Pertanika J. Sci. & Technol. 29 (1): 285 - 300 (2021)

PERTANIKA JOURNALS

SCIENCE & TECHNOLOGY

Journal homepage: http://www.pertanika.upm.edu.my/

Logic Learning in Adaline Neural Network

Nadia Athirah Norani[1], Mohd Shareduwan Mohd Kasihmuddin[2]*, Mohd. Asyraf Mansor[3] and Noor Saifurina Nana Khurizan[4]

[1,2,4]School of Mathematical Sciences, Universiti Sains Malaysia, 11800, Penang, Malaysia
[3]School of Distance Education, Universiti Sains Malaysia, 11800, Penang, Malaysia

ABSTRACT

In this paper, Adaline Neural Network (ADNN) has been explored to simulate the actual signal processing between input and output. One of the drawback of the conventional ADNN is the use of the non-systematic rule that defines the learning of the network. This research incorporates logic programming that consists of various prominent logical representation. These logical rules will be a symbolic rule that defines the learning mechanism of ADNN. All the mentioned logical rule are tested with different learning rate that leads to minimization of the Mean Square Error (MSE). This paper uncovered the best logical rule that could be governed in ADNN with the lowest MSE value. The thorough comparison of the performance of the ADNN was discussed based on the performance MSE. The outcome obtained from this paper will be beneficial in various field of knowledge that requires immense data processing effort such as in engineering, healthcare, marketing, and business.

Keywords: Adaline neural network, logic programming, logical rule

ARTICLE INFO

Article history:
Received: 4 August 2020
Accepted: 28 September 2020
Published: 22 January 2021

DOI: https://doi.org/10.47836/pjst.29.1.16

E-mail addresses:
nadiaathirahnorani@gmail.com (Nadia Athirah Norani)
shareduwan@usm.my (Mohd Shareduwan Mohd Kasihmuddin)
asyrafman@usm.my (Mohd. Asyraf Mansor)
saifurina@usm.my (Noor Saifurina Nana Khurizan)
*Corresponding author

INTRODUCTION

Artificial Neural Network (ANN) was inspired by the biological neuron model. ANN consists of interconnected neurons with distinctive input and output layer. Despite the various development of the ANN in many field of studies, ANN experience lack of representation during learning and retrieval phase. One of the unique direction in the work of ANN is the

ISSN: 0128-7680
e-ISSN: 2231-8526

use of the optimal learning system via symbolic rule. Initially, Abdullah (1992) proposed logic programming in special case of ANN called Hopfield Neural Network (HNN). The proposed network introduced an innovative connection weight by comparing cost function and energy function. This pursuit was continued by Sathasivam (2010) where this work capitalized special case of logic called Horn Satisfiability. Since then, several propositional logical rules were proposed in ANN such as 2 Satisfiability (Kasihmuddin et al., 2017), 3 Satisfiability (Mansor et al., 2017) and Random 2 Satisfiability (Sathasivam et al. 2020a). The proposed methods played an important role in the emergent of several related applications such as Very Large Scale Integration (Sathasivam et al., 2020b), Bezier reconstruction (Kasihmuddin et al., 2016), logic mining (Kho et al., 2020) and many more. In another development, Alzaeemi et al. (2020) proposed 2 Satisfiability logical rule in Radial Basis Function Neural Network (RBFNN) by utilizing the value of the parameters in the hidden layers. The comparative study between HNN and RBFNN has been reported in (Mansor et al. 2020). All the mentioned logical rule utilizes only the Satisfiable logic except in the work of Kasihmuddin et al. (2018) that proposed non-Satisfiable logic. As far as the logical rule is concern, the proposed ANN only implemented limited type of propositional logical rule. This is due to the redundant nature of the logic that prevent the previous studies to produce optimal synaptic weight that corresponds to the goal of the logic.

Popularized by Widrow and Hoff (1960), Adaline Neural Network (ADNN) is designed to cater a simple data processing unit that serves as an intermediate layer between pre-processing input and input (Sharma et al., 2019). ADNN is a single layer network with multiple nodes where each node receives multiple inputs and produces one output. ADNN is a self-adaptive algorithm that can automatically modify the current structure in order to optimize performance based on previous input (Widrow & Hoff, 1960). This network adapts to the input data, learn from the data, and produce the final output based on the minimized synaptic weight. This network is based on the Delta rule that utilizes least mean squares learning error to minimize the error during the training phase (Widrow & Lehr, 1990). Due to the simplicity of the ADNN, this network endured rapid development in term of architecture and data processing. Pajares and Jesús (2001) proposed ADNN in optimizing the stereovision matching problem. The proposed ADNN managed to increase a good decision by learning the optimal weight design when considering relative importance of the attribute. In another development, Negarestani et al. (2003) proposed ADNN to estimate the environmental parameters of the radon concentration in the soil. The proposed ADNN has a good agreement with the similar established work. Recently, Sujith and Padma (2020) utilized ADNN to estimate the harmonics for Pulse Width Modulation. The proposed ADNN optimized the crucial parameter of the system such as load voltage, current and reactive power. In some areas, ADNN became a useful method in adaptive signal processing due to the simplicity of the modelling method (Widrow &

Stearns, 1985; Wang et al., 2000; Kavak et al., 2005). Unfortunately, the above studies only utilize ADNN as a tool rather than self-adaptive ANN by using pre-defined symbolic rule. In this case, very limited effort to establish another variant of logical rule embodied into ADNN. In this paper, we implemented logical programming in ADNN by embedding several established logical rule. Hence, this combination provides a good understanding of the ADNN governed by logical structure.

