Extracting Propositions from Trained Neural Networks

Hiroshi Tsukimoto
Research & Development Center, Toshiba Corporation
70, Yanagi-cho, Saiwai-ku, Kawasaki, 210
Japan

Abstract

This paper presents an algorithm for extracting propositions from trained neural networks. The algorithm is a decompositional approach which can be applied to any neural network whose output function is monotone such as sigmoid function. Therefore, the algorithm can be applied to multi-layer neural networks, recurrent neural networks and so on. The algorithm does not depend on training methods. The algorithm is polynomial in computational complexity. The basic idea is that the units of neural networks are approximated by Boolean functions. But the computational complexity of the approximation is exponential, so a polynomial algorithm is presented. The authors have applied the algorithm to several problems to extract understandable and accurate propositions. This paper shows the results for *votes* data and *mushroom* data. The algorithm is extended to the continuous domain, where extracted propositions are continuous Boolean functions. Roughly speaking, the representation by continuous Boolean functions means the representation using conjunction, disjunction, direct proportion and reverse proportion. This paper shows the results for *iris* data.

1 Introduction

Extracting rules or propositions from trained neural networks is important[1], [6], Although several algorithms have been proposed by Shavlik, Ishikawa and others [2],[3], every algorithm is subject to problems in that it is applicable only to certain types of networks or to certain training methods.

This paper presents an algorithm for extracting propositions from trained neural networks. The algorithm is a decompositional approach which can be applied to any neural network whose output function is monotone such as sigmoid function. Therefore, the algorithm can be applied to multi-layer neural networks, recurrent neural networks and so on. The algorithm does not depend on training methods, although some other methods[2], [3] do. The algorithm does not modify the training results, although some other methods [2] do. Extracted propositions are Boolean functions. The algorithm is polynomial in computational complexity.

The basic idea is that the units of neural networks are approximated by Boolean functions. But the computational complexity of the approximation is exponential, so a polynomial algorithm is presented. The basic idea of reducing the computational complexity to a polynomial is that only low order terms are generated, that is, high order terms are neglected. Because high order terms are not informative, the approximation by low order terms is accurate[4].

In order to obtain accurate propositions, when the hidden units of neural networks are approximated to Boolean functions, the distances between the units and the functions are not measured in the whole domain, but in the domain of learning data. In order to obtain simple propositions, only the weight parameters whose absolute values are big are used.

The authors have applied the algorithm to several problems to extract understandable and accurate propositions. This paper shows the results for *votes* data and *mushroom* data.

The algorithm is extended to the continuous domain, where extracted propositions are continuous Boolean functions. Roughly speaking, the representation by continuous Boolean functions means the representation using conjunction, disjunction, direct proportion and reverse proportion. This paper shows the results for *iris* data.

Section 2 explains the basic method. Section 3 presents a polynomial algorithm. Section 4 describes the experiments. Section 5 extends the algorithm to continuous domains and applies it to *iris* data.

The following notations are used. *x,y,..* stand for variables. f,g,.. stand for functions.

2 The basic method

There are two kinds of domains, that is, discrete domains and continuous domains. The discrete domains can be reduced to {0,1} domains by dummy variables. So only {0,1} domains have to be discussed. Here, the domain is {0,1}. Continuous domains will be discussed later.

2.1 The basic method

The basic method is that the units of neural networks are approximated by Boolean functions. The output functions are sigmoid functions, so the values of the units are [0,1].

Let (f_i) be the values of a unit of a neural network. Let $(g_i)(g_i = 0 \text{ or } 1)$ be the values of Boolean functions. The basic method is as follows:

$$g_i = \begin{cases} 1(f_i \geq 0.5), \\ 0(f_i < 0.5). \end{cases}$$

This method minimizes Euclidean distance.

Fig. 1 shows a case of two variables. Crosses stand for the values of a unit of a neural network and circles stand for the values of a Boolean function. $00, 01, 10$ and 11 stand for the domains, for example, 00 stands for $x = 0, y = 0$.

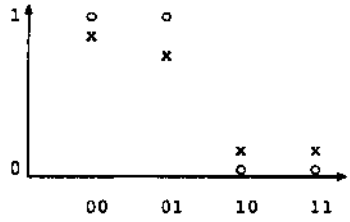

Figure 1: Approximation

In this case, the values of the Boolean function $g(x, y)$ are as follows:

$$g(0,0) = 1, g(0,1) = 1, g(1,0) = 0, g(1,1) = 0.$$

Generally, let $g(x_1, ..., x_n)$ stand for a Boolean function, and let $g_i(i = 1, ..., 2^n)$ stand for values of a Boolean function, then the Boolean function is represented by the following formula:

$$g(x_1, ..., x_n) = \sum_{i=1}^{2^n} g_i a_i,$$

where $\sum$ is disjunction, and a_i is the atom corresponding to g_i, that is,

$$a_i = \prod_{j=1}^{n} e(x_j) \ (i = 1, ..., 2^n),$$

where

$$e(x_j) = \begin{cases} \overline{x_j}(e_j = 0), \\ x_j(e_j = 1), \end{cases}$$

where $\prod$ stands for conjunction, $\overline{x}$ stands for the negation of x, and e_j is the substitution for x_j, that is, $e_j = 0$ or 1. The above formula can be easily verified. Therefore, in the case of Fig. 1, the Boolean function is as follows:

