IMPLEMENTATION OF NEURAL NETWORK BACKPROPAGATION ALGORITHM FOR THE BEHAVIOR OF THE NPC GROUP IN ACTION-RPG GAME

Reza Purbayanto

Information Engineering University Computer Indonesia

Jl.Dipatiukur 112-114 Bandung
E-mail : reza.purbayanto@gmail.com

ABSTRACK

Role Playing Game or commonly abbreviated as RPG is a type of game where the players play the role of a fictional character which aims to knit a story within the game. Usually the enemies in-game Action-RPGS created a group with job class. From some of the job class was expected to be working together fits the situation and based on job class.
Required algorithm that can provide solutions for a non player character (NPC) can conduct cooperation. Do not cover the possibility of the implementation of Artificial Neural Network (ANN) in AI Game. Backpropagation is one of the methods most used ANN, because his model almost the same with control systems in General. ANN with backpropagation properties selected in the Action-RPG game for its social interactions.
Based on the results of the research can be summed up the implementation of neural network backpropagation algorithm can be used to predict the pattern of cooperation that should be chosen in order to beat the players. From the datasets used have generated prediction accuracy of 96.66% using the parameters of the learning rate limit error of 0.00001 and 0.4 uses 8 perceptron on a hidden layer. With pretty good accuracy may be inferred that the NPC enemies can work together in accordance with the training data patterns are used and objectively conduct cooperation can be seen, this makes the game better.
Keywords: action-rpg, cooperation, neural network backpropagation, games

1. INTRODUCTION

Role Playing Game or commonly abbreviated as RPG is a type of game where the players play the role of a fictional character which aims to knit a story within the game. Players design a main character who will he play according the characteristics he wants. On the genre Action-RPG is all action is done in real-time and have full control on the character the same as hack and slash genre or shooter game but the players figure the delusion. On the type of game Action-RPG elements non player character (NPC) is very important, the NPC itself must have intelligence in order to have a behavior. Artificial intelligence (AI) was created and put into a machine (the computer) to be able to do the job like a human can do, this is one way to help a machine to give a decision for any existing conditions.

The issue of prior studies is the lack of cooperation on the behavior of NPCs. According to the dictionary meaning of the word Indonesia has cooperation activities or work done by some people (institutions, Governments, etc.) to achieve common goals [2]. On the action games-RPG, there is an important element that is the NPC enemy, enemy NPC is usually distinguished by a few job classes such as assassin, warior, archer, and NPC enemies have the same goal and that is to beat the player character. Usually the enemies in-game action-RPGS created a group with job class. From some of the job class was expected to be working together following the situation and based on his job class aims to let players lose (die) or not get experience (xp).

1.1 Action-RPG

Action-Role Playing Game or ARPG is a form of a subgenre that is defined from the role-playing video game that combines elements from the action or action-adventure games, stressed the real-time action in which the player has direct control on the character, not turn-based or menu-driven fighter. These games often use combat system similar to the hack and slash or shooter game.

1.2 Neural Network

Neural network is an artificial representation is one of the human brain which is always trying to simulate the learning process in the human brain. The term artificial neural network used because here it is implemented using a computer program which is able to solve a number of calculation process during the learning process.

1.3 Supervised Learning

Neural network learning method called terawasi if the output is expected to have been known before, e.g. as in process AND in the process of learning, a pattern of input will be provided to single neurons in the input layer. This pattern will be

dirambatkan along the network of nerves to get to the output layer neurons. This output layer will resurrect the output pattern that will be matched against the pattern output target. If there is a difference between learning outcomes output pattern with the pattern of the target, then the error will appear. If the value of the error is still quite large, indicating that still need to be done a lot more learning more.

1.4 Backpropagation

One of the artificial neural network architecture that is very popular is a multilayer feedforward networks. In General, such a network consists of a number of units of the neuron as a the input layers, one or more layers of node-node computational layer neurons, and a hidden layer node-node computational output neurons. Dipropagasikan input signals to the next direction (the direction of the output layer), layer upon layer. This type of network is the result of generalizing from a single layer perceptron architecture, so commonly referred to as the multilayer perceptron (MLPs). Error back propagation algorithm is MLPs that uses the principle of learning terawasi. Back propagation (in the direction of the input layer) occurred after the network generate output containing errors. In this phase the entire synaptic weights (which has no zero activation) in the network will be adjusted to correct/minimize the error occurred (error correction rule). For the training phase, the partners network propagation to the front and back done repeatedly for a set of data, then the exercise is repeated for a number of epoch (one for the entire lewatan session data of exercise in a training process networks) until the error happened reached the limit or zero tolerance for certain small network backpropagation can be seen in Figure 1.

