Regular Paper

How Neural Networks for Pattern Recognition Can Be Synthesized

SHIGEO ABE*, MASAHIRO KAYAMA* and HIROSHI TAKENAGA*

This paper discusses synthesis of neural networks for pattern recognition. First the saturation characteristics of the sigmoid function are shown to be essential for pattern recognition. Then neural networks for pattern recognition are proved to be synthesized by the following steps: (1) If the input data are separated by k hyperplanes, take k hidden neurons setting the weights between the input and hidden neurons as the coefficients of linear equations that define the hyperplanes. (2) If a class is separated into a single region by hyperplanes selected from the k hyperplanes, a three-layered neural network is synthesized to recognize that class; if not, a four-layered neural network is synthesized. Finally, a method to accelerate learning is proposed and is verified for a parity circuit, classification of two-dimensional patterns, and number recognition.

1. Introduction

Since the introduction of the backpropagation algorithm (BP), [1] multi-layered neural networks have been widely used for pattern recognition. Their major advantage is that they can be constructed by giving the BP a training data set that consists of the inputs and their desired outputs. Thus the pattern recognition network is generated without programming the classification algorithm. But this advantage, that a neural network can be handled as a black box, sometimes becomes a problem in actual applications. For example, if the neural network does not work properly for some unknown input data, all that can be done is to add those data to the training data set and reconstruct the network by using the BP. Moreover, there is no way of determining the structure of the network for a given problem. In [2] a three-layered neural network was proved to approximate an arbitrary continuous function, assuming an arbitrary number of hidden neurons, while in [3] a four-layered neural network was shown to be better than a three-layered neural network from the standpoint of the accuracy of approximation and the number of hidden neurons used. But these papers did not discuss how the neural networks could be synthesized. (Since neural networks for pattern recognition are trained to approximate a function whose outputs are only specified around 1 and 0, theories for function approximation also hold for pattern recognition.) For a training data set with only discrete inputs and outputs, some algorithms [4, 5] for synthesizing neural networks have been discussed, but no work has yet been reported for analog inputs [6].

*Hitachi Research Laboratory, Hitachi, Ltd., 4026 Kuji, Hitachi, 319-12, Japan.

In this paper we discuss how neural networks for pattern recognition can be synthesized. First, we study the meaning of saturation in the sigmoid function, and demonstrate that saturation is essential for pattern recognition. Then we discuss how pattern recognition networks can be synthesized, and derive an algorithm that speeds up learning. Finally, we demonstrate the validity of our results for some pattern classification problems.

2. Essentiality of Saturation to Pattern Recognition

2.1 Definition of the Network

Consider a three-layered neural network as shown in Fig. 1. The network consists of three layers of neurons. The neurons in the first layer are called input neurons, in the second, hidden neurons, and in the third, output neurons. The input neurons output the input data without modification, but input-and-output characteristics of the hidden and output neurons are specified by a sigmoid function, which has non-linearity and saturation. The neurons between the two consecutive layers are completely connected by synapses. Let the input and output of the j-th neuron in the i-th layer be $x_j(i)$ and $z_j(i)$, respectively. Then for the input layer,

$$x_j(i)=z_j(i) \quad \text{for} \quad j=1, \ldots, n(i),\ i=1 \quad \text{and}$$
$$z_{n(i)+1}(1)=1 \tag{1}$$

where $n(i)$ is the number of inputs to the i-th layer neurons and $z_{n(i)+1}(1)$ is a bias term. The output of the j-th neuron for the second and third layers is given by

$$z_j(i)=f(x_j(i)) \quad j=1, \ldots, n(i),\ i=2, 3 \quad \text{and}$$
$$z_{n(i)+1}(i)=1 \quad i=2, 3, \tag{2}$$

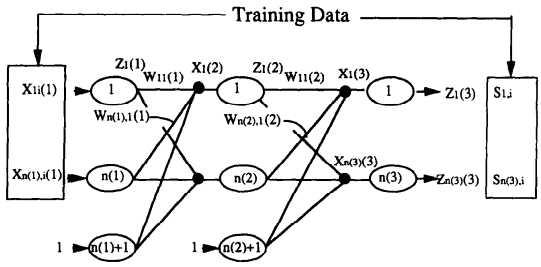


Fig. 1 A Three-layered Neural Network.

where $f(x)=1/(1+\exp(-x/T))$ is a sigmoid function,

$z_{n(i)+1}(i)$ are bias terms, and

T is a constant.