$g(x, y) = g(0,0)\bar{x}\bar{y} + g(0,1)\bar{x}y + g(1,0)x\bar{y} + g(1,1)xy,$
$g(x, y) = 1\bar{x}\bar{y} + 1\bar{x}y + 0x\bar{y} + 0xy,$
$g(x, y) = \bar{x}\bar{y} + \bar{x}y,$
$g(x, y) = \bar{x}.$

2.2 An example

We show that the proposition of Exclusive OR, that is, $x\bar{y} \vee \bar{x}y$, can be obtained by approximating a neural network which has learned Exclusive OR using the back-propagation method. First, we explain the structure of the network. Second, we analyze the learning results.

Fig. 2 shows the structure of the neural network.

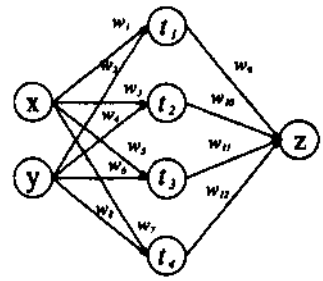

Figure 2: Neural Network

In Fig. 2, x and y are inputs, z is an output, and t_i's are the outputs of four hidden units. Each output is represented as follows:

$t_1 = S(w_1 x + w_2 y + h_1),$
$t_2 = S(w_3 x + w_4 y + h_2),$
$t_3 = S(w_5 x + w_6 y + h_3),$
$t_4 = S(w_7 x + w_8 y + h_4),$
$z = S(w_9 t_1 + w_{10} t_2 + w_{11} t_3 + w_{12} t_4 + h_5),$

where w_i's stand for weight parameters, $S(x)$ stands for sigmoid function and h_i's stand for biases.

The learning results with 1000 time repetition are as follows:

$w_1 = 2.51, \ w_2 = -4.80, \ w_3 = -4.90,$
$w_4 = 2.83, \ w_5 = -4.43, \ w_6 = -4.32,$
$w_7 = -0.70, \ w_8 = -0.62, \ w_9 = 5.22,$
$w_{10} = 5.24, \ w_{11} = -4.88, \ w_{12} = 0.31,$
$h_1 = -0.83, \ h_2 = -1.12, \ h_3 = 0.93,$
$h_4 = -0.85, \ h_5 = -2.19.$

For example,

$t_1 = S(2.51x - 4.80y - 0.83),$

and the values of $t_1(0,0), t_1(0,1), t_1(1,0)$, and $t_1(1,1)$ are as follows:

$t_1(1,1) = S(2.51 \cdot 1 - 4.80 \cdot 1 - 0.83) = S(-3.12),$
$t_1(1,0) = S(2.51 \cdot 1 - 4.80 \cdot 0 - 0.83) = S(1.68),$
$t_1(0,1) = S(2.51 \cdot 0 - 4.80 \cdot 1 - 0.83) = S(-5.63),$
$t_1(0,0) = S(2.51 \cdot 0 - 4.80 \cdot 0 - 0.83) = S(-0.83).$

$S(-3.12) \simeq 0$, $S(1.68) \simeq 1$, $S(-5.63) \simeq 0$, and $S(-0.83) \simeq 0$, therefore

$t_1 \simeq x\bar{y}.$

In a similar manner, the others are approximated by the following Boolean functions:

$t_2 = \bar{x}y,$
$t_3 = \bar{x}\bar{y},$
$t_4 = 0.$

The learning result of z is as follows:

$z = t_1 t_2 \vee \bar{t_1} t_2 \bar{t_3} \vee t_1 \bar{t_2} \bar{t_3}.$

By substituting $t_1 = x\bar{y}$, $t_2 = \bar{x}y$, and $t_3 = \bar{x}\bar{y}$, in the above formula,

$z = x\bar{y} \vee \bar{x}y$

has been obtained. This proposition is Exclusive OR.

Now we have confirmed that the logical proposition of Exclusive OR can be obtained by approximating the neural network which has learned Exclusive OR. We understand the following items.

1. The learning results of 4 hidden units and an output unit.
2. The output of hidden unit 4, t_4, is not included in the Boolean function of the output unit, z, so hidden unit 4 does not work and can be deleted.
3. The other three hidden units work, so they are necessary and cannot be deleted.

