RESEARCH ARTICLE

ScienceAsia 40 (2014): **94–100**

doi: 10.2306/scienceasia1513-1874.2014.40.094

Modelling a neural network using an algebraic method

Prompong Sugunnasil[a,*], **Samerkae Somhom**[a], **Watcharee Jumpamule**[a], **Natee Tongsiri**[b]

[a] Faculty of Science, Department of Computer Science, Chiang Mai University, Chiang Mai, Thailand 50200
[b] Faculty of Science, Department of Mathematics, Chiang Mai University, Chiang Mai, Thailand 50200

[*]Corresponding author, e-mail: p.sugunnasil@gmail.com

Received 8 Oct 2012
Accepted 3 Feb 2014

ABSTRACT: In this paper, a framework based on algebraic structures to formalize various types of neural networks is presented. The working strategy is to break down neural networks into building blocks, relationships between each building block, and their operations. Building blocks are collections of primary components or neurons. In turn, neurons are collections of properties functioning as single entities, transforming an input into an output. We perceive a neuron as a function. Thus the flow of information in a neural network is a composition between functions. Moreover, we also define an abstract data structure called a layer which is a collection of entities which exist in the same time step. This layer concept allows the parallel computation of our model. There are two types of operation in our model; recalling operators and training operators. The recalling operators are operators that challenge the neural network with data. The training operators are operators that change parameters of neurons to fit with the data. This point of view means that all neural networks can be constructed or modelled using the same structures with different parameters.

KEYWORDS: formal models, specification, system design method

INTRODUCTION

Neural networks have been widely used in various fields such as biology, economics, chemistry, and physics. Within each of these fields of study, they were also employed for different tasks such as classification, regression and clustering. Over the years, various structures and several methods of neural networks have been proposed to improve their performances. However, the ways in which neural networks have been studied were considered to be more horizontal, that is, they were designed for the specific case, than vertical, which would have seen neural networks being cumulatively studied[1].

It has also been raised that the implementation of neural networks seems to follow a trial and error principle. The nature of neural network development was considered as unpredictable and unrepeatable[2], not to mention that methods to design neural networks using the prior knowledge or hypotheses have yet to be established. Moreover, neural network simulators which were required to fine-tune neural networks themselves were designed to apply only to a specific type of neural network, despite the fact that each generic neural network model shares some of the elements which can be used as building blocks for constructing new neural networks[3].

To improve the situation mentioned above, we need a formal definition of neural networks capable of describing them in terms of both their ingredients and their relations. We introduce our algebraic framework based on the concept of building blocks which offer an inherent property of reusability. This framework will also enable practitioners to train different neural networks using the same simulator. Moreover, our proposed approach naturally offers insights into the constructive point of view of each neural network.

Many researchers have developed a unifying framework which was able to encapsulate both the architectures and the operations of the artificial network. The existing research can be categorized as non-computational or computational models.

The non-computational model is a type of model which focuses only on the architecture of the neural network. Specifically, they focus on connections between neurons in the network. Normally, graph theory is used as a guiding principle for this type of formalism. For example, in Ref. 1, the architecture of the network was depicted in terms of components and their associations. The components of the neural network were grouped into two types: the statics, and the dynamics. The statics are a group of components that remain unaltered during the operation on the network such as interconnection scheme, whereas the dynamics are a group of components which vary during the operation. Beside the graph-based approach, the set-based technique was also used to model neural networks. An interesting work in this category was the

Z notation introduced in Ref. 2.

On the other hand, the computational model includes both the architecture, and the computation. The core idea of this approach is to decompose neural networks into components where each of them cannot be modelled in separation from each other. For example, the tripartite model introduced in Ref. 4 includes the environment in which the network operates, the input/output which describes both pre- and post-processing, and the core which is the computational unit of the network. In Ref. 3, a formal framework to describe the neural network called a neural abstract machine was proposed in order to provide a platform to study the properties of neural networks. The neural abstract machine is based on the concept of the algebraic methods which separates the data and the operation on the data. The formal framework for the spiking neural network is proposed in Ref. 5. The spiking neural network is decomposed into a finer entity, such as synapses, activation functions, and neurons. Moreover, the environment can be regarded as a node. A set of node connected to each other to form the network. The execution of the model is simply done by receiving the input through ports, evaluating the input, and sending output through ports.