The value of $x_j(i)$ is given by

$$x_j(i)=W_j(i-1)z(i-1)$$
$$\text{for } i=2, 3, j=1, \ldots, n(i) \tag{3}$$

where $W_j(i-1)=(W_{j1}(i-1), \ldots, W_{j,n(i-1)+1}(i-1))$ is a weight vector and $W_{jk}(i-1)$ is a weight between the k-th neuron of the $(i-1)$st layer and j-th neuron of the i-th layer,

$z(i-1)=(z_1(i-1), \ldots, z_{n(i-1)}(i-1), 1)^t$ is an output vector corresponding to the $(i-1)$st layer neurons, and t is the transpose of a matrix.

The weights $W_{kj}(i-1)$ are determined by the BP, using the following training data set given by m pairs of inputs and desired outputs:

$$\{(x_{ij}(1), s_{jl})\}$$
$$\text{for } i=1, \ldots, n(1), j=1, \ldots, n(3),$$
$$\text{and } l=1, \ldots, m \tag{4}$$

2.2 Pattern Classification When the Number of Hidden Neurons is Smaller Than Those of Inputs and Outputs

According to the Kolmogorov theorem, any arbitrary continuous function is approximated by the composite mapping of monotonic functions [2]. Therefore, it may be better to take, as the input-output function of neurons, a non-saturated monotonic function, since saturation may slow down the convergence of learning. This saturation problem was solved during the study of a number recognition system. Using the BP to recognize numbers, we generated a three-layered neural network with 12 feature inputs, six hidden neurons, and ten outputs. A very fundamental question then arose: why were we able to generate a network for pattern recognition when the number of hidden neurons was smaller than those of input and output neurons? Let us now explain this question more specifically. From (2) the output of the j-th output neuron is given by

$$z_j(3)=f(x_j(3)), \quad j=1, \ldots, n(3) \tag{5}$$
$$x(3)=W(2)z(2) \tag{6}$$

where $W(2)=\begin{bmatrix} W_1(2) \\ W_2(2) \\ \cdots \\ W_{n(3)}(2) \end{bmatrix}$ is an $n(3)\times(n(2)+1)$ matrix,

and

$$x(3)=(x_1(3), \ldots, x_{n(3)}(3))^t.$$

For the number recognition system $n(3)=10$ and $n(2)+1=7$. Select one training datum from each number, and let ten sets of their output vectors of hidden neurons and input and output vectors of output neurons be $(z_1(2), x_1(3), z_1(3)), \ldots, (z_{10}(2), x_{10}(3), z_{10}(3))$. Then we can assume

$$(z_1(3), \ldots, z_{10}(3))=\begin{bmatrix} 10 & 0 \\ 01 & 0 \\ & \ddots \\ 00 & 1 \end{bmatrix}. \tag{7}$$

Thus the rank of the above matrix is ten, whereas the rank of

$$(x_1(3), \ldots, x_{10}(3))=W(2)(z_1(2), \ldots, z_{10}(2)) \tag{8}$$

is at most seven, according to the number of hidden neurons, since the ranks of $W(2)$ and $(z_1(2), \ldots, z_{10}(2))$ are at most seven and the rank of the matrix multiplied by two matrices does not exceed the rank of the matrix of a smaller rank. Thus the input information of rank 12, assuming independent inputs, is degenerated as hidden information of a rank of at most seven. Our question can then be rephrased as follows: why is it possible to reproduce output information of rank ten by using degenerated information of rank seven at most? (If the output function $f(x)$ is linear it is impossible.)

