Neural Computing and Applications

Vichit Avatchanakorn

Graduate School of Computer and Engineering Management
Assumption University
and
International Software Factory

1. Introduction

Imagine a computer that learns. Learning information, along with examples and conclusions that it should do, are fed into it. The computer runs through the given material again and again, with million of mistake decision, until finally it gets itself into the proper manner that can carry out the task successfully. This behavior is quite a human being. As the inspiration of the human brain structure, the design of such an information processing system, an *artificial neural network* (ANN), is concerned with nerve cells, their interconnections, and interactions.

At present, most of computers are based upon John von Neumann's concept. Such a computer, generally, executes data processing, the conversion from one format of data into another ones, through an algorithm deviced by a human. In reality, data processing in a human brain occurs from associating inputs with outputs. Such associations, usually known as mapping or transformations, are the results of learning of various pairs of input-output examples. Thus, neural computing, in formal terminology, is the engineering discipline concerned with nonprogrammed adaptive information processing systems; neural networks, that develop associations between objects in response to their environment. The neural network itself generates its own internal rules governing the association and refines those rules by comparing its results to the given examples [Neilsen,1988]. The rule generating process is called the *learning algorithm*.

It is considered that the study of artificial neural networks (ANNs) started since McCulloch and Pitts have presented the model of a neuron in 1943. Learning of neural networks was first introduced by Hebb in 1949, called as Hebbian learning rule. In the 1960s, Rosenblatt has presented the recognition machine called the perceptron, and numerous theorem on the convergence of learning algorithms have been proposed. Since the perpectron is a simple layer type neural network having the abilities of recognition closed to human beings, research of neural computing has attracted a great deal of attention. Consequently, this duration is known as the *Age of Camelot* for neural network development. However, this age does not continue so long, because in the half of 1960s, Minsky and Papert have presented the limitation of perceptron. This effected the development of neural networks in reducing the number of researchers interested in this field. The fast development of the Neumann's computers, integrated circuits, and operating systems are also the causes of the diminution. This is known as the *Dark Age*, biginning from 1969 and ended in 1982 with Hopfield's landmark paper on neural networks and physical systems.

Since Hopfield's paper, which presents the application of neural networks to the practical problems that have never been solved, i.e., the application to the traveling salesman problem, ANN development is considered in the third age, the *Renaissance*. One more interesting application in this duration is conducted by Sejnowski in 1986. He has developed the NETalk that uses the layer type neural network to create voices. The Renaissance ended in 1986. From 1987 to present is called as the *Age of Neoconnectionism*, of which the development of neural network tools for personal computers have expanded unbeleivably. Nowadays, there are numerous neural computing tools run on personal computers or workstations having higher cost performance than a mini-supercomputer. [Nakano,1991]

Generally, computers and artificial intelligent machines precisely conduct logical operations or numerical computations. In the opposition, neural conputing is good in pattern recognition and intuitive decisions, At present, the artificial intelligence is confronted by the difficulties of inference with human's common sense. It is expected that neural computing could solve real-life problems including intuitive human intelligence.

2. Artificial Neural Netwoks

The major purpose of ANNs study is to realize the information processing system that hold competence closed to human beings. Consequently, the fundamental of ANNs is the core of the information processing called the nervous system. Realization of such a processing system, has not only the practical merit, but also the significance in elucidation of information processing of the nervous system. An ANN can be considered as one type of the perallel processing machines of which processing elements operate simutaneously. A processing element is equivalent to an artificial neuron. The followings present the basic artificial neuron models and the various types of ANNs.

2.1 Artificial Neuron Models

The schematic diagram of an elementary nerve cell, called a biological neuron, is shown in Figure 2.1. The cell has three major regions: the cell body, the axon, and the dendrites.

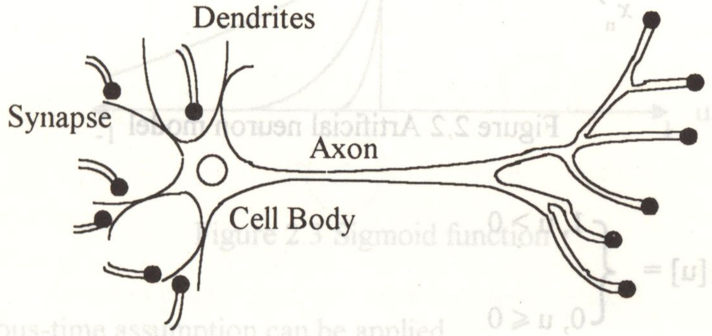


Figure 2.1 Biological neuron model

Dendrites of a neuron receive information from another neurons through axons. The axon-dendrite contact organ called a *synapse* introduces the signal to the neighboring neuron. The receiving neuron either generates a signal, which is an electrical inpulse, or produces no response. The neuron can generate a response if the total potential (the summation of its inputs) reaches a certain level called the *threshold*. The synapse connections might be excitatory if they cause the firing, or inhibitory if they hinder the firing of the neuron. Consequently, it is practical to assign, respectively, positive and negative unity weight values to such connections.