Although existing methods can capture various aspects of neural networks, there are still many desirable attributes missing. For example, in Ref. 6, Caelli, Guan, and Wen studied neural network in terms of collections of subnetwork module. The result indicated that modular neural network can reduce the computational complexity. The modularity of neural networks implies that there are groups of neurons functioning both under and above other groups of neurons - a nested structure. In this work, we study neural networks from a bottom-up fashion using algebraic methods as a modelling tool. We have noticed that all of neural networks are unique but they possess some common characteristics. They may have different types of activation function or different kind of learning mechanism but they are still based on the same structure - the neuron. The module in the neural network could be formed by the connection between neurons. The patterns of relationships between the neurons are what differentiate one model from the other. Another component of neural networks is the operation on the neural network. Operations on neurons are also included in our model in order to enable the computation of the model. As a result, we can construct any type of neural networks or even develop a new model of neural networks for a specific purpose.

The remainder of this paper is organized as follows. In the next section, the method to model neural networks is elaborated which begins with the overall idea on how to model the neural network. Then, the proposed modelling method is provided in details. Each definition is elaborated with explicit examples. The last section concludes the study and discusses the possible further research directions.

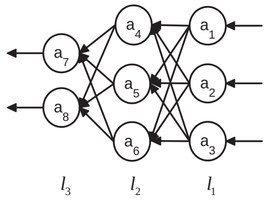

Fig. 1 A back-propagation network.

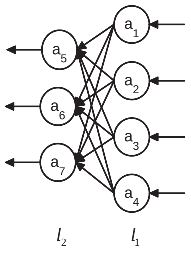

Fig. 2 A self-organizing map.

MODELLING NEURAL NETWORKS USING AN ALGEBRAIC METHOD

In this section, we describe our proposed method. We first provide our concept of modelling the neural network using algebraic approach. Then, the definitions of the fundamental elements of our modelling approach are discussed including neuron, relationships between neurons, and operations on neurons. For each definition, we provide examples of the application of the definition of neural networks using the neural networks in Fig. 1, Fig. 2, and Fig. 3. Note that we do not define a specific activation function in our example. Throughout this work, $f : K \to K$ denotes the activation function.

The general strategy is to model neural networks

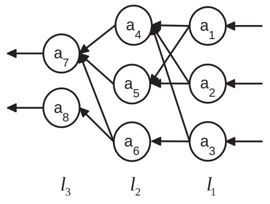

Fig. 3 A pruned neural network.

using algebraic structures and methods. We combine the formalism of algebraic system with the bottom-up concept. Modelling using algebraic structure provides a bridge between the system requirement of the target system and its formal specification[7]. Also, modelling principles based on algebraic methods provide the means to systemically analyse the modelled system[8].

In this work, the operation on a neuron is separated into two types. The first one is the structuring operator. This operator wires the neurons together to form a module of neurons. The second one is the operative operator. The operative operator is a computing operator which either uses the structure to perform calculation or adjusts the parameters of the structure. Finally, the component of the proposed method is combined into a single entity called the algebraic structure of neurons.

Throughout this paper, three neuron network models are used as an example for each definition. The back-propagation, and the self-organization map are displayed in Fig. 1, and Fig. 2, respectively. The third model displayed in Fig. 3 is the pruned version of the back-propagation.

The back-propagation is a well-known neural network with three staged training process; the feed forward of input signal, the calculation of error, and the update of the weights. In our example, there are eight components: five neurons, a_4 to a_8, and three input variables, a_1 to a_3.

For the neurons in the output layer, each output signal is compared against the corresponding target output signal in order to calculate the error information. The computation of error information is defined as

$$\delta_k = (t_k - y_k) f'(y_{in_k}) \tag{1}$$

where δ_k denotes the error information of the kth output neuron, t_k denotes the target output signal of the kth output neuron, y_k denotes the output signal of the kth output neuron, and y_{in_k} denotes the activation value of the kth output neuron.

The error information is then used to update the neuron of the output layer and propagates back to the hidden layer. The weight correction information is defined as

$$\Delta w_{jk} = \mu \delta_k z_j \tag{2}$$

where w_{jk} denotes the weight connecting jth hidden neuron and the kth output neuron, μ denotes the learning rate, and z_j denotes the output signal of jth neuron. The weight correction information used to update the neuron is

$$w_{jk}(\text{new}) = w_{jk}(\text{old}) + \Delta w_{jk} \tag{3}$$

where $w_{jk}(\text{new})$ and $w_{jk}(mathrmold)$ denote the updated weight connecting jth hidden neuron and the kth output neuron, the initial weight connecting the jth hidden neuron and the kth output neuron, respectively.