The answer to this problem lies in the saturation of the sigmoid function. In the number recognition system the outputs 1 and 0 were represented by $1-\varepsilon$, and $\varepsilon(\varepsilon>0)$, respectively, and the convergence of learning was tested by

$$1\geq z_i(3)\geq 1-2\varepsilon \quad \text{for} \quad z_i(3)=1-\varepsilon$$

and

$$2\varepsilon\geq z_i(3)\geq 0 \quad \text{for} \quad z_i(3)=\varepsilon. \tag{9}$$

Thus the finite intervals of convergence in $z_i(3)$ correspond to the infinite intervals in $x_i(3)$ as follows (see Fig. 2):

$$\infty > x_i(3)\geq\alpha \quad \text{and} \quad -\alpha\geq x_i(3) > -\infty. \tag{10}$$

where $\alpha=-T\log(1/(1-2\varepsilon)-1)=T\log(1/2\varepsilon-1)>0$.

Thus although the rank of $(x_1(3), \ldots, x_{10}(3))$ is at most seven, output information of rank ten is reproduced by saturation of the sigmoid function. Use of the

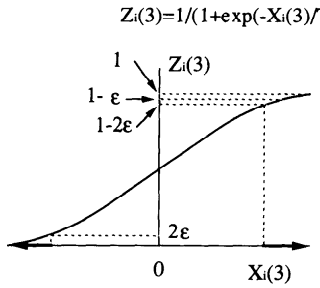
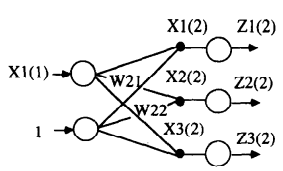


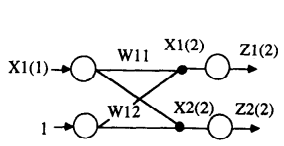
Fig. 2 Regions of Convergence for Pattern Recognition.



Input	Output		
X1(1)	Z1(2)	Z2(2)	Z3(2)
c1	1	0	0
c2	0	1	0
c3	0	0	1

c1 > c2 > c3

Fig. 3 Separation of Three Components in an Input into Three Outputs.



Input	Output	
X1(1)	Z1(2)	Z2(2)
c1	1	0
c2	0	1
c3	1	0

c1>c2>c3

Fig. 4 Classification of Different Components.

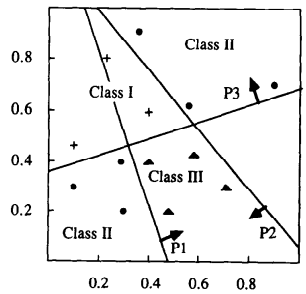


Fig. 5 Classification of Two-dimensional Points.

convergence tests given in (9) or (10) facilitates the convergence of learning. Saturation is, therefore, essential for pattern recognition, and to fully exploit saturation characteristics, we need to adopt the convergence test given by (9) or (10).

2.3 Classification Power of a Two-Layered Neural Network

By adopting the convergence test given by (9) or (10), we can clarify the separation power of a two-layered neural network with a single input and n outputs. We can show the following:

Two components in one variable $x_1(1)=[0, 1]$ can be separated by a two-layered neural network, but more than two components cannot.

Proof To simplify the proof, let us consider separation of c_1, c_2, c_3, where $1 \geq c_1 > c_2 > c_3 \geq 0$, as shown in Fig. 3. From (10) and Fig. 3, the following inequalities must hold for $x_2(2)$:

$$W_{21}c_1 + W_{22} \leq -\alpha \tag{11-1}$$

$$W_{21}c_2 + W_{22} \geq \alpha \tag{11-2}$$

$$W_{21}c_3 + W_{22} \leq -\alpha \tag{11-3}$$

where α is given by (10).

