Neural Approach to the Problem of Pattern Recognition of the Ten Digits

Chunshien Li† *and Yu-Chieh Chen*

†Department of Computer Science and Information Engineering, National University of Tainan
33, Sec. 2, Shu-Lin St., Tainan, 700, Taiwan, ROC.

Summary
For the problem of pattern recognition of the ten digits, four learning algorithms of the supervised Hebbian learning, the pseudo-inverse learning, the Widrow-Hoff learning and the back-propagation learning are studied with neural nets. Progressively, these learning algorithms as well as network structures are investigated to improve the performance of the neural networks for the recognition task. For a linear associator with the supervised Hebbian learning rule and the concept of autoassociative memory, the performance of the network is not as good as expected. Although the pseudo inverse method can improve the recognition rate, it cannot stand firm from the attack of noise. For the Widrow-Hoff learning with the Adaline network, it is able to perform the task of pattern recognition, but is still suffered from noise corruption in the input patterns. With hidden layer structure, the performance of the multi-layer neural network using the back-propagation algorithm is over that of the other three single-layer neural networks trained with the other algorithms. Noised patterns are tested. The recognition rate of one-bit corrupted patterns is 100%, and for two bits corrupted, the identification rate can still reach more than 98%. If more bits in the input patterns are corrupted, more information is lost from the patterns, and neural nets may get more difficult to perform the task of pattern recognition correctly.

Key words:
Pattern recognition, neural networks, learning algorithm, Hebbian learning, pseudo-inverse learning, Widrow-Hoff learning, back-propagation learning.

1. Introduction

There is a complex biological neural network in a human brain [12], which contains about 10^{11} neurons. The neurons in the network facilitating our thinking, reading and motion are highly interconnected. Some of our neural structure is with us at birth, and most parts have been furnished through experience. All the biological neural functionalities, including memory, are stored in the neurons and in the connections among them. The behavior of learning can be viewed as the establishment of new connections or the modification of existing connections. The neural networks considered here are not biological. They contain artificial neurons, which could be realized with computer programs or electrical circuits.

The artificial neural networks may not be as powerful as human brain, but they can be trained to perform lots of useful tasks. Many researches and applications of neural networks and fuzzy logic to control systems and pattern recognition have been studied in recent years [16]. The first practical application of neural networks was proposed in the late 1950s. Frank Rosenblatt built a perceptron network with associated rule and demonstrated its ability to perform pattern recognition [12]. In recent years, lots of researches are about pattern recognition using neural networks [5] [6], [8], [10]. They can be applied in many fields, such as intrusion detector to recognize attacks [9], analysis of cardiotocographic records [11], and underwater object recognition [3]. Mathematical models of neural networks and learning algorithms are overviewed in Section 2. Comparison of neural models and learning algorithms for pattern recognition is given in Section 3. A discussion of the learning algorithms for pattern recognition and the conclusion are given in Section 4.

2. Overview of Neural Models and Learning Algorithms

A model of a single-input neuron [4] is shown in Fig. 1, which is served as a building unit of a neural network. The neuron output is calculated as follows.

$$a = f(wp + b) \tag{1}$$

where p is the input signal, w the synaptic weight, b the bias, and f(.) the activation function. General speaking, one neuron, even with several inputs, may not be sufficient to useful application. One might need more neurons to form a network, operating in parallel, with layered structure. A single-layer network of S neurons is shown in Fig. 2. The input elements p_i, i=1, 2,…, R enter the single-layer network through the weight matrix W, given as follows.

Manuscript received January 25, 2006.
Manuscript revised February 15, 2006.

$$W = \begin{bmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,R} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,R} \\ \vdots & \vdots & & \vdots \\ w_{S,1} & w_{S,2} & \cdots & w_{S,R} \end{bmatrix} \quad (2)$$

where the row of the matrix is for a weight vector of a destination neuron, while the column is for the weight vector of destination neurons for a input signal. Note that $w_{j,i}$ represents the connection from the i^{th} input to the j^{th} neuron for i=1, 2,…, R and j=1, 2,…, S.

The one-layer network also can be drawn in abbreviated notation [4], as shown in Fig. 3.

$$\underline{p} = \begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_R \end{bmatrix}, \qquad \underline{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_S \end{bmatrix},$$

$$\underline{n} = \begin{bmatrix} n_1 \\ n_2 \\ \vdots \\ n_S \end{bmatrix}, \text{ and } \underline{a} = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_S \end{bmatrix}. \quad (3)$$

where $\underline{p}$ is the input vector, W the $S \times R$ weight matrix, $\underline{b}$ the vector of biases, and $\underline{a}$ the output vector. Elements in W and $\underline{b}$ are adjustable parameters of the network, and they can be adjusted using learning rule to perform specific tasks. Transfer functions in the neural network are chosen according to the need of the problem that the neural network is attempting to solve.

A multi-layer neural network is shown in Fig. 4. Each layer has its own weight matrix, bias vector, net-input vector and output vector. Note that the superscripts are used to identify layers. The abbreviated notation of the three-layer network is given in Fig. 5. Four learning algorithms are studied in the paper, explained as follows.

Supervised Hebbian Learning

The neural network used for Hebbian learning is a linear associator, as shown in Fig. 6. The network was proposed by James Anderson [2] and Teuvo Kohonen [7]. A linear transfer function whose output is equal to its input is used in the linear associator [17]-[18]. The algorithm of supervised Hebbian learning, where the weight matrix W is initialized to zero and each of the Q input/output pair is applied, is shown as follows.

$$W^{new} = W^{old} + \underline{t}_q \underline{p}_q^T = \underline{t}_1 \underline{p}_1^T + \underline{t}_2 \underline{p}_2^T + ... + \underline{t}_Q \underline{p}_Q^T \quad (4)$$

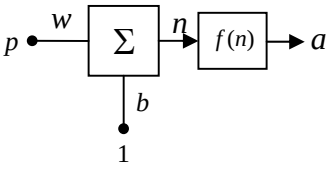


Fig.1. A single-input neuron.