The remaining part of this paper is as follows. In section 1, we explore the logical representation that consists of newly proposed logical rule. The proposed logical rule consists of both redundant and non-redundant variables. In section 2, the general structure of ADNN is explained in detail. Next, the implementation of various logical rule in ADNN is demonstrated. Finally, the performance of different logical rule in ADNN is discussed thoroughly and we conclude the paper with some remarks and future work.

MATERIALS AND METHOD

Logical Representation

Logic programming transformed the knowledge representation into mathematical derivation. Logic programming is declarative because this representation requires the user to state the desired task while putting minimal emphasis on the underlying process (Somogyi et al., 1996). In another perspective, logic programming represent fact both explicitly and implicitly depending on the nature of the desired problem. According to Kowalski (1978), logic programming is expressed symbolically according the following Equation 1:

$$Algorithm = Logic + Control. \quad [1]$$

In Equation 1, logic signifies the issue of "what" and control explains the "how" which can be determined using the method defined by the user. In this case, optimal logic programming has to be able to clearly define the end goal that corresponds to the logical statement and control the system accordingly (Hamadneh, 2013). In practice, logical representation can "communicate" with the user by assigning the logical state for each variable in the logic. Hence, listing all the possible logical instance optimize the "control" part by identifying the logical inconsistencies that deviates from the goal.

In this paper, we only consider propositional logical rule where the output consists of only bipolar (1 or -1). In this case, the output reads 1 (True) or -1 (False). The goal of each logic formulation is to produce only 1 (True) statement. One of the most prominent representations of Propositional logic is in Boolean algebra form. Boolean algebra is a logical algebra in which symbols are used to describe the complexity of the logical representation (Riche, 2011). This representation consists of individual unit called variable and connectives (Table 1). Similar to the output of the logical rule, variable is a symbolic

unit that can be defined as 1 (True) and -1 (False). The relationship of the variable is define by connectives manipulate the behaviour of the problem that leads to the final output (goal).

Table 1

List of logic programming connections

Connective	Representation
T	True
$\wedge$	False
$\wedge$	Conjunction
$\neg$	Disjunction
$\neg$	Negation
$\leftarrow$	Implication

To further illustrate the connection in Table 1, we first describe two logical variables A and B that serve as input and P as an output P where $B \in \{-1,1\}$, $B \in \{-1,1\}$ and $P \in \{-1,1\}$. The variants of the logical rule that relates A and B can be shown in Table 2. Note that, different logical formulations require different logical states which lead to $P = 1$.

Table 2

List of logical variants

Logical Statement	Conventional Formulation	Propositional Formulation
A AND B	$P = A.B$	$P = (A \wedge B)$
A OR B	$P = A + B$	$P = (A \vee B)$
NOT A	$P = \overline{A}$	$P = \overline{A}$
A NAND B	$P = \overline{A.B}$	$P = \neg(A \wedge B)$
A NOR B	$P = \overline{A + B}$	$P = \neg(A \vee B)$
A EX-OR B	$P = A \oplus B$	$P = (A \wedge \neg B) \vee (\neg A \wedge B)$
A EX-NOR B	$P = \overline{A \oplus B}$	$P = (A \wedge B) \vee \neg(A \wedge B)$

Since each variable in *P* is binary in nature, the logical structure can be implemented in ANN. In this paper, we will utilize logical structure in Table 2 as a learning rule that governs the ADNN model.

Adaline Neural Network (ADNN)

ADNN consists of single interconnected layer neurons with *n* inputs and one output. The determination of the output is based on the linear combinations of the inputs (Jannati et al., 2016). The main feature of ADNN is the ability to be self-adapting algorithm usually used for the weights training.

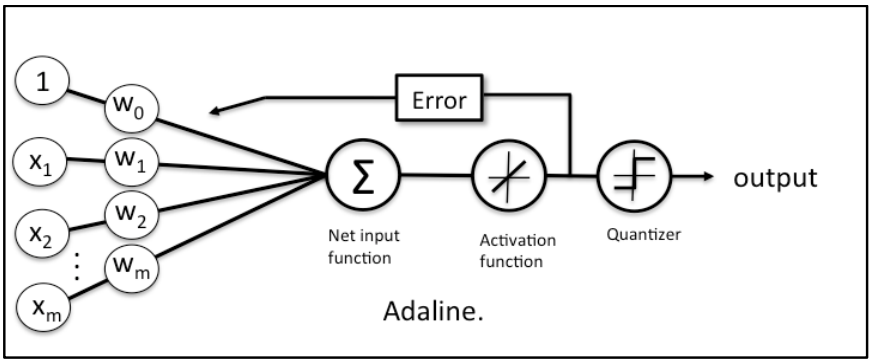

Figure 1. Symbolic illustration of Adaline Neural Network (Raschka, 2015)