A merit of the decompositional approach is that trained neural networks can be understood by the unit.

3 A polynomial algorithm

Obviously, the computational complexity of the basic method is exponential, so the basic method is not realistic. Therefore, the computational complexity should be reduced to a polynomial. This section presents the polynomial algorithm. The basic idea of reducing the computational complexity to a polynomial is that DNF formulas consisting of only low order terms are generated from the lowest order up to a certain order, that is, high order terms are neglected. Because high order terms are not informative[4j, the approximation by low order terms is accurate.

The brief outline of the algorithm follows.

1. Check the existences of terms after the approximation from the lowest order.
2. Connect the terms which exist after the approximation to make a DNF formula.
3. Execute the above two procedures up to a certain order.

In this section, first, the condition that a term $x_{i_1} \cdot \cdot x_{i_k}\bar{x}_{i_{k+1}} \cdot \cdot \bar{x}_{i_l}$ exists in the Boolean function after approximation is presented. Second, the generation of DNF formulas is explained. Third, in order to obtain accurate propositions, the condition is modified in a manner such that the distance between the hidden unit of a neural network and a Boolean function is not measured in the whole domain, but in the domain of the learning data. Fourth, in order to obtain simple propositions, the condition is modified in a way such that only the terms consisting of variables whose weight parameters' absolute values are big are generated.

3.1 The condition that $x_{i_1} \cdot \cdot x_{i_k}\bar{x}_{i_{k+1}} \cdot \cdot \bar{x}_{i_l}$ exists in the Boolean function after approximation

Let a unit of a neural network be

$$S(p_1x_1 + ... + p_nx_n + p_{n+1}),$$

where $S(\cdot)$ is an arbitrary function. Let the orthonormal expansion be

$$f_1x_1 \cdot \cdot x_n + f_2x_1 \cdot \cdot \overline{x_n} + ... + f_{2^n}\overline{x_1} \cdot \cdot \overline{x_n},$$

where "+" and "$\cdot$" are the sum and the product of elementary algebra and $\bar{x}$ is $1 - x$. Boolean functions can be represented by elementary algebra and the atoms can be the orthonormal functions. Due to space limitations, for details, see [10].

Theorem 1 *f_i's are as follows:*

$f_1 = S(p_1 + ... + p_n + p_{n+1})$,
$f_2 = S(p_1 + ... + p_{n-1} + p_{n+1})$,
...
$f_{2^{n-1}} = S(p_1 + p_{n+1})$,
$f_{2^{n-1}+1} = S(p_2 + ... + p_n + p_{n+1})$,
...
$f_{2^n-1} = S(p_n + p_{n+1})$,
$f_{2^n} = S(p_{n+1})$.

Proof f_1 can be obtained by substituting $x_1 = \cdots = x_n = 1$ in the following formula:

$$\begin{aligned} & S(p_1x_1 + ... + p_nx_n + p_{n+1}) \\ = \; & f_1x_1 \cdot \cdot x_n + f_2x_1 \cdot \cdot \overline{x_n} + ... + f_{2^n}\overline{x_1} \cdot \cdot \overline{x_n}. \end{aligned}$$

f_{2^n} can be obtained by substituting $x_1 = \cdots = x_n = 0$ in the above formula. The other coefficients can be obtained in a similar manner. □

Theorem 2 *The condition that*

$$x_{i_1} \cdot \cdot x_{i_k}\bar{x}_{i_{k+1}} \cdot \cdot \bar{x}_{i_l}$$

exists in the Boolean function after approximation is as follows:

$$S(\sum_{i_1}^{i_k} p_j + p_{n+1} + \sum_{1 \le j \le n, j \neq i_1, ..., i_l, p_j < 0} p_j) \ge 0.5,$$

where $S(\cdot)$ means a monotone increasing function.

Proof Consider the existence condition of x_1 in the Boolean function after approximation. For simplification, this condition is called the existence condition. Because

$$x_1 = x_1x_2 \cdot \cdot x_n \vee x_1x_2 \cdot \cdot \overline{x_n} \vee ... \vee x_1\overline{x_2} \cdot \cdot \overline{x_n},$$

the existence of x_1 equals the existence of the following terms:

$$x_1x_2 \cdot \cdot x_n, x_1x_2 \cdot \cdot \overline{x_n}, ..., x_1\overline{x_2} \cdot \cdot \overline{x_n}.$$