For the neurons in the hidden layer, the error information is obtained from the connected neurons in the higher layer. The incoming error information can be calculated by

$$\delta_{in_j} = \sum_{k=1}^{m} \delta_k w_{jk} \tag{4}$$

where δ_{in_j} denotes the incoming error information of the jth neuron. The error information of the neuron is calculated by

$$\delta_j = \delta_{in_j} f'(z_{in_j}) \tag{5}$$

where δ_j denotes the error information of the jth hidden neuron. The weight correction information and the weight updating process are the same as the output layer.

Fig. 2 is an example of the self-organizing map - an unsupervised neural network which brings out the underlying topological structure of the input file. The architecture of the self-organizing map consists of two layers; the cluster unit and the input unit. The cluster unit represents the topological structure of the input data which is typically one or 2-d. In this example, there are seven components including three neurons, a_5 to a_7, and four input variables, a_1 to a_4.

Unlike the back-propagation neural network, the self-organizing map is based on competitions. Each of the output signals is first compared with every cluster unit and only the cluster unit with the minimum distance is considered to be a winner. Moreover, not only the winning cluster unit is updated but also its neighbourhood cluster unit. The update function of the example is

$$w_{ij}(\text{new}) = w_{ij}(\text{old}) + \mu\Big(x_i - w_{ij}(\text{old})\Big) \tag{6}$$

where $w_{ij}(\text{new})$ and $w_{ij}(\text{old})$ denote the updated and initial weights connecting the ith input neuron and the jth cluster unit, respectively. μ denotes the learning rate, and x_i denotes the ith data value of the input pattern.

PRELIMINARIES

Definition 1 The universe of a neural network is a set that contains all of the possible values in a neural network. The universe of neural network is a ring K.

Remark 1 We have not defined what the ring K is. Basically, the universe of a neural network is a set that enables the operation between the components with some properties such as closure. Normally the universe of a neural network is a set of real numbers.

NEURON AND LAYER

In this study, the neuron is perceived as a collection of attributes, such as input channels, weights, activation functions, and learning rates. Moreover, the behaviour of a neuron is believed to be similar to a function by mapping between input and output. Essentially, the neuron is composed of an input channel, output channel, and activation function. Note that extra components such as radius of positive reinforcement, radius of negative reinforcement, and learning rate, can be included in the definition of the neuron later if it is necessary.

Definition 2 Given a universe of a neural network K, a set of outputs $K_{\text{out}} \subset K$, the arity $\tau \in \mathbb{N}$, and the activation function f mapping from $\sum_{i=1}^{\tau} w_i x_i$ to K_{out}, a neuron is a function $a : K^{\tau} \to K_{\text{out}}$ which can be more elaborate by formalizing it as a three-tuple (X, W, δ) where
(i) $X = \{x_j | x_j \in K \wedge j \in \mathbb{N} \wedge 0 < j \leqslant \tau\}$ is a finite set called the *input variables*,
(ii) $W = \{w_j | w_j \in K \wedge j \in \mathbb{N} \wedge 0 < j \leqslant \tau\}$ is a finite set called the *weight*, and
(iii) $f : K \to K_{\text{out}}$ is the activation function.

For simplicity, we denote the neuron by

$$a(x_1, x_2, \ldots, x_{\tau}).$$

The set of neurons is denoted by A.

Example 1 The neuron in Fig. 1 and Fig. 2 can be modelled as $a = (X, W, f, \mu)$ where
(i) X denotes a set of input variables. In this case, $X = \{x_1, x_2, x_3\}$ for a_4, a_5, a_6, a_7, and a_8 in Fig. 1, and $X = \{x_1, x_2, x_3, x_4\}$ for a_5, a_6, and a_7 in Fig. 2,
(ii) W denotes a set of weights where $W = \{w_1, w_2, w_3\}$,
(iii) f denotes an activation function mappings from $\sum_{i=1}^{3} x_i w_i$ to K_{out}, and
(iv) $\mu \in K$ denotes a learning rate.

Definition 3 Given a universe of neural network, K, and a set of neurons, A, a layer is formally defined as a tuple $\langle t_1, \ldots, t_n \rangle$ where $t_1, \ldots, t_n$ are neurons, data, or input variables. The set of all layers is denoted by $W(A, K)$.