Figure 1 shows an ADNN with multiple nodes where each node receives multiple inputs and generate one output, *y(k)*. Note that, k is the sampling time index (k = 0, 1, ...). The updating rule of the output is based on the following Equation 2:

$$y(k) = b + \sum_{i=0}^{r} w_i x(k-i), \qquad i = 0,1,2,...,r, \qquad [2]$$

where b, w_i, r are denoted as bias, weight and the order of adaptive linear combiner respectively. In ADNN, the activation function must be non-linear (such as sigmoid activation function) to ensure nonlinearity of the output. Sigmoid activation will give smooth gradient and help to adapt with variety of data (Sharma, 2017). The output classification of $y(k)$ is based on the following Equation 3:

$$y(k) = \begin{cases} 1 & , \ y(k) \geq 0 \\ -1 & , \ y(k) < 0 \end{cases}. \qquad [3]$$

Optimal value of w_i is crucial to ensure the input will always approach the optimal $y(k)$. The training pattern will undergo pre-defined number of iteration until the optimal value of were obtained. In this case, the weight updating is carried out by applying Delta rule.

Delta rule use in ADNN will evaluate the difference between $d(k)$ with the target output $d(k)$, in order to adjust the weight and threshold of the neuron. The error $e(k)$ is given by Equation 4:

$$e(k) = d(k) - y(k). \quad [4]$$

The value of $e(k)$ is vital for the change of weight and bias (Equation 5):

$$\begin{aligned} w_{i+1} &= w_i + \alpha e(k) x(k-i) \\ b_{i+1} &= b_i + \alpha e(k), \end{aligned} \quad [5]$$

where, w_{i+1} and b_{i+1} are the new value of weights and bias respectively. α is the learning rate that accelerate the searching of optimal value for $w_{i+1}$and b_{i+1} where α∈ (0, 1). $x(k-i)$ is set of activation input at time k. The proposed ADNN is structurally different than traditional Hebb Rule since the value of weights is adjusted as the training progress. In general, ADNN is based upon an idea that decrement in the value of $e(k)$ will reinforce the input-output connection. By reducing the value of $e(k)$, the back-spreading from one layer to the other layer can be reduced dramatically. Algorithm 1 shows the step based training algorithm for ADNN.

Algorithm 1: Adaline Neural Network

Step 1: Initialize weights w_i with small random value that is not equal to zero (Sivanandam & Deepa, 2006). Assign learning rate α.

Step 2: Set activation of input $x(k-i) = s(k-i)$, for i = 1 to r from Equation (3).

Step 3: With every bipolar training pair conduct Step 4-6.

Step 4: Measure the net input to output unit from Equation (2).

Step 5: Update bias and weights, $i = 1$ to r from Equation (5).

Step 6: Calculate the error from Equation (4).

Step 7: Test for stopping condition. Depending the stopping conditions, the number of iterations and epochs take place.

Logic Programming in ADNN

Logic programming can play a part as a representational knowledge that can be "learned" by ADNN. Apparent knowledge should be held as well as recuperated on the off chance that essential. The most thought here is to break the situation into smaller parts and after that hunt for logical rules which can demonstrate with a computer. This neuron model is oversimplified, and it has substantial computing potential and bipolar in nature. Based on Table 2, this could conduct the basic logic operations NOT, OR, and AND, with suitable chosen weights and thresholds. Every multivariable combinational function could be done using either the NOT and OR, or on the other hand the NOT and AND logical rules.

Coherent run this logical rule is one of the perfect ways to clarify the black box model of neural network. Neural network is a black box throughout the way that while it would approximate any function, studying the structure will not provide any insight into the nature of the function being approximated. The poor design practices which unintentionally result in logically redundant and may cause an unnecessary increase in network complexity.

The list of logical rules through Table 2 are used to develop logic programming in ADNN. These logical rules utilized as bipolar inputs and bipolar targets. The conventional ADNN utilizes bipolar $\{1,-1\}$ neuron representation that corresponds to $\{TRUE, FALSE\}$. Figure 2 shows the training process of logical rules can be implemented in ADNN. ADNN is capable of performing only a small subset of function known as linearly separable (Widrow & Lehr, 1990).