The existence of the above terms means that all coefficients of these terms $f_1, f_2, ..., f_{2^{n-1}}$ are greater than or equal to 0.5 (See 2.1). That is, the existence condition is

$$min\{f_i\} \ge 0.5 (1 \le i \le 2^{n-1}).$$

Because f_i's $(1 \le i \le 2^{n-1})$ are
$f_1 = S(p_1 + ... + p_n + p_{n+1})$,
$f_2 = S(p_1 + ... + p_{n-1} + p_{n+1})$,
...
$f_{2^{n-1}} = S(p_1 + p_{n+1})$,

each $f_i(1 \leq i \leq 2^{n-1})$ contains p_1. If each p_j is non-negative, $f_{2^{n-1}}(= S(p_1+p_{n+1}))$ is the minimum because the other f_i's contain other p_j's, and therefore the other f_i's are greater than or equal to $f_{2^{n-1}}(= S(p_1+p_{n+1}))$. Note that the above argument holds, because $S(\cdot)$ is monotone increasing. Generally, since each p_j is not necessarily non-negative, the $min\{f_i\}$ is f_i which contains all negative p_j. That is,

$$min\{f_i\} = S(p_1 + p_{n+1} + \sum_{1 \leq j \leq n, j \neq 1, p_j < 0} p_j),$$

which necessarily exists in $f_i(1 \leq i \leq 2^{n-1})$, because $f_i(1 \leq i \leq 2^{n-1})$ is

$$S(p_1 + p_{n+1} + (\text{arbitrary sum of } p_j(2 \leq j \leq n))).$$

From the above arguments, the existence condition of x_1, $min\{f_i\} \geq 0.5$, is as follows:

$$S(p_1 + p_{n+1} + \sum_{1 \leq j \leq n, j \neq 1, p_j < 0} p_j) \geq 0.5.$$

Since $S(p_1x_1 + ... + p_nx_n + p_{n+1})$ is symmetric for x_i, the above formula holds for other variables; that is, the existence condition of x_i is

$$S(p_i + p_{n+1} + \sum_{1 \leq j \leq n, j \neq i, p_j < 0} p_j) \geq 0.5.$$

Similar discussions hold for $\overline{x}_i$, so we have the following formula:

$$S(p_{n+1} + \sum_{1 \leq j \leq n, j \neq i, p_j \leq 0} p_j) \geq 0.5.$$

Similar discussions hold for higher order terms $x_{i_1} \cdot \cdot x_{i_k}\overline{x}_{i_{k+1}} \cdot \cdot \overline{x}_{i_l}$, so we have the following formula:

$$S(\sum_{i_1}^{i_k} p_j + p_{n+1} + \sum_{1 \leq j \leq n, j \neq i_1, ..., i_l, p_j \leq 0} p_j) \geq 0.5. \quad (1)$$

□

Since the output function is monotone increasing, this theorem cannot be applied to units whose output function is a Gaussian function. Moreover, it can be easily verified that a similar theorem holds when the output function is monotone decreasing.

3.2 Generation of DNF formulas

The algorithm generates terms using (1) from the lowest order up to a certain order. A DNF formula can be generated by taking the disjunction of the terms generated by (1). A term whose existence has been confirmed does not need to be re-checked in higher order terms. For example, if the existence of x is confirmed, then it also implies the existence of xy, xz ,..., because $x = x \vee xy \vee xz \vee ...$; hence, it is unnecessary to check the existence of xy, xz,.... As can be seen from the above discussion, the generation method of DNF formulas includes reductions such as $xy \vee xz = x$. An example follows. Let

$$S(0.65x_1 + 0.23x_2 + 0.15x_3 + 0.20x_4 + 0.02x_5 - 0.5)$$

be a unit of a neural network.
For $x_i(i = 1, 2, 3, 4, 5)$,

$$S(p_1 + p_6) \geq 0.5,$$

and therefore x_1 exists.
For $x_ix_j(i, j = 2, 3, 4, 5)$,

$$S(p_i + p_j + p_6) < 0.5,$$

and therefore no x_ix_j exists.
For $x_ix_jx_k(i, j, k = 2, 3, 4, 5)$,

$$S(p_2 + p_3 + p_4 + p_6) \geq 0.5,$$

and therefore $x_2x_3x_4$ exists.
Because higher order terms cannot be generated from x_5, the algorithm stops. Therefore, x_1 and $x_2x_3x_4$ exist and the DNF formula is the disjunction of these terms, that is, $x_1 \vee x_2x_3x_4$.

3.3 A modified condition

This subsection discusses the improvement of the accuracy of the proposition obtained from a hidden unit. In the domain outside the domain of the learning data, the values of a trained neural network are predicted values. So the values in the domain are not reliable. If we have a trained network and no learning data, we cannot know which is the domain of the learning data. Thus only (1) is available. If we have a trained network and the learning data, we can use the domain of the learning data in order to improve the accuracies of the propositions obtained. In order to obtain accurate propositions, when the units of neural networks are approximated to Boolean functions, the distances between the units and the functions are not measured in the whole domain, but only in the domain of the learning data. That is, the left-hand side of (1), which means the minimum value for a term over the whole domain, is modified to the minimum value for a term in the domain of the learning data. Theorem 2 is modified as shown below.