From (11-2)

$$W_{22} \geq \alpha - W_{21}c_2. \tag{12}$$

Also from (11-1), and (11-3),

$$W_{22} \leq -\alpha - W_{21}c_1 \tag{13}$$

$$W_{22} \leq -\alpha - W_{21}c_3. \tag{14}$$

Thus from (12) and (13),

$$W_{21} \leq -2\alpha/(c_1 - c_2) < 0. \tag{15}$$

Also, from (12) and (14),

$$W_{21} \geq 2\alpha/(c_2 - c_3) > 0. \tag{16}$$

Since (15) and (16) contradict each other, we cannot separate three or more components by a two-layered neural network.

Next let us consider classification of two points c_1 and c_2. In the same way as above, (11-1) and (11-2) must hold for $x_2(2)$. Thus (15) needs to be satisfied. Then one solution is

$$W_{21} = -2\alpha/(c_1 - c_2) \tag{17}$$

$$W_{22} = \alpha(c_1 + c_2)/(c_1 - c_2). \tag{18}$$

(End of proof)

The above property is similar to the proof by Minsky of the separation power of a perceptron, and can be rephrased as follows (see Fig. 4):

Different components in a variable $x_1(1)=[0, 1]$ cannot be classified into the same class by a two-layered neural network.

3. Pattern Recognition by a Three-layered Neural Network

The weights of the neural network are interpreted as the coefficients of a hyperplane. Assuming $x_j(i)=0$ in (3),

$$W_j(i-1)z(i-1)=0 \tag{19}$$

represents a hyperplane in $n(i-1)$-dimensional space. The change of weight $W_{j,n(i-1)+1}(i-1)$ causes a parallel displacement of the hyperplane. From (2), the value of $z_j(i)$ corresponding to $x_j(i)$ satisfying (19), which is on the hyperplane, is 1/2. We say that the $n(i-1)$-dimensional point $(z_1(i-1), \ldots, z_{n(i-1)}(i-1))^t$ is on the positive side of the hyperplane if

$$x_j(i)>0 \text{ or } z_j(i)>1/2 \tag{20}$$

and on the negative side if

$$x_j(i)<0 \text{ or } z_j(i)<1/2. \tag{21}$$

A class is said to be **singly separated** by k hyperplanes if all the training data in the class are on the same side of the hyperplanes and no training data in other classes exist in the separated region. If training data in a class are divided into subsets such that each subset of the data is singly separated by hyperplanes, the class is said to be **plurally separated.**

Consider classification of three classes in the two-dimensional space shown in Fig. 5. The arrows attached to the three hyperplanes P1, P2, and P3 designate the positive sides of the hyperplanes, and each dot in the figure denotes a training datum. Since all the training data for class I are on the positive sides of P2 and P3, and no other data exist in this region, class I is singly separated by planes P2 and P3. Likewise class III is singly separated by hyperplanes P1, P2 and P3. Class II is plurally separated.

We can now prove the following:

Consider classification of the $n(1)$-dimensional data into $n(3)$ classes. If there are $n(2)$ hyperplanes in $n(1)$-dimensional space such that all the learning data in any class can be singly separated by a subset of the $n(2)$ hyperplanes, the classifier can be synthesized by the three-layered neural networks with $n(1)$ input, $n(2)$ hidden, and $n(3)$ output neurous.

Proof Let the $n(2)$ hyperplanes be

$$W_j(1)z(1)=0 \quad \text{for} \quad j=1, \ldots, n(2) \tag{22}$$

where $W_j(1)$ and $z(1)$ are defined by (3). Now let

$$z_j(2)=1/(1+\exp(-x_j(2)/T)),$$

and

$$x_j(2)=W_j(1)z(1) \quad \text{for} \quad j=1, \ldots, n(2) \tag{23}$$

and let them correspond to the first and second layers of the neural network shown in Fig. 1. According to the assumption, $z_i(2)$ corresponding to the training data is either

$$z_i(2)>1/2 \quad \text{or} \quad z_i(2)<1/2. \tag{24}$$

Thus by multiplying the weight vector $W_j(1)$ by a positive constant, we can set the values of $z_i(2)$ for the training data as either

$$z_i(2)=1 \quad \text{or} \quad z_i(2)=0. \tag{25}$$

(The above restriction is imposed to simplify the proof.)

