International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)
Volume 7, Issue 2, February 2018, ISSN: 2278 – 1323

Computing the Time Complexity of ANFIS Algorithm

Retantyo Wardoyo[1], Linda Nur Afifa[2]
[1] *Computer Science, Universitas Gadjah Mada Yogyakarta*
[2] *Informatics Engineering, Universitas Darma Persada Jakarta*

***Abstract*—The effectiveness of an algorithm is measured from the amount of time and space required by the algorithm. The objective of this study was to measure the time complexity of the ANFIS algorithm. The time complexity is calcultaed by counting the number of loops and operators used in a procedure. In addition, calculate the time required to running the ANFIS algorithm with given a number of input. The size of the input (n) is very influential as it states the amount of data processed. The results of the profiling showed that ANFIS has the asymptotic time complexity O(n).**

***Index Terms*—ANFIS, running time, time complexity, asymptotic notation.**

I. Introduction

An issue may have many solutions, and for this, an algorithm used must be proper and efficient [1]. The complexity of an algorithm is a measure of to what extent a computation is required to resolve a problem, commonly expressed in time complexity T(n) and space complexity S(n). The time complexity of an algorithm is expressed by T (n) that contains the expressions of numbers and the number of steps required as a function of the problem level [2]. The time complexity will increase along with the increase of the input size (n). Time used to run an algorithm must be faster; hence, time complexity becomes essential.

For the quite large value (n), even unlimited, analysis on the asymptotic time efficiency of an algorithm is called as the asymptotic time complexity, expressed in three kinds: the best-case denoted by Ω (g (n)) (Big-Omega), the state average (average case) denoted by Θ (g (n)) (Big-Theta) and circumstances the worst (worst case) denoted by O (g (n)) (Big-O) [3].

Asymptotic time complexity can be explained using Figure 1. Figure 1 shows the notation O to be the limit of a function f (n). it is stated as f (n) = O (g (n)) if there is a positive constant $n_0$and c such that in the $n_0$and in the right side of n_0, the value f (n) is always right on c (g (n) or under c (g (n). the time complexity of the algorithm is expressed with O (g (n)) or read with the "big- O of (g (n))”.

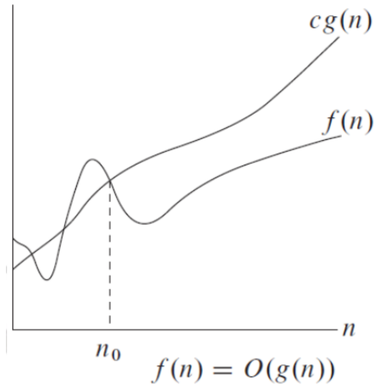

Figure 1. The Graph of Asymptotic Notation O [1]

Adaptive neuro fuzzy inference system (ANFIS) is a method that combines the mechanism of *fuzzy inference system* with neural networks [4].

In this paper, a calculation of how much the time complexity of the algorithm is. Furthermore, this paper discusses about ANFIS (Section 2). Section 3 presents the time complexity of algorithm; Section 4 and Section 5 present the simulation resultsand conclusion, respectively.

II. Adaptive neuro fuzzy inference system (anfis)

*A. ANFIS*Structure

The structure of ANFIS with the structure of the first-order Sugeno model used the common form with two rules of if-then fuzzy as stated as follows[2],

Rule 1：
If $x\ is\ A_1$and $y\ is\ B_1$, then$f_1 = p_1x + q_1y + r_1$ (1)

Rule 2：
If $x\ is\ A_2$and $y\ is\ B_2$, then$f_2 = p_2x + q_2y + r_2$ (2)

In the inference of first-order fuzzy Sugeno with two inputs, the rules used were equalized with the structure of the network structure with five layers as shown in Figure 2.

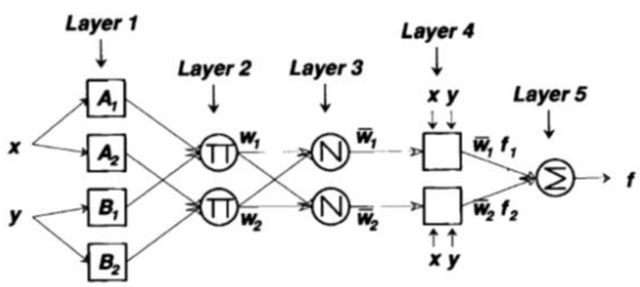

Figure 2. ANFIS Structure[1]

In layer 1, fuzziness process was conducted followed by the layer 2 in performing the AND operation of the part of the antecedent from the fuzzy rules. In layer 3, the normalization of the membership function (MF) was given and layer 4 was to execute the part of *consequent*of fuzzy rules and layer 5 calculated the output to sum the result from layer 4. The following is the formula used in each layer [3][4][5].