Theorem 3 *Let a unit of a neural network be*

$$S(p_1x_1 + ... + p_nx_n + p_{n+1})$$

then, the existence condition of a term

$$x_{i_1} \cdot \cdot x_{i_k}\overline{x}_{i_{k+1}} \cdot \cdot \overline{x}_{i_l}$$

after the approximation is as follows:

$$S(\sum_{i_1}^{i_k} p_j + p_{n+1} + min\{\sum_{1 \leq j \leq n, j \neq i_1, ..., i_l} p_je_j\}) \geq 0.5,$$

where e_j is 0 or 1 and $S(\cdot)$ means a monotone increasing function. The domain used for calculating the minimum value in the above formula is limited to the domain of the learning data.

For example, let three attributes of a learning data be (x_1, x_2, x_3) and the attribute values be binary. Assume that the learning data are in the table below.

data	x_1	x_2	x_3	class
i	0	0	1	0
2	1	0	0	0
3	1	1	0	1

In this case, the domain of the learning data is (0,0,0), (1,0,0) and (1,1,0), and a unit of a neural network is

$$S(p_1x_1 + p_2x_2 + p_3x_3 + p_4).$$

The existence condition of x_1 after the approximation is as follows:

$$S(p_1 + p_4 + min\{\sum_{1 \le j \le 3, j \ne 1} p_j e_j\}) \ge 0.5.$$

But, the checking range is limited to the domain of the learning data, so the checking range is limited to the domain of the data 2 and data 3 where $X_1 = 1$. Therefore the existence condition is as follows:

$$S(p_1 + p_4 + min\{p_2 \cdot 0 + p_3 \cdot 0, p_2 \cdot 1 + p_3 \cdot 0\}) \ge 0.5$$
$$\rightarrow S(p_1 + p_4 + min\{0, p_2\}) \ge 0.5.$$

Of course, this modified condition is not applied to the output units in 3-layer networks, because the inputs of the output units come from the hidden units. Therefore, (1) is applied to the output units.

3.4 Weight parameters

Let's sort weight parameters p_is as follows:

$$|p_1| \ge |p_2| \ge \cdots \ge |p_n|.$$

When terms are generated, if all weight parameters are used, the propositions obtained are complicated. Therefore unimportant parameters should be neglected. In this paper, p_i's whose absolute value is small are regarded as the unimportant parameters. We will use p_i's up to a certain number, that is, we neglect small p_i's.

How many weight parameters are used is the next problem. Here, the weight parameters $p_1,...,p_k$ are used, where k is determined by a value based on Fourier transform of logical functions[4]. Due to space limitations, the explanation is omitted, which will be presented in another paper.

3.5 Computational complexity of the algorithm and error analysis

The computational complexity of generating the mth order terms is a polynomial of *nCm,* that is, a polynomial of *n.* Therefore, the computational complexity of generating DNF formulas from neural networks is a polynomial of n. Usual generations will be terminated up to a low order, because understandable propositions are desired. Therefore, the computational complexity is usually a polynomial of a low order.

In the case of the domain {0,1}, Linial showed the following formula [4]:

$$\sum_{|S|>k} \hat{f}(S)^2 \le 2M2^{-k^{1/2}/20},$$

where / is a Boolean function, 5 is a term, \S\ is the order of S, k is any integer, *f(S)* denotes the Fourier transform of / at *S* and *M* is the circuit's size of the function. The above formula shows the high order terms have little power; that is, low order terms are informative. Therefore, a good approximation can be obtained by generating up to a certain order.

3.6 Comparisons

This subsection briefly compares the algorithm with other algorithms. Algorithms may be evaluated in terms of five aspects: expression form, internal structure, network type and training method, quality, computational complexity[I]. Here, comparisons focus on internal structure, training method and network type.

internal structure : There are two techniques, namely decompositional and pedagogical. The decompositional algorithms obtain rules by the unit and aggregate them to a rule for the network. The pedagogical algorithms generate examples from the trained network and obtain rules from the examples. Obviously, the decompositional algorithms are better than the pedagogical algorithms in understanding the internal structures. Therefore, for example, the decompositional algorithms can be also used for the training control.

training method : The algorithm does not depend on training methods, although some other methods do. For example, [2] and [3] use special training methods and their algorithms cannot be applied to networks trained by the back-propagation method. The algorithm does not modify the training results, although some other methods [2] do.

network type : The algorithm does not depend on network types, although some other methods do. For example, [2] cannot be applied to recurrent neural networks.