Based on the biological neuron model described above, the first formal artificial neuron model was introduced by McCulloch and Pitts in 1943. Figure 2.2 shows the McCulloch-Pitts Model that is the multi-input-single-output nonlinear element.

A neuron having n inputs, $x_1, x_2, \ldots, x_i, \ldots, x_n$, and one output, y, is defined as the following discrete-time difference equation

$$y(k+1) = 1[\ \Sigma^i w_i x_i(k) - h\ ] \qquad k = 0,1,2,\ldots \qquad (2.1)$$

where, k = 0,1,2,... denotes the discrete-time instant, w_i is the weight connecting the i-th input with the neuron, also called the *synapse weight*, h is the neuron's *threshold value*. 1[u] is the step function defined as

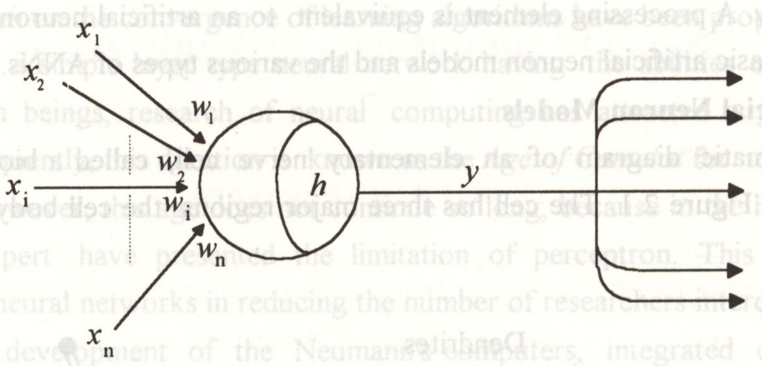


Figure 2.2 Artificial neuron model

$$1[u] = \begin{cases} 1, u > 0 \\ 0, u \leqslant 0 \end{cases} \tag{2.2}$$

The inputs and output of a neuron are 1 or 0, which represent the firing or non-firing state, respectively. In (2.1), if there exists the i-th input, the potential of a neuron will change by w_i. w_i is positive for excitatory synapses, w_i is negative or inhibitory synapses, and w_i = 0 for no existing synapse connection. If the total potential of the neuron (summation of $w_i x_i$, $\Sigma^i w_i x_i$) is greater than the threshold h, the neuron fires and generates an electrical impluse.

The McCulloch-Pitts model of a neuron makes use of several drastic simplification, such as it allows binary 0, 1 states (discrete-information) only, and operates under a discrete-time assumption. A general artificial neuron model is defined as the following function :

$$y = f(\Sigma^i w_i x_i - h) \tag{2.3}$$

The function f(u) is an activation function and the sigmoid function

$$f(u) = 1 / \{ 1+\exp(-u/T) \} \tag{2.4}$$

is generally used. Figure 2.3 shows the sigmoid function for various T. Notice that $T \rightarrow 0$, the limit of the function becomes the step function defined in (2.2). From the model in (2.3), the neuron's output y is a continuous value between 0 and 1, i.e.,

$$y = \{ x / x \in R, x \in (0,1)\} \tag{2.5}$$

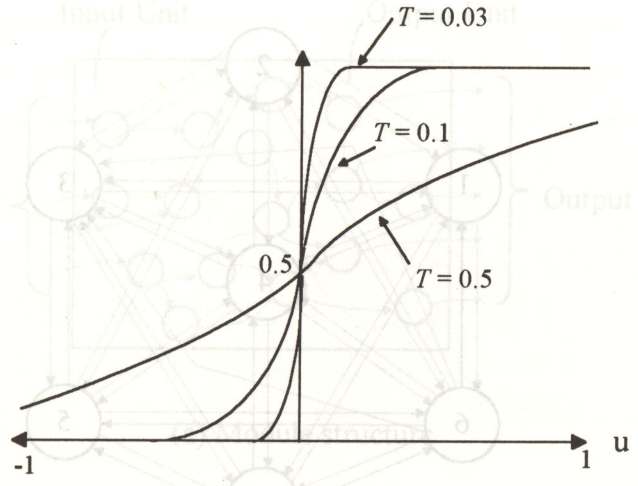


Figure 2.3 Sigmoid function

and the continuous-time assumption can be applied.

2.2 Models of Neural Networks

There are numerous artificial neural network models proposed by various researchers. However, most of the models are based on the biological consideration of the nervous system. The followings indicate some configuration and behavior of ANN models charaterized from the real nervous system.