Now determine the weights between output and hidden neurons. Let the inputs to the output neuron $x_i(3)$ be given by

$$x_i(3)=W_i(2)z(2) \quad \text{for} \quad i=1, \ldots, n(3). \tag{26}$$

The weight vector $W_i(2)$ should be determined so that $x_i(3)$ is $1-\varepsilon$ for the i-th class and ε for classes other than i. This can be achieved by satisfying

$$W_i(2)x(2)\geq\alpha \qquad \text{for class } i \tag{27}$$

$$W_i(2)z(2)\leq-\alpha \qquad \text{for classes other than } i. \tag{28}$$

If class i is separated by less than $n(2)$ hyperplanes as class I in Fig. 5 we can cancel the effect of the planes not contributing to the separation by setting the corresponding weights to zero. Thus we assume that class i is singly separated by $n(2)$ planes. According to this assumption, the outputs $z_1(2), \ldots, z_{n(2)}(2)$ for class i are uniquely determined. Therefore, assume that

$$(z_1(2), \ldots, z_{n(2)}(2))=(1, \ldots, 1, 0, \ldots, 0) \quad \text{for class } i, \tag{29}$$

namely, assume that the first s outputs are 1, and the remaining outputs are 0.

Now let

$$W_{ij}(2)=2\alpha \qquad \text{for } j=1, \ldots, s \tag{30}$$

$$W_{ij}(2)=-2\alpha \quad \text{for } j=s+1, \ldots, n(2). \tag{31}$$

Thus (27) and (28) become

$$W_{i,n(2)+1}(2)\geq\alpha-2s\alpha \quad \text{for class } i \tag{32}$$

$$-\alpha-2\alpha\sum_{j=1}^{s}z_j(2)+2\alpha\sum_{j=s+1}^{n(2)}z_j(2)\geq W_{i,n(2)+1}(2) \quad \text{for classes other than } i. \tag{33}$$

Since class i is singly separated, $z_j(2)=0$ holds for some j, $j=1, \ldots, s$ or, $z_j(2)=1$ for some j, $j=s+1, \ldots, n(2)$. Therefore, we can select weights $W_{i,n(2)+1}$ that satisfy (32) and (33); for example, $W_{i,n(2)+1}(2)=\alpha(1-2s)$. (End of proof)

From the proof we can see that **if class i is plurally separated, we can synthesize a four-layered neural network where the third layer classifies each of the separated regions and the fourth layer simply sums up the outputs of the third layer.**

To simplify the proof, the outputs of the hidden neurons are assumed to take the values 1 and 0. Allowing the hidden neurons to take analog values, we can extend the scope of the solution. Therefore, it may be possible to generate a three-layered network even if a class is plurally separated.

Now apply the above procedure to the classification of the two-dimensional points shown in Fig. 5. Determine the weights between the input and hidden neurons as the coefficients of the hyperplanes shown in the figure. The signs of the weights are taken so that the

Table 1 Hidden Neuron Outputs for Three Classes.

Output	Class I	Class II	Class III
Z1(2)	10	01	1
Z2(2)	11	10	1
Z3(2)	11	01	0

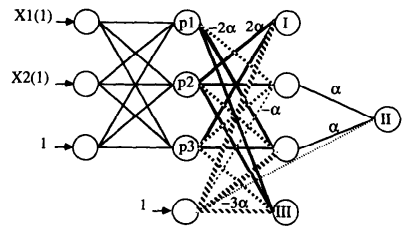


Fig. 6 A Neural Network Classifier for Fig. 5.

positive sides of the planes correspond to the regions indicated by arrows in the figure. Then by multiplying the weights by positive numbers, the outputs of the hidden neurons for each class are as shown in Table 1. For class I, $z_1(2)$ assumes the values 1 and 0, but since $z_2(2)$ and $z_3(2)$ do not change, $z_1(2)$ does not contribute to separation. Thus class I is singly separated by planes P2 and P3. Likewise, class III is singly separated by planes P1, P2, and P3, while class II is plurally separated. Thus the neural network is synthesized as shown in Fig. 6.