4 Experiments

The training method is the back-propagation method. The repetition is stopped when the error is less than 0.01. Therefore, the error after the training is less than 0.01. The data used for the training are also used for the prediction. Therefore, the accuracy of the trained neural networks is 100%. Usual prediction experiments use data different from those used for the training. But, in this case, it is desired that the accuracy of neural networks be 100%, because we want to see how Boolean functions can approximate the neural networks. Generating terms of propositions are terminated up to second order, because simple propositions are desired.

4.1 *votes* data

This data consists of the voting records of the U.S. House of Representatives in 1984. There are 16 binary attributes. Classes are *Democrat* and *Republican:* The number of samples used for the experiment is 232. The

accuracies of propositions extracted from the trained neural networks are shown in the table below. In the table, i.w.p. stands for initial weight parameter and the numbers in the column of hidden layer mean the numbers of hidden units.

hidden layer	i.w.p.l	i.w.p.2	i.w.p.3
5	0.974	0.974	0.974
3	0.983	0.970	0.978
4	0.978	0.974	0.974

In the case of 3 hidden units and i.w.p.l, the following propositions have been obtained:

Democrat (physician-fee-freeze:n) V (adoption-of-the-budget-resolutipnry) (anti-satellite- test-ban:n) (synfuels-corporation-cutbackry),

Republican: (physician-fee-freeze: y) ((adoption-of-the-budget-resolution:n)V (anti-satellite-test-ban:y)V (synfuels-corporation-cut back: n)).

In the other cases, similar results have been obtained. The results for *votes* data by C4.5[5], which is a typical algorithm for machine learning, are as follows:

(physician-fee-freeze:n) V (adoption-of-the-budget-resolution:y) (synfuels-corporation-cut back:y) —► *Democrat,*

(adoption-of-the-budget-resolution:n)(physician-fee-freeze:y) V (physician-fee-freeze:y)(synfuels-corporation-cutback:n) —► *Republican.*

The accuracy of the result of C4.5 is 97.0%, so the propositions extracted from trained neural networks by the algorithm are a little better than the results of C4.5 in accuracy and almost the same as the results of C4.5 in understandability.

4.2 *mushroom* data

There are 22 discrete attributes concerning mushrooms such as the cap-shapes. Classes are *edible* or *poisonous.* The number of samples is 4062. The accuracies of propositions extracted from the trained neural networks are shown in the table below.

hidden layer	i.w.p.l	i.w.p.2	i.w.p.3
0	0.930	0.973	0.985
3	0.956	0.983	0.959
4	0.961	0.952	0.985

In the case of 3 hidden units and i .w.p.l, the following propositions have been obtained:

edible:(gill-size:broad)((odor:almond)V(odor:anise)V (odonnone)),

poisonous:(gill-size:narrow)V-»(odor:almond) -(odor:anise) -(odor:none).

In the other cases, similar results have been obtained. The results for *mushroom* data by C4.5 are as follows:

(odonnone) V (odonalmond) V (odonanise) —► *edible,*

(odonfoul) V (odonspicy) V (odonfishy) V (odor:pungent) V (odorxreosote) —► *poisonous.*

The accuracy of the result of C4.5 is 98.7%, so the propositions extracted from trained neural networks by the algorithm are a little worse than the results of C4.5 in accuracy and almost the same as the results of C4.5 in understandability.

5 Extension to the continuous domain

5.1 The basic idea

In this section, the algorithm is extended to continuous domains. Continuous domains can be normalized to [0,1] domains by some normalization method. So only [0,1] domains have to be discussed. First, we have to present a system of qualitative expressions corresponding to Boolean functions, in the [0,1] domain. The author presents the expression system generated by direct proportion, reverse proportion, conjunction and disjunction. Fig.3 shows the direct proportion and the inverse proportion. The inverse proportion *(y = 1 - x)is* a little different from the conventional one *(y = -x)*, because *y = 1 - x* is the natural extension of the negation in Boolean functions. The conjunction and disjunction will be also obtained by a natural extension. The functions generated by direct proportion, reverse proportion, conjunction and disjunction are called continuous Boolean functions, because they satisfy the axioms of Boolean algebra.

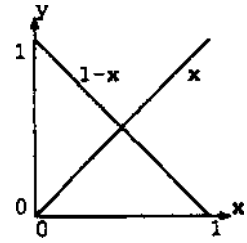

Figure 3: Direct Proportion and Reverse Proportion

Since it is desired that a qualitative expression be obtained, some quantitative values should be ignored. For example, two functions "A" and "B" in Fig. 4 are different from direct proportion *x* but the two functions are proportions. So the three functions should be identified as the same one in the qualitative expression. That is, in

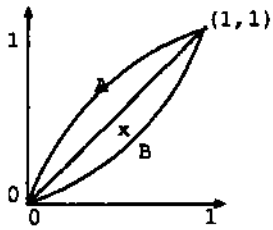

Figure 4: A Qualitative Norm

$[0,1]$, $x^k (k \geq 2)$ should be identified with x in the qualitative expression. Mathematically, a norm is necessary, by which the distance among the three functions is 0. The qualitative norm can be introduced.