<u><Network model></u>

Generally, a neural network can be considered as an interconnection of neurons through synapse weights, as shown in Figure 2.4. This is the interconnection type neural network. All neuron can operate simultaneously in this type of ANNs.

<u><Connection constraint></u>

A human brain consists of approximately 10^{11}neurons, which in reality all of neurons is not connected to each other. A number of connections for a neuron are approximately about 10^4 order in the human brain.

<u><Synchronizing behavior></u>

Characteristics of an ANN depend on the timing of the neurons' operation. Without any specification, all of the neurons in the ANN operate synchronously with a constant clock.

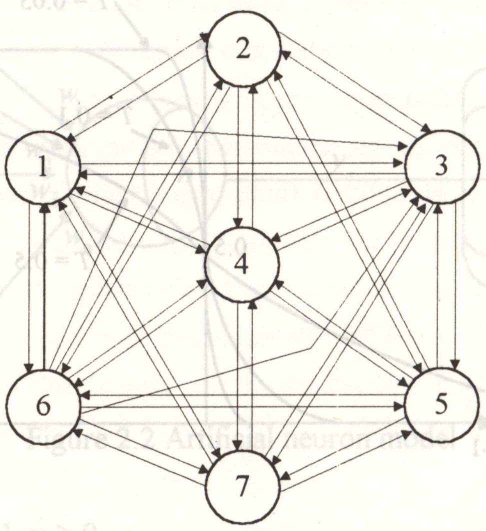


Figure 2.4 Interconnection type neural network

<u><Module structure></u>

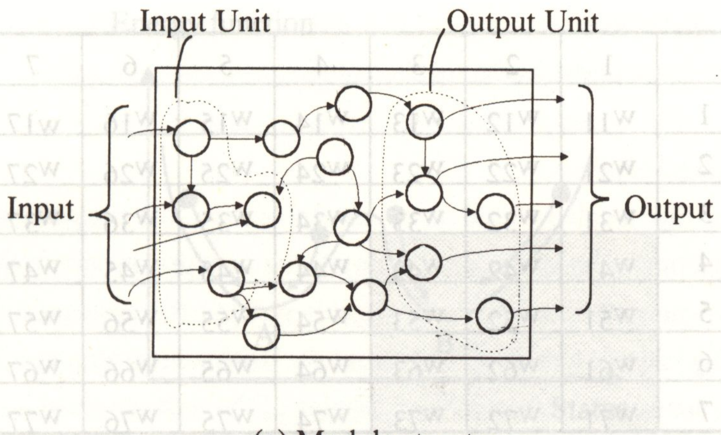
A concept of the module structure is very important for the analysis of a large scale system. It is known that a human brain consists of many modules with various functions. In a high level parallel processing system, a module is considered as the fundamental processing element instead of a neuron as shown in Figure 2.5.

<u><Layer structure></u>

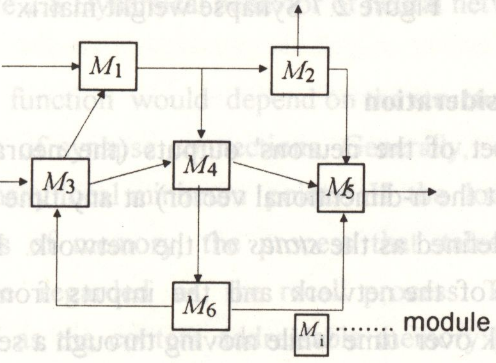
Figure 2.6 shows a layer type ANN of which the neurons are grouped in the layer structure and the signals can only flow in the direction between the specific groups. If the matrix of the synapse connections of the networks in Figure 2.4 is shown in Figure 2.7, the matrix of the synapse connections of Figure 2.6 is also shown in Figure 2.7 within the shaded area. As the result, a layer type ANN can be considered as a special case of the interconnected ANN.

<u><Hopfield network: asynchronizing behabior></u>

In Figure 2.4, if the network is constrained with the conditions that (1) the synapse weights of the interconnected neurons are equivalent and (2) the behavior of all neurons is completely asynchronous, the network is called as the Hopfield Network. The Hopfield network is applied to utilize as an associative memory, to solve the optimization problems, etc.