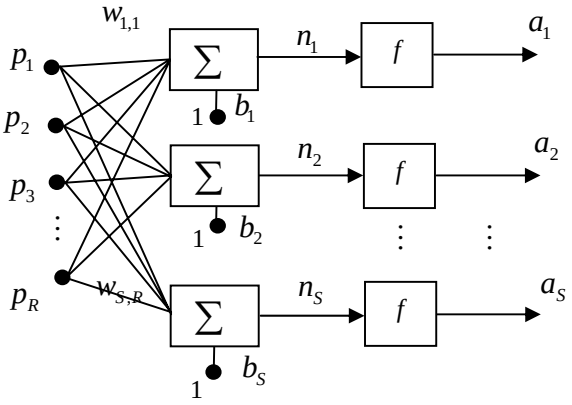


Fig.2. A layer of neural network with S neurons.

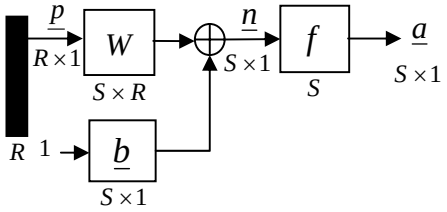


Fig.3. Abbreviated notation of one-layer network.

where $\underline{t}_q$ is the target corresponding to the input vector $\underline{p}_q$, which can be represented in matrix form as follows.

$$W^{new} = \begin{bmatrix} \underline{t}_1 & \underline{t}_2 & ... & \underline{t}_Q \end{bmatrix} \begin{bmatrix} \underline{p}_1^T \\ \underline{p}_2^T \\ \vdots \\ \underline{p}_Q^T \end{bmatrix} \equiv TP^T \quad (5)$$

which is called the supervised Hebbian learning rule [4].

Pseudo-Inverse Learning

Assume that W is the weight matrix after training. The linear associator shown in Fig. 6 is expected to produce the output $\underline{t}_q$ for a corresponding input $\underline{p}_q$, given as

$$W \times \underline{p}_q = \underline{t}_q \quad (6)$$

for $q = 1,2,...,Q$. Eq. (6) can be represented in matrix form as follows.

$$WP = T \quad (7)$$

where

$$P = [\underline{p}_1 \quad \underline{p}_2 \quad \cdots \quad \underline{p}_Q]$$
$$T = [\underline{t}_1 \quad \underline{t}_2 \quad \cdots \quad \underline{t}_Q] \quad (8)$$

If the matrix *P* has an inverse, the solution can be written as follows.

$$W = TP^{-1} \quad (9)$$

However, it is hardly possible to find a weight matrix to satisfy Eq. (7). What one can do is to minimize the following equation.

$$F(W) = \sum_{q=1}^{Q} \| \underline{t}_q - W \underline{p}_q \|^2 \quad (10)$$

The Moore-Penrose pseudo-inverse [1] of the matrix *P* can be computed to minimize *F(W)* given in Eq. (10), given as

$$P^+ = (P^T P)^{-1} P^T \quad (11)$$

The weight matrix for the linear associator is calculated as follows to minimize *F(W)*.

$$W = TP^+ \quad (12)$$

Widrow-Hoff Learning

Bernard Widrow and Marcian Hoff introduced the ADALINE (ADAptive LInear NEuron) network and a learning rule, called the LMS (Least Mean Square) algorithm [8], which is a gradient-based algorithm. The ADALINE network is shown in Fig. 7. The output of the ADALINE network is given as follows.

$$\underline{a} = purelin(W \underline{p} + \underline{b}) = W \underline{p} + \underline{b} \quad (13)$$

In Widrow-Hoff learning rule, error signal used to update the weight matrix is defined as follows.

$$\underline{e} = \underline{t} - \underline{a} \quad (14)$$

The learning rule of Widrow-Hoff method is given as

$$W(k+1) = W(k) + 2 \cdot \alpha \cdot \underline{e}(k) \cdot P^T \quad (15)$$

where α is the learning rate of the LMS algorithm and the index *k* indicates the k^{th} learning iteration.

Back-Propagation Learning Rule

The perceptron learning rule in Eqs. (5) and (12) and the LMS (Least Mean Square) algorithm in Eq. (15) are used to train single-layer neural networks. These single-layer networks are able to solve linearly separable problems only [4]. Although multilayer neural networks are powerful for dealing with nonlinear problems, there was no algorithm to train these kinds of networks until 1974 [15]. Since 1980s, back-propagation algorithm has

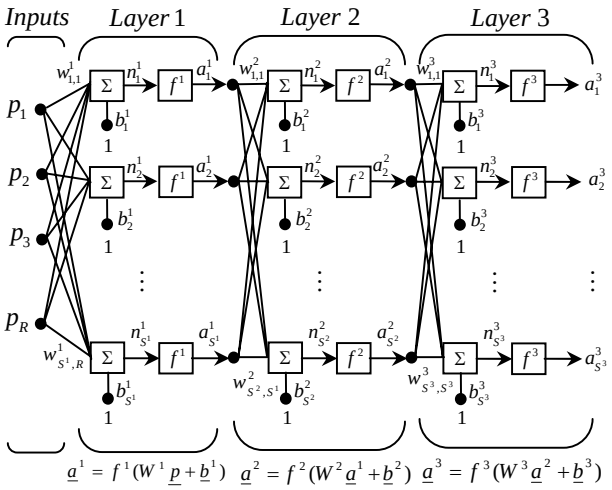


Fig.4. Three-layer neural network.

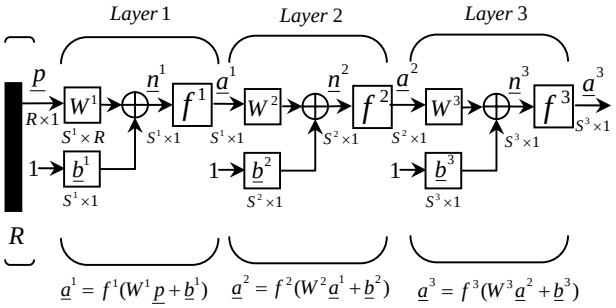


Fig.5. Abbreviated notation of a three-layer network.