Example 2 In Fig. 1, there are three layers, l_1 to l_3. Each of the layers is formalized as follows. The purpose of this example is to demonstrate the appearance of the layer. Thus we only focus on the architecture of each layer. $l_1 = \langle a_1, a_2, a_3 \rangle$, $l_2 = \langle a_4, a_5, a_6 \rangle$, and $l_3 = \langle a_7, a_8 \rangle$.

The input data in the neural network are a layer whose members is all data from the universe of a neural network. The set of all data is denoted by a set $D = \{\langle d_1, \ldots, d_p \rangle | d_1, \ldots, d_p \in K \wedge p \in \mathbb{N}\}$.

OPERATIONS ON NEURONS

In this section, we introduce the operations on the neuron. There are two types of operation: *recalling operation*, and *training operation*. The training operation is a process of changing the parameters of the structure in relation to the training data. The training operation is modelled as a function mapping from a neuron to another neuron. The recalling operation is a process of challenging a neural network with a data, and yielding the output. In this work, the recalling operation is modelled as a function mapping from a neural network and data to data. The training operation is supervised or unsupervised. The supervised approach includes the target output signal to guide the training, while the unsupervised approach does not. Furthermore, the data are inherently given to the structure at the input channel which is represented by the input variables.

Structuring operator

The neurons can be assembled to function as a module in the neural network. In this paper, we call the module of neurons a *structure*. The structure in the neural network represents how information flows in the neural network. Additionally, structures can also coordinate with one another to construct a new structure. The types of behaviour of the structure are the same as the neurons' except that the end product of the structure's computation is a product of computation of the members in the structure. In this study, the essence of the structure is captured by using the composition of neurons where the outputs of neurons in the lower level functions as the inputs for the neurons in the higher level. In other word, the concept of the composition is used to model the flow of information in the neural network. The beneficial point in using the composition is that the layer can be designed in a separate manner.

As mentioned in Definition 3, there are two types of components existing at a time period, the input variable and the neuron. We perceive them as a term of the neuron. The term of neurons which combines the input variable and the neuron is denoted by $V(X, A) = X \cup A$.

Definition 4 Let K be a universe of a neural network, A be a set of neurons, and $W(A, K)$ be a set of layer. The composition of neuron and layer is given as

$$S : V(X, A) \times W(A, K) \to V(X, A)$$

by inductively setting the following steps. If $s = x_j$ where x_j denotes the jth input variable, then $S(s, \langle t_1, t_2, \dots, t_n\rangle) = t_j$. If $s = a(a^1, \dots, a^\tau)$ and $a^1, \dots, a^\tau \in A$, then $S(s, \langle t_1, t_2, \dots, t_n\rangle) = a(S(a^1, \langle t_1, \dots, t_n\rangle), \dots, S(a^{\mathrm{r}}, \langle t_1, \dots, t_n\rangle))$.

Definition 5 Let K be a universe of neural network, A be a set of neurons, and $W(A, K)$ be a set of layer. The composition of layers is given as

$$\oplus : W(A, K) \times W(A, K) \to W(A, K)$$

by setting $\langle t_1, \dots, t_n\rangle \oplus \langle t'_1, \dots, t'_m\rangle$ equal to $\langle S(t_1, \langle t'_1, \dots, t'_m\rangle), \dots, S(t_n, \langle t'_1, \dots, t'_m\rangle)\rangle$.

Example 3 An illustration of composition between neurons in l_2 and the input variables in l_1 of the neural network in Fig. 1 is given as

$$S\left(a_4(x_1, x_2, x_3), \langle a_1, a_2, a_3\rangle\right) = a_4(a_1, a_2, a_3).$$

An illustration of composition between neurons in l_3 and the neurons in l_2 is given by

$$\begin{aligned} &S(a_7(x_1, x_2, x_3), \langle a_4(a_1, a_2, a_3), a_5(a_1, a_2, a_3),\\ &\qquad a_6(a_1, a_2, a_3)\rangle)\\ &\quad a_7(a_4(a_1, a_2, a_3), a_5(a_1, a_2, a_3), a_6(a_1, a_2, a_3)).\end{aligned}$$

