Neural Networks as Nonlinear Structure–Activity Relationship Analyzers. Useful Functions of the Partial Derivative Method in Multilayer Neural Networks

Tomoo Aoyama† and Hiroshi Ichikawa*,‡

Hitachi Computer Engineering Company, Ltd., Horiyamashita, Hadano, Kanagawa 259-13, Japan, and Hoshi College of Pharmacy, Shinagawa, Tokyo 142, Japan

Received March 20, 1992

The operation of the perceptron-type neural network can be regarded as a function which transforms an input vector to another (output) vector. We have presented the analytical formula for the partial derivative of this function with respect to the elements of the input vector. Using numerical data, we have examined the accuracy, independency of the elements of the input vector, and recognition ability of a function in mixed functions. It was shown that the partial differential coefficients of output strength with respect to input parameters were useful to analyze the relationship between inputs and outputs in the neural network and characteristics of each data in the set of data.

INTRODUCTION

The neural network (or so-called neuro-computer) is a data-processing method which was suggested by the neuron's operation in the brain[1] as information is accumulated in terms of the connection strength between neurons. The characteristics of the operation of the neural network have been found to be suitable for data processing in which the relationship between the cause and its results cannot be defined.

The operation involves parallel-distributed processing and has recently attracted considerable attention since this method may break through the bottleneck of Neumann-type digital computer to jump to a new generation of computer architecture. There are a number of types of network connections such as the perceptron type,[2] the Hopfield type,[3] and the Boltzmann-machine type.[4,5] Since the learning procedure, which is called the back-propagation method, has been established for the perceptron-type network,[6] most studies for chemical application have dealt only with this structure.

The perceptron was first proposed by Rosenblatt in 1957 as a model of self-organization, i.e. learning, in recognition.[2] Although this method had attracted considerable attention and was studied by a number of researchers, in 1969 Minsky and Papert pointed out the limit of the two-layer perceptron's operation in linear separation:[7] for example, even the simple exclusive-OR function cannot be performed by this method. This problem was solved recently by constructing a multilayer network with a back-propagation algorithm.[6] In addition, the perceptron-type neural network was found to perform far more functions than expected.[8-10]

"Neural networks" are now regarded as hardware structures, as compared to the usual serial processing unit. Fortunately, most of the operations of a neural network can be simulated using a digital computer. Besides, such a simulation is more convenient in application studies than rigid hardware construction, since one can easily change the structure and the modes of operation.

In an attempt to apply the perceptron-type neural networks to the structure–activity relationship (SAR) studies, we have investigated the operation of the neural networks. So far the following characteristics have been revealed. (1) The neural networks give an excellent performance in pattern recognition; such an ability can readily be applied to the SAR studies of drugs resulting in a better graded classification and prediction than the other conventional methods such as the linear-learning machine, cluster analysis, and adapted least-squares (ALS) methods.[11,12] (2) Usually, the number of neurons of the output layer equals that of the classification. We have found that if the number of neurons of the output layer is set to be one, a neuron with the linear activation function which is allowed to take an analog value, this type of neural network (named the MR-type neural network) operates as a generalized, i.e., nonlinear, multiregression analysis. The MR-type neural network is easily applied to the QSAR analysis. Because of its nonlinear operation, the results were remarkable in both fitting and prediction.[13] (3) The operation of the neural network is basically carried out in a nonlinear manner.[14] The nonlinearity makes it difficult to correlate the output strength with any input parameters. It has been seriously pointed out that although the classification is beautifully carried out, there is no way to know why such a decision was made. This problem was solved by two ways. (4) Since the operation of the neural network can be defined mathematically, it is possible to trace the propagation of the infinitesimal change of the input values to the neurons in the output layer. Namely, the partial derivatives of the output intensity with respect to the input parameter ($\partial O/\partial x$) can be obtained.[15] (5) Another way was accidentally brought forth while we tried to find what happens to the network structure if a forgetting effect is incorporated in the learning phase. The results were remarkable: The connections between neurons are greatly simplified and one can easily trace the flow of the input information to the output.[16]

Meanwhile, we have discussed the basic operation characteristics of the neural networks and correlated the new methods to the conventional linear multiregression analysis and adapted least-squares method in fitting and classification.[13,15] As we have shown,[15] it is possible to take the partial derivative of the operation. Although the application of this method seems to be wide-spread, the basic characteristics including the reliability was not known. This paper discloses those problems and gives some applications of the derivative method to (graded) classification in SAR studies.

MATHEMATICAL FOUNDATION

Basic Theory. Shown in Figure 1 is the perceptron-type three-layer network: The circles are neurons which are, in simulation, variables taking a value ranging from 0 to 1. The data is input to A and output from B. Usually, the number of neurons in the input layer is set to be equal to that of the

† Hitachi Computer Engineering Co., Ltd.
‡ Hoshi College of Pharmacy.

0095-2338/92/1632-0492$03.00/0

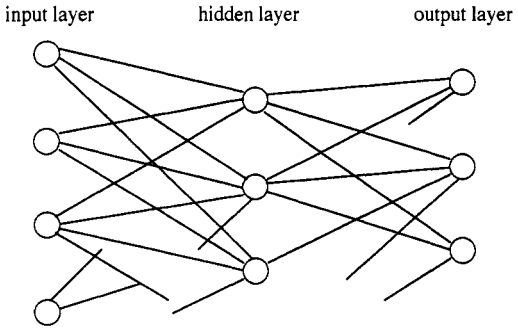


Figure 1. A three-layer perceptron-type (hierarchy) network.

parameters (descriptors) plus 1 (bias), while that of the output layer is equal to the number of categories. The number of neurons in the second layer is arbitrary. However, a small number may reduce the resolution ability of the network whereas a large number consumes considerable time in the training phase. The recommended number is somewhere between the number of input neurons and double their number, unless the number of the input parameters is small (e.g. less than ca. 5).

The value of a neuron (O_j) at the second or third layer can be expressed by eq 1

$$O_j = 1/[1 + \exp(-\alpha y_j)] = f(y_j) \tag{1}$$

$$y_j = (\sum W_{ij}x_i) - \theta_j$$

where x_i is one of the values of a neuron at the first or second layer; W_{ij}, an element of the weight matrix, expresses the weight value of the connection between neurons i and j, and takes either a positive or a negative value; α is a parameter which expresses the nonlinearity of the neuron's operation; θ_j is a threshold value for neuron j. Usually, one needs not to set this value if constant 1 is always given as a bias to one of the neurons in the input layer. The reason is that the connection weights between the neuron with constant 1 and any of the neurons in the second layer are optimized during the learning (training) phase to play the same role as that of the optimized θ. The sigmoid function is not unique as the activation function, $f(y)$, and can be replaced by any function with an appropriate operation if it is differentiable. On feeding the input data, the value of each neuron expressed by eq 1 is synchronously renewed.

