การจำแนกการกระทำของตัวละครในโปรแกรมจำลองเกมเล่นตามบทบาทโดยใช้โครงข่ายประสาทแบบแพร่กระจายย้อนกลับที่คืนสภาพได้

นายปิยชัย เอี่ยมสุขวัฒน์

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต

สาขาวิชาวิทยาการคอมพิวเตอร์และสารสนเทศ ภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์

คณะวิทยาศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย

ปีการศึกษา 2555

ลิขสิทธิ์ของจุฬาลงกรณ์มหาวิทยาลัย

CLASSIFYING CHARACTER'S ACTION IN ROLE-PLAYING GAME SIMULATION USING RESILIENT BACKPROPAGATION NEURAL NETWORK

Mr.Piyachai Eamsukawat

A Thesis Submitted in Partial Fulfillment of the Requirements

for the Degree of Master of Science Program in Computer Science and Information

Department of Mathematics and Computer Science

Faculty of Science

Chulalongkorn University

Academic Year 2012

ปิยชัย เอี่ยมสุขวัฒน์ : การจำแนกการกระทำของตัวละครในโปรแกรมจำลองเกมเล่นตามบทบาทโดยใช้โครงข่ายประสาทแบบแพร่กระจายย้อนกลับที่คืนสภาพได้. (CLASSIFYING CHARACTER’S ACTION IN ROLE-PLAYING GAME SIMULATION USING RESILIENT BACKPROPAGATION NEURAL NETWORK) อ.ที่ปรึกษาวิทยานิพนธ์หลัก : ผศ.ดร.ศรันญา มณีโรจน์, อ.ที่ปรึกษาวิทยานิพนธ์ร่วม : ผศ.ดร.ศุภกานต์ พิมลธเรศ, 63 หน้า.

งานวิจัยที่ใช้การเรียนรู้ของเครื่องจักรกลเพื่อสร้างเนื้อหาสาระที่ออกแบบใหม่ในเกมคอมพิวเตอร์มีอยู่มากมาย สิ่งที่ท้าทายคือการจำแนกการกระทำของตัวละครโดยใช้การเรียนรู้ของเครื่องจักรกลเพราะสามารถนำไปใช้ในเกมคอมพิวเตอร์โดยตรงและปรับปรุงการเรียนรู้ของตัวละครเกี่ยวกับการใช้กลยุทธ์ภายใต้สภาพการณ์ที่แตกต่างกันของเกม สิ่งนี้ทำให้เกมน่าตื่นเต้นมากขึ้น ต้นไม้การตัดสินใจเร็วมาก สามารถจำแนกการกระทำของตัวละครในโปรแกรมจำลองเกมเล่นตามบทบาทคอมพิวเตอร์ แต่ความแม่นยำนั้นไม่เพิ่มขึ้นมากเมื่อจำนวนการกระทำของตัวละครเพิ่มมากขึ้น ในงานวิจัยนี้โครงข่ายประสาทแบบแพร่กระจายย้อนกลับที่คืนสภาพได้ถูกใช้เพื่อจำแนกการกระทำของตัวละครในโปรแกรมจำลองเกมเล่นตามบทบาทคอมพิวเตอร์และเปรียบเทียบความแม่นยำกับต้นไม้ความคิดไวมาก กลยุทธ์คงทีและกลยุทธ์ที่เปลี่ยนแปลงได้ถูกทดสอบในการทดลองเหล่านี้ ผลการทดลองแสดงให้เห็นว่าข้อมูลฝึกฝนจำนวนมากตามข้อมูลเกมคอมพิวเตอร์ที่สอดคล้องกันนั้นวิธีการที่เสนอนี้ทำงานได้ประสิทธิภาพดีกว่าวิธีการที่มีอยู่ โครงข่ายประสาทแบบแพร่กระจายย้อนกลับที่คืนสภาพได้สามารถถูกออกแบบให้ใช้ในเกมคอมพิวเตอร์เพื่อลดความซับซ้อนในการเขียนข้อกำหนดทางโปรแกรมและเพิ่มความสนุกในคอมพิวเตอร์เกมโดยเพิ่มทางเลือกให้กับผู้เล่น

ภาควิชา คณิตศาสตร์และวิทยาการคอมพิวเตอร์ ลายมือชื่อนิสิต..........

สาขาวิชา วิทยาการคอมพิวเตอร์ และเทคโนโลยีสารสนเทศ ลายมือชื่อ อ.ที่ปรึกษาวิทยานิพนธ์หลัก..........