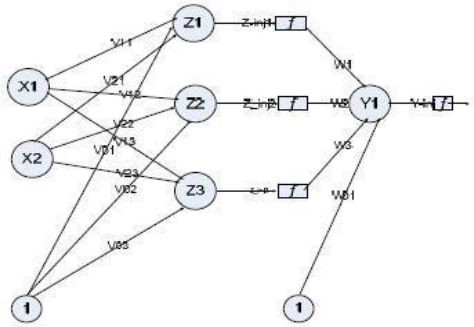


Figure 1. Architecture Of Multi Layer Perceptron

1.5 Stages of Neural Network Backpropagation Algorithman Algoritma Neural Network Backpropagation

Step 0:
The awarding of the weights initialization (rated small randomly)
Step 1:
If the condition stops Reached not, repeat steps 2 through 9 until the end of the iteration condition is fulfilled (condition if it stops looping looping Reaches maximum or mean square error (MSE) is smaller than the ratio study)
Step 2:
For each pair of training data (training data) do steps 3 through 8 Propagation forward (Feedforward)
Step 3:
Each unit of input (Xi, I = 1, ..., n) receives the signal the input signal Xi and disseminated to the units the next section (hidden layer units)
Step 4:
Each unit is hidden layer of Multiplied by the factor weights and aggregated and added to the bias:

$$Z_{in_j} = V_{oj} + \sum_{I=1}^{n} X_i V_{ij} \quad (1)$$

Then calculate in accordance with the activation functions are used:

$$Z_j = f(Z_{in_j}) \quad (2)$$

When used is a sigmoid function, the form of these functions are:

$$Z_j = \frac{1}{1+exp^{-z_in_j}} \quad (3)$$

Each unit output (Yk, k = 1,2,3, .. m) multiplied by a factor weighing and summed:

$$Y_{in_k} = W_{ok} + \sum_{j=1}^{p} Z_j W_{jk} \quad (4)$$

Calculated in accordance with the activation function

$$y_k = f(y_{in_k}) \quad (5)$$

Step 6:
Each unit output (Yk, k = 1, ..., m) received a target pattern corresponding to the input pattern while training and the error count:

$$\delta_k = y_k(1 - y_k) \quad (6)$$

Calculating the weighing factor improvement later to fix w_{jk}

$$\Delta W_{kj} = \alpha . \partial_k . Z_j \quad (7)$$

Calculating the correction fixes:

$$\Delta W_{0j} = \alpha . \partial_k \quad (8)$$

And use the value on all units the previous layer.
Step 7:
Each weighing connecting units l; apisan output with units in the hidden layer (delta multiplied and summed as an input to the next layer units.

$$\delta_in_j = \sum_{k=1}^{n} \delta_k W_{jk} \quad (9)$$

Furthermore, multiplied by the derivative of the function aktifasinya to calculate the error.

$$\delta_j = \delta_in_j f'(y_in_j) \quad (10)$$

Then calculate the weights improvement (used to fix V_{ij})

$$\Delta V_{ij} = \alpha \delta_j X_i \quad (11)$$

Then calculate the bias improvement (to improve V_{oj})

$$\Delta V_{oj} = \alpha \delta_j \quad (12)$$

Fixing weights and biases
Step 8:
Each output unit (yk, k = 1, ..., m) fixed bias and penimbangnya (j = 0, ..., p)

$$W_{jk}(baru) = W_{jk}(lama) \Delta W_{jk} \quad (13)$$

Each unit is hidden (Zj, j: 1, ..., p) fixed bias and penimbangnya (j = 0, ..., n)

$$V_{jk}(baru) = V_{jk}(lama)\Delta V_{jk} \quad (14)$$

Step 9:

