International Journal of
Intelligent Systems and Applications in Engineering

ISSN:2147-6799 www.ijisae.org Original Research Paper

Only One Neuron either N-bit Parity Rule Based Modified Translated Multiplicative or McCulloch-Pitts Models for Some Machine Learning Problems

Ali Özdemir [1], Melih İnal *[2]

Accepted 3rd September 2016

Abstract: In this study, solutions to machine learning problems such as Monk's 2 (M_2), Balloon and Tic-Tac-Toe problems employing a single neuron dependent on rules which use either modified translated multiplicative (π_m) neuron or McCulloch-Pitts neuron model is proposed. Since M_2 problem is similar to N-bit parity problem, translated multiplicative (π_t) neuron model is modified for M_2 problem. Also, McCulloch-Pitts neuron model is used to increase classification performance. Then either πm or McCulloch-Pitts neuron model is applied to Balloon and Tic-Tac-Toe problems. When the result of proposed only one π_m neuron model that is not required any training stage and hidden layer is compared with the other approaches, it shows satisfactory performance.

Keywords: *Machine learning; Modified translated multiplicative neuron model; Monk's and Balloon problems; N-bit parity problem; Translated multiplicative neuron model.*

1. Introduction

Translated multiplicative neuron (πt- neuron) is primarily used to the N-bit parity problem. N-bit parity problem is an approach to test neural network architectures and learning algorithms. The N-bit parity problem is considered as a very hard problem to be solved by neural networks, because a single 'flip' of a bit in the input string requires a complementary classification. The N-bit parity problem is a generalization of the 'eXclusive-OR' (XOR) problem. N-bit parity problem can be explained as follows. Let x = $[x_1,\ldots,x_N]^T$ is N-bit binary vector and $x_i \in \{0,1\}$ (i = 1, …, N). The parity generator function which is stated as shown in Eq.1, can be determined the parity as follows:

$$p(x)=\begin{cases}0, \text{ if } \sum_{i=1}^{N} x_i \text{ is even}\\ 1, \text{otherwise.}\end{cases} \quad (1)$$

There are many neural network architectures applied in N-bit parity problem [1-8]. (Kim et al. 2005) proposed a method of improving the learning time and convergence rate to exploit the advantages of ANN and fuzzy theory to neuron structure. Their method is applied to the XOR and N-bit parity problems. But, (Iyoda et al. 2003) make a comparison between neural architectures for the N-bit parity problem. The comparison result shows that πt neuron model is not required any hidden neurons and learning algorithm. πt neuron model is called translated multiplicative neuron model. It uses threshold activation function.

[1] *Kocaeli University, Graduate School of Natural and Applied Sciences, Electronics and Computer Department, 41380 Kocaeli, Turkey*
[2] *Kocaeli University, Informatics Department, Umuttepe Campus, 41380 Kocaeli, Turkey*
* *Corresponding Author: Email: minal@kocaeli.edu.tr*
Note: This paper has been presented at the 3rd International Conference on Advanced Technology & Sciences (ICAT'16) held in Konya (Turkey), September 01-03, 2016.

Since π_t neuron model stems from multiplicative neuron model, then several multiplicative neurons which have been proposed [9, 10, 11] can be examined to comprehend π_t neuron model. The model is defined as follows:

$$v = b\prod_{i=1}^{N}(x_i - t_i) \quad y=f_{th}(v) \quad (2)$$

where, $b \in R$ and $t_i \in R$ (i = 1,…,N) which are the neuron's adjustable parameters, are bias and weights, respectively. The neuron's output is defined as y.

The threshold activation function fth: R → {0, 1} is defined as follows:

$$f_{th} = \begin{cases}1, & v \geq 0\\ 0, & v < 0\end{cases} \quad (3)$$

In fact, πt- neuron model which is shown in Figure 1 is inspired from McCulloch-Pitts. McCulloch-Pitts neuron model is given by the following equation:

$$v_m = w_0 + \sum_{i=1}^{N} x_i w_i . \quad y=f_{th}(v_m) \quad (4)$$

where, w0 is bias and wi are the weights.

Comparing Eq. 4 with Eq. 2, w0 is equivalent to b and wi are equivalent to ti parameters. The parameters of multiplicative πt neuron, which uses threshold activation function, are defined as:
$0 < t^i < 1$ (i = 1,…,N);
If N is even then b < 0, If N is odd then b > 0.