Layer 1
In this layer is adaptive nodes and generates a membership grade of a linguistic label for every node 'i'. X, y are input of the nodes. A_1, A_2, B_1, B_2 are the linguistic labels use in fuzzy theory for dividing the membership function. The relationship between the output and input functions of this layer can be expressed as:

$O_{Ii} = \mu_{Ai}(A); \quad i = 1{,}2(3)$
$O_{Ij} = \mu_{Bj}(B); \quad j = 1{,}2(4)$

Where O_{Ii} and O_{Ij} denote the ouput functions, μ_{Ai} and μ_{Bj} denote the membership function (MF).

Layer 2
Every node 'i' in this layer is a fixed node, whose output is the product of all the incoming signal referred to 'firing strenght'.

$O_{2,i} = w_i = \mu_{Ai}(A)\mu_{Bj}(B); \quad i = 1{,}2(5)$

Where $O_{2,i}$ denotes the output of layer 2.

Layer3
The node of this layer calculates ratio of the i[th] rules strenght to the sum of all rules firing strenght. Hereafter, $O_{3,i}$will be called normalized firing strength.

$O_{3,i} = \overline{w}_i \frac{w_i}{w_1+w_2}, \quad i = 1, 2(6)$

Layer 4
This layer provides output that resulting from the inference of rule. The resulting output obtainable from simply a product of normalized firing rule strengh and first order polynomial.

$O_{4,i} = \overline{w_i} f_i = \overline{w_i}(p_i A + q_i B + r_i), \ i = 1{,}2(7)$

In this layer p_i, q_i and r_i are consequence parameter, while $O_{4,i}$ is the output layer 4.

Layer 5
This layer consist of single node that computes of all incoming signal from layer 4, calculated using equation below.

$O_{5,i} = \sum_i \overline{w_i} f_i = \frac{\sum_i w_i f_i}{w_1+w_2}, \quad i = 1, 2 \ (8)$

The first and fourth layer are adaptive layer . They have modifiable parameter in layer 1 and consequent parameter in fourth layer. Furthermore, tuning modifiable parameter to macthing ANFIS output with the training data used leraning process [6].

B. Learning Algorithm

To obtain the optimum rule, several algorithms of learning rules can be applied. To fix the parameters of the adaptive network, a hybrid algorithm as a combination of propagation method and gradient-descent method can be used[4]. Hybrid learning procedure is depicted on Figure 3.

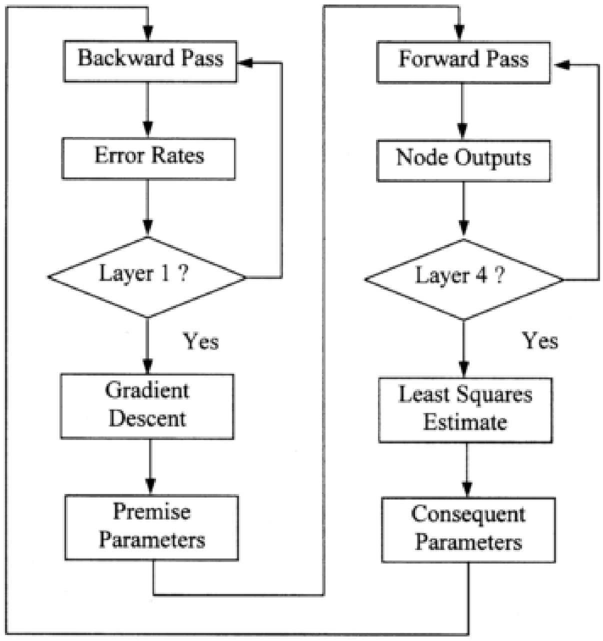

Figure 3. Hybrid Learning Procedure of ANFIS [6]

Here are some rules that are often used, and affect the computational complexity, that is gradient descent and Least Square Estimate (LSE) [5]. Gradient descent or steepest descent is an algorithm used to find minimum the value of a function by using the negative value of the function gradient at a point. While the LSE is a method to estimates of parameter consequent every rules using linier regression.

Overall, hybrid learning procedure of ANFIS are using LSE to forward pass and Gradient descent for backward pass as depicted in Figure 3. To optimize the consequences parameter the Least Square method is used. After optimal consequent parameter are found, adjust optimally the premise parameter corresponding to the fuzzy set in input domain. Error ouput is used to adapt premise parameter using back propagation. Backward learning being done if the square of error is smaller or equal to predefined error criterion [7].

III. ALGORITHM TIME COMPLEXITY

To determine the complexity of an algorithm requires a size of input (n) and running time. The length of the running time commonly will increase along with the increase of input (n). Running time of algoritm in certain n is an operation or step executed. The amount of constant time is required to execute each line of pseudocode. One line can have the amount of time that is different from others.