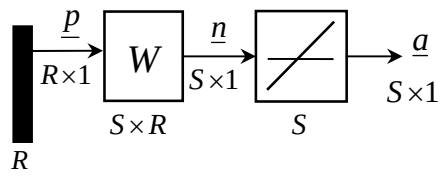


Fig.6. Linear associator network.

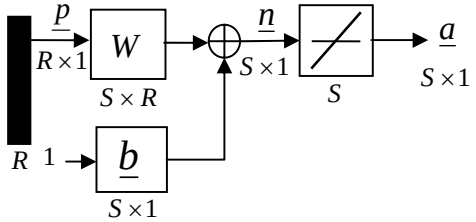


Fig.7. ADALINE network.

been widely publicized [13] for training multilayer neural networks. In Fig. 8, the outputs of the first and second layers are given as follows.

$$\underline{a}^1 = f^1(W^1\underline{p} + \underline{b}^1) \quad (16)$$

where

$$\underline{p} = \begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_R \end{bmatrix}, \underline{a}^1 = \begin{bmatrix} a_1^1 \\ a_2^1 \\ \vdots \\ a_{S^1}^1 \end{bmatrix}, \underline{b}^1 = \begin{bmatrix} b_1^1 \\ b_2^1 \\ \vdots \\ b_{S^1}^1 \end{bmatrix}, \underline{b}^2 = \begin{bmatrix} b_1^2 \\ b_2^2 \\ \vdots \\ b_{S^2}^2 \end{bmatrix}. \ \underline{a}^2 = \begin{bmatrix} a_1^2 \\ a_2^2 \\ \vdots \\ a_{S^2}^2 \end{bmatrix}$$

The back-propagation learning algorithm for multi-layer neural network is a generalization of the LMS algorithm. The mean square error is used in the algorithm. The network is provided with input/output pairs. As each input is applied to the network, the network output is compared to the corresponding target. The network parameters are adjusted with learning algorithm to minimize the mean square error defined below.

$$F(\underline{x}) = E[\underline{e}^2] = E[(\underline{t} - \underline{a})^2] = E[(\underline{t} - \underline{a})^T(\underline{t} - \underline{a})]. \quad (18)$$

where $\underline{x}$ is the vector of the network synaptic weights and biases and $\underline{a}$ is the network output. As for the LMS algorithm, the mean square error can be approximated by

$$\hat{F}(\underline{x}) = (\underline{t}(k) - \underline{a}(k))^T(\underline{t}(k) - \underline{a}(k)) = \underline{e}^T(k)\underline{e}(k) \quad (19)$$

where the expectation of the square error has been replaced by the square error at iteration k. The steepest descent algorithm for the approximate mean square error is given as follows for the weights an biases of the m^{th} layer for m=1, 2, …, M.

$$w_{i,j}^m(k+1) = w_{i,j}^m(k) - \alpha\frac{\partial\hat{F}}{\partial w_{i,j}^m},$$

$$b_i^m(k+1) = b_i^m(k) - \alpha\frac{\partial\hat{F}}{\partial b_i^m} \quad (20)$$

for i=1, 2, …, S^m and j=1, 2, …, S^{m-1} where α is learning rate. For a multi-layer network, the error is not an explicit function of the parameters in the hidden layers, whose derivatives can not be computed directly. The chain rule of calculus is used to calculate the derivatives, given as

$$\frac{\partial\hat{F}}{\partial w_{i,j}^m} = \frac{\partial\hat{F}}{\partial n_i^m} \times \frac{\partial n_i^m}{\partial w_{i,j}^m},$$

$$\frac{\partial\hat{F}}{\partial b_i^m} = \frac{\partial\hat{F}}{\partial n_i^m} \times \frac{\partial n_i^m}{\partial b_i^m}. \quad (21)$$

The net input to layer m is a function of the weights and bias in the layer, as shown below.

$$n_i^m = \sum_{j=1}^{S^{m-1}} w_{i,j}^m a_j^{m-1} + b_i^m \quad (22)$$

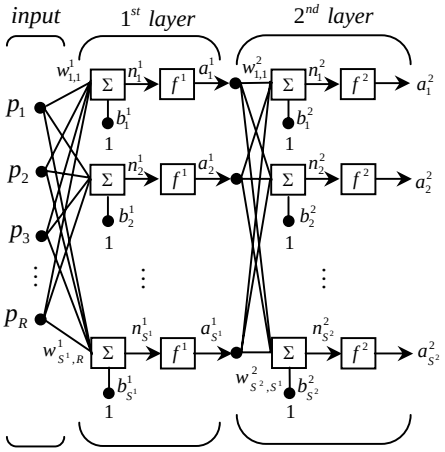


Fig.8. Two-layer neural network.

Thus,

$$\frac{\partial n_i^m}{\partial w_{i,j}^m} = a_j^{m-1}, \quad \frac{\partial n_i^m}{\partial b_i^m} = 1. \quad (23)$$

The sensitivity [4] of $\hat{F}$ for the i^{th} node of layer m is defined as follows.

$$\mu_i^m \equiv \frac{\partial\hat{F}}{\partial n_i^m} \quad (24)$$

With Eqs. (21)-(23), Eq. (24) can be written as follows.

$$\frac{\partial\hat{F}}{\partial w_{i,j}^m} = \mu_i^m a_j^{m-1}, \ \frac{\partial\hat{F}}{\partial b_i^m} = \mu_i^m. \quad (25)$$

Thus, Eq. (20) can be expressed as follows.

$$w_{i,j}^m(k+1) = w_{i,j}^m(k) - \alpha\mu_i^m a_j^{m-1},$$

$$b_i^m(k+1) = b_i^m(k) - \alpha\mu_i^m \quad (26)$$

whose matrix form can be expressed as follows.