If the same activation function is used in the Eq.2 and Eq.4, the mathematical procedure's complexity is equivalent of these two models. Table 1 and Table 2 show the solutions of 2-bit XOR parity problem and 10x10-bit parity problem's. b and t_i ([t_1, ... ,t_N]) are selected as constants: -24 and 0.8, respectively. (Iyoda et al. 2003) proved that the translated multiplicative π_t neuron model can solve the N-bit parity problem for $\forall$ N ≥ 1.

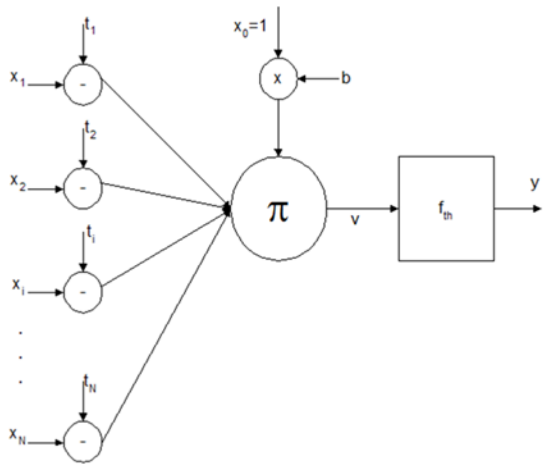

Figure 1. Translated multiplicative neuron model

An N-bit Parity problem can easily be solved by only one π_t neuron using threshold activation function and also parameters defined in certain intervals. This approach has the lowest process complexity, which is presented between neural network solutions so far [1]. Therefore, modified translated multiplicative (π_m) neuron or McCulloch-Pitts neuron model, is proposed for solution of Monk's M_2 problem which has a nonlinear relationship similar to XOR problem. In Eq. 2 and Eq. 4, all biases and weights are chosen as constants with their optimum values. Since, they are chosen as constants; there is no need any learning stage for the both networks. The main contribution of the study that it presents modified translated multiplicative neuron model to solve Monk´s M_2 problem by expressing it as an N-bit parity problem.

Table 1. Solution of 2–bit XOR Parity Problem

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0

In the following section, Monk's problem is presented. Previous studies in Monk's problem are given in Section 3. The π_m neuron model and results obtained from the application of either one π_m neuron model or one McCulloch-Pitts neuron model to Monk's M_2 problem are given in Section 4. The other results of the machine learning problems are given are given in Section 5-7.

Table 2. Solution of 10x10-bit Parity Problem

x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	y
0	0	0	0	0	1	1	1	0	0	1
1	0	0	0	1	0	1	1	0	1	1
1	0	1	0	0	1	0	1	0	1	1
0	1	1	0	0	0	0	0	1	1	0
1	0	1	1	1	0	0	1	1	0	0
0	0	1	1	1	0	0	1	1	1	0
1	1	1	0	0	1	0	0	1	0	1
0	0	0	0	1	0	0	0	1	1	1
1	1	0	1	1	0	1	1	0	0	0
1	1	1	1	0	0	0	1	1	0	0

2. Description of Monk's Problem

An artificial robot named as Monk's problem has six attributes that is defined by [12]:

x1: head_shape ∈ round, square, octagon
x2: body_shape ∈ round, square, octagon
x3: is_smiling ∈ yes, no
x4: holding ∈ sword, balloon, flag
x5: jacket_color ∈ red, yellow, green, blue
x6: has_tie ∈ yes, no

The Monk's Problem is classified as M_1, M_2 and M_3 problems according to above attributes. These three problems are defined by:

Problem M_1: If head shape and body shape have same value or jacket color is red, this problem belongs to M_1 (head shape = body shape or jacket color = red).
Problem M_2: If exactly two of six attributes of robot have their first value, this problem belongs to M_2.
Problem M_3: If jacket_color is green and holding is sword or jacket_color isn't blue and body_shape isn't octagon, it belongs to M3 ((jacket_color = green and holding = sword) or (jacket_color ≠ blue and body_shape ≠ octagon)).