ปีการศึกษา 2555 ลายมือชื่อ อ.ที่ปรึกษาวิทยานิพนธ์ร่วม..........

5472629423 : MAJOR COMPUTER SCIENCE AND INFORMATION TECHNOLOGY
KEYWORDS : RESILIENT BACKPROPAGATION/RESILIENT PROPAGATION (RPROP)/ NEURAL NETWORK/ARTIFICIAL INTELLIGENCE (AI)/GAME/COMPUTER ROLE-PLAYING GAME (CRPG)/PLAYER MODELING/STRATEGY/VERY FAST DECISION TREE (VFDT)/ MACHINE LEARNING(ML)

PIYACHAI EAMSUKAWAT : CLASSIFYING CHARACTER'S ACTION IN ROLE-PLAYING GAME SIMULATION USING RESILIENT BACKPROPAGATION NEURAL NETWORK. ADVISOR : ASST. PROF. SARANYA MANEEROJ, Ph.D., CO-ADVISOR : ASST. PROF. SUPHAKANT PHIMOLTARES, Ph.D., 63 pp.

There are many researches using Machine Learning (ML) to create new design contents in computer game. The challenging task is to classify game character's action using ML because it can be straightforwardly implemented in the game, thereby enhancing character learning about how to deploy strategies under different game situations. This makes the game more exciting. Very Fast Decision Tree (VFDT) can classify character's actions in computer role-playing game (CRPG) simulation but the accuracy is not much improved when the number of character's actions is increased. In this research, Resilient Backpropagation (RPROP) can improve such accuracy when the character's actions increase, so RPROP is implemented to classify character's action in the CRPG simulation and compared the accuracy with VFDT. The static strategies and the changing strategies are tested in these experiments. The results show that at the high number of training data corresponding to the computer game data, the proposed scheme performs better than the existing method. RPROP can be designed to use in computer game to decrease the complexity of programming script and improve the excitement of the computer game by giving the player more alternatives.

Department : Mathematics	Student's Signature
Field of Study : Computer Science and Information Technology	Advisor's Signature
Academic Year : 2012	Co-advisor's Signature

ACKNOWLEDGEMENTS

I would like to acknowledge my advisor, Assistant Professor, Saranya Maneeroj, Ph. D. and my co-advisor Assistant Professor, Suphakant Phimoltares, Ph. D., at The Advanced Virtual and Intelligent Center (AVIC) for their support. In fact, this thesis is never going to complete without them. They always give me the new bright ideas as much as I need and also alert me, every time I am careless.

I also would like to acknowledge R. M. M. Vallim and J. Gama for giving me with the data sets which are very benefit for this research. The data sets are very important. By receiving them instead of generating new data sets, save a lot of researching time.

I would like to appreciate Ms. Nonglak Puttacharoen, faculty staff who always gives the very important information and help me solve a lot of problems as my another advisor.

My gratitude has to belong to my seniors, Ph.D. student at AVIC. They generously teach me a lot of knowledge and kindly take care of me.

Thank you my parents for my financial support and their kindness together with my family. They always back me up with their love.

CONTENTS

	Page
ABSTRACT IN THAI	iv
ABSTRACT IN ENGLISH	v
ACKNOWLEDGEMENTS	vi
CONTENTS	vii
LIST OF TABLES	ix
LIST OF FIGURES	x
CHAPTER I INTRODUCTION	1
1.1 Objectives	6
1.2 Scope	6
1.3 Research methodology	6
1.4 Benefits	7
CHAPTER II THEORITICAL BACKGROUND	10
2.1 Multilayer Feed Forward Artificial Neural Network	10
2.2 Decision Tree	12
2.3 Reinforcement Learning	13
2.4 Greedy Algorithms	14
CHAPTER III RELATED WORKS	15
3.1 Dyna-H	16
3.2 Action-dependent Learning Rates with Trends (ALeRT)	20
3.3 Cross Entropy Method (CEM)	24
3.4 Very Fast Decision Tree (VFDT)	26
3.5 Resilient Backpropagation (RPROP)	33