Given N neurons at the first layer. A set of the input data can be expressed by a vector with N elements for the input neurons, which is called here an "input pattern". Likewise, the output data can also be regarded as a vector and be called an "output pattern" (O_j). The vector which is compared with an output pattern to fix W_{ij} is called a "training pattern" (t_j).

The training of the network is based on the following equations:

$$\delta W_{ij}^{(n-1,n)} = -d_j^{(n)} x_i \epsilon \tag{2}$$

$$d_j^{(3)} = (O_j - t_j) g'(y_j) \tag{3a}$$

$$d_j^{(2)} = (\sum W_{jl}^{(2,3)} d_l^{(3)}) f'(y_j) \tag{3b}$$

Here, $f'(\,)$ and $g'(\,)$ are the derivative functions of the activation functions for the second- and third-layer neurons, $f(\,)$ and $g(\,)$, respectively, and ϵ is a parameter which determines the shift for correction in back-propagation. The superscripts on W and d indicate the relevant layer(s), and thus eq 3a is used only to correct the connection between the second and third (output) layers while 3b is for another connection. If f and g are the sigmoid functions, the derivative functions, f' and g' in eq 3 are

$$f'(y_j) = g'(y_j) = f(y_j)[1 - f(y_j)]\alpha \tag{4}$$

In the above equations, both ϵ and α can be set independently of the layer.

Since the value of each neuron is defined between 0 and 1, the given data must be appropriately scaled within the defined region. Here it should be noted that if the value of a neuron in the input layer is zero, the connections from such a neuron are always null, i.e., the information from that neuron is not propagated to the second and the third layers. To avoid this difficulty, the smallest value should be set at slightly larger than zero, typically 0.1. We, therefore, used the following scaling equation:

$$\tilde{p} = \frac{(q_{max} - q_{min})p + q_{min}p_{max} - q_{max}p_{min}}{p_{max} - p_{min}} \tag{5}$$

where q_{max} and q_{min} are the maximum and minimum values of scaling, p is the element of the input pattern to be scaled, and p_{max} and p_{min} are the maximum and minimum values of the elements of the actual input patterns. Finally, each element of the training pattern should also be between 0 and 1.

There may be a number of ways to express the graded categories. For example, we simply used the pattern (0,0,0,0,1) for the first and (1,0,0,0,0) for the fifth grade in the five-graded classification. By doing so, one can observe the degree of contamination in the determined class from other classes.

Training is carried out according to the above back-propagation algorithm until the error function

$$E = \sum (O_j - t_j)^2 \tag{6}$$

becomes small enough. The initial weight matrices are created by using random numbers as all elements take the values between −1 and 1. Even when M sets of the input and training patterns are given, all of the output patterns can be made close enough to the training patterns by iteration through eqs 1 and 2 owing to the convergence theorem of perceptron.[15] If convergence is attained, then the neural network has an ability to classify the input patterns into M groups.

Introduction of Linear Operation into the Neural Network. The neural network based on eq 1 performs a nonlinear operation. As already pointed,[13] a nonlinear operation is not always convenient in practical application, and therefore, it may be preferable if a linear operation can be introduced into the neural network. This is possible, defining a new activation function as a combination of eq 1 and y_j as

$$O_j = \beta h(y_j) + (1 - \beta)y_j = f(y_j) \tag{7}$$

$$h(y_j) = 1/[1 + \exp(-\alpha y_j)]$$

$$y_j = (\sum W_{ij}x_i) - \theta_j$$

Here, parameter β expresses the mixing degree of the linear operation to the nonlinear operation, and therefore, by changing β, one can mix the linear operation into the network at any level. If β is set to be 0, the network can be expected to perform the linear operation correctly. In practice, however, a problem arises: if β is set close to 0, the neurons in the second layer easily exceed the defined region (0–1), resulting in a destruction of learning. This difficulty can be removed

Table I. Calculated Derivatives of Simple Functions by the Neural Network[a]

$y = 2x$			$y = x^2$			
x	y	y'	x	y_{calc}	y_{calc}	y'_{theort}
0	0.14	1.93	−10	99.97	−18.00	−20
1	2.08	1.94	−9	82.14	−17.53	−18
2	4.03	1.96	−8	65.15	−16.33	−16
3	5.99	1.97	−7	49.67	−14.55	−14
4	7.96	1.98	−6	36.18	−12.40	−12
5	9.94	1.99	−5	24.93	−10.10	−10
6	11.93	1.99	−4	15.98	−7.83	−8
7	13.93	2.00	−3	9.24	−5.68	−7
8	15.93	2.01	−2	4.56	−3.70	−4
9	17.94	2.01	−1	1.80	−1.84	−2
10	19.95	2.02	0	0.85	−0.07	0
11	21.97	2.02	1	1.67	1.71	2
12	23.99	2.02	2	4.30	3.57	4
13	26.01	2.02	3	8.85	5.56	6
14	28.02	2.02	4	15.49	7.74	8
15	30.04	2.01	5	24.38	10.06	10
16	32.05	2.01	6	35.64	12.45	12
17	34.05	2.00	7	49.24	14.72	14
18	36.05	1.99	8	64.95	16.62	16
19	38.03	1.97	9	82.27	17.89	18
20	40.00	1.96	10	100.45	18.32	20

[a] The network structure was $N(2,10,1)$, where $\alpha = 4$, $\beta = 1$, and the threshold of the convergence $<10^{-5}$ in terms of the scaled units.

Table II. Derivatives in Classification[a]

	variable	
point	x	y
1 (1,10)	0.00 (0.00)	0.00 (0.00)
2 (0.83,9.17)	0.00 (0.00)	0.00 (0.00)
3 (1.67,8.33)	0.00 (0.00)	0.00 (0.00)
4 (2.50,7.50)	0.00 (0.00)	0.00 (0.00)
5 (3.33,6.67)	−0.03 (0.03)	0.03 (−0.03)
6 (4.17,5.83)	−0.75 (0.74)	0.75 (−0.74)
7 (5.83,4.17)	−0.74 (0.73)	0.74 (−0.73)
8 (6.67,3.33)	−0.04 (0.03)	0.04 (0.03)
9 (7.50,2.50)	0.00 (0.00)	0.00 (0.00)
10 (8.33,1.67)	0.00 (0.00)	0.00 (0.00)
11 (9.17,0.83)	0.00 (0.00)	0.00 (0.00)
12 (10,1)	0.00 (0.00)	0.00 (0.00)

[a] The network structure was $N(3,10,2)$, where $\alpha = 1$, $\beta = 1$, and the threshold of the convergence $<10^{-3}$ in terms of the scaled units. Points from 1 to 6 were used to train as class 1 while those from 7 to 12, as class 2. The values in parentheses are the derivatives for class 2.

by introducing a concept of "neuron fatigue" into the usual back-propagation learning method.[14] Our experiences tell us that the smallest value of β is around 0.5 by the learning procedure without the "neuron fatigue" procedure.