Data which are taken from 6 different places of the artificial robot must be rearranged to simulate to the parity rule. The arrangement is done by the following statements:

x1: head_shape ∈ 1,2,3 (0000, 0001, 0010)
x2: body_shape ∈ 1,2,3 (0000, 0001, 0010)
x3: is_smiling ∈ 1,2 (0000, 0001)
x4: holding ∈ 1,2,3 (0000, 0001, 0010)
x5: jacket_color ∈ 1,2,3,4 (0000, 0001, 0010, 0011)
x6: has_tie ∈ 1,2 (0000, 0001)

As it is seen above, decimal values are given to attributes for each point of the robot. These decimal values are converted to 4 bit binary number system. To adjust the data according to N-bit parity rule, each decimal value is decreased for one, and then the decreased decimal number is converted to its binary value.

3. Other Studies Related With Monk's Problem

(Thurn et al., 1991) summarize the comparison of different learning techniques in a report which was performed at the 2nd European Summer School on Machine Learning, held in Belgium. A variety of symbolic and non-symbolic learning techniques are compared on Monk's problems. In Table 3, only the comparison results of Monk's M_2 problem with this study are given. One significant characteristic of this comparison is that it was performed by a collection of researchers, each of whom was an advocate of the technique they tested. Here some algorithms that have recognition rate more than this study are explained in this section. Since Thrun et al. give brief description about their studies [12], special properties of the algorithms e.g. rules that are used in their algorithms or other things which are based on neural networks will reemphasize in the following:

AQ17 Algorithms. AQ17-DCI algorithm is based on AQ learning programs. Here is a brief description of the AQ algorithm:

1. Select a seed example from the set of training examples for a given decision class.
2. Using the extend against operator, generate a set of alternative most general rules (a star) that cover the seed example, but do not cover any negative examples of the class.
3. Select the “best" rule from the star according to a multi-criteria rule quality function (called LEF – the Lexicographical Evaluation Function), and remove the examples covered by this rule from the set of positive examples yet to be covered.
4. If this set is not empty, select a new seed from it and go to step 2. Otherwise, if another decision class still requires rules to be learned, return to step 1, and perform it for the other decision class.

Table 3. The performance sorting for Monk’s M2 problem of different methods

Performance Sequence	Method and Reference	System Performance (%)
1	AQ17-DCI / Bala et al.	100.00
2	Backpropagation / Thrun	100.00
3	Backpropagation with weight decay / Thrun	100.00
4	Cascade Correlation / Fahlman	100.00
5	**π_m & McC.-P. neuron models / our study**	**96.45**
6	AQ17-HCI / Bala et al.	93.10
7	AQ17-FCLS / Bala et al.	92.60
8	AQ15-GA / Bala et al.	86.80
9	Assistant Professional /Cestnik et al.	81.30
10	AQR / Kreuziger et al.	79.70
11	Prism / Keller	72.70
12	Ecobweb l.p. & information utility / Van de Welde	71.30
13	Mfoil / Dzeroski	69.20
14	ID5R / Kreuziger et al.	69.20
15	ID3, no windowing / Kreuziger et al.	69.10
16	CN2 / Kreuziger et al.	69.00
17	ID3 / Kreuziger et al.	67.90
18	Ecobweb leaf prediction / Reich et al.	67.40
19	TDIDT / Van de Welde	66.70
20	IDL / Van de Welde	66.20
21	ID5R-hat / Van de Welde	65.70
22	Classweb 0.10 / Kreuziger et al.	64.80
23	ID5R / Van de Welde	61.80
24	Classweb 0.15 / Kreuziger et al.	61.60
25	Classweb 0.20 / Kreuziger et al.	57.20

AQ17-DCI uses 2 rules for Class 0 and 1 rule for Class1.

Backpropagation and Backpropagation with weight decay. There were 17 input units, all having either value 0 or 1 corresponding to which attribute-value was set. All input units had a connection to 2 hidden units, which itself were fully connected to the output unit. An input was classified as class member if the output, which is naturally restricted to (0; 1), was ≥ 0.5. Training took between ten and thirty seconds on a SUN Sparc Station. On a parallel computer, namely the Connection Machine CM-2, training time was further reduced to less than 5 seconds for each problem. The following results are obtained by the plain, unmodified backpropagation algorithm. After 90 training epochs, the system performance was reached to 100% accuracy. Weight decay widely used technique often prevents backpropagation nets from overfitting the training data and thus improves the generalization. With weight decay α=0.01 Thrun improved the classification accuracy on this third set for M_3 problem significantly and, moreover, the concept learned was the same for all architectures he tested (i,e, 2, 3, or 4 hidden units).