Page

CHAPTER IV PROPOSED METHOD 35

4.1 Database of Character's Action 35

4.2 Learning Strategies 41

4.2.1 Learning Static Strategies 42

4.2.2 Learning Changing Strategies 44

CHAPTER V RESULTS 48

5.1 Results of Learning Static Strategies 48

5.2 Results of Learning Changing Strategies 52

CHAPTER VI DISCUSSION 56

6.1 Discussion of Learning Static Strategies 56

6.2 Discussion of Learning Changing Strategies 57

CHAPTER VII CONCLUSION 59

REFERENCES 61

BIOGRAPHY 63

LIST OF TABLES

Table Page

1.1 Task schedule 8

4.1.1 Examples of offensive fighter database 38

4.1.2 Examples of defensive fighter database 39

4.1.3 Examples of offensive wizard database 40

4.1.4 Examples of defensive wizard database 41

4.2 Pattern of the changing strategies training 46

5.1.1 Comparison between RPROP and VFDT for offensive fighter 49

5.1.2 Comparison between RPROP and VFDT for defensive fighter 50

5.1.3 Comparison between RPROP and VFDT for offensive wizard 51

5.1.4 Comparison between RPROP and VFDT for defensive wizard 52

5.2.1 Comparison between RPROP and VFDT for Changing Strategies Fighter by starting from Offensive Fighter 54

5.2.2 Comparison between RPROP and VFDT for Changing Strategies Fighter by starting from Defensive Fighter 55

6.1 Examples of offensive fighter database 56

LIST OF FIGURES

Figure Page

1.1 Spronck's Minigate environment 2007 3

3.1 diagram of multilayer feed forward artificial neural network 11

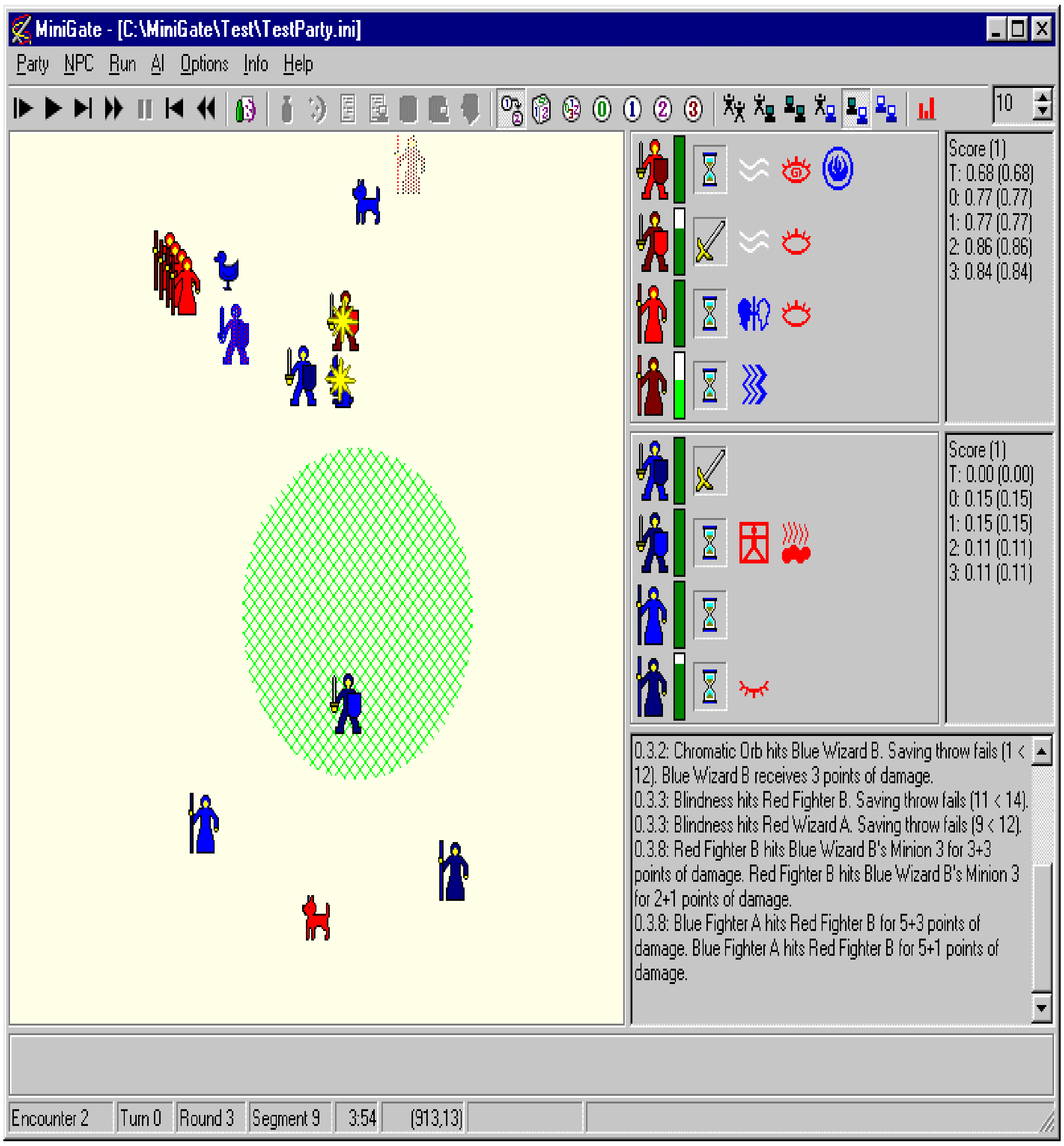
3.2 diagram of decision tree 13

CHAPTER I

INTRODUCTION

Computer Role Playing Game (CRPG) is the type of computer game. Player has to play as a role of a main character or a group of main characters. CRPG aims to present the story of the characters – such as fairy tales, historical stories, or fantasy stories. CRPG normally is turn-based Gameplay which is a playing style in the computer game. About the turn-based Gameplay, two or more characters fight in two teams – one team is controlled by the player and the other is controlled by computer. The characters have their own turn. The player can play by selecting character's action when the characters in the player team have their turn. In the opposite way, some developers want to create different CRPG to present in the computer game market, so CRPG also has the other type of Gameplay – such as side-scrolling action style or first person shooting style. The simple thing to do in CRPG over fight, is questing. The questing can be anything which the computer game can do; for example, the game asks the player some questions or gives player some puzzle games to solve. Machado et al. (2011) stated the objectives in CRPG are fighting, questing, finding items and leveling – leveling is the content that makes player's character stronger by fighting and questing. Finding items can be the objective but it is always optional