The training of the network is similarly carried out based on eqs 2 and 3 until the sum of the squared errors, $\sum(O_j - t_j)^2$, becomes small enough. If the new activation function is adopted, the derivative function, f', in eq 3 is

$$f'(y_j) = \alpha\beta h(y_j)[1 - h(y_j)] + 1 - \beta \tag{8}$$

In the above equations, both ϵ and α, and even β, can be set independently of the layer.

How To Obtain Partial Differential Coefficients by the Neural Network. Since the operation of the perceptron-type neural networks is completely defined by the mathematical formula, it is possible to take the partial derivative of an output to any input parameters. Since

$$\delta y_j = W_{ij}^{(1,2)} \delta x_i \tag{9}$$

and

$$\delta O_j = f'(y_j)\delta y_j \tag{10}$$

the partial derivative of the output in the second layer becomes

$$\frac{\partial O_j^{(2)}}{\partial x_i} = f'(y_j) W_{ij}^{(1,2)} \tag{11}$$

Likewise, the partial derivatives of the output in the third layer with respect to an input parameter is given by

$$\frac{\partial O_j^{(3)}}{\partial x_i} = \sum_k f'(y_k) W_{ik}^{(1,2)} g'(y_j) W_{kj}^{(2,3)} \tag{12}$$

where f' and g' are, respectively, the differential functions of the activation function in the second and third layers while the superscript on W expresses the layer's order. The expression using eq 7 as the activation function turns out to be

$$\frac{\partial O_j^{(3)}}{\partial x_i} = \sum_k [\beta^{(2)}\alpha^{(2)} h(y_k)\{1 - h(y_k)\} + (1 - \beta^{(2)})] W_{ik}^{(1,2)} \times [\beta^{(3)}\alpha^{(3)} h(y_j)\{1 - h(y_j)\} + (1 - \beta^{(3)})] W_{kj}^{(2,3)} \tag{13}$$

Note that when β is near 1, the partial derivative, $\partial O_j/\partial x_i$, approaches 0 as the output value nears 0 or 1. This character stems from the sigmoid function. The computer program has been submitted to QCPE.[16]

RESULTS AND DISCUSSION

Unlike the linear multiregression analysis, we cannot find the analytical method to investigate the reliability of the results by the neural network. We may, therefore, discuss the tendencies of the fitting curve or separation surface by using concrete numerical data, although such a tendency is not a proof of the reliability. To express the network structure we used notation $N(l,m,n)$, where l, m, and n are the numbers of neurons in the first, second, and third layers. The β value was always set to be 1, i.e., the pure sigmoid function is used as the activation function of the neurons in the second layer. The function of the neurons in the third layer depends on the network's operation. Namely, if it was fitting, the linear function was adopted (the MR-type network) while it was classification the sigmoid function was used. The α value was fixed to be 1 (classification) or 4 (fitting). The α value and the number of neurons in the second layer are not very essential if the convergence in the learning phase is attained. The threshold of the convergence in the learning phase was less than 10^{-5} (fitting) or 10^{-3} (classification) in terms of the scaled units.

a. Accuracy of the Derivatives Obtained by the Neural Network. Table I shows the analytical and calculated derivatives for $y = x$ and $y = x^2$ functions together with the reproduced values by the network. The network structure was $N(2,10,1)$, where, as a rule, one of the neurons in the first layer was used as a bias. The 21 points, 0, 1, 2, ..., 20, were used to train to simulate the $y = 2x$ function and the 21 points, −10, −9, ..., −1, 0, 1, ..., 9, 10, were used for $y = x^2$ functions.

One may understand that the network well reproduced the function's value. Except for the terminal points, the calculated derivatives are within the error around 5%, and such errors increase at the terminal and extreme points. According to our experience, this is observed very generally and must be regarded as a defect by the usual back-propagation learning method.

Table III. Fluctuation of Predicted Function's Values and Their Derivatives by Change of the Dummy Neuron's Value in $y = 2x$ [a]

	function's values					derivatives				
x[b]	−10[c]	−5	0	5	10	−10	−5	0	5	10
0.5	1.17	0.67	0.44	0.59	1.19	1.86	1.90	1.90	1.88	1.83
1.5	3.05	2.58	2.36	2.49	3.04	1.89	1.93	1.94	1.92	1.87
2.5	4.95	4.53	4.32	4.42	4.93	1.92	1.97	1.98	1.95	1.90
3.5	6.89	6.52	6.31	6.39	6.85	1.95	2.00	2.01	1.99	1.94
4.5	8.85	8.53	8.34	8.40	8.80	1.97	2.03	2.04	2.02	1.96
5.5	10.83	10.57	10.40	10.43	10.77	1.99	2.05	2.07	2.05	1.99
6.5	12.83	12.64	12.48	12.50	12.77	2.01	2.07	2.10	2.07	2.01
7.5	14.85	14.72	14.59	14.58	14.79	2.03	2.09	2.12	2.09	2.03
8.5	16.88	16.82	16.71	16.68	16.83	2.04	2.11	2.13	2.11	2.04
9.5	18.92	18.93	18.84	18.79	18.87	2.04	2.11	2.14	2.12	2.05
10.5	20.97	21.04	20.99	20.91	20.92	2.05	2.12	2.15	2.12	2.05
11.5	23.02	23.16	23.13	23.04	22.98	2.05	2.12	2.15	2.13	2.06
12.5	25.07	25.28	25.28	25.16	25.03	2.04	2.12	2.14	2.12	2.05
13.5	27.11	27.39	27.42	27.28	27.08	2.04	2.11	2.13	2.11	2.05
14.5	29.14	29.49	29.55	29.39	29.12	2.03	2.09	2.12	2.10	2.03
15.5	31.16	31.58	31.66	31.48	31.15	2.01	2.08	2.10	2.09	2.02
16.5	33.16	33.64	33.75	33.56	33.16	1.99	2.05	2.08	2.06	2.00
17.5	35.14	35.68	35.82	35.61	35.15	1.97	2.03	2.06	2.04	1.98
18.5	37.09	37.70	37.86	37.64	37.11	1.94	2.00	2.02	2.01	1.95
19.5	39.02	39.68	39.87	39.63	39.05	1.91	1.97	1.99	1.98	1.92

[a] The network structure was $N(3,10,1)$, where one of the neurons in the first layer was used as the dummy input. Other network parameters and convergence condition were the same as those in Table I. [b] Predicted point. [c] Indicates that the value of −10 was input to the dummy input.