The Cascade Correlation Algorithm. Cascade Correlation is a supervised neural network learning architecture that builds a near-minimal multi-layer network topology in the course of training. Initially the network contains only inputs, output units, and the connections between them. This single layer of connections is trained (using the Quickprop algorithm) to minimize the error. When no further improvement is seen in the level of error, the network's performance is evaluated. If the error is small enough, training stage stops. Otherwise a new hidden unit is added to the network in an attempt to reduce the residual error. The result of the Cascade Correlation algorithm for M2 problem: After 82 epochs, 1 hidden unit: 0 Errors on training set and 0 Errors on test set. Elapsed real time: 7.75 seconds.

4. Modified Translated Multiplicative (π_m) Neuron Model

Only M_2 problem is similar to parity problem among these three Monk’s problems. So, π_m neuron model that is formed by the algorithm of π_t is applied to M_2 problem. When π_m neuron model is used stand alone, no good classification performance is obtained. Therefore, π_m or McCulloch-Pitts neuron models alternatively are used according to the rules.

Data matrix that has size of 169x7 is obtained from ftp server of University of California, Irvine [13]. According to robot’s attributes, 64 of the data produced the output 1 while the rest produced output 0. The first experiment is done for examining the π_t neuron model using 169 data matrix. The b and t_i parameters of π_t neuron model are chosen -1 and 0.5, respectively (b=-1, t_1,...,t_N = 0.5, where N=24). Since the robot has 6 attributes and each of them is represented by 4-bit binary number, the model has 24 inputs. For the first classification, 105 of the data have been correctly classified with 62.130% success. While 115 out of 169 data are already 0, the 62.130% system performance is not satisfying for the classification given above. If all the outputs of the model are assumed to be 0, anyway 68.047% performance is obtained.

The input data are examined to get a better solution than above. When any of x1, x2, x4, and x5 has the value “3” in decimal number system, it is observed that π_t neuron model is not good in classifying according to N-bit parity rule. So, some changes in algorithm are made by adding rules to multiplicative π_t neuron model. This neuron model is named as modified translated multiplicative (π_m) neuron model. Here, b parameter in π_m neuron model is chosen different from π_t that is used for N-bit parity problem.

The following rules and threshold activation function given in Eq.3 are used for both πm and McCulloch-Pitts neuron models:

Rule 1: IF (x1=3 or x2=3 or x4=3 or x5=3) THEN b=2 use Eq.2 ELSE b=-2 use Eq.4

If only Rule 1 is used, the 125 of 169 data are correctly classified. The system performance is 73.964%.

Rule 2: IF x5=4 THEN b=-2 use Eq.2

If Rule 1 and Rule 2 are used together, 143 of 169 data are correctly classified. The system performance is 84.615%.

Rule 3: IF x5=4 and ((x1 = x2 = x3 = 1 and x4 ≠ 1) or (x1=3 and x2 ≠1 and x3 ≠1 and x4 ≠1 and x6 ≠ 1) or (x1=2, 3 and x2 =2, 3 and x3 =2, 3 and x4 =2, 3 and x6=2)) THEN b=2 use Eq.2 ELSE b=-2 use Eq.2

If Rule 1, Rule 2 and Rule 3 are used together, 149 of 169 data are correctly classified. The system performance is 88.166%.

Rule 4: IF (x1=3 or x2=3 or x4=3 or x5=3) and ((x1= x2= x3 = x6 =1) or (x1=3 and x2 = 2,3 and x3 = 2,3 and x4 = 2,3 and x5 = 2,3 and x6=2) or (x1 = 2,3 and x2 ≠ 2 and x3 ≠ 2 and x4 = x5 = x6 =1) or (x1 = 2 and x2 = 3 and x3 = 2 and x4 ≠1 and x6 ≠ 1) or (x1 ≠ 3 and x2 ≠ 2 and x3 = 1 and x4 = 1 and x6 ≠ 2) or (x1 = 3 and x2 = x5 = x6 = 1)) THEN b=-2 use Eq.2 ELSE b=2 use Eq.2

If Rule 1, Rule 2, Rule 3 and Rule 4 are used together, 154 of 169 data are correctly classified. The system performance is 91.124%.

Rule 5: IF (x1≠3 or x2≠3 or x4 ≠3 or x5≠3) THEN b=-2 use Eq.4

If Rule 1, Rule 2, Rule 3, Rule 4 and Rule 5 are used together, 163 of 169 data are correctly classified. The system performance is 96.45%. In the all rules, weights of the neuron models are chosen as 0.5.