4. Speed-up of Learning

The synthesizing algorithm discussed in the previous section suggests the following new learning algorithm:

1) Determine the weights between the input and hidden neurons by finding the hyperplanes that separate classes.

2) Determine the weights between the hidden and output neurons by solving a set of simultaneous linear inequalities.

There is an algorithm [7] for determining the hyperplanes that separate classes, but it is not clear whether it is more powerful than the BP. A general way of solving a set of simultaneous linear inequalities is to use linear programming, but again it is not clear whether this is more powerful than the BP. Therefore, in this section we discuss a method of speeding up learning that is applicable to the BP, and the parallel forward-propagation algorithm (PFP) that we have proposed [8].

The easiest way to speed up learning is as follows:

If

$$z_{il}(3) > 1/2 + \varepsilon_1 \quad \text{or} \quad x_{il}(3) > \varepsilon_2 \quad \text{for } s_{il} = 1 \tag{34}$$

$$z_{il}(3) < 1/2 - \varepsilon_1 \quad \text{or} \quad x_{il}(3) < -\varepsilon_2 \quad \text{for } s_{il} = 0 \tag{35}$$

hold for all $i=1, \ldots, n(3)$, $l=1, \ldots, m$, where ε_1 and ε_2 are zero or positive values, and s_{il} is a desired output defined in (4), stop learning.

Let

$$M_i = \min_{l=1,\ldots,m} \{|x_{il}(3)|\} \tag{36}$$

and if $M_i < \alpha$, change $W_{ki}(2)$ as follows:

$$W_{ki}(2) \leftarrow (\alpha / M_i) W_{ki}(2) \quad k = 1, \ldots, n(2)+1. \tag{37}$$

If $M_i > \alpha$, there is no need to modify $W_{ki}(2)$.

If (34) and (35) hold, all the classes are separated. Therefore, by multiplying positive values with the weights between hidden and output neurons as shown in (37), we can satisfy the convergence criterion. This method is applicable to the BP and the PFP.

5. Numerical Calculations

We now show how the PEP is speeded up by using the convergence test given in (34) and (35), setting $\varepsilon_1 = \varepsilon_2 = 0$. (Hereafter we call this algorithm the MPFP.) The initial values of the weights between inputs and hidden neurons are taken so that the corresponding hyperplanes do not overlap, as follows:

$$|W_{kj}(1)| = Bias + \beta Rand \tag{38}$$

where *Bias* and β are constants, *Rand* is a uniformly distributed random variable in $[-0.5, 0.5]$, and the signs of the weights are given as follows:

$$W_{1,1}(1), \ldots, W_{1,n(1)}(1) = +, -, +, -, +, \ldots$$

$$W_{2,1}(1), \ldots, W_{2,n(1)}(1) = -, +, -, +, -, \ldots$$

Table 2 Average Numbers of Steps for a Parity Circuit (100 Trials).

β	Number of Steps for PFP	Number of Steps for MPFP
0.001	3.97	6.45
0.01	5.47	6.38
0.1	6.77	5.51
0.2	5.79	4.40
0.3	5.49	3.83
0.4	4.82	3.39
0.5	4.60	3.41

Bias=0.5, Converged 100 times

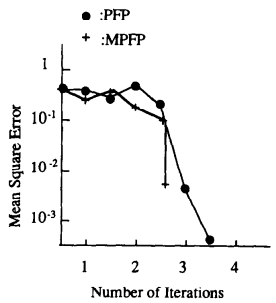


Fig. 7 Convergence Process of a Parity Circuit.