Table IV. Fluctuation of Predicted Function's Values and Their Derivatives by Change of the Dummy Neuron's Value in $y = x^2$ [a]

	predicted values					derivatives				
x[b]	−10[c]	−5	0	5	10	−10	−5	0	5	10
−9.5	90.93	90.89	90.76	90.66	90.69	−17.89	−17.98	−18.00	−17.96	−17.87
−8.5	73.44	73.32	73.19	73.14	73.25	−16.99	−17.04	−17.04	−16.99	−16.92
−7.5	57.17	57.01	56.89	56.88	57.05	−15.47	−15.49	−15.47	−15.43	−15.39
−6.5	42.65	42.49	42.39	42.41	42.60	−13.51	−13.50	−13.48	−13.45	−13.44
−5.5	30.24	30.09	30.02	30.05	30.24	−11.31	−11.28	−11.25	−11.24	−11.25
−4.5	20.06	19.96	19.91	19.95	20.12	−9.04	−9.00	−8.97	−8.97	−9.00
−3.5	12.14	12.08	12.05	12.09	12.22	−6.83	−6.78	−6.76	−6.76	−6.81
−2.5	6.37	6.36	6.35	6.38	6.46	−4.73	−4.68	−4.66	−4.68	−4.73
−1.5	2.63	2.67	2.68	2.69	2.72	−2.76	−2.72	−2.70	−2.72	−2.78
−0.5	0.81	0.89	0.91	0.90	0.87	−0.89	−0.85	−0.84	−0.87	−0.93
0.5	0.84	0.96	0.98	0.94	0.85	0.93	0.97	0.97	0.94	0.88
1.5	2.68	2.84	2.87	2.79	2.64	2.77	2.80	2.80	2.77	2.72
2.5	6.40	6.59	6.61	6.51	6.31	4.68	4.71	4.71	4.69	4.63
3.5	12.08	12.31	12.33	12.21	11.96	6.71	6.75	6.75	6.73	6.68
4.5	19.86	20.12	20.16	20.02	19.73	8.87	8.92	8.93	8.92	8.88
5.5	29.86	30.17	30.23	30.07	29.75	11.14	11.19	11.22	11.21	11.19
6.5	42.15	42.52	42.60	42.45	42.11	13.43	13.49	13.52	13.52	13.51
7.5	56.67	57.09	57.21	57.07	56.72	15.57	15.63	15.67	15.68	15.67
8.5	73.16	73.65	73.81	73.68	73.33	17.34	17.41	17.45	17.46	17.46
9.5	91.16	91.71	91.90	91.80	91.44	18.53	18.59	18.63	18.65	18.65

[a] The network structure was $N(3,10,1)$, where one of the neurons in the first layer was used as the dummy input. Other network parameters and convergence condition were the same as those in Table I. [b] Predicted point. [c] Indicates that the value of −10 was input to the dummy input.

Table II shows the derivatives in case of classification. We used the $y = -x + 10$ function; 12 equally divided points of the region from (0,10) to (10,0) on the line were selected and points from (0,10) to 4.17,5.83) were trained as class 1, other points as class 2. The network structure was $N(3,10,2)$. As the center (5,5) is the inflection point, derivatives around the center appear to be steep gradients, while those around the trained points are zeros. The results are reasonable and satisfactory.

b. Is the Independency between Input Neurons Kept? When the input parameters are independent of each other, one may wonder whether or not the decision or derivative given by the neural network is influenced by the values of other input neurons. This should be severely checked in the interpolated values, since they are not the trained points. To know such an independency we introduced dummy input neurons by which the predictions and decisions are checked. We used the same network structures as above.

This time the 21 points with −10 for the dummy neuron in addition to those with 10 for the dummy neuron were trained at the same time. By doing so, one can teach the network that the dummy neuron is independent of the decision. Then intermediate points between the training points were predicted by the network, where various values for the dummy neuron, −10, −5, 0, 5, and 10, were input to see the influence of the dummy neuron. Table III shows the results. The predicted values are not very accurate at both ends, and fluctuations by the dummy neuron becomes larger near both ends, especially at the low end.

Table IV shows the case of the $y = x^2$ function. Since the neural network is good at treating the nonlinear correlation, the fluctuation in both the predicted and derivative values is smaller than that of linear relation. However, one can see that as a rule, near the inflection point the fluctuation tends to be large.

Table V. Fluctuation of Predicted Values and Derivatives by Change of the Dummy Neuren's Value in Classification[a]

	predicted values					derivatives				
point[b]	10[c]	5	0	−5	−10	10	5	0	−5	−10
(0,10)	0.978	0.978	0.978	0.977	0.977	0.065	0.065	0.064	0.064	0.064
(0.833,9.167)	0.966	0.966	0.966	0.966	0.965	0.118	0.117	0.116	0.115	0.114
(1.667,8.333)	0.944	0.944	0.944	0.944	0.943	0.223	0.220	0.218	0.215	0.213
(2.5,7.5)	0.901	0.902	0.902	0.902	0.902	0.428	0.422	0.417	0.412	0.407
(3.333,6.667)	0.821	0.823	0.823	0.824	0.825	0.784	0.774	0.765	0.756	0.748
(4.167,5.833)	0.686	0.687	0.689	0.691	0.692	1.227	1.219	1.211	1.202	1.192
(5.0,5.0)	0.500	0.502	0.504	0.506	0.508	1.446	1.448	1.449	1.449	1.448
(5.833,4.167)	0.315	0.316	0.318	0.319	0.320	1.209	1.219	1.229	1.238	1.246
(6.667,3.333)	0.180	0.180	0.180	0.181	0.182	0.766	0.775	0.784	0.793	0.802
(7.5,2.5)	0.100	0.100	0.100	0.100	0.100	0.419	0.424	0.429	0.434	0.439
(8.333,1.667)	0.058	0.057	0.057	0.057	0.057	0.220	0.221	0.224	0.226	0.228
(9,167,0.833)	0.035	0.035	0.034	0.034	0.034	0.117	0.118	0.118	0.119	0.120
(10,0)	0.023	0.023	0.022	0.022	0.022	0.065	0.066	0.066	0.066	0.066

[a] Only two points, (0,10) and (10,0), were used to train as classes 1 and 2, respectively. The network structure was $N(4,10,2)$. One of the first-layer neurons was used as the dummy input. Other network parameters and convergence conditions were the same as those in Table II. [b] Predicted point. [c] Indicates that the value of −10 was input to the dummy input.

Table VI. Example of Input and Training Data To Isolate Individual Functions Out of Their Combined Function

	input data		training data
sample no.	x	y	$(x + 2y)$
1	8.5	0.6	9.7
2	0.1	2.0	4.1
3	9.3	4.1	17.5
4	0.0	0.6	1.2

The results for the classification were shown in Table V where only two points, (0,10) and (10,0), were trained. As one can see, the fluctuations in both cases are small enough to say that the independency is virtually maintained.

From above results, it may be generally said that if the independency of input parameters is properly incorporated in the network by training, the predicted values and their derivatives are not influenced by the values of other input parameters. However, a small amount of fluctuation appears at/near the terminal and extreme points.

c. Isolation of Functions Out of the Mixed Functions. As already described, it is possible to take the partial derivatives of the output strength with respect to each input parameter. This means that isolation of individual linear functions out of the mixed function is possible. We performed this separation test to figure out the minimum sample number to isolate according to the following procedures. For example, let us consider the isolation of x and $2y$ out of the $x + 2y$ function. As shown in Table VI,[9] using random numbers for x and y which are the input pattern, the values for $x + 2y$ are obtained and used as the training pattern. By obtaining the partial derivatives, one can know the relationship, $x + 2y$. The derivatives are averaged by the number of sample data.