objective and leveling is reward of fighting and questing. The finding items and leveling are important because the player's characters have to be strong enough to fight bosses who are stronger than normal enemies. With good strategies, the boss fight will be easier. CRPG allows the player to decide to use some different strategies. The strategy planning is the exciting part of CRPG because the player involves with a lot of decisions, so the player has choices which sometimes bad strategies can satisfy the player more than the better strategies. And by choosing the choice, the player also involves the story more too.

In this thesis, CRPG is used to generate character's action. The CRPG is Spronck's Minigate environment (2007) which is the simulation of fighting in CRPG with turn-based table style – also called tactical CRPG. The difference from turn-based CRPG is the battle field. The turn-based CRPG does not design the distance on the battle field and all the field areas are the same area for every characters on the field. But the tactical CRPG has the field that separates into many areas like hexes. Every hex has its own type of field that can be different from others, so some characters can be on the grass field when some of them are on the dessert field but actually they are on the same battle field. The different areas can have different effects which give the benefit for the characters and the player has to plan the strategies involving with the hexes on battle field as well. However in Minigate environment, every area is the same, so this is not important. The other thing about the tactical CRPG's battle field is the distance. In turn-based CRPG, the distance is not calculated but in tactical CRPG, there are distances in the battle field. The character can move and the character's actions have range limit. In Minigate environment, two teams of four characters have to fight each other till death. The survivor team is the winner. In each team, there are two fighter characters and two wizard characters. The fighter has strong melee attack and a lot of hit point but wizard has a lot of tricky spells, so the common strategy is using the fighter attack while keep enemy busy with the wizard's spells. The other strategy is using the range attack or spells to eliminate the wizard enemy first because the wizard has not much hit point. However, there is no best strategy – that makes the balance in the game. The other key to victory in this game is randomness. There are many random elements which are the hit rate of the attack, the effect of spell and then potions and wizard's spell can become random set too, if the rule setup for it to be random sets. In this way, the strategies have to adapt to these random elements which are randomly occur during the game too, so the static strategy is not going to be the best.