Test conditions of termination (end of iteration) dismissal testing is done by calculating the value of the error. If you are looking for the error value that does not have a target value then the formula to find the value of error is as follows:

$$1 - (y)^2 \quad (15)$$

If you already have a target value then the formula to find the value of error is as follows:

$$\frac{1}{2}\sum k\,(Tk - Yk)^2 \quad (16)$$

2. THE CONTENT OF RESEARCH

Mount & Blade is a type of game with action-rpg developed by Taleworlds Entertainment where the game using your desktop or game console to play it. On the game where the player plays the main character with the action directly and each player character or NPC has different characteristics and can evolve to follow the course of the game. As for the rules in simulation games are as follows:

1. The player must defeat all NPC enemy, if all the NPC enemy died or fled or dead main character then the game finished.
2. The NPC enemy can escape if out of the arena for the game.
3. the NPC enemy have 3 kinds of job class (Warior, Assassin, Archer).
4. the NPC enemy respawn Points and the main character is determined by the system.
5. Enemy with job class warior and assassin attack near distance.
6. Enemy with job class archer has the distance of attack much.
7. There are 4 main attribute table in every character that is strength, agility, speed and attack.
8. Each of the points of strength adds 1 point attack.
9. any points of agility adds 0.5% speed.

The size of the gaming arena that is 300x300 pixels with a scale of 1:3779.

2.1 Determination of Input & Decision

In this game a decision taken is influenced by the value of the character in the game so that the data input to the network architecture used was as follows:

1. Input 1 (x 1): the value of HP (health points) npc warrior
2. Input 2 (x 2): the value of HP (health points) npc assassin
3. Input 3 (x 3): the value of HP (health points) npc archer
4. Input 2 (x 4): the value of HP (health point) player
5. Input 3 (x 5): value to the distance towards the warrior npc player
6. Input 4 (x 6): value to the distance towards the assassin npc player
7. Input 5 (x 7): value to the distance npc archer leading to player

Determination of the decisions will be made based on the reference to tactics. As for some of the tactics used by the Romans while fighting as follows:

a. the fight will begin by soldiers over long distances (the archers), they begin to attack when the enemy is at the distance of 18 meters.
b. when the enemy had already reached 2 meters then the warrior with the gladius (a short range weapon) and began attacking soldiers with long distance backwards.
c. When the enemy will use the javelin (remote weapon) then it will stick with the formations formed a cordon of soldiers who usually brings a more solid shield to protect the troops behind him.
d. Soldier survived also protects the weaker soldier when an enemy more powerful strikes.

Therefore the decision was made some conditions resembled the tactics above in decision making that exists are as follows:

Table 1. Decision

Decision	Warior	Assasin	Archer
1	Escape	Escape	Escape
2	Escape	Escape	Attack
3	Attack	Attack	Hold
4	Attack	Hold	Hold
5	Hold	Attack	Hold
6	Hold	Hold	Attack

Here is a sample training data:

Table 2. Data Training

No	Data Input							Target Output
	X1	X2	X3	X4	X5	X6	X7	O1
1	100	100	100	100	33	31	35	0.92
2	100	100	100	100	25	24	28	0.92
3	100	100	100	100	19	18	22	0.92
4	100	100	100	100	13	11	15	0.92
5	100	100	100	95	6	6	10	0.75
6	100	70	100	90	2	1	8	0.75
7	100	70	100	90	2	1	8	0.75
8	100	40	100	80	3	2	7	0.75
9	100	30	100	85	3	2	7	0.58
10	100	30	100	85	2	1	8	0.58
11	80	30	100	70	1	3	9	0.58
12	60	30	100	50	2	2	10	0.58
13	40	30	100	45	2	3	5	0.42
14	40	30	70	45	3	3	1	0.42
15	40	30	70	40	3	2	2	0.42
16	40	30	40	40	2	3	1	0.42

17	40	0	100	40	1	0	11	0.25
18	0	0	100	40	1	0	9	0.25
19	0	0	100	40	0	0	6	0.25
20	0	0	100	40	0	0	2	0.25
21	0	0	70	40	0	0	1	0.08
22	0	0	40	40	0	0	1	0.08
23	0	0	10	40	0	0	1	0.08
24	0	0	0	40	0	0	0	0.08