Table VII shows the results for $x + 2y$, $x - y$, $x + 2y + 3z$, $x + y^2$, and $x - y + z^2$, where the values in parentheses show the maximum deviations from the mean values. Although these derivatives may vary according to the random numbers, one can see the neural network can detect the individual functions by a relatively small number of data. Of course, such a number depends on the complexity of the mixed function. In a simple relationship like $x + 2y$, only 5 sets of data seems to be enough to separate them. However, in the rather complex $x - y + z^2$ function, 15 sets of random numbers were needed.

d. Recognition of Two Similar Functions. As mentioned in the former section, the neural network easily recognizes the two functions, $x + 2y$. We then examined the limit of such separation ability using the function, $x + ay$, where a was changed from 0.6 to 0.9. In order to avoid the problem of scarcity of data, we used 40 sets of random numbers independently for x and y. The results are shown in Table VIII. One may understand that if the data is sufficient, the

Table VII. Isolation of Individual Functions out of the Mixed Function[a]

sample no.	5	10	15	20	30
$x + 2y$					
x'	0.953 (0.188)	0.979 (0.121)	0.977 (0.143)	0.985 (0.159)	0.987 (0.141)
y'	1.946 (0.273)	1.953 (0.137)	1.949 (0.161)	1.949 (0.134)	1.970 (0.136)
$x - y$					
x'	1.433 (0.388)	0.972 (0.077)	0.973 (0.093)	0.985 (0.066)	0.984 (0.111)
y'	−0.674 (0.197)	−0.962 (0.085)	−0.976 (0.101)	−0.981 (0.086)	−0.996 (0.126)
$x + 2y + 3z$					
x'	0.848 (0.411)	0.979 (0.254)	0.968 (0.200)	0.989 (0.254)	1.001 (0.168)
y'	1.636 (0.690)	1.928 (0.230)	1.949 (0.260)	1.931 (0.339)	1.980 (0.301)
z'	3.263 (1.269)	2.915 (0.558)	2.962 (0.460)	2.921 (0.441)	2.960 (0.410)
$x + y^2$					
x'	1.646 (4.014)	1.111 (0.528)	1.055 (0.081)	1.007 (0.109)	1.009 (0.072)
$x - y^2$					
x'	1.656 (3.953)	0.963 (0.622)	1.073 (0.442)	1.010 (0.446)	0.994 (0.366)
$x - y + z^2$					
x'	−0.350 (1.052)	0.449 (0.370)	1.065 (0.188)	1.008 (0.315)	1.051 (0.157)
y'	0.005 (0.965)	−0.406 (0.343)	−0.936 (0.430)	−1.025 (0.169)	−1.009 (0.193)

[a] The network structure was $N(n,10,1)$; $\alpha = 4$, $\beta = 1$, and the convergence condition $<10^{-5}$ in terms of the scaled units. Averaged derivatives are shown. The values in parentheses are the maximum deviations from the mean values.

Table VIII. Recognition Test in $x + ay$ [a]

a	1	2	ratio
0.6	0.982	0.595	0.606
0.7	0.991	0.689	0.695
0.8	0.989	0.795	0.804
0.9	0.998	0.895	0.897

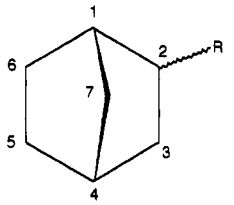
[a] The network structure was $N(3,10,1)$. Other network parameters and the convergence condition were the same as those in Table I.

Table IX. Recognition of Two Similar Linear Functions by Correlated Input Data[a]

	trial					
	1	2	3	4	5	6
			$\sigma^2 = 0.05$ ($a = 1.1$)			
x	1.009	1.058	1.034	1.082	1.103	1.112
y	1.079	1.034	1.052	1.005	0.981	0.981
			$\sigma^2 = 0.1$ ($a = 1.1$)			
x	1.022	1.025	0.953	0.976	0.977	1.022
y	1.070	1.064	1.141	1.115	1.105	1.072
			$\sigma^2 = 0.2$ ($a = 1.1$)			
x	1.009	1.039	0.988	0.977	1.012	0.978
y	1.082	1.055	1.105	1.115	1.074	1.104
			$\sigma^2 = 0.05$ ($a = 1.3$)			
x	1.006	1.082	1.110	1.120	1.115	1.067
y	1.281	1.209	1.174	1.116	1.168	1.225
			$\sigma^2 = 0.05$ ($a = 2.0$)			
x	1.085	1.106	1.153	1.178	1.130	1.230
y	1.900	1.881	1.829	1.807	1.845	1.764

[a] All conditions concerning the network were the same as those in Table VIII.

Table X. Derivatives of Norbornane and Norbornene[a]



substituent	compd no. exo	compd no. endo
CH_3	1	14
NH_2	2	15
OH	3	16
COOH	4	17
CH_2OH	5	18
CH_3, =O(3)[b]	6	19
CH_3, =O(5)	7	20
CH_3, F_2(6)	8	21
CH_3, 5=6[c]	9	22
OH, 5=6	10	23
CH_3, CH_3(4)	11	36
CH_3, $=CH_2$(3)	12	24
OH, CH_3(1), $(CH_3)_2$(7)	13	25
CN	26	27
$COOCH_3$	28	29
CH_3, =O(6)	30	31
CH_3, F_2(5)	32	33
CH_2OH, 5=6	34	35
Cl, CH_3(1), $(CH_3)_2$(7)	37	38

[a] The data are quoted from the literature (ref 17). [b] Indicates that the attached substituent at position 3. [c] Indicates the double bond between positions 5 and 6.

neural network can distinguish the two functions x and $0.9y$ out of the function, $x + 0.9y$.

Table XI. Relative ^{13}C NMR Chemical Shifts and Conformations in Norbornanes and Norbornenes[a]

