 1, pp. 75-89, Pergamon Press, Elmsford, NY, 1988.

[6] D. E. Rumelhart, J. L. McClelland, and The PDP Group (ed.), *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, MIT Press, Cambridge, MA, 1986.