Table 3 Convergence Characteristics of Two-dimensional Points (Three Hidden Neurons, 100 Trials).

Bias	β	Three Classes		Four Classes	
		Convergence	Steps	Convergence	Steps
0.3	0.5	2	33.5	3	32.7
0.5	0.5	0	—	8	28.5
0.7	0.5	3	32.7	1	41.0
0.5	0.9	4	26.5	2	22.0
0.5	0.3	1	12.0	2	30.0
0.5	0.1	0	—	7	30.3
0.5	0.001	3	27.3	2	37.0

Convergence: The number at which the solution was obtained within 50 iterations
Steps: Average number of steps needed for convergence

Table 4 Convergence Characteristics of Two-dimensional Points (Four Hidden Neurons, 100 Trials).

Bias	β	Three Classes		Four Classes	
		Convergence	Steps	Convergence	Steps
0.5	0.5	17	23.3	8	24.6
0.5	0.3	13	26.8	14	32.3
0.5	0.1	14	22.3	21	28.9
0.5	0.001	13	27.1	18	34.7

Table 5 Convergence Characteristics of Two-dimensional Points (Five Hidden Neurons, 100 Trials).

Bias	β	Three Classes		Four Classes	
		Convergence	Steps	Convergence	Steps
0.5	0.5	40	21.9	40	21.6
0.5	0.3	36	26.3	42	27.3
0.5	0.1	47	23.1	41	24.1
0.5	0.001	39	24.0	61	19.3

$$W_{3,1}(1), \ldots, W_{3,n(1)}(1) = +, +, -, -, +, \ldots$$
$$W_{4,1}(1), \ldots, W_{4,n(1)}(1) = -, -, +, +, -, \ldots$$
$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$

The weights between hidden and output neurons are given by

$$W_{jk}(2) = Bias + \beta Rand \tag{39}$$

This initial value selection makes the PFP converge in one iteration under the best condition for the *exclusive-or* circuit. In the following, ε in (9) is set to 0.01 and the calculation is stopped when (34) and (35) are satisfied.

5.1 Parity Circuit

The parity circuit [9] outputs one when the inputs are all zero or the number of inputs that are one is even. Otherwise it outputs zero. Thus it can be viewed as a pattern classifier. Table 2 shows the average numbers of steps needed for convergence for the PFP and the MPFP in 100 trials with two hidden neurons. Solutions are obtained for all the initial values tried. The convergence characteristics are almost the same for the two methods. Figure 7 shows the best convergence process for the two methods. One iteration consists of corrections of the weights between input and hidden neurons followed by those of the weights between hidden and output neurons. Using the MPFP, the convergence test given by (34) and (35) holds at the second iteration.

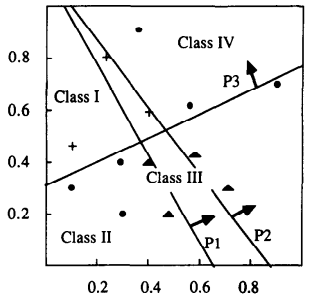


Fig. 8 Hyperplanes Learned with Four Patterns and Three Hidden Neurons.

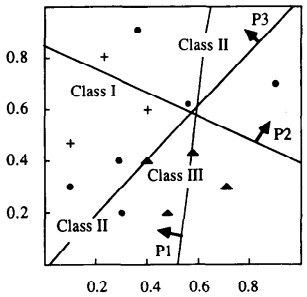


Fig. 9 Hyperplanes Learned with Three Patterns and Three Hidden Neurons.

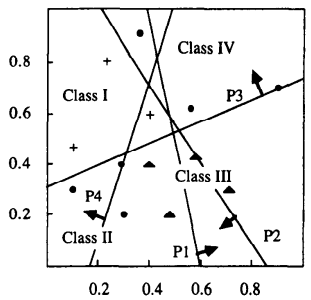


Fig. 10 Hyperplanes learned with Four Patterns and Four Hidden Neurons.