$$W^m(k+1) = W^m(k) - \alpha\underline{\mu}^m(\underline{a}^{m-1})^T,$$

$$\underline{b}^m(k+1) = \underline{b}^m(k) - \alpha\underline{\mu}^m. \quad (27)$$

The sensitivity vector at the output layer (that is layer M) can be calculated as follows.

$$\underline{\mu}^M = -2\begin{bmatrix} \dot{f}^M(n_1^M) & 0 & 0 & 0 \\ 0 & \dot{f}^M(n_2^M) & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & \dot{f}^M(n_{S^M}^M) \end{bmatrix}(\underline{t} - \underline{a}^M) \quad (28)$$

And then the sensitivity vector of the output layer is propagated backward layer by layer through out the

network. The sensitivity at the first layer can be calculated as follows.

$$\underline{\mu}^1 = -2\begin{bmatrix} \dot{f}^1(n_1^1) & 0 & 0 & 0 \\ 0 & \dot{f}^1(n_2^1) & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & \dot{f}^1(n_{S^1}^1) \end{bmatrix} \cdot (W^2)^T \cdot \underline{\mu}^2 \qquad (29)$$

3. Pattern Recognition with Neural Nets

With the four learning methods mentioned, the neural networks are trained for the task of pattern recognition, for which the ten digits are chosen to be the patterns. The algorithms are provided with input/output pairs,

$$\{\underline{p}_1, t_1\}, \{\underline{p}_2, t_2\}, ..., \{\underline{p}_Q, t_Q\}$$

where $\underline{p}_q$ is the q^{th} input vector to the network, and t_q is the corresponding target, q=1, 2, …, Q. As an input pattern is applied to the network, the network output is compared to the corresponding target.

Learning algorithms are to update the synaptic weights of the networks, which are viewed as the memories of the networks. After training, the networks are tested with patterns which are similar to the training patterns. The patterns representation and the training processes of the four learning algorithms are studied as follows.

Representation of Patterns

The patterns of the ten digits for recognition training are shown in Fig. 9. They represent the digits 0~9, which are characterized with a 6×5 grid array. These digits are converted into vectors, and become the training patterns for Hebbian, Pseudo-Inverse and Widrow-Hoff learnings. Each white square is represented by a "-1", and each black square by a "1". To create pattern vector, each 6×5 grid array is scanned one row at a time so that each pattern becomes a 30×1 vector. For example, the prototype vector of the digit "0" is expressed as follows.

$$p_0 = [\ -1\ \ 1\ \ 1\ \ 1\ \ -1\ \ 1\ \ -1\ \ -1\ \ -1\ \ 1\ \ 1\ \ -1\ \ 1\ \ -1\ \ 1\ \ 1\ \ -1\ \ 1\ \ -1\ \ -1\ \ -1\ \ 1\ \ -1\ \ 1\ \ 1\ \ 1\ \ -1\ ]^T$$

The autoassociative memory, a special type of associative memory, is used for Hebbian, pseudo-inverse, and Widrow-Hoff learnings. With the autoassociative memory, the desired output vector is equal to the input vector.

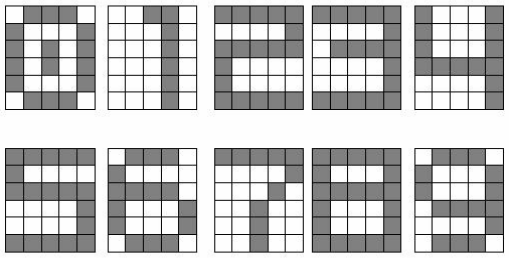


Fig.9. Patterns of the ten digits.

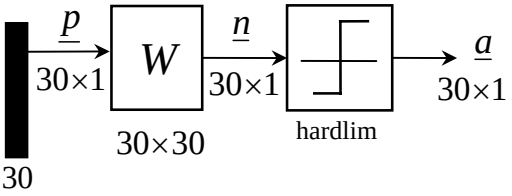


Fig.10. Autoassociative network for digit recognition.

Supervised Hebbian Learning

Since the autoassociative memory is used, there are only two possible values, "1" and "-1", for an output vector, and symmetrical hard limit function is used for an autoassociative network, shown in Fig. 10. The autoassociative network is trained with the supervised Hebbian learning rule given in Eqs. (4) to (5). The performance of this network is then investigated. The network is provided with the patterns shown in Fig. 9 to check the network output. The test result is shown in Fig. 11. There are only two digits, "4" and "8" recognized correctly, using the test patterns the same as the original patterns. The original patterns are changed one bit randomly and then are provided to the network for the output. It was found that the network could only recognize two or three of ten patterns. Because the input prototype patterns are not orthogonal to each other, some errors are produced with the Hebbian rule [4]. To reduce these errors, the pseudo-inverse learning is used and discussed in the following.

Pseudo-Inverse Learning

With the same network structure used in the Hebbian learning, the neural network is trained with the pseudo-inverse learning, given in Eqs. (10) to (12). The network is then tested with the original patterns shown in Fig. 9. The ten digits can be recognized very well, with 100% recognition rate. If a bit changed from each original pattern for noise corruption, there are 30 different

situations. We found that when the 6th, 10th or 11th element only was changed from each original pattern, the network produced errors. Fig. 12 is the test result when the 6th element was changed from each pattern vector. It is interesting to note that the test output is the same as the input vector to the network. This is also occurred when either the 10th or the 11th element is changed. With the pseudo-inverse learning, the network still cannot do a good job for the test patterns with noise.

Widrow-Hoff Learning

In Widrow-Hoff learning, the linear associator network is used, as shown in Fig. 6, and the error between target and network output is used to update the synaptic weights of the network. The concept of autoassociative memory is used in training the network. With the Widrow-Hoff learning given in Eq. (15), the network is trained with learning rate α=0.01.