The layer composition between layer l_1 and layer l_2 is given by

$$\begin{aligned} &\langle a_4(x_1, x_2, x_3), a_5(x_1, x_2, x_3), a_6(x_1, x_2, x_3)\rangle \oplus\\ &\qquad \langle a_1, a_2, a_3\rangle\\ &= \langle a_4(a_1, a_2, a_3), a_5(a_1, a_2, a_3), a_6(a_1, a_2, a_3)\rangle.\end{aligned}$$

The total structure of the example in composition form is given by

$$\begin{aligned} &\langle a_7(x_1, x_2, x_3), a_8(x_1, x_2, x_3)\rangle \oplus\\ &\quad (\langle a_4(x_1, x_2, x_3), a_5(x_1, x_2, x_3), a_6(x_1, x_2, x_3)\rangle \oplus\\ &\qquad \langle a_1, a_2, a_3\rangle)\\ &= \langle a_7\left(a_4(a_1, a_2, a_3), a_5(a_1, a_2, a_3), a_6(a_1, a_2, a_3)\right),\\ &\quad a_8\left(a_4(a_1, a_2, a_3), a_5(a_1, a_2, a_3), a_6(a_1, a_2, a_3)\right)\rangle.\end{aligned}$$

Example 4 An illustration of composition between neurons in l_2 and the input variables in l_1 of the neural network in Fig. 2 is given by

$$S\left(a_5(x_1, x_2, x_3, x_4), \langle a_1, a_2, a_3, a_4\rangle\right).$$

The total structure of the example in composition form is given by

$$\begin{aligned} &\langle a_5(x_1, x_2, x_3, x_4), a_6(x_1, x_2, x_3, x_4),\\ &\quad a_7(x_1, x_2, x_3, x_4)\rangle \oplus \langle a_1, a_2, a_3, a_4\rangle.\end{aligned}$$

Example 5 As an extension of the neural network in Fig. 1, we provide a more complicated neural network model by randomly pruning the network. The figure of the extended example is shown in Fig. 3. The total structure of example in composition form is given by

$$\begin{aligned} &\langle a_7(x_1, x_2, x_3), a_8(x_3)\rangle \oplus\\ &\quad \langle a_4(x_1, x_2, x_3), a_5(x_1, x_2), a_6(x_3)\rangle \oplus \langle a_1, a_2, a_3\rangle\\ &\qquad = \langle a_7(a_4(a_1, a_2, a_3), a_5(a_1, a_2), a_6(a_3)),\\ &\qquad\qquad a_8(a_6(a_3))\rangle.\end{aligned}$$

We list some of the fundamental properties of the composite neuron term. First, the composition of neural network is not commutative. The flow of information in the neural network is one direction. The order of the neurons contributes to the overall function of the network. Hence it is interchangeable. The composition of the neural network preserves the associativity. Within the same term of composition, the order of operations does not have an influence on either the performance or the result as long as the sequence is preserved.

The measurement of the structure's complexity is also examined in this paper. We identify the structure with trees. Typically, the tree is a connected graph without any cycles. The concept of complexity in terms of the tree is the height of a tree that represents the structure. Since the member of a certain structure is a nested structure of the layer, the complexity of the structure is determined through the maximum complexity of the member in the structure.

Definition 6 Let $a_\tau(t_1, \dots, t_\tau)$ be a neuron. The depth of a is determined by the maximum depth of its members. We formally define the depth of neuron's structure by

$$\mathrm{depth}(a) = \max\{\mathrm{depth}(t_1), \dots, \mathrm{depth}(t_\tau)\} + 1$$

where if $t = x \in X$ then $\mathrm{depth}(t) = 0$, and if $t = a'(t'_1, \dots, t'_{\tau'})$, then $\mathrm{depth}(t) = \max\{\mathrm{depth}(t'_1), \dots, \mathrm{depth}(t'_{\tau'})\} + 1$.

Example 6 Consider the back-propagation neural network in Fig. 1, there are eight components: $a_1, a_2, a_3, a_4, a_5, a_6, a_7$, and a_8 where a_1, a_2, and a_3 are input variables, a_4, a_5, and a_6 are neurons in the hidden layer, and a_7, and a_8 are neurons in the output layer. Each of the neurons has its own input pattern. The depth of each component can be determined as $\operatorname{depth}(a_1) = \operatorname{depth}(a_2) = \operatorname{depth}(a_3) = 0$, $\operatorname{depth}(a_4) = \operatorname{depth}(a_5) = \operatorname{depth}(a_6) = 1$, and $\operatorname{depth}(a_7) = \operatorname{depth}(a_8) = 2$.