MiniGate - [C:\MiniGate\Test\TestParty.ini]
Party NPC Run AI Options Info Help
10

```
Score (1)
T: 0.68 (0.68)
0: 0.77 (0.77)
1: 0.77 (0.77)
2: 0.86 (0.86)
3: 0.84 (0.84)

Score (1)
T: 0.00 (0.00)
0: 0.15 (0.15)
1: 0.15 (0.15)
2: 0.11 (0.11)
3: 0.11 (0.11)

0.3.2: Chromatic Orb hits Blue Wizard B. Saving throw fails (1 <
12). Blue Wizard B receives 3 points of damage.
0.3.3: Blindness hits Red Fighter B. Saving throw fails (11 < 14).
0.3.3: Blindness hits Red Wizard A. Saving throw fails (9 < 12).
0.3.8: Red Fighter B hits Blue Wizard B's Minion 3 for 3+3
points of damage. Red Fighter B hits Blue Wizard B's Minion 3
for 2+1 points of damage.
0.3.8: Blue Fighter A hits Red Fighter B for 5+3 points of
damage. Blue Fighter A hits Red Fighter B for 5+1 points of
damage.

Encounter 2 | Turn 0 | Round 3 | Segment 9 | 3:54 | (913,13)
```

Figure 1.1 : Spronck's Minigate environment 2007

There are a lot of computer games nowadays. Every computer game company wants to create a new product and get more customers which can create the large competitive market. In order to survive in the market, a new computer game must have some new contents. In this way, a lot of computer researchers also want to develop the new contents. In this research area, Machine Learning (ML) is now very popular because there are a lot of ML techniques to use with many different types of computer games. The types of computer game can be action game or real-time strategy game which has a lot of actions in every second. However, in some types of computer game like turn-based game, there is some amount of time when the game stops and waits for players to decide their action. In these types, there are different in computational time for ML techniques because in the action game or the real-time strategy game, the game time cannot stop until the game ends but in the turn-based game, the game time stops in a lot of phases. And the ML techniques have different computational time, so the computer researcher has to consider the suitable ML techniques for the type of computer game. And then, there are some different ideas to use ML in the computer game. R. Lopes el at. 2011 separated the ideas into five adaptable contents which include game worlds, Gameplay mechanics, Non-playing characters (NPC) and Artificial Intelligent (AI), game narratives, and game scenarios (include quests). In these contents, some adaptive ideas can be about personal preference or about playing experience. This thesis uses the adaptive AI content to adapt with the playing experience. And to use the combination, there is a model of playing which is called player modeling. When people have some experience in the game, they will create their sets of strategies. And when the sets of strategies come with specific environment in the game, they can be classified into types of playing strategy which is the player modeling. The player modeling alone cannot do anything because it is just the set of strategy without someone using it, so the computer game must have some functions to adapt to it which are the five adaptable contents.

In the traditional of game developing, CRPG always creates for single player mode – a player plays against the computer AI. The computer game character which is played by the computer AI is called non player character (NPC) (S. Yildirim et al. 2009). The computer AI is a programming language and writes by a lot of rule-based script – one type of static AI. For examples, in this thesis, the character's actions were generated by using rule-based script of R. Vallim el at. (2010), such as; if the hit point is lower than half and have potion of the hit point, use drink potion of the hit point action – the rests are showed in Chapter 3.4. Very Fast Decision Tree. However, the static AI has problems. First, it is easy to recognize by human because it is created by simple rules and never changes. Second, it always has weakness. In the previous example when the hit point is lower than half with the potion, the computer AI will use the potion, so if player make its hit point drop to just a half and then kill it in next turn, it will never have a change to use the potion. The example shows that the weakness of the static AI makes the computer AI not good enough in the game. Then third, it is easy for human to predict what it is going to do. After its script is understood; its weakness is also found and then every action of its can be predicted. Finally, the rule-based script become more complicated when the computer AI has to be developed more in order to be good enough to play with human. All four problems of the static AI are presented by I. Szita et al. (2009). Now a lot of researchers present using ML instead to solve all of the problems of static AI and the computer AI with ML is called adaptive AI. There are some works of the researchers with the adaptive AI which are showed in the next section.

In the researching area, R. Vallim et al. (2010) have used Very Fast Decision Tree (VFDT) to classify character's action in Minigate environment. VFDT can predict the character's action in some quality accuracy but when there are more number of the data, the accuracy raise slowly because VFDT use information gain and entropy which are not change more when the number of the data increase. Read more details of VFDT in the next section.