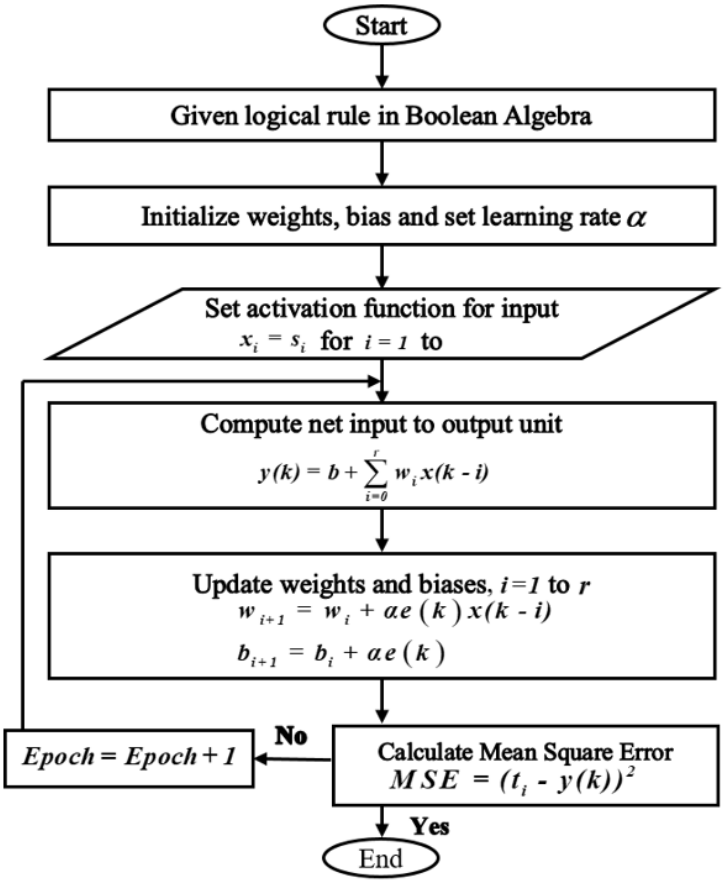

Figure 2. Flowchart of logic programming in Adaline Neural Network

With two inputs, ADNN can realize possible bipolar logical rules. Combination of elements and network of components can be utilized to perform functions which are not linearly separable.

Figure 3 demonstrates the distinctive situation that can be considered when logical rule in ADNN configuring the learning rate. However, learning rate will have an impact on how successfully ADNN will converge to arrive at the finest possible accuracy. The determination value of learning rate in the training process has a significant impact on the learning process. The learning rate decides how quickly ADNN is adjusted to the logical rule. ADNN was trained with diverse optimizer which is logical rule. For each optimizer it was trained with distinctive learning rate between $0 \leq \alpha \leq 1$ at logarithmic intervals. ADNN is trained until it achieves minimum mean square error. According to Smith (2017), we can approximate the optimal learning rate by increasing exponentially the learning rate at each iteration. In this case, the initial learning rate must be low in order to cater the increment of the α.

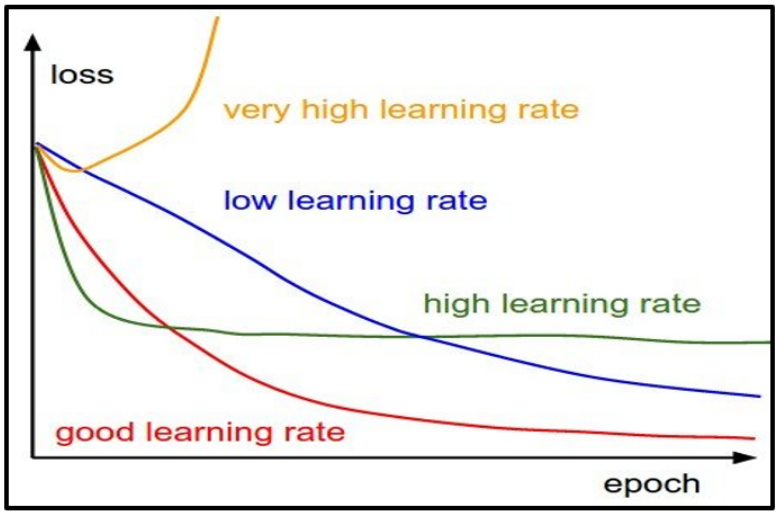

Figure 3. Symbolic illustration effect of different learning rate (Zulkifli, 2018)

IMPLEMENTATION

Performance Evaluation

In this section, ADNN is evaluated based on mean square error. During the simulation, the value of learning rate would be introduced during the learning phase of the logical rule in ADNN. The proposed ADNN is compared with logical rule and learning rate.

Mean Square Error (MSE)

Mean square error (MSE) was used as a standard metric to evaluate the correlation between the target value and the net input to the output (Chen et al., 2010). This is supposed to minimize average of the squared difference between the estimated value and the target value. Least mean square error (LMS) algorithm proposed by Widrow and Hoff in 1959, is an adaptive algorithm. The LMS algorithm is relatively straightforward and LMS also an example of supervised training. The supervised training that uses MSE cost function may use formal statistical methods to determine the confidence of a trained model. The value can be used to calculate the confident interval of the output of the network (Equation 6).

$$\{p_1, t_1\}, \{p_2, t_2\}, ..., \{p_q, t_q\}. \quad [6]$$

Here, $p_i = y(k)$ is an input to the network and t_i is the corresponding target output. The error is calculated as the difference between the target output and network output (Equation 7).

$$MSE = (t_i - y(k))^2. \quad [7]$$

The LMS algorithm will adjust the weights and bias of the ADNN to minimize the mean square error which is the difference between the target output t , and the net input y_{in} . The mean square error performance index for ADNN is quadratic. The performance index will either have a global minimum, a weak minimum or no minimum depending on characteristic of the input vector (Asha & Anuja, 2010).

Experimental Setup

This research based on simulated data set, where the initial neuron state was generated randomly. Based on Table 2, the variant of the logical rule would have applied in ADNN with several conditions that would take place where the learning process may run 50 epochs and 4 number of iteration. As stated in Table 3, a small random value of initial weights and bias was chosen because this may impact the error factor (Sivanandam & Deepa, 2006). There was no optimal number of iteration. In order to avoid overfitting, need to iterate until the error does not significantly decrease. The learning rate is hyperparameter may control how much the model can react to the evaluated error each time the weight is updated. The output from logical rules in ADNN would be compared with the target output until the weight change came to a small value.