(a) Module structure



(b) Module network

Figure 2.5 Module structure of neural network

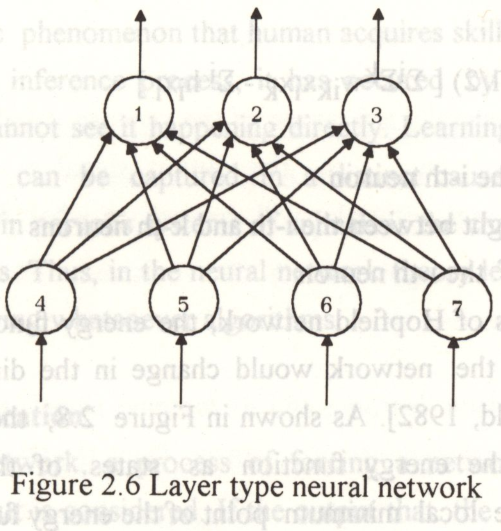


Figure 2.6 Layer type neural network

	1	2	3	4	5	6	7
1	w_{11}	w_{12}	w_{13}	w_{14}	w_{15}	w_{16}	w_{17}
2	w_{21}	w_{22}	w_{23}	w_{24}	w_{25}	w_{26}	w_{27}
3	w_{31}	w_{32}	w_{33}	w_{34}	w_{35}	w_{36}	w_{37}
4	w_{41}	w_{42}	w_{43}	w_{44}	w_{45}	w_{45}	w_{47}
5	w_{51}	w_{52}	w_{53}	w_{54}	w_{55}	w_{56}	w_{57}
6	w_{61}	w_{62}	w_{63}	w_{64}	w_{65}	w_{66}	w_{67}
7	w_{71}	w_{72}	w_{73}	w_{74}	w_{75}	w_{76}	w_{77}

Figure 2.7 Synapse weight matrix

2.3 Dynamical Consideration

In this section, a set of the neurons' outputs (the neural network consists of n neurons would construct the n-dimentional vector) at any time within the time interval under consideration is defined as the *states* of the network. From the neural network theory, the present state of the network and the intputs from external would determine behavior of the network over time while moving through a sequence of states. For an example, if the neural network is utilized as an associative memory, the steady-state (the state that the outputs may not change although the time has elapsed) output is determined form the given initial state. Thus, the study of behavior of the network is conducted by following the change of states over the time.

For investigation, the energy function E(x) of the discrete-information neural network is defined as

$$E(\mathbf{x}) = -(1/2)\ [\ \Sigma^i \Sigma^k w_{ik} x_i x_k - \Sigma^i h_i x_i\ ] \qquad (2.6)$$

where x_i : output of the i-th neuron

w_{ik} : synapse weight between the i-th and k-th neurons

h_i : threshold of the i-th neuron

Based on the conditions of Hopfield network, the energy function defined by (2.6) is existed and states of the network would change in the direction that reduces the energy function [Hopfield, 1982]. As shown in Figure 2.8, the network would move toward the valley of the energy function as states of the network change. The equilibrium state is at the local minimum point of the energy function which the network move toward.

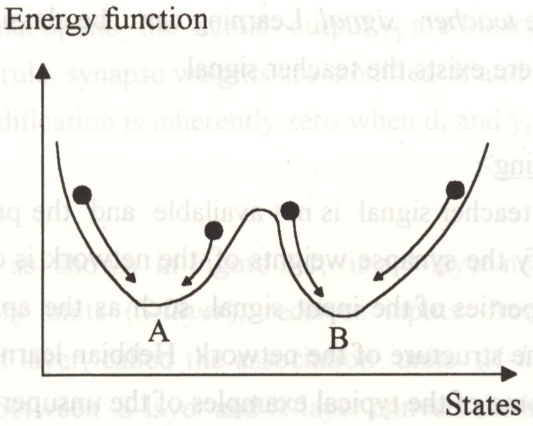


Figure 2.8 Dynamical behavior of neural network

Shape of the energy function would depend on the structure of a neural network, such as the distribution of synapse connections. Generally, energy function is multi-stable function that has many local minimum points. If the local minimum points are considered as the contents of memory, the process that states move toward their local minimum points can be regarded as the recall process. This type of the neural network can be utilized as the content addressable memory [Hopfield, 1982]. Time required for recalling the contents is constant and not varied to the number of recorded memory.

3. Neural Network Learning

Learning is the basic phenomenon that human acquires skills and abilities. Learning in human beings is an inference process; it has occured by observing changes in performance and we cannot see it happening directly. Learning in neural networks is a more direct process that can be captured in a distinct cause-effect relationship. It is considered that learning in nervous systems of animals is the change in synapse weights of interconnected neurons. Thus, in the neural network theory learning means the change in synapse weights following whatsoever algorithms.

3.1 Learning Classification

For learning in a network, a process of forcing a network to yield a particular response to a specific input is considered. If the output that the network should respond, when the input is applied, is provided by the external source, the desired response of the network is called the *teacher signal*. Learning can be classified as the followings according to whether there exists the teacher signal.

<Unsupervised learning>

Learning that the teacher signal is not available and the properties of environment are introduced to modify the synapse weights of the network is called the unsupervised learning. Statistical properties of the input signal, such as the appearance frequency, are introduced to reflect the structure of the network. Hebbian learning rule, self-organizing feature extraction are some of the typical examples of the unsupervised learning.