By assuming the first line, it needs a time as large as c_i , in which $c_i$is a Constanta. To determine the running time of a line of pseudocode is by multiplying the Constanta $c_i$with the time required to execute the line.

For the case where there is a loop while or for with the length of n, then the command is executed with the time of $n+1$. Meanwhile, for the line of comment, it is computing as a line that is not executed. Thus, the amount of time for the line is zero.

Furthermore, the running time of the algorithm is the running time of each command executed. A command requires $c_i$the step of $n$time to be executed to have an effect as much as $c_i n$in the total running time ($T(n)$).

Once $T(n)$has been formed, it could determine the form of algorithm using the asymptotic notation$O$, thus the level of the increase of the running time can be predicted if the size of the input $n$is increased.

In most cases analysis time complexcity used for very large input-size and worst case scenario. The worst case scenario commonly expressed with Big-O notation or O(g(n)) [8], defined at equation below.

$$O(g(n)) = \left\{ \begin{matrix} f(n): there\ is\ exist\ contant\ C\ and\ n_{0,} \\ f(n) \leq Cg(n), for\ n \geq n_0 \end{matrix} \right\} (9)$$

Furthermore Big-O formula will be used to calculate time complexcity or running time of ANFIS algorithm. The general rule to calculate running time of the large number of computer program is summing running time of all fractions, as in the following equation.

$$running\ time = \sum running\ time\ of\ all\ fragments (10)$$

For example there are three fragment of program and they have $O(1), O(n) and\ O(n^2)$ time complexity. According to equation 10, the running time is

$$T(n) = O(1) + O(n) + O(n^2) \approx O(n^2)$$

IV. Results and Discussion

To compute the time complexity of ANFIS in this paper was performed based upon the computation as explained in sectionII and III, stated that the gradient descent and LSE are affected to computational complexity. In this paper, learning process of ANFIS algorithm was run by calling some functions in accordance to procedure as percept on figure 3.

In figure 4 process learning start from data input then process in layer 1 to layer 5 using equation 1-8. After process in layer 5 and the output is obtained, error will be calculated to compare with the squared error of training data pair. Back propagation learning will do if value of square error has not reached the expected.

Furthermore, learning process in figure 4 translated to the code program for running time calculation. Running time calculated with asymtotic time complexity O(g(n)) every fragment of programs.

Furthermore, learning process in figure 3 translated to the code program for running time calculation. Running time calculated with asymtotic time complexity O(g(n)) every fragment of programs.

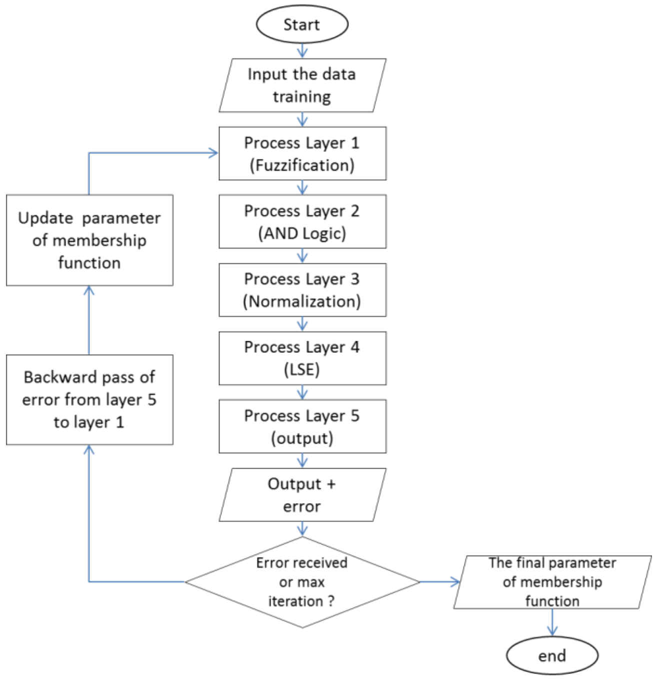

Figure 4. Learning process of ANFIS [9]

*Running time*in the equation can be stated with the equation of $an + b$, in which the value of $a$and $b$dependent upon the amount of the value c_i. Based on the results above, $T(n)$can be stated with the asymptotic notation. To all value of $n \geq 1$and $c \geq 1$equation 4.1 can be changed into

$$\sum_{i=1}^{14} c_i\, n + c_{11} + c_{12} = an + b = O(n) (11)$$

Based upon the equation 11, the asymptotic value is (n) , this means that the algorithm is linear. It means if n if n becomes$2n$, then the running time of algorithm will be 2 folds from the beginning.

Furthermore, simulation was conducted by giving a number of data. The result of the simulation showed that *Total Running Time* ($T(n)$) also experienced the increase along with the increase of the number of the inputs as shown in Table 1.