A study is carried out to examine the performance of π_m neuron model parameters b and ti as shown in Table 4. To get the best performance, the parameters b and ti are to be chosen in M_2 problem as follows:

Table 4. Performances due to different b and ti values

b	t_i	Perfor mance (%)	b	t_i	Perfor mance (%)
	0.1	94.675		0.1	94.675
	0.2	94.083		0.2	94.675
	0.3	**96.450**		0.3	94.675
	0.4	93.491		0.4	94.675
± 1	0.5	89.941	± 4	0.5	94.675
	0.6	89.941		0.6	94.675
	0.7	89.941		0.7	94.083
	0.8	89.941		0.8	94.083
	0.9	89.941		0.9	93.491
	0.1	94.675		0.1	94.675
	0.2	94.675		0.2	94.675
	0.3	94.675		0.3	94.675
	0.4	94.083		0.4	94.675
± 2	**0.5**	**96.450**	± 5	0.5	94.675
	0.6	**96.450**		0.6	94.675
	0.7	93.491		0.7	94.675
	0.8	93.491		0.8	94.675
	0.9	93.491		0.9	94.083
	0.1	94.675		0.1	94.675
	0.2	94.675		0.2	94.675
	0.3	94.675		0.3	94.675
	0.4	94.675		0.4	94.675
± 3	0.5	94.083	± 6	0.5	94.675
	0.6	93.491		0.6	94.675
	0.7	93.491		0.7	94.675
	0.8	**96.450**		0.8	94.675
	0.9	**96.450**		0.9	94.675

- b: ± 1 and ti: 0.3
- b: ± 2 and ti: [0.5-0.6]
- b: ± 3 and ti: [0.8-0.9]

In addition to performance sequence of previous studies on Monk's problem, π_m and McCulloch-Pitts neuron models proposed in this study are given in Table 3. The results obtained in this paper have higher performance when compared to the some of the studies given in Table 3. Studies supplying 100% performance for M_2 problem are already well known. This paper proposes a new approach which is called π_m neuron model. Moreover, 6 individual rules can be defined for the remaining 6 data, which are not correctly classified to make system performance 100%.

5. Results of the Proposed Model for Balloon Problem

The application of either π_m or McCulloch-Pitts neuron model to Balloon problem and results are presented in this section. The data sets of Balloon problems are given in Table 5.

Table 5. The Data Sets of Balloon Problems

DATA A					DATA B					DATA C					DATA D				
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	0	1	1	1	1	0	0	1	1	1	0	1	1	1	1	0	1
1	1	0	1	1	1	1	0	1	0	1	1	0	1	1	1	1	0	1	1
1	1	0	0	0	1	1	0	0	0	1	1	0	0	1	1	1	0	0	1
1	0	1	1	1	1	0	1	1	1	1	0	1	1	0	1	0	1	1	1
1	0	1	0	1	1	0	1	0	0	1	0	1	0	0	1	0	1	0	0
1	0	0	1	1	1	0	0	1	0	1	0	0	1	0	1	0	0	1	0
1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0
0	1	1	1	1	0	1	1	1	1	0	1	1	1	0	0	1	1	1	1
0	1	1	0	1	0	1	1	0	0	0	1	1	0	0	0	1	1	0	0
0	1	0	1	1	0	1	0	1	0	0	1	0	1	0	0	1	0	1	0
0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0
0	0	1	1	1	0	0	1	1	1	0	0	1	1	0	0	0	1	1	1
0	0	1	0	1	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0
0	0	0	1	1	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

When the following individual rules are defined for data sets of Balloon, the classification performance of the proposed model is 100% for each data set. As indicated in the following rules, there is no need to implement McCulloch-Pitts neuron model except for Data Set D.

Rule for Data Set A: IF x3=0 or x4=0 THEN b=-2 use Eq.2 ELSE b=2 use Eq.2

Rule for Data Set B: IF x3=0 and x4=0 THEN b=-2 use Eq.2 ELSE b=2 use Eq.2.

Rule for Data Set C: IF x1=0 and x2=0 THEN b=-2 use Eq.2 ELSE b=2 use Eq.2.