compd	C_1	C_2	C_3	C_4	C_5	C_6	C_7	exo/endo
1	6.7	6.7	10.1	0.5	0.2	−1.1	−3.7	exo
2	8.9	25.3	12.4	−0.4	−1.2	−3.1	−4.4	exo
3	7.7	44.3	12.3	−1.0	−1.3	−5.2	−4.4	exo
4	4.6	16.7	4.4	−0.2	−0.3	−1.0	−1.8	exo
5	1.8	15.1	4.4	−0.2	0.2	−0.7	−3.3	exo
6	5.7	3.0	2.6	−0.5	−0.4	0.7	−3.5	exo
7	6.1	5.9	10.6	0.6	0.2	0.2	−3.7	exo
8	6.5	6.3	10.4	0.3	−0.8	−0.1	−3.5	exo
9	6.5	7.5	9.5	0.5	1.7	0.7	−3.8	exo
10	7.8	47.0	11.7	−1.3	3.9	−2.7	−3.2	exo
11	6.9	6.4	10.1	0.7	−1.2	0.1	−3.9	exo
12	5.6	4.9	7.0	0.2	−1.1	0.2	−3.9	exo
13	2.5	42.5	11.9	−0.8	−1.1	−2.4	1.4	exo
14	5.4	4.5	10.6	1.4	0.5	−7.7	0.2	endo
15	6.8	23.3	10.5	1.2	0.6	−9.5	0.3	endo
16	6.3	42.4	9.5	0.9	0.2	−9.7	−0.9	endo
17	4.2	16.2	2.1	0.9	−0.6	−4.8	1.9	endo
18	1.7	12.8	4.0	0.4	0.2	−7.2	1.4	endo
19	4.7	3.1	2.2	0.3	1.3	−6.5	−0.6	endo
20	4.7	5.3	9.2	1.3	−0.4	−6.5	1.4	endo
21	4.6	11.5	8.9	−0.1	0.8	0.4	1.8	endo
22	5.6	7.5	8.7	1.4	1.7	−3.0	1.7	endo
23	7.1	47.8	13.3	2.2	3.6	−3.4	0.3	endo
24	4.1	4.2	7.0	0.7	0.5	−7.4	0.0	endo
25	3.2	40.2	10.4	−0.5	0.0	−10.3	3.1	endo
26	5.5	1.0	6.3	−0.3	−1.5	−1.6	−1.3	exo
27	3.4	0.1	5.5	0.2	−0.7	−4.9	0.0	endo
28	5.1	16.4	4.2	−0.4	−1.1	−1.4	−2.1	exo
29	4.0	15.9	2.2	0.7	−0.7	−5.0	1.7	endo
30	6.6	7.0	10.1	0.2	−1.2	0.5	−3.7	exo
31	6.0	8.4	11.2	−0.1	0.7	−1.5	−1.6	endo
32	6.3	7.2	9.8	0.7	−0.1	0.8	−3.5	exo
33	5.1	4.8	8.4	1.1	−0.1	−7.3	1.6	endo
34	1.9	17.1	5.2	−0.1	0.9	0.9	−3.4	exo
35	2.3	18.3	5.0	0.3	1.3	−2.9	1.4	endo
36	5.1	4.0	8.4	1.1	0.2	−7.7	1.6	endo
37	2.9	30.3	13.4	−0.5	−2.1	−0.7	2.0	exo
38	3.7	29.8	10.8	−1.6	−1.1	−9.0	2.2	endo

[a] The data are quoted from the literature (ref 17).

e. Recognition of Two Similar Functions by Correlated Input Data. So far, we used completely random numbers independently for variables x, y, or z. Actual data for a QSAR analysis, however, have some kinds of correlation among them. It is, therefore, necessary to know the relationship between the separability and the dispersion among input data. To this end, we again used the $x + ay$ function, where the input data for x and y are correlated by the following way. The similarity of a data set is measured using the dispersion, σ^2 ($=\Sigma(x_i - y_i)^2/N$ (N is the total number of the data sets (=40)). The correlation was made by discarding the set of data (x_i, y_i) if $(x_i - y_i)^2 > j$. If j is chosen to be 0.82, 0.59, or 0.37, σ^2 is approximately 0.2, 0.1, or 0.05. The cases with the coefficient, a, being 1.1, 1.3, and 2.0 were examined. Table IX shows the results. When a is greater than 1.3, the neural network can distinguish the two functions even at σ^2 being 0.05. However, when $a = 1.1$, the separation cannot be carried out unless σ^2 is greater than 0.1.

APPLICATION OF THE PARTIAL DERIVATIVE METHOD IN THE NEURAL NETWORK

The recognition tests of individual functions out of the combined function indicate that the recognition ability of the perceptron-type neural network seems to be powerful, in fact, far better than we had expected, although the reliability is not given in terms of any mathematical formula.

As examples of application, we have used two kinds of well-investigated chemical problems. The first problem is con-

Table XII. Correlation Analysis between the Configuration and ^{13}C NMR Chemical Shifts[a]

config	C_1	C_2	C_3	C_4	C_5	C_6	C_7
exo	−0.009	0.088	0.031	−0.111	−0.107	0.141	−0.213
endo	0.010	−0.088	−0.032	0.110	0.109	−0.141	0.214

[a] The network structure was $N(8,14,2)$, where $\alpha = 1, \beta = 1$, the threshold of the convergence $<10^{-3}$. Averaged values are shown.

cerning the relationship between the ^{13}C NMR chemical shift and the configuration of the substituent in norbornanes or norbornenes.[17] We adopted this as the representative of problems that may not contain much discrepancy in the data. The other problem is that of QSAR; this may contain discrepancy and/or ambiguity between the data since they involve biological responses.

a. The Relationship between the ^{13}C NMR Chemical Shift and the Configuration of Norbornanes and Norbornenes. Shown in Tables X and XI are the endo/exo configurations and the relative ^{13}C NMR chemical shifts in the derivatives of norbornane and norbornene, quoted from the literature,[17] where the same compound numbers are used. In accord with the former studies,[11] we used 25 (no. **1–25**) out of 38 data as training data. The network structure was set to be $N(8,14,2)$.

Table XII[16] shows the partial derivatives. The reconstruction–learning method for this data showed that the input parameters 7, 6, 5, 4, and 2 are important to decide exo or endo configuration and the parameters 1 and 3 have no contribution to the decision.[15]

Let us look at the averaged derivatives. The absolute values for each set of parameters are nearly the same, but the signs are opposite each other. This means that each input parameter oppositely contributes to the exo/endo decision and its magnitude is the same. The absolute values for parameters 1 and 3 are negligibly small enough to say that these parameters have nothing to do with the exo/endo decision. The largest absolute values are found in parameter 7 showing that this parameter has the major contribution to the decision. These results are in good accord with the former ones obtained by the reconstruction–learning method.[15] The merit of the present method, however, is that one can quantitatively handle the degree of the contribution of each input parameter.

b. Graded Classification of Activities in Mitomycins. Table XIII shows the input parameters and observed and calculated strengths in mitomycins. The experimental data and structural descriptors were taken from the literature,[18] and the same compound numbers were used. The scaled structural descriptors, i.e., F_x, $\sigma_{m\text{-}X}$, $V_{w\text{-}X}$, Y_{OMe}, Y_{OH}, and $E_{S\text{-}Z}$, are input using the network structure $N(7,12,5)$, where neurons 1–6 were given for the structural descriptors and neuron 7 for the bias. Neurons 1, 2, 3, 4, and 5 in the third layer correspond to the biological activities 3+, 2+, +, +/−, and −, respectively.

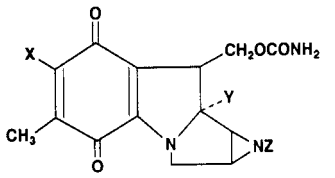
Table XIV shows the derivatives for each compound and the average for all compounds. First look at the averaged derivatives. The strongest positive contribution to the strongest group of biological activity is determined to be descriptor 6, i.e., $E_{S\text{-}Z}$ while the negative contribution is found in parameters 1, 3, and 4. In this way, one can easily know how to select the substituents to obtain a stronger mitomycin. One thing to be noticed, however, is that this analysis can apply only when the relationship between the class and the intensity of a descriptor is linear. If the relationship is nonlinear, the degree of contribution must be scanned to see how the nonlinearity takes place.