RESULT AND DISCUSSION

Efficiency can be demonstrated by examining the value of MSE produced by ADNN with different logical rule. In term of stability, learning rate can be a metric to examine the stability of the proposed logical rule. Figures 4-8 reveal the MSE result for logical rule in ADNN with different value of learning rate respectively. The MSE was minimized

Table 3
Table of parameters for logical rules in ADNN

Parameter	Value
Initial Weight (w_i)	0.1
Bias (b)	0.1
Stopping Criteria	1. Number of iteration $= 4$ 2. Number of Epoch ≤ 50
Learning Rate	$0.01 \leq \alpha \leq 0.4$
Number of Layer	Single Layer
Neuro State	$S_i = \{-1, 1\}$

based on the number of registered epochs. In order to make a fair comparison among all logical rule, the simulations are carried out until 50 epochs. As shown in Figure 4, logical rule in ADNN such as OR, AND, NAND, and NOR take fewer number of epoch during the learning phase to minimized the error. Conventional ADNN that employed a higher number of epochs during the learning phase might come to a point where the network became over-adapted to training data and losses performance in terms of generalization such as in Figure 6-8. Based on the value of MSE, the solution in NAND-Function in ADNN revealed the lowest error compared to the other logical rule. Based on Figure 5, the value of learning rate $\alpha = 0.01$ is lower compared to Figure 4 value of learning rate $\alpha = 0.1$. The value of learning rate will determine how quickly or slowly ADNN can learn from the problem. EXOR-Function and EXNOR-Function in Figure 5 showed the lowest learning rate but the training process took significantly longer to train and the error was not well minimized. Very small learning rate will not only be time consuming but will increase the probability for the ADNN to stuck in local minima MSE value (Goodfellow et al., 2016). Nevertheless, another four logical rules in ADNN revealed that the network took a shorter time to train and the error was optimally minimized. The result demonstrates that the training algorithm is precisely designed to find an approximate solution to minimize errors although there is no guarantee that the proposed ADNN will always arrive to the optimal solution. Hence according to the experimental value, we obtained $\alpha = 0.1$ as an optimal learning rate.

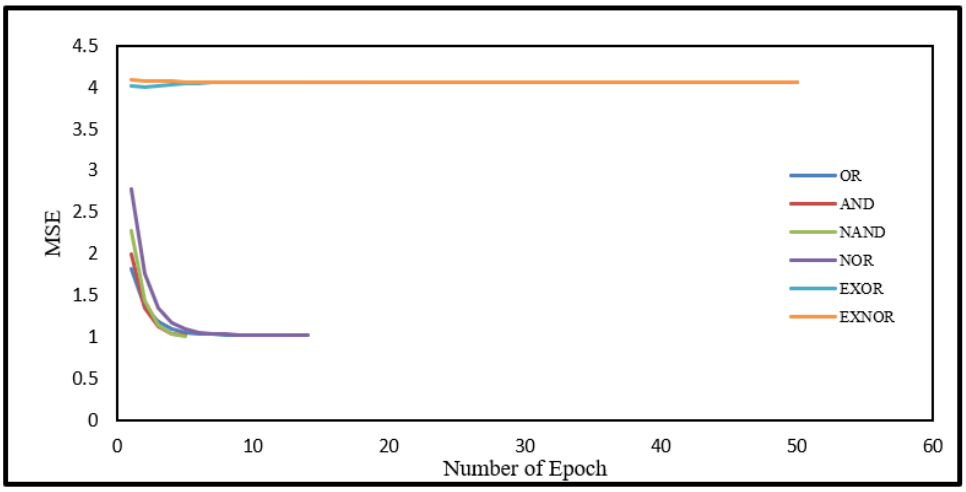

Figure 4. MSE for OR, AND, NAND, NOR, EXOR and EXNOR Function in ADNN with α = 0.1

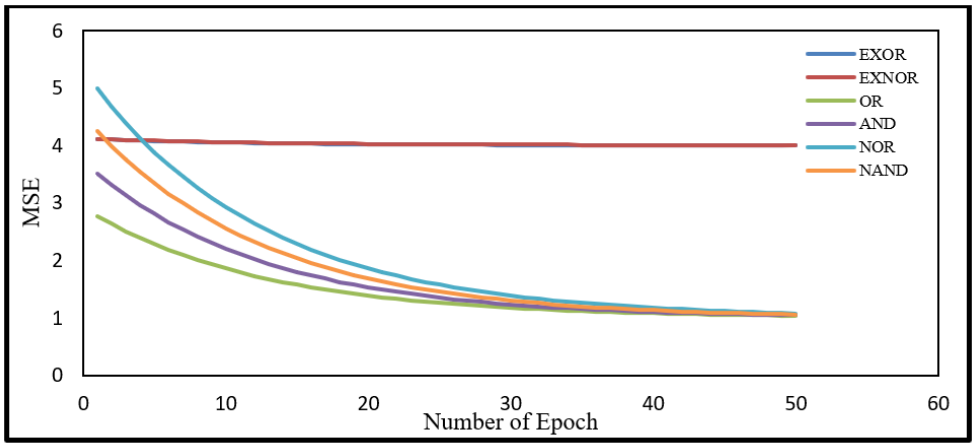

Figure 5. MSE for OR, AND, NAND, NOR, EXOR and EXNOR Function in ADNN with α = 0.01