Rules for Data Set D:

Rule 1: IF ((x1=1 and x2=1) or (x3=1 and x4=1)) and ((x1=1 and x2=0) or (x1=0 and x2=1) or (x3=1 and x4=0 or x3=0 and x4=1)) THEN use b=-2 Eq.2 ELSE b=2 use Eq.2

Rule 2: IF (x1≠1 and x2≠1) or (x3≠1 and x4≠1) THEN use b=-2 Eq.4.

A study, which is implemented for Data Set D, is carried out to examine the performance of π_m neuron model parameters b and t_i. To get the 100% performance, the parameters b and t_i are to be chosen for Data Set D as follows:

b: ± 1 and ti: [0.3-0.4] or b: ± 2 and ti: [0.1-0.9]

When we compare the result which are performed for Balloon problems, of (Solorio et. al., 2002) introduce an algorithm called Ordered Classification (OC) with our proposed model, while OC has 0.27938, ours has 0.0 classification error.

6. Application of π_m to Tic-Tac-Toe Problem

Tic-Tac-Toe is formed data which is taken from a game. The game, made from nine squares, is defined by (Pilgrim, 1995). Every one of these squares takes symbol of 'x', 'o', 'b'. 'x' and 'o' show the first and second player, respectively. The symbol 'b' shows the space squares in the game.

x1	x2	x3
x4	x5	x6
x7	x8	x9

Figure 2. Tic-Tac-Toe game

Each player individually marks their symbol in any square for writing respectively own letter in any square in Figure 2.

If any player signs with his/her letter with 3 successive places, the player wins. These successive places can be in column, row or diagonal. There are 958 data in this database [14, 15]. But the winner is determined according to first player in this data. Winner is described with positive, loser is described with negative. 626 of 958 databases are positive, it means that 'x' is won, 332 are negative and it means that 'x' is lost.

'x' value is selected 1; o and b are selected 0 so that this data translated to binary. In this way, it simulated to N-bit parity problem. If first player wins, result of related data groups is 1, otherwise 0: (x_{10}: positive, negative (1, 0)).

This data matrix of tic-tac-toe is applied to π_t neuron model. The 658 of 958 data is correct classified and the system performance is 63.466% (b<0, t_i=0.5). Then, πm neuron model is used for this database and the above rule is written. The 942 of 958 data is correct classified, the system performance is 98.330%.

Rule: If space number=2 or 3 Then b <0 use Eq. 2 Else b>0 use Eq. 2 (b=±2, t_i=0.5).

Performance's row of the other algorithm which solve this problem and algorithm used in this study is shown in Table 6.

The best algorithm is Newboole and second is IB3-CI (Instance Based 3-Constructive Induction). In 1991, Pierre Boneli and Alexander Parodi originated Stewart W. Wilson's Boole classification system and developed Newboole. This classification system is based on genetic and it uses supervised learning as learning algorithm [14, 15]. In 1991, Aha developed ib3-ci (1991) algorithm. It is another construction algorithm that generates Boolean features based on the conjunction operator. Tic-tac-toe is a simple game often used as a programming assignment for computer-science students or as an in-class example of how to develop software [17-20]. Every tic-tac-toe program should include a way of representing the board and evaluating the board for a win. Often, this evaluation is done by checking all eight possibilities (on the traditional 3x3 tic-tac-toe board).

7. Conclusion

Modified translated multiplicative neuron which is inspired by the architecture of translated multiplicative neuron that is an effective ANN model in solving N-bit parity problem and McCulloch-Pitts neuron models are applied to Monk's M_2 and Balloon problems. The 100% classification performances of studies given in Table 3 utilize hidden layer in ANN architecture such as the Backpropagation and Cascade Correlation models. The proposed model consists of only one neuron. While AQ17-DCI algorithm uses 3 rules for obtaining 100% system performance, the proposed model uses 5 rules with 96.45% performance. Six additional rules can be individually defined for the remaining 6 data, which are not correctly classified for accomplishing 100% performance. Also the comparison with OC algorithm shows that the proposed neuron model π_m can be an alternative model. Once the weights and bias and also proper rule(s) are optimally selected, the proposed model can be classified any desired data without learning stage. The evaluations of πm model give satisfy result for the three machine learning datasets such as Monk's M_2, Balloon and Tic-tac-toe.