<Supervised learning>

Learning that the teacher signal is provided from the external source for the specific input is called the supervised learning. The network will adjust the synapse weights so that it responds the given input with the exact corresponding output (the desired output). Perceptron, error back-propagation, correlation learning rules are some typical examples of this learning type.

For both the supervised and unsupervised learning, the following general learning rule is adopted for the discrete-time neural network model [Amari, 1990]:

$$w_i(k+1) = w_i(k) + cr(k)x_i(k) \qquad (3.1)$$

where x_i : i-th input of a neuron

w_i : synapse weight of the corresponding input x_i

c : learning constant

r : learning signal

The learning signal is generally a function of the synapse weights w_1, w_2, ..., w_i, ..., w_n, the input x_1, x_2, ..., x_i, ..., x_n, the output signal y, and the teacher signal d. The learning rules of neural networks will depend on what function that the learning signal is defined as.

3.2 Perceptron

Perceptron is supervised learning and its learning signal is the difference between the desired and the actual response of a neuron. Thus, the learning signal is equal to

$$r = d_i - y_i \qquad (3.2)$$

where the teacher signal d_i and the actual output y_i are binary values, 0 and 1. Under the perceptron learning rule, synapse weights are modified if and only if y_i is incorrect. The synapse weight modification is inherently zero when d_i and y_i agree.

<u><Simple perceptron></u>

Simple perceptron, as shown in Figure 3.1, is a 3-layer neural network. The first layer, called the sensory units (*S-layer*), receives inputs from external sources and transfers to the second layer, called the association units (*A-layer*), just as it received them. The connections between S-layer and A-layer convert the inputs into the signals for the A-layer. The third layer, called the response units (*R-layer*), consisted of the basic perceptron elements, responds the processed information result according to the input. Furthermore, the learning is conducted at only the R-layer.

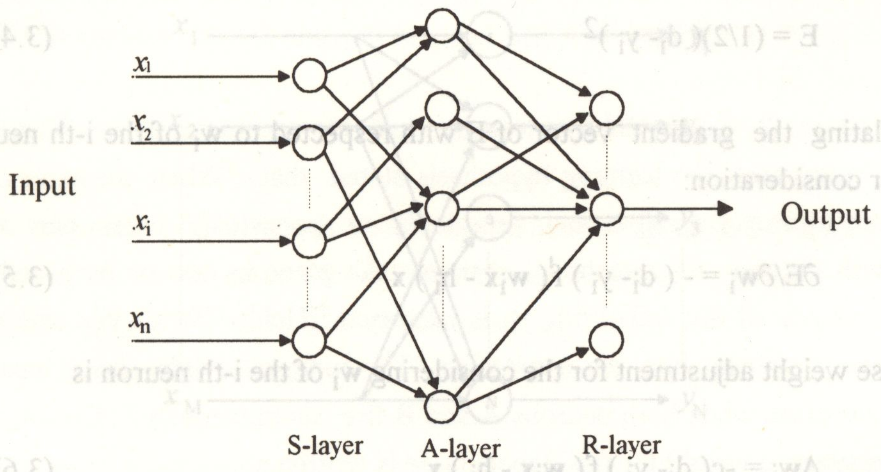


Figure 3.1 Simple perceptron

Generally, the simple perceptron is utilized as the linear classifier, i.e., to classify what inputs at the S-layer should be between the two classes of categories. However, most of problems in reality are nonlinear classification problems. As an example, the simple perceptron cannot solve the exclusive OR (XOR) logic, which is not the linear classification problem. This is the limitation of the simple perceptron.

3.3 Delta Learning Rule

The delta rule is only valid for continuous activation functions as defined in (2.4) and in the supervised learning. The learning signal is defined as

$$r = [\, d_i - f(\mathbf{w}_i\mathbf{x} - h_i)\,]\, f'(\mathbf{w}_i\mathbf{x} - h_i) \qquad (3.3)$$

where $\mathbf{x} = [x_1, x_2, \ldots, x_n]^t$ is the vector of inputs to the i-th neuron
$\mathbf{w}_i = [w_{i1}, w_{i2}, \ldots, w_{in}]^t$ is the vector of synapse weights connected to the i-th neuron

The term $f'(u)$ is the deriverative of the activation function f(u). As indicated in (3.3), the delta learning rule will continuously modify the synapse weights of the network whenever the actual output is not exactly equal to the teacher signal.