Then look at the derivatives for individual compounds. Contributions to a class from each parameter can be determined. As for compound **1**, which belongs to group 5, the derivatives for grades 4 and 5 result in significant values. On the other hand, those for grades 1, 2, and 3 are almost null. The same thing can be said to compound **2**. If one looks at those derivatives this way, one can determine the characteristics of the input data to the output or training data.

CONCLUDING REMARKS

We have introduced a general method to analyze the relationship between the input parameter and output results

Table XIII. Structures and Observed Activities in Mitomycins and Input Parameters, Output Patterns, and Calculated Classifications by the Neural Network[a]



										output pattern						
no.	X	Y	Z	F_x	$\sigma_{m\text{-}X}$	$V_{w\text{-}X}$	Y_{OMe}	Y_{OH}	$E_{S\text{-}Z}$	3+	2+	+	+/−	−	obs	calcd
1	NH_2	OMe	H	0.02	−0.16	0.177	1	0	1.24	0.95	0.08	0.00	0.00	0.00	3+	3+
2	NHEt	OMe	H	−0.11	−0.24	0.493	1	0	1.24	0.95	0.05	0.00	0.00	0.00	3+	3+
3	NH_2	OMe	Me	0.02	−0.16	0.177	1	0	0	0.01	0.98	0.06	0.00	0.00	2+	2+
4	NH_2	OMe	Et	0.02	−0.16	0.177	1	0	−0.07	0.00	0.98	0.10	0.00	0.00	2+	2+
5	NH_2	OMe	Ac	0.02	−0.16	0.177	1	0	−0.47	0.03	0.94	0.56	0.00	0.00	2+	2+
6	NH_2	OH	Me	0.02	−0.16	0.177	0	1	0	0.03	1.00	0.00	0.02	0.00	2+	2+
7	NMe_2	OMe	H	0.10	−0.15	0.441	1	0	1.24	0.02	0.92	0.03	0.00	0.00	2+	2+
8	NH_2	OMe	COPh-*o*-Cl	0.02	−0.16	0.177	1	0	−1.19	0.00	0.04	0.84	0.00	0.07	+	+
9	NH_2	OMe	COPh-*p*-Cl	0.02	−0.16	0.177	1	0	−1.19	0.00	0.04	0.84	0.00	0.07	+	+
10	NHPh	OMe	H	−0.02	−0.12	0.892	1	0	1.24	0.00	0.08	0.96	0.00	0.00	+	+
11	OMe	OMe	H	0.26	0.12	0.304	1	0	1.24	0.00	0.08	0.99	0.00	0.00	+	+
12	OMe	OMe	Me	0.26	0.12	0.304	1	0	0	0.00	0.07	1.00	0.00	0.00	+	+
13	OMe	OH	Me	0.26	0.12	0.304	0	1	0	0.00	0.05	0.01	0.96	0.03	+/−	+/−
14	NH_2	H	Me	0.02	−0.16	0.177	0	0	0	0.00	0.00	0.00	0.00	0.94	−	−
15	NH_2	OMe	SO_2Me	0.02	−0.16	0.177	1	0	−1.54	0.00	0.00	0.45	0.03	0.89	−	−
16	OMe	H	Me	0.26	0.12	0.304	0	0	0	0.00	0.00	0.06	0.00	0.99	−	−

[a] The network structure was $N(7,6,5)$, in which $\alpha = 1, \beta = 1$, and the threshold of the convergence $<10^{-3}$. The data were quoted from the literature (ref 18).