Table 6. Order of performance of different algorithms for Tic-tac-toe problem

Order	Algortihms	Performance(%)
1	NEWBOOLE	100.00
2	IB3-CI	99.10
3	**π_m Model**	**98.33**
4	IB1	98.10
5	CN2	98.10
6	IB3	82.00
7	MBRtalk	88.40
8	NewID	88.00

References

[1] Iyoda, E. M., Nobuhara, H. and Hirota, K.: A Solution for the N-bit Parity Problem Using a Single Translated Multiplicative Neuron, Neural Processing Letters, vol.18, pp. 213-218, 2003.

[2] Arslanov, M.Z., Ashigaliev, D.U. and Ismail, E. E.: N-bit Parity Ordered Neural Network, Neurocomputing 48 (2002), 1053-1056

[3] Al-Rawi, M.: A Neural Network to Solve the Hybrid N-parity: Learning with Generalization Issues, Neurocomputing, vol.68, pp. 273-280, 2005

[4] Hohil, M. E., Liu, D., Smith, S. H., Solving the N-bit parity problem using neural networks, Neural Networks, vol.12, pp.1321-1323, 1999.

[5] Li, D., Hirasawa, K., Hu, J., Murata, J., Studying the effects on multiplication neurons for parity problem, 41st Society of Instrument and Control Engineers-SICE,2002.

[6] Kim, K., Kim, S., Joo, Y., Oh, A.S.: Enhanced fuzzy single layer perceptron, Advances in Neural Networks, vol.3496,pp. 603-608, 2005.

[7] Setino, R.: On the solution of the parity problem by a single hidden layer feedforward neural network, Neurocomputing, vol.16 (3), pp. 225-235, 1997.

[8] Setiono, R., Hui, L. C. K.: Some N-bit parity problems are

solvable by feed-forward networks with less than n hidden units, Int. Joint Conf. on Neural Networks, 1993, pp. 305-308.

[9] Schmitt, M.: On the complexity of computing and learning with multiplicative neurons, Neural Computation, vol.14(2), pp. 241-301, 2002.

[10] Zhang, B.-T.: A Bayesian Evolutionary Approach to The Design and Learning of Heterogeneous Neural Trees, Integrated Computer-Aided Engineering, vol. 9(1), pp. 73-86, 2002.

[11] Bas, E., Uslu, V. R. and Egrioglu, E.: Robust learning algorithm for multiplicative neuron model artificial neural networks, Expert Systems with Applications, vol.56, pp. 80-88, 2016.

[12] Thrun, S.B., Bala, J., Bloedorn, E., Bratko, I., Cestnik, B., Cheng, J., De Jong, K., Dzeroski, S., Fahlman, S.E., Fisher, D., Hamann, R., Kaufman, K., Keller, S., Kononenko, I., Kreuziger, J., Michalski, R.S., Mitchell, T., Pachowicz, P., Reich, Y., Vafaie, H., Van de Welde, W., Wenzel, W., Wnek, J., and Zhang, J.: The Monk's Problems: A Perfor. Comparison of Different Learning Algorithm, a Report, Carnegie Mellon University CMU-CS-91-197, 1991.

[13] University of California, Irvine Dataset: [Online]. Available: ftp://ftp.ics.uci.edu/pub/machine-learning-databases/monks-problems/, retrieved November 16 2016.

[14] Pilgrim, R., A., "Tic-Tac-Toe: Introduction Expert Systems to Middle School Students", Acm Sigcse Bulletin, Vol. 27, 340–344, (1995).

[15] Gordon, A., "A General Algorithm for Tic-Tac-Toe Board Evaluation", Journal of Computing Sciences in Colleges, Vol. 21, 42-46, (2006).

[16] Solorio, T. and Fuentes, O.: Taking Advantage of Unlabelled Data with the Ordered Classification Algorithm, ACTA, Proc. of AI and Soft Computing ASC 2002, 357-200, (2002).

[17] Noughts And Crosses - The oldest graphical computer game, http://www.pong-story.com/1952.htm, retrieved November 16, 2016.

[18] Wachsmuth, B. G., Tic-tac-toe game, http://pirate.shu.edu/~wachsmut/Teaching/CSAS1111/Assigns-CPP/assign7.html, retrieved November 17, 2016.

[19] Massey, B., Tic-tac-toe board evaluation, http://web.cecs.pdx.edu/~bart/cs541-fall2001/homework/3-learn.html, retrieved November 17, 2016.

[20] Appel, A. W., Game player programs, http://www.cs.princeton.edu/courses/archive/spr05/cos217/asgts/gameplayer/, retrieved November 16, 2016.