<Error Back-propagation>

The error back-propagation learning rule is proposed by Rumelhart in 1986 applied to a multi-layer neural network. The basic concept of this learning rule is based on the delta learning rule derived from the condition of least squared error between the actual output y_i and the teacher signal d_i:

$$E = (1/2)(d_i- y_i)^2 \tag{3.4}$$

and calculating the gradient vector of E with respected to $\mathbf{w}_i$ of the i-th neuron at the layer under consideration:

$$\partial E/\partial \mathbf{w}_i = - (d_i- y_i) f'(\mathbf{w}_i\mathbf{x} - h_i) \mathbf{x} \tag{3.5}$$

The synapse weight adjustment for the considering $\mathbf{w}_i$ of the i-th neuron is

$$\Delta\mathbf{w}_i = -c(d_i- y_i) f(\mathbf{w}_i\mathbf{x} - h_i) \mathbf{x} \tag{3.6}$$

It is known that the back-propagation learning rule guarantee for convergence of error to its local minimum point. The initial synapse weights are needed to be set with a small random values.

The 3-layer neural network with the back-propagation learning rule applied to the 3-rd and 2-nd layer can solve the XOR logic and another nonlinear classification problems. This shows the powerfulness of the back-propagation algorithm.

3.4 Correlation Learning and Associative Memory

The correlation learning is one of the supervised learning of which the learning signal is defined as

$$r = d_i \tag{3.7}$$

Namely, this rule states that the synapse weight increase is proportional to the product of the teacher signal and the input (refer to (3.1)). This rule is typically applied to the neural network utilized as an associative memory.

Utilization of a neural network as an associative memory can be described by considering the network consisting of N neurons and M inputs shown in Figure 3.2, and assumming that there are P pairs of input-output (also called as key-recall) patterns, ((**x**(1),**y**(1)), (**x**(2),**y**(2)), ..., (**x**(P),**y**(P))). An associative memory is the memorizing of input-output patterns so that the output pattern is recalled whenever the corresponding input pattern is given. The learning algorithm allowing the above process is called the recording algorithm. The associative memory whose key and recall patterns are equivalent (**y**(r) = **x**(r), r = 1,2,...,P) is called the *autoassociative memory*, while the memorizing different key and recall patterns is called the *heteroassociative memory*.

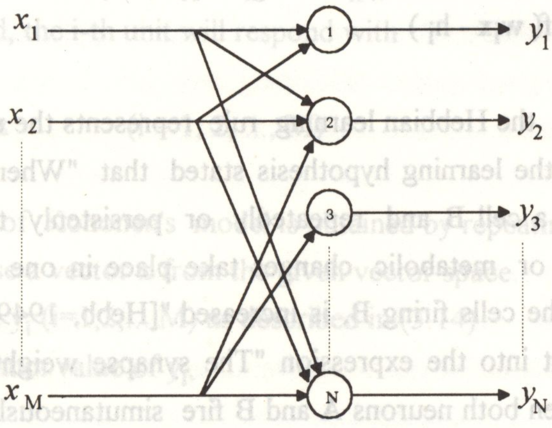


Figure 3.2 Neural network associative memory

If the correlation learning (3.7) is applied to an interconnection type neural network as the recording algorithm, the synapse weight adjustment is written as

$$w_{ij}(k+1) = w_{ij}(k) + cy_i^{(r)}x_j^{(r)}(k) \qquad (i = 1, ..., N, j = 1, ..., M) \qquad (3.8)$$

which is classified as the *recurrent associative memory*, because of its dynamical behavior. Retrieval process of a recurrent associative memory can be regarded as the moving of states of the neural network toward their local minimum point where the recall patterns are stored. The followings indicate distinctive features of a neural network associative memory

(1) Information stored in a neural network is distributed over synapse weights, which can be considered as a *distributed memory*. Thus, locally whatever faults happening in the system will not affect the overall stored information.

(2) Since the retrieval process is performed parallelly, required time is constant and not varied to the number of stored information.

(3) Since a neuron has threshold operation, a neural network associative memory can recall the accurate output although its exciting inputs containing ambiguous information.

3.5 Hebbian Learning and Self-Organizing Networks

<ins>&lt;Hebbian learning rule&gt;</ins>

The learning signal of the Hebbian learning rule is equal to the neuron's output, i.e.,

$$r = y_i = f(\mathbf{w}_i\mathbf{x} - h_i) \qquad (3.9)$$

As indicated in (3.9), the Hebbian learning rule represents the unsupervised learning. The rule interpretes the learning hypothesis stated that "When an axon of cell A is near enough to excite a cell B and repeatedly or persistenly takes part in firing it, some growth process or metabolic changes take place in one or both cells such A's efficiency, as one of the cells firing B, is increased."[Hebb, 1949]. We can change the above classic statement into the expression "The synapse weight from neuron A to neuron B increases when both neurons A and B fire simutaneously." Consequently, the change in a synapse weight for the Hebbian rule is written as

$$w_j(k+1) = w_j(k) + \alpha\, x_j(k)\, y(k) \qquad (j = 1, \ldots, n) \qquad (3.10)$$

where $\alpha \geqslant 0$ is the learning speed parameter. The synapse weight adjustment in (3.10) need to be modified with the following methods to counteract unconstrained growth of weight values.