According to Figures 4-8, the lowest MSE value that governs the ADNN is NAND function. In the case of $P = 1$, NAND function works effectively in producing the final neuron state $P = 1$. In this case, ADNN produces the least MSE value in order to achieve the optimal weight for the output neurons. This is considered as an interesting result because this is the first logical that utilized redundant variable and being implemented in ANN. This is a major breakthrough over the result obtained by Kasihmuddin et al. (2017) that considered the redundant logical with $P = 1$ as an outcome. As of simple propositional logical rule

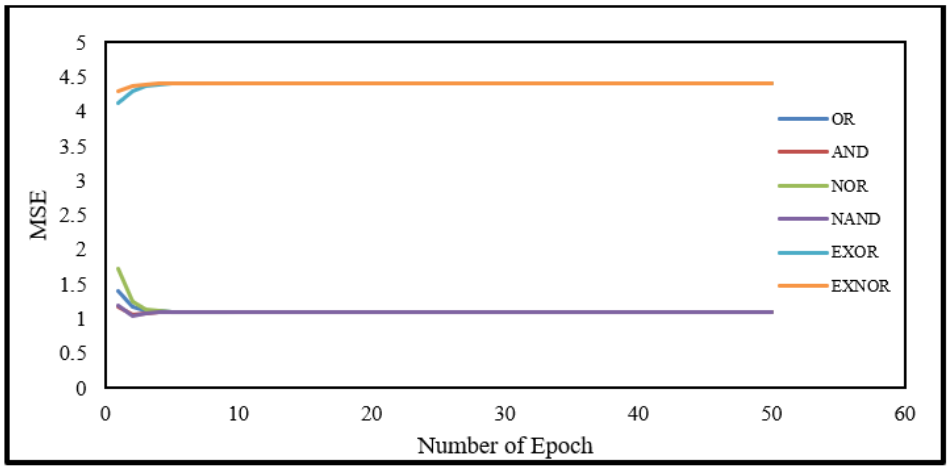

Figure 6. MSE for OR, AND, NAND, NOR, EXOR and EXNOR Function in ADNN with α = 0.2

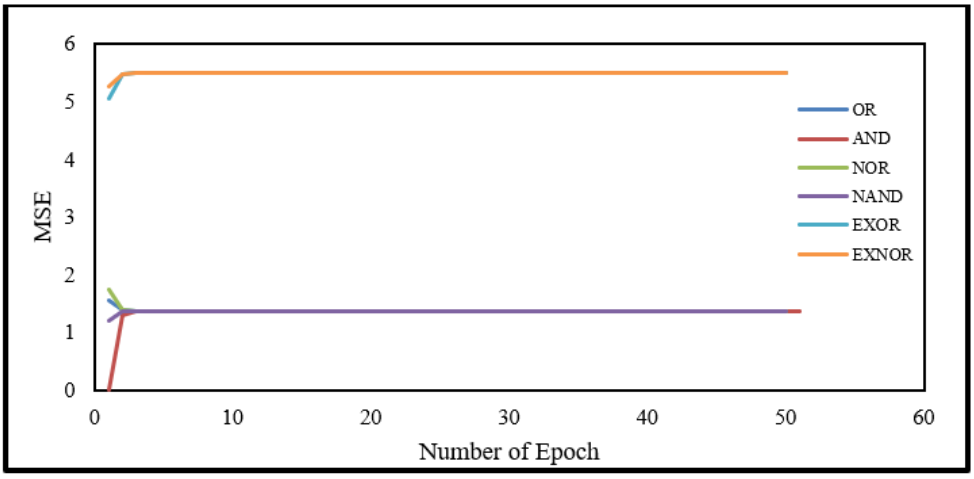

Figure 7. MSE for OR, AND, NAND, NOR, EXOR and EXNOR Function in ADNN with α = 0.3

such as OR and AND had converge effectively and has a good agreement with the work of Mansor et al. (2020) that used similar logical rule. The proposed ADNN with NAND logical rule is shown to be more effective than the work of Alzaeemi et al. (2020) because ADNN incorporated with logical rule does not requires any complex hidden layer and specialized parameters where Alzaeemi et al. (2020) presented 2SAT logic programming in Radial Basis Function Neural Network. The proposed ADNN also does not require any rigid cost function that leads to $P=1$ and computational clause checking. Hence,