The test result, using the original patterns, is shown in Fig. 13. The network can recognize the ten patterns very well. If the network is provided with the noised patterns in which one or two bits are changed from the original patterns in Fig. 9, the test result is no longer as good as Fig. 13. The test results, when patterns changed one and two bits from original patterns, are shown in Figs. 14 and 15.

For the autoassociative memory in the network, the results are admitted to be correct only when they are completely equal to the targets. With an idea different from the autoassociative memory, the amount of neurons is reduced from 30 to 10, so that input patterns to the network are recognized using 10 distinguishing targets, each of which is a 10×1 vector. The modified structure of network for Widrow-Hoff learning is shown in Fig. 16, in which *log-sigmoid* transfer functions are used.

Note that the linear transfer function is replaced with a log-sigmoid transfer function in the network, which squashes the output in between 0 and 1. The output of the network is a 10×1 vector, given as

$$\underline{a} = [a_0 \quad a_1 \quad a_2 \quad a_3 \quad a_4 \quad a_5 \quad a_6 \quad a_7 \quad a_8 \quad a_9]^T$$

The ten distinguishing targets are shown in Table 1. With the learning rate α=0.01, the network shown in Fig. 16 is trained for 1,000 times with the patterns shown in Fig. 9. The performance of this revised network is then checked. Tables 2, 3, and 4 are some of the outputs of the network when the original patterns are used for testing. In Table 2, for the digit "2", the test result of the network is excellent. The value of a_2 is superior to other values in the output vector $\underline{a}$. The results show that the network can identify these patterns very well. The test results for others of the ten digits are very good, shown in Tables 3 to 4.

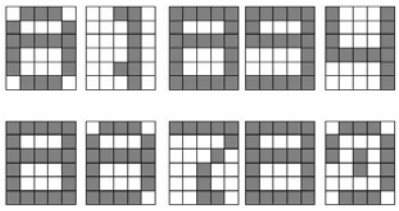


Fig.11. Test result of Hebbian learning.

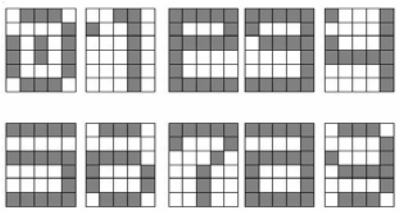


Fig.12. Test result of pseudo-inverse learning when the 6th bit in each original pattern is changed.

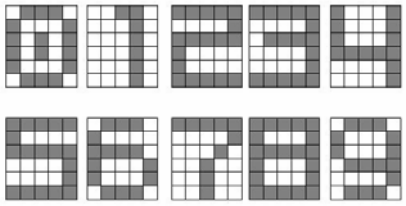


Fig.13. Test result of the Widrow-Hoff learning with original patterns.

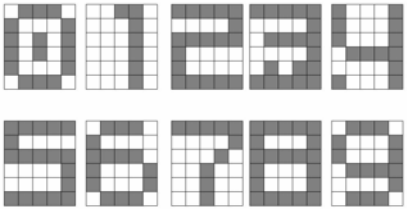


Fig.14. Test result of the Widrow-Hoff learning with one-bit changed patterns.

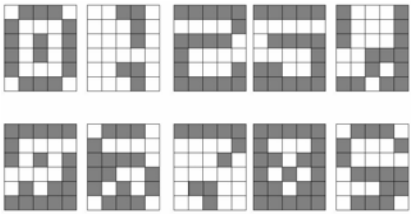


Fig.15. Test result of the Widrow-Hoff learning with two-bit changed patterns.

In Table 4, the test result can be viewed in such a way that the testing pattern is 99.66% similar to digit "7" and 0.11% similar to digit "0". A restriction is placed on the recognition performance of the network that if the largest element value in the output vector is not more than 50%, the network is not considered recognizing the pattern correctly. It may be interested to see the performance of the network by the Widrow-Hoff learning if a bit is corrupted from original pattern. There are 30 variations if concerning about one bit changed for a digit pattern, and there are 300 different kinds of conditions for the ten digit patterns. Two test results of the 300 noised patterns are listed in Tables 5 and 6. In Table 6, the pattern is 92.82% similar to the digit of "5", and 35.4% similar to the digit of "8", and the result is considered successful in this testing. After testing the 300 noised patterns, there are only 4 patterns that can not be recognized correctly by the network. The identification rate of one-bit changed patterns is considered as 98.67%.

If two bits are changed from each original pattern, there are C_2^{30} different test patterns to be obtained. 300 patterns are selected from the C_2^{30} patterns for testing, the identification rate of this testing is 96%. Similarly, for 3 bits changed in each pattern, the identification rate can still reach more than 94 %. A test result is shown in Table 7 for a test pattern with 3 bits noised. If the amount of noise-corrupted bits is increased, the identification rate is getting worse. This urges us to use a back-propagation neural network for the pattern recognition task.

Back-Propagation Learning

A two-layer neural network shown in Fig. 17 is used for the pattern recognition. Each layer has the same structure as the Widrow-Hoff network shown in Fig. 16. Input pattern to this network is a 30×1 vector, and the output is a 10×1 vector. 10 neurons in the output layer are served to distinguish input patterns to 10 different classes. The amount of the neurons in the hidden layer is given to be 20. Furthermore, the amount of training data for the network is increased from 1 set of 10 patters shown in Fig. 9 to 3 sets of 30 patterns shown in Figs. 18 to 20, with targets listed in Table 1. Three different versions of patterns of the ten digits are used for the network to expect improving the performance of the network for pattern recognition. The network is trained for 1,000 iterations with back-propagation learning algorithm. The initial conditions of the two weight matrices W^1 and W^2 are set to 20-by-30 matrix and 10-by-20 matrix with random entries of a uniform probability distribution in the interval [0, 1]. Besides, since the weight matrices are updated by the BP

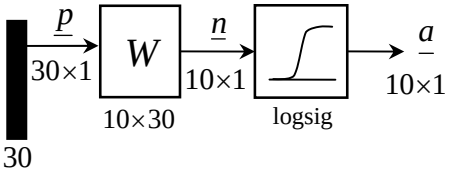


Fig.16. Widrow-Hoff network with logsig transfer function.