2.2 Decision Threshold Value

On the value of the output, there is value to determine the decision of the enemy. The value of the normalization of the decision obtained using the equation. The existing values can be seen in table 3.6

Table 3. Threshold value decision

Warior	Assasin	Archer	Nilai Keputusan	Value normalization decision	Treshold	classification Cooperation
Escape	Escape	Escape	1	0.16	0-0.16	All Retreat
Escape	Escape	Attack	2	0.33	0.17-0.33	Defend Retreat
Attack	Attack	Hold	3	0.5	0.34-0.5	Full Attack
Attack	Hold	Hold	4	0.66	0.51-0.66	Warrior Attack
Hold	Attack	Hold	5	0.83	0.67-0.83	Assassin Attack
Hold	Hold	Attack	6	1	0.84-1	Archer Attack

2.3 Training & Testing Neural Network Backpropagation

The following is the value weights used:

Tabel 4. Random value in the input network

Input	Hidden 1	Hidden 2	Hidden 3
1	0.7195	0.6621	0.96
2	0.4957	0.6251	0.3561
3	0.7862	0.81	0.2136
4	0.1623	0.2011	0.1116
5	0.5151	0.4327	0.6204
6	0.1756	0.4083	0.564
7	0.4051	0.3564	0.294
bias	0.5469	0.8274	0.9169

Table 5. The value of random weighting networks on hidden

Hidden	Output
1	0.6807
2	0.9666
3	0.647
bias	0.2964

Here is the value of inputs used:

Table 6. Input Value

Input	Description	Value
x1	Nilai HP Warior	100
x2	Nilai HP Assassin	30
x3	Nilai HP Archer	100
x4	Nilai HP Player	85
x5	Nilai jarak warior ke player	2
x6	Nilai jarak assassin ke player	1
x7	Nilai jarak archer ke player	8

Here are the values of the input that has been on the normalization:

Table 7. Value of the normalization of the input

Input	Value Normalization
x1	1
x2	0.3
x3	1
x4	0.85
x5	0.067
x6	0.033
x7	0.026

Backpropagation neural network parameter used is as follows:

Learning Rate: 0.5

Minimum Error: 0.0001

Of the value of the input architecture used for neural backpropagation network are as follows:

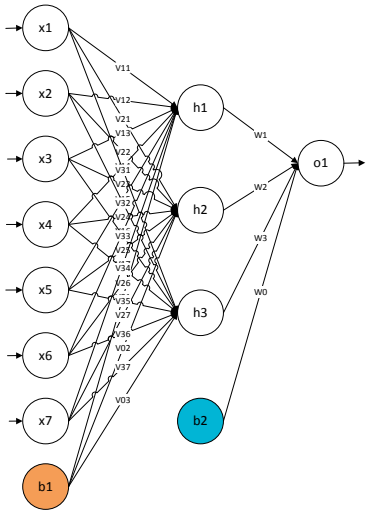


Figure 2. Arcitecture *neural network*

From the results of the calculations are done using the parameters and values for the input above obtained a result of error output generated. Since the resulting error is still too large of a defined target error of 0.0001 and the output values generated much of the value of targets set then it will be done the learning process by using the backpropagation method to change the weighting of existing networks, the backpropagation calculation process is done next to generate output values approach or equal to the value of the targets set and to minimize the error value of the output produced.

2.4 Testing of C-Fold Cross Validation

Based on the results of the test scenarios 1 to 10 IE testing as many as 330 test data obtained from 10 times a game done by the player.