In $\{0,1\}$, a unit of a neural network is a function $f : \{0,1\} \rightarrow R$. In $\{0,1\}$, $x^k = x$, because $0^k = 0$ and $1^k = 1$, where $k \geq 2$. Therefore, these functions do not contain any terms such as

$$x_1^{k_1} x_2^{k_2} \cdots x_n^{k_n},$$

where $k_i \geq 2$. These functions are called multilinear functions. Therefore, units of neural networks are multilinear functions. The space of multilinear functions is Euclidean space. See 2.1. So the unit can be approximated to a Boolean function by Euclidean norm. In $[0,1]$, similar facts hold, that is, a unit of a neural network is a multilinear function in the qualitative expression, that is, the qualitative norm, and the space of multilinear functions is Euclidean space in the qualitative norm. Thus the unit can be approximated to a continuous Boolean function by Euclidean norm.

5.2 Multilinear functions

Definition 4 *Multilinear functions of n variables are as follows:*

$$\sum_{i=1}^{2^n} a_i x_1^{e_{i1}} \cdots x_n^{e_{in}},$$

where a_i is real, x_j is variable, and e_{ij} is 0 or 1.

For example, multilinear functions of 2 variables are as follows:

$$axy + bx + cy + d.$$

In other words, multilinear functions are functions which are linear when only one variable is considered and the other variables are regarded as parameters.

Multilinear functions of the domain $[0,1]$ are considered. In the domain $[0,1]$, a qualitative norm has to be introduced. First, τ is defined, which is necessary for the definition of the qualitative norm.

Definition 5 *Let $f(x)$ be a real polynomial function. Consider the following formula:*

$$f(x) = p(x)(x - x^2) + q(x),$$

where $q(x) = ax + b$, where a and b are real, that is, $q(x)$ is the remainder. τ_x is defined as follows:

$$\tau_x : f(x) \rightarrow q(x).$$

The above definition implies the following property:

$$\tau_x(x^k) = x,$$

where $k \geq 2$. In the case of n variables, τ is defined as follows:

$$\tau = \prod_{i=1}^{n} \tau_{x_i}.$$

For example, $\tau(x^2y^3 + y + 1) = xy + y + 1$.

Definition 6 *An inner product is defined as follows:*

$$< f, g >= 2^n \int_0^1 \tau(fg)dx,$$

where f and g are multilinear functions. The above definition can satisfy the properties of inner product[7].

Definition 7 *Norm $|f|$ is defined as follows:*

$$|f| =< f, f >^{1/2}.$$

The above norm is the qualitative norm. τ is necessary for the qualitative norm, because identifying several functions with a typical function, for example, x^n with x, can be realized by τ.

Theorem 8 *The space of multilinear functions of $\{0,1\}$ domain can be a Euclidean space[10]. The space of multilinear functions in the domain [0,1] is a Euclidean space with the above inner product: Proof can be found in [7].*

5.3 Continuous Boolean functions

This subsection briefly describes continuous Boolean functions [8], [9].

Theorem 9 *The functions obtained from Boolean functions by extending the domain from $\{0,1\}$ to $[0,1]$ can satisfy all axioms of Boolean algebra with the logical operations defined below. Proof can be found in [9].*

AND : $\tau(fg)$,

OR : $\tau(f + g - fg)$,

NOT : $\tau(1 - f)$.

Therefore, the functions obtained from Boolean functions by extending the domain from $\{0,1\}$ to $[0,1]$ are called continuous Boolean functions. For example, xy and $1 - y (= \bar{y})$ are Boolean functions, where $x, y \in [0,1]$. We show a simple example for logical operation. $(X \vee Y) \wedge (X \vee Y) = X \vee Y$ is calculated as follows:

$$\begin{aligned} & \tau((x + y - xy)(x + y - xy)) \\ = \; & \tau(x^2 + y^2 + x^2y^2 + 2xy - 2x^2y - 2xy^2) \\ = \; & x + y + xy + 2xy - 2xy - 2xy \\ = \; & x + y - xy. \end{aligned}$$

In the continuous domain, fuzzy rules can be obtained from trained neural networks by some algorithms [1]. The expression by continuous Boolean functions is more understandable than fuzzy rules, whereas continuous Boolean functions are worse than fuzzy rules in accuracy. The extension of continuous Boolean functions will be included in future work.