Table XIV. Partial Derivatives of Each Set of Input Data in Mitomycin

	1	2	3	4	5	6		1	2	3	4	5	6
						Compound **1** (5)[a]							
1	−0.029	−0.070	−0.078	−0.125	−0.115	0.114	4	0.713	0.048	0.311	0.228	0.280	−0.716
2	0.000	0.000	0.000	−0.000	0.000	−0.000	5	−0.497	−0.266	−0.315	−0.056	−0.076	0.545
3	0.000	0.000	0.000	0.000	−0.000	−0.000							
						Compound **2** (5)							
1	−0.036	−0.052	−0.063	−0.072	−0.083	0.083	4	0.193	−0.016	0.072	0.070	0.089	−0.190
2	0.000	0.000	0.000	−0.000	0.000	−0.000	5	−0.117	−0.066	−0.082	−0.022	−0.022	0.142
3	0.000	0.000	0.000	0.000	−0.000	−0.000							
						Compound **3** (4)							
1	−0.010	−0.003	−0.011	−0.012	−0.015	0.015	4	0.910	−0.397	0.238	0.356	0.607	−0.542
2	0.000	0.000	0.000	−0.000	0.000	−0.000	5	−0.340	−0.262	−0.345	−0.064	−0.031	0.531
3	0.010	0.013	0.018	0.005	−0.001	−0.023							
						Compound **4** (4)							
1	−0.008	−0.001	−0.009	−0.011	−0.014	0.011	4	0.738	−0.434	0.115	0.304	0.552	−0.339
2	0.000	0.000	0.000	−0.000	0.000	−0.000	5	−0.206	−0.165	−0.215	−0.039	−0.016	0.331
3	0.023	0.030	0.042	0.011	−0.003	−0.054							
						Compound **5** (4)							
1	−0.003	0.005	0.000	−0.007	−0.012	−0.003	4	0.483	−1.255	−0.632	0.322	0.863	0.722
2	0.001	0.001	0.001	−0.000	0.000	−0.001	5	−0.010	−0.011	−0.013	−0.002	0.001	0.020
3	1.357	2.561	3.189	0.750	−0.571	−3.983							
						Compound **6** (4)							
1	−0.036	−0.033	−0.073	−0.064	−0.047	0.096	4	0.386	0.031	0.273	0.153	0.138	−0.434
2	0.289	0.280	0.330	−0.028	0.142	−0.479	5	−0.139	−0.085	−0.120	−0.015	−0.024	0.191
3	0.000	0.000	0.000	0.000	0.000	−0.000							
						Compound **7** (4)							
1	−0.013	−0.010	−0.022	−0.026	−0.026	0.034	4	0.898	−0.131	0.425	0.352	0.477	−0.839
2	0.000	0.000	0.000	0.000	0.000	−0.000	5	−0.693	−0.460	−0.618	−0.104	−0.077	0.959
3	0.002	0.002	0.003	0.001	0.000	−0.004							
						Compound **8** (3)							
1	0.189	0.310	0.212	−0.027	−0.379	−1.705	4	−0.116	−1.009	−0.703	0.084	0.523	1.882
2	0.001	0.002	0.001	−0.001	0.001	−0.001	5	−0.000	−0.000	−0.000	−0.000	0.000	0.000
3	−0.053	1.475	1.675	0.148	−0.385	1.836							
						Compound **9** (3)							
1	0.189	0.310	0.212	−0.027	−0.379	−1.705	4	−0.116	−1.009	−0.703	0.084	0.523	1.882
2	0.001	0.002	0.001	−0.001	0.001	−0.001	5	−0.000	−0.000	−0.000	−0.000	0.000	0.000
3	−0.053	1.475	1.675	0.148	−0.358	1.836							
						Compound **10** (3)							
1	−0.007	0.009	0.003	−0.012	−0.022	−0.002	4	0.220	−0.795	−0.473	0.173	0.541	0.513
2	0.001	0.001	0.001	−0.000	0.001	−0.001	5	−0.006	−0.006	−0.007	−0.001	0.000	0.011
3	0.244	0.504	0.598	0.125	−0.102	−0.805							
						Compound **11** (3)							
1	0.004	0.017	−0.003	−0.055	−0.064	0.005	4	0.044	−0.521	−0.343	0.122	0.376	0.407
2	0.003	0.003	0.002	−0.001	0.008	−0.005	5	−0.000	−0.000	−0.000	0.000	0.000	0.000
3	0.320	0.985	1.355	0.645	−0.200	−1.781							
						Compound **12** (3)							
1	0.039	0.017	0.008	−0.075	−0.065	−0.060	4	−0.001	−0.005	−0.003	0.002	0.002	0.006
2	0.004	0.004	0.000	−0.008	0.013	−0.005	5	−0.000	−0.000	−0.000	0.000	0.000	0.000
3	−0.002	0.001	0.003	0.005	−0.002	0.000							
						Compound **13** (2)							
1	0.165	0.235	0.192	−0.244	−0.300	−0.368	4	−0.053	−0.253	−0.224	0.031	0.140	0.341
2	0.016	−0.036	−0.093	−0.068	0.265	0.047	5	−0.000	−0.000	−0.000	0.000	0.000	0.000
3	−0.007	0.121	0.149	0.088	−0.053	−0.161							
						Compound **14** (1)							
1	0.008	0.016	−0.155	−0.656	−0.568	0.223	4	0.524	−0.376	0.038	0.282	0.414	−0.152
2	0.001	0.001	0.001	−0.000	0.001	−0.001	5	−0.595	−0.431	−0.494	0.027	−0.042	0.788
3	0.000	0.000	0.000	0.000	−0.000	−0.000							
						Compound **15** (1)							
1	0.390	0.424	0.286	0.055	−0.542	−3.301	4	−0.001	−0.002	−0.002	0.000	0.001	0.006
2	0.001	0.002	0.001	−0.001	0.001	−0.000	5	−0.000	−0.000	−0.000	−0.000	0.000	0.000
3	−0.598	0.707	0.911	−0.084	0.006	5.062							
						Compound **16** (1)							
1	0.030	0.005	−0.015	−0.110	−0.084	−0.007	4	0.000	−0.000	−0.000	0.000	0.000	0.000
2	0.131	0.048	−0.018	−0.193	0.364	−0.131	5	−0.000	−0.000	−0.000	0.000	0.000	0.000
3	−0.114	0.162	0.426	0.635	−0.150	−0.339							
						Averaged Derivatives							
1	0.055	0.074	0.030	−0.092	−0.170	−0.411	4	0.301	−0.383	−0.101	0.160	0.345	0.159
2	0.028	0.019	0.014	−0.019	0.050	−0.036	5	−0.163	−0.110	−0.138	−0.017	−0.018	0.220
3	0.070	0.502	0.628	0.155	−0.112	0.099							

[a] The network structure was $N(7,6,5)$, in which $\alpha = 1$, $\beta = 1$, and the threshold of the convergence $<10^{-3}$. The value in parentheses shows the class of its biological intensity. See Table XII.

input to the output, the reconstruction–learning method was introduced. This eliminates the unnecessary input parameters and minimizes the number of the neurons in the hidden layer(s) without lessening the network's ability.

Finally, a direct method of analyzing the input–output relationship was introduced. The operation of the perceptron-type neural network can be regarded as a function which transforms an input vector to another (output) vector. We derived the analytical formula for the partial derivative of this function with respect to each element of the input vector. Using numerical data, we examined the accuracy, independency of the elements of the input vector, and separation ability of a function out of the mixed function and obtained enough evidence to conclude that the method is very useful in the analysis of the structure–activity relationship (SAR) in both fitting and classification.

REFERENCES AND NOTES

(1) Rosenblatt, F. *Principles of Neurodynamics; Perceptrons and the Theory of Brain Mechanics*; Spartan Books: Rockelle Park, 1961.
(2) Hopfield, J. J. *Proc. Natl. Sci. U.S.A.* **1982**, *79*, 2554.
(3) Ackley, D. H.; Hinton, G. E.; Sejnowski, T. J. *Cognit. Sci.* **1985**, *9*, 147.
(4) Hinton, G. E.; Sejnowski, T. J. *Parallel Distributed Processing*; Rumelhart, D. E., McClelland, Eds.; MIT Press: Cambridge, MA, 1986; Vol. 1, Chapter 7.
(5) Rumelhart, S. E.; Hinton, G. E.; Williams, R. J. *Nature (London)* **1986**, *323*, 533.
(6) Minsky, M.; Papert, S. *Perceptron—An Essay in Computational Geometry*; MIT Press: Cambridge, MA, 1969.
(7) Sejnowski, T. J.; Rosenberg, C. R. *The Johns Hopkins University Electrical Engineering and Computer Science Technical Report*, JHU/EECS-86/01, 1986.
(8) Sejnowski, T. J.; Rosenberg, C. R. *Complex Syst.* **1987**, *1*, 145.
(9) Gorman, R. P.; Sejnowski, T. J. *Neural Networks* **1988**, *1*, 75.
(10) Aoyama, T.; Suzuki, Y.; Ichikawa, H. *Chem. Pharm. Bull.* **1989**, *37*, 2558.
(11) Aoyama, T.; Suzuki, Y.; Ichikawa, H. *J. Med. Chem.* **1990**, *33*, 905.
(12) Aoyama, T.; Suzuki, Y.; Ichikawa, H. *J. Med. Chem.* **1990**, *33*, 2583.
(13) Aoyama, T.; Ichikawa, H. *Chem. Pharm. Bull.* **1991**, *39*, 358.
(14) Aoyama, T.; Ichikawa, H. *Chem. Pharm. Bull.* **1991**, *39*, 372.
(15) Aoyama, T.; Ichikawa, H. *Chem. Pharm. Bull.* **1991**, *39*, 1222.
(16) *QCPE* **1992**, *No. 615*.
(17) Kowalski, B. R. *Chemometrics: Theory and Applications*; ACS Symposium Series 53; American Chemical Society: Washington, D.C., 1977; p 423.
(18) Moriguchi, I. In *Structure–Activity Relationship—Quantitative Approaches*; Fujita, T., Ed.; Nankodo: Tokyo, Japan, 1987; Chapter 9. See also ref 12.