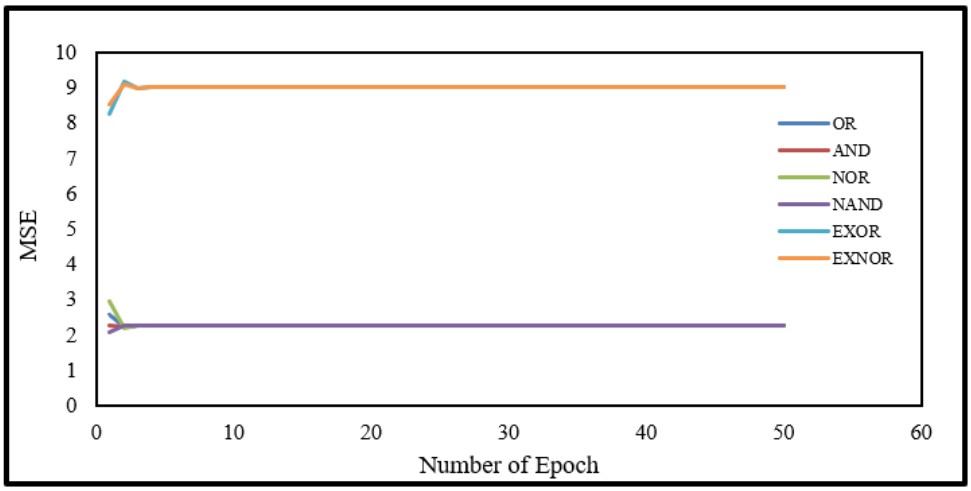

Figure 8. MSE for OR, AND, NAND, NOR, EXOR and EXNOR Function in ADNN with α = 0.4

parameter tuning is required to reduce the error accumulation for other logical rule such as EXOR and EXNOR. ADNN is tends to converge to suboptimal weight when dealing with EXOR and EXNOR due to the structure of the clause involved. Although the value of error is high, the final output of the neuron is consistently $P = 1$. In practice, the training of EXOR and EXNOR can be further optimized by implementing accelerating algorithm such as Metaheurictics Algorithm. The limitation of logic programming in ADNN are the logical rule and the learning rate depending on the complexity of the learning process.

CONCLUSION

The primary aim for logic programming in ANN is to create flexible ANN that can govern both non-redundant and redundant logical rule. In this paper, the implementation of various logical rule in ADNN is proposed. The result obtained demonstrates the effectiveness of ADNN in governing redundant logical rule. This study has successfully uncovered the best logical rule that can be governed in ADNN with the lowest MSE value. In this case, NAND has been observed to has the lowest MSE value which leads to $P = 1$ with regards to other redundant logic. Hence, this profound logical rule is only a tip of the iceberg in creating ANN that is governed by logical rule. For future work, ADNN can utilize the non-differentiable signum function for non-linearity feature or Madaline network. In this network, multiple unit of ADNN in parallel will be utilized to process the training data. Despite the success by the proposed paradigms, robust and continuous efforts are needed especially on the complexity and application of these paradigms to obtain more feasible solutions.

ACKNOWLEDGEMENT

This research was funded by Fundamental Research Grant Scheme (FRGS), Ministry of Education Malaysia, grant number 203/PJJAUH/6711751 and Universiti Sains Malaysia.

REFERENCES

Abdullah, W. A. T. W. (1992). Logic programming on a neural network. *International Journal of Intelligent Systems, 7*(6), 513-519. doi: https://doi.org/10.1002/int.4550070604

Alzaeemi, S., Mansor, M. A., Kasihmuddin, M. S. M., Sathasivam, S., & Mamat, M. (2020). Radial basis function neural network for 2 satisfiability programming. *Indonesian Journal of Electrical Engineering and Computer Science, 18*(1), 459-469. doi: 10.11591/ijeecs.v18.i1.pp459-469

Asha, P., & Anuja, J. (2010). *Benchmarking the competency mapping process with special reference to manufacturing industry.* Retrieved November 25, 2019, from https://www.asmedu.org/incon/INCON6HRRP.pdf

Chen, B., Zhu, Y., & Hu, J. (2010). Mean-square convergence analysis of ADALINE training with minimum error entropy criterion. *IEEE Transactions on Neural Networks*, *21*(7), 1168-1179. doi: 10.1109/TNN.2010.2050212

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. Cambridge, USA: MIT press. doi: https://doi.org/10.4258/hir.2016.22.4.351

Hamadneh, N. (2013). *Logic programming in radial basis function neural networks* (PhD thesis). Universiti Sains Malaysia, Malaysia.

Jannati, M., Hosseinian, S. H., Vahidi, B., & Li, G. J. (2016). ADALINE (Adaptive Linear Neuron)-based coordinated control for wind power fluctuations smoothing with reduced BESS (battery energy storage system) capacity. *Energy*, *101*, 1-8. doi: https://doi.org/10.1016/j.energy.2016.01.100

Kasihmuddin, M. S. M., Mansor, M. A., & Sathasivam, S. (2016). Bezier curves satisfiability model in enhanced Hopfield network. *International Journal of Intelligent Systems and Applications, 8*(12), 9-17. doi: 10.5815/ijisa.2016.12.02

Kasihmuddin, M. S. M., Mansor, M. A., & Sathasivam, S. (2017a). Hybrid genetic algorithm in the Hopfield network for logic satisfiability problem. *Pertanika Journal of Science and Technology, 25*(1), 139-152.