Table 1. Running Time Total

Percoba-an ke	Jml_data	Running time/fungsi (s)				Total (T(n))
		f_fcm	f_anfis	main_anfis	recursife_lse	
1	25	0,045	23,705	23,770	0,868	48,388
2	50	0,055	22,248	22,293	1,899	46,495
3	100	0,120	25,308	25,363	2,946	53,737
4	150	0,125	25,981	26,021	3,200	55,327
5	200	0,156	31,736	31,892	8,167	71,951
6	250	0,187	31,767	31,814	6,413	70,181
7	300	0,203	34,358	31,000	7,025	72,586
8	350	0,218	38,177	38,208	6,365	82,968
9	400	0,250	40,338	40,369	6,925	87,882
10	450	0,296	44,968	44,999	8,066	98,329
11	500	0,312	48,410	48,441	12,182	109,345
12	550	0,339	51,555	51,586	12,630	116,110
13	600	0,359	54,215	54,247	11,9640	120,785
14	650	0,375	57,846	57,877	11,8000	127,898
15	700	0,374	60,791	60,854	12,7950	134,814

As shown in Table 1, it was obtained by doing the *profiling*of the program execution. The experiment was conducted for 15 times with the more increasing number of data. The total of time needed to execute the function as a whole also increased. The increase of the *running time*of all functions in algorithms of ANFIS is shown in Figure 3.

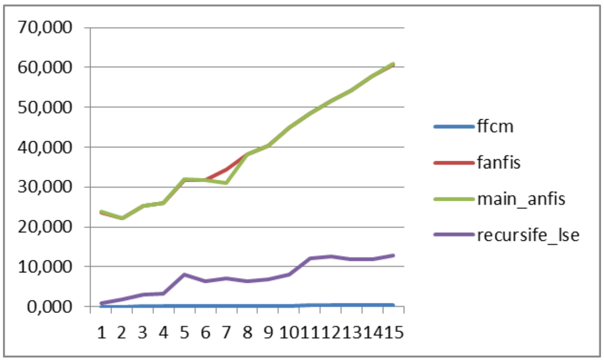

Figure 5. The increase of *Running Time*

V. Conclusion

In this paper, the simulation and calculation of running time of ANFIS have been conducted. The result of this simulation showed that the *running time*had the asymptotic time complexity as much as $O(n)$ or frequently called to increase linearly. The *Running time*increases along with the increase of input in the simulation. The ANFIS algorithm combined the fuzzy algorithm and nervous networks; thus when the program was given a number of inputs, then it would be continued by conducting the weight update to obtain the most optimum parameter. The rules used to change the weight were very influential in the complexity of the computation of ANFIS algorithm. There was an increasing time on the *running time*of rules (rekursife_lse).

REFERENCES

[1] U. N. Azizah, "Comparison of Laplacian Edge Detector Based on Time Complexity and Image Result," Universitas Pendidikan Indonesia, 2013.
[2] Ariesta damayanti, "ANFIS and Genetic Algorithm Method For Early Detection Toddler Development Deviations," Gadjag Mada, 2015.
[3] N. Walia, "ANFIS : Adaptive Neuro-Fuzzy Inference System- A Survey," vol. 123, no. 13, pp. 32–39, 2015.
[4] B. Han and H. Ouyang, "A Novel Speed Identification Method of Induction Motor Based ANFIS," pp. 53–58, 2013.
[5] L. B. Fazlic, Z. Avdagic, and I. Besic, "GA-ANFIS expert system prototype for detection of tar content in the manufacturing process," *2015 38th Int. Conv. Inf. Commun. Technol. Electron. Microelectron. MIPRO 2015 - Proc.*, no. May, pp. 1194–1199, 2015.
[6] S. S. Roy, "Design of adaptive neuro-fuzzy inference system for predicting surface roughness in turning operation," *J. Sci. Ind. Res.*, vol. 64, pp. 653–659, 2005.
[7] "ANFIS," UMY.
[8] M. Sipser, "Introcuction to Theory of computation by Micheal Sipser Ist Ed..pdf." 1997.
[9] R. Hardi, "Intensity of Rainfall Prediction Using ANFIS," Universitas Islam Indonesia, 2006.

Retantyo Wardoyo, He had his undergraduate (Drs) in Mathematics from UniversitasGadjahMada, Yogyakarta, Indonesia in 1982, and Master (MSc.) in Computer Science from the University of Manchester, UK in 1990. He had his Doctoral degree (PhD) in Computation from University of Manchester Institute of Science and Technology, UK in 1996.
Dr. Wardoyo's research area of interests are database systems, operating systems, management information systems, fuzzy logics, and software engineering

Linda Nur Afifa, obtained her title in engineering (MT) at the Sekolah Teknik Elektro dan Informatika (STEI) Institut Teknologi Bandung, and graduated in 2010. Currently, she is a lecturer in Universitas Darma Persada, Jakarta.