5.4 On the approximation

The units of neural networks are not multilinear functions when the domain is [0,1]. Thus the units of neural networks should be approximated to multilinear functions using $x^n = x$. Note that the distance between a unit (f) and the nearest continuous Boolean function (g) equals the distance between the multilinear function (f') obtained by the approximation $x^n = x$ from the unit and the nearest continuous Boolean function (g). That is, $|f - g| = |f' - g|$, where $|f|$ means the qualitative norm. Thus, the approximation of the unit by a multilinear function does not degrade the approximation of the unit by a continuous Boolean function.

In the domain of $\{0,1\}$, we have Linial's theorem for error analysis, while, in the domain of [0,1], we do not have a similar theorem. A theoretical analysis of the error in the case of the [0,1] domain will be included in future work.

5.5 *iris* data

iris data consists of four continuous attributes, sepal length(=a), sepal width(=b), petal length(=c) and petal width(=d), and three classes, *setosa*, *versicolour* and *virginica*. Continuous inputs are normalized to [0,1] with maximum data and minimum data. The number of samples is 150. The experimental conditions are the same as in Section 4. Accuracy is calculated in a way such that a class whose continuous Boolean function's value is the biggest is chosen. The accuracies of propositions extracted from the trained neural networks are shown in the table below.

hidden layer	i.w.p.1	i.w.p.2	i.w.p.3
2	0.947	0.947	0.947
3	0.947	0.953	0.947

In the case of 3 hidden units and i.w.p 1, the following propositions have been obtained:

setosa: $\bar{c}\bar{d}$;petal length is small and petal width is small,

versicolour: $c\bar{d} \vee \bar{c}d$; (petal length is big and petal width is small) or (petal length is small and petal width is big),

virginica: cd; petal length is big and petal width is big.

Note that the above propositions are continuous Boolean functions. For example, $\bar{c}\bar{d}$ has been obtained for *setosa*, $\bar{c}\bar{d}$ means $(1-c)(1-d)$. In the other cases, similar results have been obtained. The learning results of C4.5 are as follows:

(petal length $\leq$ 1.9) $\rightarrow$ *setosa*,

(petal length $>$ 1.9)(petal length $\leq$ 5.3)(petal width $\leq$ 1.7) $\rightarrow$ *versicolour*,

(petal width $>$ 1.7)$\vee$ (petal length $>$ 4.9) $\rightarrow$ *virginica*.

The accuracy of the results of C4.5 is 97.3%, so the propositions extracted from trained neural networks are a little worse than the result of C4.5 in accuracy.

6 Conclusions

This paper has presented an algorithm for extracting propositions from trained neural networks. The algorithm does not depend on structures of networks and training methods. The algorithm is polynomial in computational complexity. The algorithm has been extended to the continuous domain. This paper has shown the experimental results for *votes* data, *mushroom* data and *iris* data. The algorithm has been compared with C4.5, but when classes are continuous, decision tree learning algorithms such as C4.5 cannot work. Therefore, extracting propositions from trained neural networks is an important technique. The algorithm has been implemented as a module NNE(Neural Networks with Explanation) in the data mining system KINO (Knowledge INference by Observation).

References

[1] R. Andrews, J. Diederich and A.B. Tickle: Survey and critique of techniques for extracting rules from trained artificial neural networks, *Knowledge-Based Systems,* Vol.8, No. 6, pp.373-189, 1995.

[2] M.W. Craven and J.W. Shavlik: Learning symbolic rules using artificial neural networks, *Proceedings of the Tenth International Machine Learning Conference,* pp.73-80,1993.

[3] M. Ishikawa: Structural Learning with Forgetting, *NEURAL NETWORKS,* Vol.9, No.3, pp.509-521, 1996.

[4] N. Linial, Y. Mansour and N. Nisan: Constant Depth Circuits, Fourier Transform, and Learnability, *Journal of the ACM,* Vol.40, No.3, pp.607-620, 1993.

[5] J.R. Qumlan: *C4-5: Programs for machine learning,* Morgan Kaufmann Pub., 1993.

[6] J.W. Shavlik: Combining Symbolic and Neural Learning, *Machine Learning,* 14, pp.321-331, 1994.

[7] H. Tsukimoto and C. Morita: The discovery of propositions in noisy data, *Machine Intelligence 13,* pp. 143-167, Oxford University Press, 1994 .

[8] H. Tsukimoto: The discovery of logical propositions in numerical data, *AAAI'94 Workshop on Knowledge Discovery in Databases,* pp.205-216, 1994.

[9] H. Tsukimoto: On continuously valued logical functions satisfying all axioms of classical logic, *Systems and Computers in* Japan,Vol.25, No. 12, pp.33-41, SCRIPTA TECHNICA, INC., 1994.

[10] H. Tsukimoto and C. Morita: Efficient algorithms for inductive learning-An application of multi-linear functions to inductive learning, *Machine Intelligence 14,* pp.427-449, Oxford University Press, 1995.