Operative operator

The recalling operation is an operation to calculate the output signal of the input signal based on the parameter of the structure of neuron. In Definition 2, each neuron is already incorporates the activation function which enables the neuron to recall the data. Thus the definition of the recalling operation can focus solely on the behaviour. We give the formal definition of the recalling operation in terms of the composite form.

Definition 7 We say that a function O_r is a recalling operator, if $O_\mathrm{r} : A \times D \to K_t$ where A denotes a set of neurons, D denotes a set of data, and $K_t \subseteq K$ denotes a set of target output signal. Suppose that $a(t_1, t_2, \ldots, t_\tau)$ is a neuron with τ input variables. The recalling operator can be formally defined as

$$O_\mathrm{r}\left(a(t_1, t_2, \ldots, t_\tau), d\right) = \begin{cases} f\left(\sum_{i=1}^{\tau} d_i w_i\right), & \operatorname{depth}(a) = 0, \\ f\left(\sum_{i}^{\tau} O_\mathrm{r}(t_i, d) w_i\right), & \operatorname{depth}(a) \neq 0. \end{cases}$$

Definition 8 We say that a function O_{t_s} is a supervised training operator if $O_{t_s} : A \times D \times K_t \to A$ where A denotes a set of neurons, D denotes a set of data, and $K_t \subseteq K$ denotes a set of target output signal.

Example 7 In this example, the training operation of the back-propagation neural network in Fig. 1 and Fig. 3 is used as a demonstration. Basically, the weight updating process of our modelling method is the same as the traditional method except that it is rewritten in an algebraic representation. If the neuron is in the output layer, the weight of the neuron is updated by

$$w_j(\text{new}) = w_j(\text{old}) + \mu\left(\left(k - f\left(\sum_{i=1}^{\tau} w_i x_i\right)\right) f'\left(\sum_{i=1}^{\tau} w_i x_i\right)\right)(x_j)$$

where w_j denotes the weight connecting the neuron and the jth input, μ denotes the learning rate, k denotes the target output signal, f denotes the activation function, f' denotes the derivative activation function, and x_k denotes the kth input. If the neuron is in the hidden layer, the weight of the neuron is updated by

$$w_j(\text{new}) = w_j(\text{old}) + \mu\left(\delta_j f'\left(\sum_{i=1}^{\tau} w_i x_i\right)\right)(x_j)$$

where δ_j denotes the sum of error information gathered from neurons in higher layers which uses information from the jth neuron.

Definition 9 We say that a function O_{t_u} is an unsupervised training operator if $O_{t_\mathrm{u}} : A \times D \to A$ where A denotes a set of neurons, and D denotes a set of data.

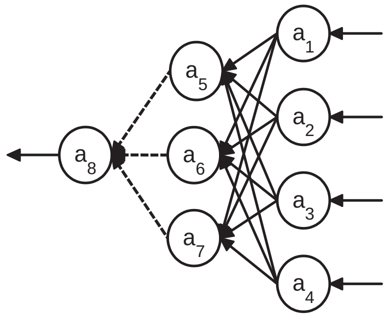

Fig. 4 A neural network model.

Example 8 Like Example 7, the function of the neural network is preserved. The self-organizing map begins its weight updating process by determining the winner node. The winner node is the least different neuron compared to the input signal. Then, weights of both the winner node and its neighbour are updated. When the self-organizing map is updated, the process involves with not only the winner node but also the surrounding neurons. One way to solve the problem is to perceive the self-organizing map as a whole. Moreover, the process to determine the winner node is not the individual level process, but it is the whole level process which requires the information of every element. Hence the additional neuron has to be included to represent the entire neural network. In this case, the neuron a_8 is added to fill the requirement. The illustration of the new neural network is shown in Fig. 4. The weight training of the self-organizing map is given separately.

(i) If the neuron is the representation unit, the training process of this neuron is to call the recalling function on every neuron in the network and determines the winner node. Then, the representation unit performs the training operation on all of the neuron in the cluster unit.

(ii) If the neuron is in the cluster layer, the weight of the neuron is updated using

$$w_j(\text{new}) = w_j(\text{old}) + \mu h(j)(x_j - w_j)$$

where w_j is the jth weight, x_j denotes the jth data, and $h(j)$ denotes the update information in relation to the distance from the winner node.