Table 8. Total Data Testing

Data	Total Data
A1	33
A2	33
A3	33
A4	33
A5	33
A6	33
A7	33
A8	33
A9	33
A10	33

Table 9. Scenario testing

Scenario testing	Data testing	Data training
Pengujian 1	A1	A2, A3, A4, A5, A6, A7, A8, A9, A10
Pengujian 2	A2	A1, A3, A4, A5, A6, A7, A8, A9, A10
Pengujian 3	A3	A1, A2, A4, A5, A6, A7, A8, A9, A10
Pengujian 4	A4	A1, A2,A3, A5, A6, A7, A8, A9, A10
Pengujian 5	A5	A1, A2, A3, A4, A6, A7, A8, A9, A10
Pengujian 6	A6	A1, A2, A3, A4, A5, A7, A8, A9, A10
Pengujian 7	A7	A1, A2, A3, A4, A5, A6, A8, A9, A10
Pengujian 8	A8	A1, A2,A3, A4, A5, A6, A7, A9, A10
Pengujian 9	A9	A1, A2, A3, A4, A5, A6, A7, A8,A10
Pengujian 10	A10	A1, A2, A3, A4, A5, A6, A7, A8, A9

Scenario testing is testing the use of parameters number of perceptron on a hidden layer of 1-10. Following are the results of the test are obtained

Table 10. Comparison of the accuracy of the hidden layer

Total hidden layer	Accuracy
1 hidden layer	55.75%
2 hidden layer	74.84%
3 hidden layer	94.54%
4 hidden layer	94.54%
5 hidden layer	93.93%
6 hidden layer	95.45%
7 hidden layer	95.15%
8 hidden layer	96.06%
9 hidden layer	95.45%
10 hidden layer	95.75%

2nd test scenario is testing using different learning rate parameter. Parameter learning rate used in testing the value of 0.1 – 1. Here are the test results obtained

Table 11. Comparison of the accuracy of learning rate

Learning Rate	Accuracy
0.1	94.84%
0.2	94.54%
0.3	95.75%
0.4	96.36%
0.5	96.06%
0.6	95.75%
0.7	95.75%
0.8	95.45%
0.9	95.15%

1	94.84%

3rd test scenario is testing using the parameter values of a different error. Error value parameters used in testing the value of 0.1 – 0.00001. Here are the test results obtained

Table 12. Comparison of the accuracy of the error value

Error Value	Accuracy
0.1	94.84 %
0.01	95.15 %
0.001	96.06 %
0.0001	96.36 %
0.00001	96.66 %

Based on the results of the test scenarios 1 to 10 IE testing as many as 330 test data obtained from 10 times a game done by players, can be drawn the conclusion that the number of perceptron on a hidden layer affects the level of error is generated on a network architecture that is used, in table 53 can be seen the test results cross validation is done, on testing conducted using a combination of parameters 8 perceptron on a hidden layer , the learning rate is 0.4 and the value of error accuracy 96.66% 0.00001, the accuracy value indicates the magnitude of the level of cooperation right from the predictions of the cooperation in the generate model used by.

2.5 Testing Questionnaire

Testing is done by exposing the game before and after neural network backpropagation algorithm implemented after it was given a detailed questionnaire on the respondents. Questionnaire consists of 3 questions and answers using a scale of 1 to 3 which was brought against the respondent, while the results are attached. Here are the questions from the questionnaire:

a) Whether the behavior of the cooperation between the npc enemy backpropagation algorithm was implemented after a look at the game?

b) what is the backpropagation algorithm was implemented after the game becomes more difficult?

c) Do with the behaviour of the cooperation at the npc enemies games be better?

Here are the categories of answers to a questionnaire:

Table 13. Category Answers Questionnaire

Answer Value	Description
3	Ya
2	Sama Saja
1	Tidak

Tabel 14. Score Category Answer

Number	Explanation
0%-33%	Tidak
34%-66%	Sama Saja
67%-100%	Ya

The No. 1 question asked the respondent to produce answers to the following:

Table 15. Processing the first question

Answer Value	Description	Respondents	Total Scor Respondents
2	Ya	10	30
1	Sama Saja	0	0
0	Tidak	0	0
	Jumlah	10	30

The number of items for the whole ideal score is 3 x 10 = 30 (Yes). The number of low scores 1 x 10 = 10 (not). So based on that data then the levels are visible on game cooperation behavior made is (30:30) x 100% = 100%.