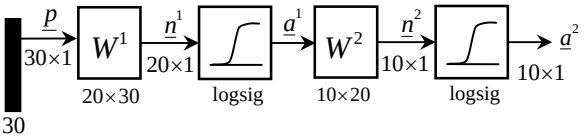


Fig.17. Two-layer network for pattern recognition.

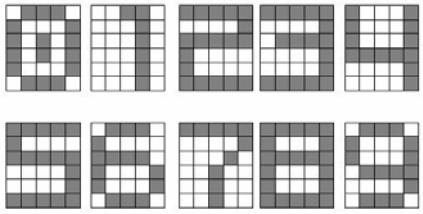


Fig.18. Training patterns of set 1.

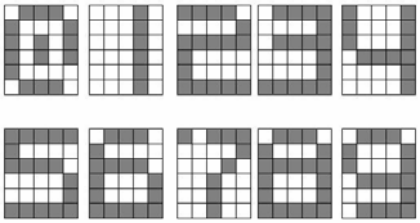


Fig.19. Training patterns of set 2.

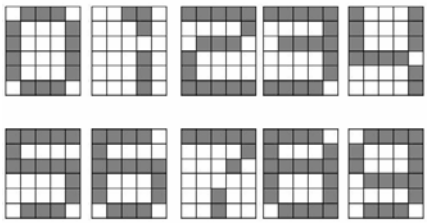


Fig.20. Training patterns of set 3.

learning rule, the value of learning rate α becomes an important factor for the stability and speed of learning. The learning rate used is designed to be decreased gradually with learning iterations, given in following equation.

$$\alpha = 0.5 \times e^{(\frac{-k}{1000})}$$

where k indicates the k^{th} learning iteration.

Testing with the 30 original patterns perfect without error. Two of these test results are shown in Tables 8 and 9. Table 8 shows the test result for the pattern of the digit "0" in the training patterns of set 1, and Table 9 is the test result for the pattern of "7" shown in Fig. 19. Moreover, we test the network with the testing pattern in Table 7, which can not be recognized correctly by the Widrow-Hoff network. The two-layer network recognizes the pattern very well, as shown in Table 10.

To test the performance of the network for noise robustness, the digit of "9" in the training patterns of set 1 is 5-bit corrupted and the test result is shown in Table 11. The performance of this testing is pretty good, even there are 5 bit changed. After testing the 300 one-bit changed patterns, all of the testing patterns are recognized correctly by the network and the identification rate of one-bit corrupted patterns is 100%. If two bits are changed from each original pattern, there are C_2^{30} different test patterns from the original patterns shown in Fig. 9 to be obtained. 300 patterns are selected from the C_2^{30} patterns for testing. Only 4 patterns that can not be recognized correctly by the network, thus the identification rate of this testing is 98.67%. Similarly, for 3 bits changed in each pattern, the identification rate can still reach 96.33%, and for 5 bits changed in each pattern, the identification rate is 91.33%.

4. Discussions and Conclusions

Models of neural networks and training methods have been overviewed and studied for their capability and performance in pattern recognition. Single-layer neural nets such as linear associator and Adaline are able to perform the pattern recognition of the ten digits but are fragile in recognition performance when noises are attacking the input patterns. Multi-layer neural net is superior to single-layer neural nets in the performance of pattern recognition of the ten digits. And it is much more robust standing for noise corruption in the input patterns. For a linear associator with the supervised Hebbian learning rule and the concept of autoassociative memory, the performance of the network is not as good as expected.

Table 1: Targets of the ten digits for Widrow-Hoff learning.

Digit	Target
0	1 0 0 0 0 0 0 0 0 0
1	0 1 0 0 0 0 0 0 0 0
2	1 0 0 0 0 0 0 0 0 0
3	0 1 0 0 0 0 0 0 0 0
4	1 0 0 0 0 0 0 0 0 0
5	0 1 0 0 0 0 0 0 0 0
6	1 0 0 0 0 0 0 0 0 0
7	0 1 0 0 0 0 0 0 0 0
8	1 0 0 0 0 0 0 0 0 0
9	0 1 0 0 0 0 0 0 0 0

Table 2: Test result of the pattern "2".

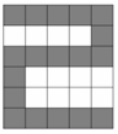
Test pattern	Output vector $\underline{a}$		Target
	a_0	0.00201219516138	0
	a_1	0.00039147767143	0
	a_2	0.98972741903177	1
	a_3	0.00301141806624	0
	a_4	0.00033066543309	0
	a_5	0.00056993723520	0
	a_6	0.00048076700437	0
	a_7	0.00184060791137	0
	a_8	0.01436147485882	0
	a_9	0.00003954185133	0

Table 3: Test result of the pattern "3".

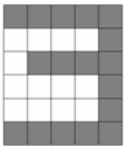
Test pattern	Output vector $\underline{a}$		Target
	a_0	0.00004466879181	0
	a_1	0.00022689077386	0
	a_2	0.00108251128144	0
	a_3	0.98731478033793	1
	a_4	0.00013725185118	0
	a_5	0.00780721158314	0
	a_6	0.00004494518130	0
	a_7	0.00116343046601	0
	a_8	0.00417556428794	0
	a_9	0.00074007425914	0

Table 4: Test result of the pattern "7".

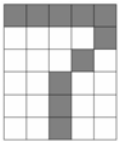
Test pattern	Output vector $\underline{a}$		Target
	a_0	0.00106729238450	0
	a_1	0.00193640454340	0
	a_2	0.00253779099154	0
	a_3	0.00314544436438	0
	a_4	0.00094923447830	0
	a_5	0.00018866740566	0
	a_6	0.00022092764568	0
	a_7	0.99663596242401	1
	a_8	0.00000118457743	0
	a_9	0.00110908538684	0

Table 5: Test result with one-bit changed pattern of "1".