(1) Introducing the natural damping constant to the synapse weights

$$w_j(k+1) = \beta\, w_j(k) + \alpha\, x_j(k)\, y(k) \qquad (j = 1, \ldots, n) \qquad (3.11)$$

where β is the damping constant, $0 < \beta < 1$

(2) Providing the inhibitive input and increasing its synapse weight w_0 simutaneously.

$$w_j(k+1) = w_j(k) + \alpha\, x_j(k)\, y(k) \qquad (j = 1, \ldots, n) \qquad (3.12)$$

$$w_0(k+1) = w_0(k) + \alpha_0 y(k) \qquad (3.13)$$

<Self-organizing feature mapping>

The self-organizing feature map is an example of self-organizing neural networks presented by Kohonen in 1982 [Kohonen, 1982]. Kohonen has expanded the idea of topological mapping from an information processing point of view, and presented the network model that has the feature space instead of an input layer. The model has N input nodes in space and M output nodes arranged in a two dimentional array. Output nodes are extensively interconnected with numerous local connections and every input connected to every output node via variable connection weights. If the weight vector for the i-th output node $\mathbf{w}_i = [w_{i1}, w_{i2}, \ldots, w_{iN}]^t$, and the input vector $\mathbf{x} = [x_1, x_2, \ldots, x_N]^t$ is applied, the i-th unit will respond with

$$y_i = \mathbf{w}_i{}^t \mathbf{x} \qquad (i = 1, 2, \ldots, M) \qquad (3.14)$$

The feature mapping of Kohonen's model is obtained by repeating the algorithm:

(1) Randomly choose a vector $\mathbf{x}$ from the given vector space.

(2) Calculate output y_i (i=1,2,...,M) as described in (3.14)

(3) Select the maximum value of y_i,

$$y_k = \max_i(y_i) \qquad (3.15)$$

(4) Update the weights for the k-th node and all nodes in the neighborhood as

$$w_i(k+1) = [\, w_i(k) + \alpha x(k)\,] \,/\, \big|\big|\, w_i k) + \alpha x(k) \,\big|\big| \qquad (3.16)$$

With the recursive computation of (1)~(4), the weight vectors of neighborhood nodes become parallel, and the model is completely constructed. Namely, when the given signal vector is presented, the neighborhood units would generate the similar outputs.

Numerous technical reports have been written about successful applications of the feature map algorithm [Kohonen, 1984]. The algorithm has been applied in sensory mapping, robot control, vectorquantizer, and speech recognition.

4. Applications

Having understood the fundamental principles of massively parallel interconnected simple neuron, we may now put them to good use in the design of practical systems. Every year, the published technical references on application of neural networks well over a thousand papers are printed. As we shall see in the followings, neural computing architectures are successfully applicable to many real-life problems.

Application of neural computing is expected to be the fields that relate with complicated systems requiring high speed parallel distribution processing and including vagueness. The scales of being developed neural networks are very small comparing with modules of the nervous system. Most of the learning algorithms are overwhelming the back-propagation technique.

To construct application systems of neural networks, we need to gather the learning data (input and teacher data). Obtaining the successful application system depends on that we must gather the good quality of learning data.

The followings describe more closely utilization of neural computing for various application areas.

<Pattern recognition>

It is considered that most of application for which neural computing has been suggested are the utilization of neural networks as the recognition machines. The first application presented by F. Rosenblatt in 1958 is the use of the perceptron as a pattern classifier. Consecutively, this prosperity of neural networks is applied to various areas such as character recognition, speech recognition, robot control, process control, fault detection, behavior forecasting, etc.

Layer type neural networks with the back-propagation learning algorithm are the most popular for the use. It is considered that since the structure of the networks and the learning algorithm are rather easy for implementation. However, to obtain the success application system, we must collect the good quality of learning data. It is difficult for some cases. As an example, neural computing applied to nonlinear process control of which the teacher signals are sequences of operations of an operating personnel. The difficulty is to eliminate scattering data which sometimes being used in some situations. However, with the techniques that researchers have proposed for determining the good teacher signal, the application of neural networks to pattern recognition is still popular.

<Expert systems>

Constructing of an expert system, knowledge engineers must interview the experts of the related field. Then they conclude the results of interviews and compose the rules in the form that an expert system shell or a computer can process them. Whether the constructed expert system works successfully depends on the rules composed which are different from one knowledge engineer to anothers.

Knowledge in neural networks composed by learning algorithms that use examples and conclusions as learning data. Today, a large number of data are going to available at the occurrence of events because of the advanced technology. If neural networks are constructed in the form that knowledge rules can be extracted, neural computing would become one of the knowledge acquisition supporting tools for expert systems.