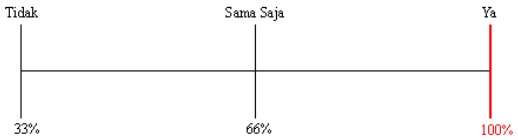


Figure 3. Scale processing of the first question

From question No. 2 raised to the respondents to produce answers to the following:

Table 16. Processing the second question

Category Answers	Description	Respondents	Total Scor Respondents
3	Ya	8	24
2	Sama Saja	1	2
1	Tidak	1	1
	Jumlah	10	27

The number of items for the whole ideal score is 3 x 10 = 30 (Yes). The number of low scores 1 x 10 = 10 (not). So based on the data level whether the game becomes more difficult is (27:30) x 90% = 100%.

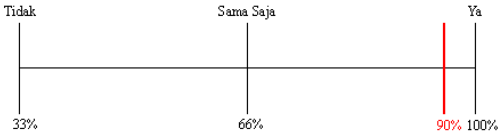


Figure 4. Scale processing of the second question

From question No. 3 posed to respondents to produce answers to the following:

Table 17. Processing third question

Category Answers	Description	Respondents	A Total Score Of Respondents
3	Ya	10	30
2	Sama Saja	0	0
1	Tidak	0	0
	Jumlah	10	30

The number of items for the whole ideal score is 3 x 10 = 30 (Yes). The number of low scores 1 x 10 = 10 (not). So based on the level of data does better with the gaming behaviour of cooperation on a game that is made (30:30) x 100% = 100%.

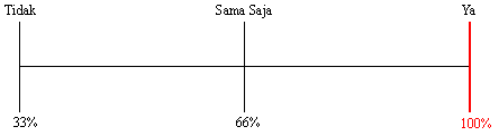


Figure 5. Scale processing third question

Berdasarkah beta test results it can be concluded that the visibility of the cooperation behavior in games that have been implemented by neural network backpropagation algorithm, also with the cooperation behavior in the npc enemies make players more difficult fortunately defeated npc enemies and making the game better.

3. CLOSING

3.1 Conclusion

Based on the results obtained in the research and drafting of this thesis as well as adapted to the purpose, then it can be concluded that:

Implementation of backpropagation neural network algorithm for the behavior of the npc group in game action-rpg can be seen in the behavior of cooperation between npc enemies that are used to predict the decisions that must be taken to choose an appropriate cooperation pattern with training data are used. From the results of research conducted with results didapatkah parameters used 8 perceptron on a hidden layer, the learning rate is 0.4, the value of the limit of error of 0.00001 has the most accuracy than any other network architecture model. The value of the accuracy of this data showed lower are tested the most value approaches the value of the output data training and have a level of accuracy prediction of classification decision of 96.66% also in the results of a detailed questionnaire to 10 respondents had done 100% of respondents stated a game simulation in cooperation has implemented algorithms BPNN can be seen. With pretty good accuracy may be inferred that the npc enemies can work together in accordance with the training data patterns are used and objectively conduct cooperation can be seen, this makes the game better and challenging.

3.2 Advice

Based on the results of the research that has been done, then can be given suggestions for future development are:

1. for the further development of input parameters used can be more diversified in order to adapt the development of the genre action-rpg games today.
2. Implementation of neural network on npc enemies must not be limited to just fight.

BIBLIOGRAPHY

[1] R. P. Kristo, N. H. Rini dan M. M.Azis, “Implementasi Logika Fuzzy Untuk Mengatur Perilaku Musuh dalam Game,” *Jurnal EECCIS*, vol. 7, no. 1, pp. 15-20, Juni 2013.T. Callahan, J. Hauser, and J. Wawrzynek, “The Garp Architecture and C Compiler,” Computer, vol. 33, no. 4, pp. 62-69, 2000.

[2] Kemdikbud, "Kamus Besar Bahasa Indonesia (KBBI)," [Online]. Available: http://kbbi.web.id/kerja. [Accessed 5 Januari 2017].

[3] J. A. Freeman and D. M. Skapura, Neural Networks, Houston: Addison-Wesley Publishing Company, 1991.

[4] J. Heaton, "Heaton Research," April 2012. [Online]. Available: http://www.heatonresearch.com/book/introduction-neural-network-math.html. [Accessed November 2016].

[5] J. Schneider, "Cross Validation," 1997. [Online]. Available: https://www.cs.cmu.edu/~schneide/tut5/node42.html.

[6] T. Entertainment, "Mount & Blade : Warband," Paradox Interactive, 2010.