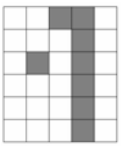
Test pattern	Output vector $\underline{a}$		Target
	a_0	0.0092%	0
	a_1	99.8077%	1
	a_2	1.0961%	0
	a_3	1.3805%	0
	a_4	0.15%	0
	a_5	0.6765%	0
	a_6	5.6731%	0
	a_7	0.3425%	0
	a_8	0.0002%	0
	a_9	0.7343%	0

Table 6: Test result with one-bit changed pattern of "5".

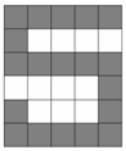
Test pattern	Output vector $\underline{a}$		Target
	a_0	0.0135%	0
	a_1	0.0190%	0
	a_2	0.1805%	0
	a_3	0.3105%	0
	a_4	0.0357%	0
	a_5	92.8218%	1
	a_6	1.2476%	0
	a_7	0.0235%	0
	a_8	35.4024%	0
	a_9	0.0704%	0

Table 7: Test result with three-bit changed pattern of "4".

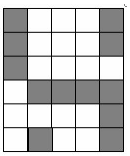
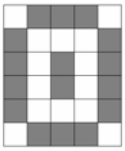
Test pattern	Output vector $\underline{a}$		Target
	a_0	0.3139%	0
	a_1	1.0815%	0
	a_2	0.2507%	0
	a_3	2.8352%	0
	a_4	45.0819%	1
	a_5	0.0149%	0
	a_6	3.0970%	0
	a_7	2.0566%	0
	a_8	0.0665%	0
	a_9	11.0714%	0

Table 8: Test result with the digit of "0" in set 1.

Test pattern	Output vector $\underline{a}$		Target
	a_0	95.0366%	1
	a_1	3.6343%	0
	a_2	1.3811%	0
	a_3	1.4427%	0
	a_4	0.0141%	0
	a_5	1.4955%	0
	a_6	2.7093%	0
	a_7	0.0481%	0
	a_8	0.0154%	0
	a_9	1.6394%	0

The recognition rate is poor for the pattern vectors of the ten digits are not orthonormal [4].

Using the pseudo inverse learning algorithm [1], the recognition performance of the ten digits is much improved, as shown in Fig. 12. Although the pseudo inverse method can improve the recognition rate, it cannot stand firm from the attack of noise, even for one bit change only in input pattern, as shown in Fig. 12. The learning method is to make the linear associator network produce the network output as close to the target as possible. When input pattern is corrupted, it is interesting to see that the network produces the output almost the same as the corrupted input, as shown in Fig. 12.

For the Widrow-Hoff learning with the autoassociative memory, the linear associator network is able to perform excellently the job, shown in Fig. 13, but is still suffered from noise corruption in the input patterns as shown in Figs. 14 and 15. If the autoassociative memory is discarded and is replaced with simpler coding patterns as the targets of the ten digits, the network performance of pattern recognition of the ten digits is improved. Log-sigmoid transfer functions are used in the single-layer neural network for more robust in noise corruption. The performance of the modified single-layer neural net with the Widrow-Hoff learning is much improved, compared to that of the original linear associator using the idea of autoassociative memory. The recognition rate can be as high as more than 98% for noised version of test patterns with one bit corrupted. If more bits in the input patterns are corrupted, more information is lost from the patterns, and neural nets may get more difficult to perform the job.

For the neural network with one hidden layer and the standard BP learning algorithm, the recognition performance is much improved in both correct recognition rate and noise robustness. The simplified version of coding for the target patterns of the ten digits is used in the multilayer, and obviously it improves the performance of training and recognition. Training patterns are important information to the neural network for pattern recognition. The training data is increased from one set of ten patterns to three sets of 30 patterns, in which some noised versions of the digit patterns are involved. This increases both the recognition rate and the robustness for noise corruption. The hidden layer is very important to the network for performing the task, for the recoding ability of the network is augmented [13].

Both single-layer and multi-layer neural networks with learning algorithms have been studied progressively for the performance comparison in the problem of pattern recognition of the ten digits. The result of pattern recognition can be affected by several factors, such as model structure, transfer function, amount of layers and nodes in each layer, learning algorithm, pattern definition, noise, training patterns, and training process.

Table 9: Test result with the digit of "7" in set 2.

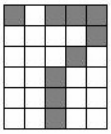
Test pattern	Output vector $\underline{a}$		Target
	a_0	1.7733%	0
	a_1	4.3424%	0
	a_2	0.7677%	0
	a_3	1.9739%	0
	a_4	2.3498%	0
	a_5	1.6274%	0
	a_6	0.0987%	0
	a_7	94.9723%	1
	a_8	1.1572%	0
	a_9	1.7974%	0

Table 10: Test result with three-bit changed pattern of "4".

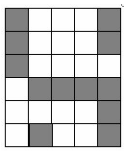
Test pattern	Output vector $\underline{a}$		Target
	a_0	0.9613%	0
	a_1	1.1368%	0
	a_2	0.0015%	0
	a_3	3.2144%	0
	a_4	91.2105%	1
	a_5	2.3362%	0
	a_6	0.0017%	0
	a_7	1.0149%	0
	a_8	2.2584%	0
	a_9	34.1968%	0

Table 11: Test result with five-bit changed pattern of "9".

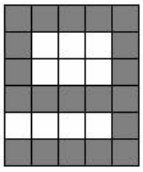
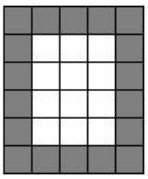
Test pattern	Output vector $\underline{a}$		Target
	a_0	0.0589%	0
	a_1	0.0017%	0
	a_2	0.0046%	0
	a_3	1.7290%	0
	a_4	8.6045%	0
	a_5	4.4975%	0
	a_6	0.0633%	0
	a_7	0.0106%	0
	a_8	21.2919%	0
	a_9	94.0201%	1

Table 12: Test result with six-bit changed pattern of "0"..