The above shows an ideas using neural networks to improve defects in expert systems. There are many papers present the merger of neural computing and expert systems, called as neural-expert systems, such as the use of learning algorithm in tuning certainty factor, in extracting rules, etc. Today, the knowledge systems including human vagueness, called as fuzzy-expert systems, also merge neural networks for the purposes of extraction of fuzzy rules, tuning membership functions, infering certainty grade, etc.

<Optimization calculation>

Among numerous technical papers showing the application of neural networks in various areas, the application for approximate solution to optimization problems is one of the interesting fields. Interconnection type neural networks minimize their energy function value in time, such that the networks stay whensoever at steady state. Thus, optimizing a system for the given objective function is considered as the designing a neural network that minimizes its energy function. The approximate solution obtained when the corresponding network reaches its local minimum point. This idea was first presented by J.Hopfield with the application of Hopfield network to the traveling-salesman problem (TSP). The great advantage of this idea is that a physical device for solving the problem can be constructed, generating the solution without the any learning iterations.

There are many researchers present their papers following the Hopfield's idea, such as the application to a solution of load flow problems, unit commitment problems, dynamical systems, etc. The difficulty is the formulation of a given problem into a form of an energy function.

5. Concluding Remarks

From the first neuron model was developed, nowadays neural networks have the age of about 50 years. Comparing with the human's history which is extremely much more than the number 50, it can be said that neural network research is now just starting. The evolution of neural networks supported from computer technology is expected to be rapidly advanced in the future. The followings indicate the future prospects of neural computing:

<Improvement of nervous system models>

The nervous system modeled from artificial neurons is at present very simple. It is possible that a neuron possesses more high level processing and undiscovered mechanisms. Furthermore, the biological nervous system has both electrical and chemical activities, both analog and digital processing.

To construct the nervous system closed to the human brain, knowledge of nervous phisiology is important. Hereafter, with the development of measurement equipments, it is expected that research in nervous phisiology is advanced and results in new knowledge of neural computing.

<High speed processing>

Accompany with the enlargement of scales of neural networks, enormous quantity of learning data becomes one of difficulties. The following hardware and software development are expected to improve the processing speed.

-Hardware: neuron chips, optical elements, parallel processing machines, etc.

-Software : improvement of learning algorithms, introducing knowledge for individual application

<Learning data>

To build a large scale neural system, it is necessary to provide a great volume of learning data together with the high speed processing machines. Since there is the limitation in human's capability, whatever automatic inputs of data are required. Learning data preparation is very important when constructing an application system. It is also expected for research and development in this prospect.

<Development of new generation computer>

Neural networks use a human brain as the model and aim at having characteristics closed to human beings. If a computer possessing such characteristics, called as a neural computer, has been realized, the computer can analyze and process information including vagueness, and make an intuitive decision.

References

[Amari, 1990], Mathematical foundations of neural computing, *IEEE Proc.* 78(9)

[Eberhart, Dobbins, 1990], *Neural network PC tools: A practical guide*, Academic Press, 1990

[Hirafuji, 1989], Neural-expert system (M-NCS), *Inter AI*, 2 (in Japanese)

[Hopfield, 1982], Neural network and physical systems with emergent collective computational abilities, *Proc. Natl. Acad. Sci. USA*, 79

[Hopfield, Tank, 1985], Neural computation of decisions in optimazation problems, *Biol.Cybern.*, 52

[Kohonen, 1984], *Self-organization and associative memory*, Springer-Verlag

[Lippmann, 1987], An introduction to computing with neural nets, *IEEE Magazine on Acoustics, Signal and Speech Processing* (April), 4

[McClelland, Rumelhart and the PDP Research Group, 1986], *Parallel distributed processing*, Cambridge, The MIT Press

[Nakano, 1991], *Neurocomputer: An introduction and practical*, Gijutsu-hyoron-sha (in Japanese)

[Narendra, Parthasarathy, 1990], Identification and control of dynamical systems using neural networks, *IEEE Trans. on Neural Networks*, 1(1), 4

[Neilsen, 1988], Neurocomputing: picking the human brain, *IEEE Sprectrum*, 3

[Pao, Sobajic, 1990], Nonlinear process control with neural nets, *Neurocomputing*, 2(2)

[Zurada, 1992], *Introduction to artificial neural systems*, West Publishing Company

Biography

Vichit Avatchanakorn received his B.Eng. from Kasetsart University in 1985, M.Eng. and D.Eng. from Tokai University, Japan, in 1988 and 1991 respectively, all in Electrical Engineering. He is now with International Software Factory (ISOFAC) Co., Ltd., as a director of MIS division, in charge of supervising development of computerization systems. He is also an Associate Dean of the Graduate School of Computer and Engineering Management, Assumption University. His current interested research fields are the application of fuzzy logic, neural networks to systems control and planning, system self-organization, and pattern recognition.