Definition 10 Let $W(A, K)$ denotes a set of layers, A denotes a set of neurons, D denotes a set of data, and $K_t \subseteq K$ denotes a set of target output signals. The extended supervised training operator is defined as

$$\hat{O}_{t_s} : W(A,K) \times D \times K_t \to W(A,K)$$

by setting

$$\begin{aligned}\hat{O}_{t_s}\left(\langle t_1,\dots,t_n\rangle, \langle d_1,\dots,d_n\rangle, k\right) = \\ \langle O_{t_s}\left(t_1, \langle d_1,\dots,d_n\rangle, k\right), \ \dots, \\ O_{t_s}\left(t_n, \langle d_1,\dots,d_n\rangle, k\right)\rangle .\end{aligned}$$

Definition 11 $W(A,K)$ denotes a set of layer, A denotes a set of neurons, and D denotes a set of data. The extended unsupervised training operator is given by

$$\hat{O}_{t_\mathrm{u}} : W(A,K) \times D \to W(A,K)$$

by setting $\hat{O}_{t_\mathrm{u}}(\langle t_1,\dots,t_n\rangle, \langle d_1,\dots,d_n\rangle) = \langle O_{t_\mathrm{u}}(t_1, \langle d_1,\dots,d_n\rangle), \dots, O_{t_\mathrm{u}}(t_n, \langle d_1,\dots,d_n\rangle)\rangle$.

Remark 2 In this study, we regard the order of the data given to the neural network as a training schema. The training schema is varied depending on the designer.

The set of operators is denoted by O.

ALGEBRAIC STRUCTURE OF NEURONS

Definition 12 A denotes a set of neurons, D denotes a set of data, and O denotes a set of operations. We define the neuron algebra using $\mathcal{A} = (A; D, O)$ where
(i) A is a set denoted the neurons,
(ii) D is a set denoted the data, and
(iii) O is a set denoted the operation on the neuron.

CONCLUSION AND DISCUSSION

Using the proposed framework, the artificial neural network can be designed in both directions: the top-down approach and the bottom-up approach. The network can be decomposed into several subnetwork modules due to their functions by focusing on the layer with the maximum depth, and moving down the network hierarchy. The neurons in the layer with the maximum depth represent the output layer of the network which is the function of the network. On the other hand, the designer can also focus the detail on the neurons in the layer with the maximum depth which represent the first group of the neurons which come in contact with the input. Consider

$$\langle x_1^1, x_2^1, \dots, x_{n_1}^1\rangle \oplus \dots \oplus \langle x_1^m, x_2^m, \dots, x_{n_m}^m\rangle .$$

where the x_n^m represents the nth neurons of the mth layer. The top-down approach is to design the network from the first layer to the mth layer, and the bottom-up approach is to work from the mth layer to the first layer. Moreover, since the proposed method uses the fundamental mathematical notation, the framework can be implemented in any formal language. The future extension of the proposed framework includes the incorporation of the temporal components which lead to a more expressive way to specify the artificial neural network.

Acknowledgements: The first author was supported by the Graduate School, Chiang Mai University. The fourth author was supported by the Department of Mathematics, Faculty of Science, Chiang Mai University.

REFERENCES

1. Fiesler E (1994) Neural network classification and formalization. *Comput Stand Interfac* **16**, 231–9.
2. Senyard A, Kazmierczak E, Sterling L (2003) Software engineering methods for neural networks. In: Proceedings of the Tenth Asia-Pacific Software Engineering Conference Software Engineering Conference, IEEE Computer Society, Washington, DC, pp 468–77.
3. Borger E, Sona D (2001) A neural abstract machine. *J Univers Comput Sci* **7**, 1006–23.
4. Dorffner G, Wiklicky H, Prem E (1994) Formal neural network specification and its implications on standardization. *Comput Stand Interfac* **16**, 205–19.
5. Zaharakis ID, Kameas AD (2008) Modeling spiking neural networks. *Theor Comput Sci* **395**, 57–76.
6. Caelli T, Guan L, Wen W (1999) Modularity in neural computing. *Proc IEEE* **87**, 1497–518.
7. Goguen JA, Thatcher JW, Wagner EG, Wright JB (1977) Initial algebra semantics and continuous algebras. *J ACM* **24**, 68–95.
8. Loegel G, Ravishankar CV (1994) An algebraic approach to modeling in software engineering. In: Proceedings of the 3rd International Conference on Methodology and Software Technology: Algebraic Methodology and Software Technology, Springer-Verlag, London, pp 385–92.