5.2 Separation of Points in a Plane

The original PFP does not converge for pattern classification of the two-dimensional points shown in Fig. 5. A test case is added in which class II is divided into two different classes, so that all four classes become

Table 6 Convergence Characteristics of Number Recognition (100 Trials).

Bias	β	No. of Hidden Neurons	Convergence	Steps
0.3	0.001	4	0	—
0.3	0.001	6	21	26.1
0.3	0.001	8	49	21.8
0.3	0.001	10	93	15.1

singly separated. Tables 3 and 4 show the convergence characteristics for three and four classes, changing the hidden neurons from three to five, where the maximum number of iterations is 50. When the number of hidden neurons is three, the convergence is very bad, but a solution is obtained for three classes. Figure 8 shows hyperplanes obtained by learning for four classes and three hidden neurons. By moving planes P1, P2, and P3 in parallel, that is, by changing the weights corresponding to the bias terms, the planes become separation planes. Figure 9 shows the hyperplanes for three classes and three hidden neurons. In this case the planes do not correspond to the separation planes. Figure 10 shows the hyperplanes for four classes and four neurons. The hyperplanes P1 to P3 become separation planes by changing the weights which correspond to the bias terms.

5.3 Number Recognition

To recognize ten different numbers, 12 features are extracted from the original image. The neural network with 12 inputs and 10 outputs are learned by using 100 training data, changing the number of hidden neurons from four to 10. For this problem the original PFP does not converge. Table 6 shows the results. The maximum number of iterations was set to 50. With four hidden neurons we cannot obtain a solution. As the number of hidden neurons increases, solutions are obtained more frequently.

The size of the problem used in our study is relatively small. In a future study we need to clarify the convergence characteristics for large networks.

6. Conclusions

We discussed how to synthesize multi-layered neural networks for pattern recognition. First the saturation characteristics of the sigmoid function were shown to be essential for pattern recognition. It was then shown that, if a class is separated by hyperplanes into a single region, three-layered neural networks can be synthesized to recognize that class, and that if a class is separated by hyperplanes into several regions, the class can be recognized by a four-layered neural network. As a result of this analysis, a method to accelerate learning was proposed and its validity was tested by a parity circuit, classification of two-dimensional points, and number recognition.

Acknowledgements

We are grateful to Dr. J. Kawakami and Dr. Y. Morooka for their helpful discussions and encouragement.

References

1. RUMELHART, D. E. et al. Parallel Distributed Processing, **1** and **2**, *MIT Press*, Cambridge, MA., 1986.
2. FUNAHASHI, K. On the Approximate Realization of Continuous Mapping by Neural Networks, *Neural Networks*, **2**, 3 (1989), 183–192.
3. CHESTER, D. L. Why Two Hidden Layers Are Better Than One, *Proc. IJCNN-90-WASH-DC*, **1** (January 1990), 265–268.
4. MAKHOUL, J. et al. Formation of Disconnected Decision Regions with a Single Hidden Layer, *Proc. IJCNN-89*, **I** (June 1989), 455–460.
5. RUJAN, P. A Geometric Approach to Learning in Neural Networks, *Proc. IJCNN-89*, **II** (June 1989), 105–109.
6. LIPPMANN, R. P. An Introduction to Computing with Neural Nets, *IEEE ASSP Magazine* (April 1987), 4–22.
7. SIMURA, M. Pattern Recognition and Learning Machines, Syoukoudou (in Japanese), Tokyo, 1972.
8. ABE, S. Learning by Parallel Forward Propagation, *Proc. IJCNN-90*, **3**, San Diego (June 1990), 99–104.
9. BABA, N. A New Approach for finding the Global Minimum of Error Function of Neural Networks, *Neural Networks*, **2** (1989), 367–373.

(Received September 25, 1990; revised January 24, 1991)