Kasihmuddin, M. S. M., Mansor, M. A., & Sathasivam, S. (2018). Discrete Hopfield neural network in restricted maximum k-satisfiability logic programming. *Sains Malaysiana, 47*(6), 1327-1335. doi: http://dx.doi.org/10.17576/jsm-2018-4706-30

Kavak, A., Yigit, H., & Ertunc, H. M. (2005). Using ADALINE neural network for performance improvement of smart antennas in TDD wireless communications. *IEEE Transactions on Neural Networks, 16*(6), 1616-1625. doi: 10.1109/TNN.2005.857947

Kho, L. C., Kasihmuddin, M. S. M., Mansor, M., & Sathasivam, S. (2020). Logic mining in league of legends. *Pertanika Journal of Science and Technology, 28*(1), 211-225.

Kowalski, R. (1978). Logic for data description. In *Logic and data bases* (pp. 77-103). Boston, MA: Springer. doi: https://doi.org/10.1007/978-1-4684-3384-5_4

Mansor, M. A., Jamaludin, S. Z. M., Kasihmuddin, M. S. M., Alzaeemi, S. A., Basir, M. F. M., & Sathasivam, S. (2020). Systematic boolean satisfiability programming in radial basis function neural network. *Processes, 8*(2), 1-16. doi: https://doi.org/10.3390/pr8020214

Mansor, M. A., Kasihmuddin, M. S. M., & Sathasivam, S. (2017). Artificial immune system paradigm in the Hopfield network for 3-satisfiability problem. *Pertanika Journal of Science and Technology, 25*(4), 1173-1188.

Negarestani, A., Setayeshi, S., Ghannadi-Maragheh, M., & Akashe, B. (2003). Estimation of the radon concentration in soil related to the environmental parameters by a modified Adaline neural network. *Applied Radiation and Isotopes, 58*(2), 269-273. doi: https://doi.org/10.1016/S0969-8043(02)00304-4

Pajares, G., & Jesús, M. (2001). Local stereovision matching through the ADALINE neural network. *Pattern Recognition Letters, 22*(14), 1457-1473. doi: https://doi.org/10.1016/S0167-8655(01)00097-6

Raschka, S. (2015). *Adaptive linear neuron*. Retrieved July 13, 2019, from https://rasbt.github.io/mlxtend/user_guide/classifier/Adaline/

Riche, J. (2011). Logic in Whitehead's universal algebra. *Logique and Analyse*, *214*(6), 135-159.

Sathasivam, S. (2010). Upgrading logic programming in Hopfield network. *Sains Malaysiana, 39*(1), 115-118.

Sathasivam, S., Mamat, M., Mansor, M., & Kasihmuddin, M. S. M. (2020b). Hybrid discrete Hopfield neural network based modified clonal selection algorithm for VLSI circuit verification. *Pertanika Journal of Science and Technology, 28*(1), 227-243.

Sathasivam, S., Mansor, M. A., Kasihmuddin, M. S. M., & Abubakar, H. (2020a). Election algorithm for random k satisfiability in the Hopfield neural network. *Processes, 8*(5), 1-19. doi: https://doi.org/10.3390/pr8050568

Sharma, P., Raghuvanshi, A., & Pachori, R. (2019). *Artificial intelligence and soft computing: Soft computing techniques: Artificial intelligence, neural networks, fuzzy logic and genetic algorithm.* Chhattisgarh, India: Educreation Publishing.

Sharma, S. (2017). *Activation functions in neural networks*. Retrieved September 8, 2020, from https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6

Sivanandam, S. N., & Deepa, S. N. (2006). *Introduction to neural networks using Matlab 6.0*. New Delhi, India: Tata McGraw-Hill Education.

Smith, L. N. (2017, March 24-31). Cyclical learning rates for training neural networks. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)* (pp. 464-472). Santa Rosa, CA, USA. doi: 10.1109/WACV.2017.58

Somogyi, Z., Henderson, F., & Conway, T. (1996). The execution algorithm of mercury, an efficient purely declarative logic programming language. *The Journal of Logic Programming*, *29*(1), 17-64. doi: https://doi.org/10.1016/S0743-1066(96)00068-4

Sujith, M., & Padma, S. (2020). Optimization of harmonics with active power filter based on ADALINE neural network. *Microprocessors and Microsystems, 73*(2020), 1-17. doi: https://doi.org/10.1016/j.micpro.2019.102976

Wang, Z. Q., Manry, M. T., & Schiano, L. (2000). LMS learning algorithms: Misconceptions and new results on convergence. *IEEE Transactions on Neural Network, 11*(1), 47-56. doi: 10.1109/72.822509

Widrow, B., & Hoff, M. E. (1960). *Adaptive switching circuits*. Stanford, California: Stanford University CA Stanford Electronics Labs.

Widrow, B., & Hoff, M. E. (1959). Adaptive sampled-data systems - A statistical theory of adaptation. *I.R.E. Wescon Convention Record, 4*, 96-104.

Widrow, B., & Lehr, M. A. (1990). 30 years of adaptive neural networks: Perceptron, madaline, and backpropagation. *Proceedings of the IEEE*, *78*(9), 1415-1442. doi: 10.1109/5.58323

Widrow, B., & Stearns, S. D. (1985). *Adaptive signal processing*. Englewood Cliffs, N.J: Prentice-Hall.

Zulkifli, H. (2018). *Understanding learning rates and how it improves performance in deep learning*. Retrieved July 16, 2019, from https://towardsdatascience.com/understanding-learning-rates-and-howit-improves-performance-in-deep-learning-d0d4059c1c10.