Test pattern	Output vector $\underline{a}$		Target
	a_0	44.7165%	1
	a_1	0.0032%	0
	a_2	13.5847%	0
	a_3	4.0557%	0
	a_4	1.3201%	0
	a_5	0.0099%	0
	a_6	0.0598%	0
	a_7	0.3538%	0
	a_8	43.7357%	0
	a_9	0.3852%	0

Among these factors, model structure, hidden layer, and learning algorithm are considered as the most significant. Four learning algorithms, which are the supervised Hebbian rule, the pseudo-inverse method, the Widrow-Hoff learning algorithm, and the back-propagation algorithm, are studied for single-layer neural networks with linear transfer function and with log-sigmoid transfer function and for the multi-layer neural network with one hidden layer. The neural network with hidden layer is much powerful than the one-layer neural networks in the problem of pattern recognition. A multi-layer neural network with more hidden neurons may possess recoding capability to tolerate more patterns in it so that the performance can be increased.

References

[1] A. Albert, *Regression and the Moore-Penrose Pseudo-inverse*, Academic Press, New York, USA, 1972.

[2] J. Anderson, "A Simple Neural Network Generating An Active Memory," *Mathematical Biosciences*, vol. 14 pp. 197-220, 1972.

[3] D. Boulinguez and A. Quinquis, "3-D underwater object recognition," *IEEE Journal of Oceanic Engineering*, vol. 27, no.4, pp. 814-829, Oct., 2002.

[4] M. T. Hagan, H. B. Demuth and M. Beale, *Neural Network Design*, Thomson Learning, Singapore, 1996.

[5] A. K. Jain, R. P. W. Duin and Jianchang Mao, "Statistical pattern recognition: a review," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 1, pp. 4-37, Jan., 2000.

[6] K. G. Khoo and P. N. Suganthan, **"**Structural pattern recognition using genetic algorithms with specialized operators," *IEEE Transactions on Systems, Man and Cybernetics-Part B*, vol. 33, no. 1, pp. 156-165, Feb., 2003.

[7] T. Kohonen, "Correlation matrix memories," *IEEE Transaction on Computers*, vol. 21, pp. 353-359, 1972.

[8] F. Lampariello, M. Sciandrone, "Efficient training of RBF neural networks for pattern recognition", *IEEE Transactions on Neural Networks*, vol. 12, no. 5, pp.1235 - 1242, Sep., 2001.

[9] S.C. Lee, D.V. Heinbuch, "Training a neural-network based intrusion detector to recognize novel attacks" *IEEE Transactions on Systems, Man and Cybernetics*, Part A, vol. 31, no. 4. pp. 294 -299, July 2001.

[10] M. Pardo and S. G.bereglieri, "Learning from data: a tutorial with emphasis on modern pattern recognition methods," *IEEE Sensors Journal*, vol. 2, no. 3, pp.203-217, Jun., 2002.

[11] O. Fontenla Romero, A. Alonso-Betanzos, B. Guijarro Berdinas, "Adaptive pattern recognition in the analysis of cardiotocographic records", *IEEE Transactions on Neural Networks*, vol. 12, no. 5, pp. 1188 -1195, Sept. 2001.

[12] F. Rosenblatt, "The perceptron: A probabistic model for information storage and organization in the brain," *Psychological Review*, vol. 65, pp.386-408, 1958.
[13] D. E. Rumelhart, G. E. Hinton and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol.323, pp.533-536, 1986.
[14] Y. Suzuki, K. Itakura, S. Saga and J. Maeda, "Signal processing and pattern recognition with soft computing," *Proceedings of the IEEE*, vol. 89, no. 9, pp. 1297-1317, Sep., 2001.
[15] P. J. Werbos, "Beyond regression: New tools for prediction and analysis in the behavioral sciences," Ph.D. Thesis, Harvard University, Cambridge, MA, 1974.
[16] D. White and D. Sofge (Eds.), *Handbook of Intelligent Control*, New York: Van Nostrand Reinhold, 1992.
[17] B. Widrow, M. E. Hoff, "Adaptive switching Circuits," *IRE WESCON Convention Record*, New York: IRE, part 4, pp. 96-104, 1960.
[18] B. Widrow and R. Winter, "Neural nets for adaptive filtering and adaptive pattern recognition," *IEEE Computer Magazine*, pp. 25-39, March 1988.
[19] W. Ziomek, M. Reformat and E. Kuffel, "Application of genetic algorithms to pattern recognition of defects in GIS," *IEEE Transactions on Dielectrics and Electrical Insulation*, vol. 7, no. 2, pp. 161-168, Apr., 2000.

Chunshien Li received the Ph.D. degree in Electrical Engineering and Computer Science in 1996 from University of Illinois at Chicago, USA. He is currently with Department of Computer Science and Information Engineering, National University of Tainan, Taiwan. He was with the Department of Electrical Engineering, Chang Gung University, Tao-Yuan, Taiwan. He received the Research Award from the National Science Council, Taiwan, in 1999 and the Research Award of Excellent Teacher from Chang Gung University, Taiwan, in 2000. He has been the first author of more than 60 technical papers. He is biographized in Marquis Who'sWho in Science and Engineering, in Medicine and Healthcare, and in the World. He has served as a referee of paper review for the journals of *IEEE Transactions on Industrial Electronics*, *IEEE Transactions on Systems, Man and Cybernetics-Part B: Cybernetic*, *IEEE Transactions on Fuzzy Systems*, and *Fuzzy Sets and Systems*, and other international journals. His current research interests include soft computing, computational intelligence, neuro-fuzzy systems, pattern recognition, intelligent signal processing, intelligent systems and control, machine learning, and chip/processor-based real-time systems.

Yu-Chieh Chen received the BS degree in Electrical Engineering in 2001 from Yuan-Zue University, Taiwan, and the MS degree in Electrical Engineering in 2003 from Chang Gung University, Taiwan. Her research interests are in fuzzy logic, neural networks, soft computing, and intelligent systems.