The other one of ML is artificial neural network (ANN). ANN has been used for classification and clustering purpose. The idea of ANN is to translate input into output in the way that human's neurons use its signals. By using all of input to compute the accuracy, ANN can classify the character's action in some amount of number better than VFDT In ANN area, there are two problems which are; first, it takes too long to process learning time which is how ANN improves the translation to be more accurate and second, how to improve the accuracy. So M. Riedmiller et al. (1992) has presented Resilient Backpropagation (RPROP) to make the learning time process faster. However, RPROP is not only faster but it also improves the accuracy of ANN in some cases when the decreasing factor allow the ANN to find the highest accuracy when the slope between error percentages and the changing weight value is very high. The more details of ANN and RPROP are stated in the next section. In this thesis, RPROP is used to solve the slow improvement of accuracy when use VFDT to classify the increasing number of character's actions.

1.1. Objectives

In this study, the main objective is to classify character's action of computer role playing game with high accuracy by using neural network.

1.2. Scope

In this study, the identification system is constrained as follows:

1.2.1. Using RPROP neural network with three layers and total of five hidden nodes to classify the character's actions.

1.2.2. Using the character's actions which were generated by R. Vallim el at. (2010).

1.2.2. Comparing VFDT and neural network in classifying the character's action

1.3. Research Methodology

In the procession, there are several tasks as following:

- Literature review: In order to understand this area of researching, a lot of paper must be review. The importance key is about understanding the trend of using ML in the computer game. Start review survey paper, understand meaning of technical words and then focus on some topics.
- Collect data: The technical term of the research must be used to test the methodology. So data are needed for the experiment. At the beginning, the data are generated but the problem is the data preparation was described in the reference unclearly. So asking for help from the other researchers is a best way to go around the problem. By sending Email to R. Vallim and J. Gama for help,

the data were given kindly. Observe VFDT: The replication of the reference experiment was setup. In this time, the data were the problem, because by using Massive Online Analysis (“MOA” is an application that performs VFDT with the data.), the calculation could not be observed, so the problem was how the data were calculated. To find the answer, the MOA’s programming script must be understood.

- Test ANN with data: For ANN, it cannot use MOA to do the job. So other application has to be prepared to test the data. The formation of the data has to be changed in order to use with this application.
- Configuration of ANN: Because ANN can be used with any data, to make the ANN becomes more specific to these data, some configurations have to be tested. And in this process, RPROP had been decided to be used as ANN that was fit for the data.
- Collect and discuss the results: This process did not take too long because RPROP was good in time consumption, so the taken time came from some discussions and tried to improve the results.
- Report the experiment: This thesis and conference paper was written. This is very important because the proposed scheme and the experiment must be published in order to give the knowledge in another point of view.

Table 1.1 : Task schedule

Step	Tasks	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
1	Literature review	■	■	■	■	■	■												
2	Collect data							■	■										
3	Observe very fast decision tree method									■	■	■							
4	Test ANN with data												■						
5	Configuration of ANN													■	■	■			
6	Collect and discuss the results																■		
7	Report the experiment																	■	■

1.4. Benefits

There are two benefits in this thesis. First, some ML techniques can be used to create new contents in the computer game. In this thesis, RPROP was used in CRPG to classify the character's action and the reason to use RPROP because it can classify the character's action better than VFDT when the total number of the action increase too some amount of numbers. It is shown that the way to use ML in the computer game has to use ML that is compatible.

The second benefit is the example of using the computer game in the computer researching. Actually, there are many researches using some computer games or physical games to explain about social, psychological or teaching; however, using computer game is able to generate a lot of data by using only a few computers and there are a lot of computer games to use too. Other than using new algorithms to create new contents in the game, some computer games can be used to test the new algorithms.

CHAPTER II

THEORITICAL BACKGROUND

As mention in previous section, the proposed method is using RPROP to classify the character's action in CRPG. RPROP is a multilayer feed forward ANN, so in the section Multilayer Feed Forward ANN is explained with the other important algorithms which are decision tree, reinforcement learning, and greedy algorithm.

2.1. Multilayer Feed Forward Artificial Neural Network

Basic of ANN come from linear function.

$$y = a_1x_1 + a_2x_2 + a_3x_3 + ... + a_nx_n + b \qquad (1)$$

And then when there are more than one output (y), the formula moves to matrix.

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ ... \\ y_m \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & ... & a_{1n} \\ a_{21} & a_{22} & a_{23} & ... & a_{2n} \\ a_{31} & a_{32} & a_{33} & ... & a_{3n} \\ ... & ... & ... & ... & ... \\ a_{m1} & a_{m2} & a_{m3} & ... & a_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ ... \\ x_n \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ ... \\ b_m \end{bmatrix} \qquad (2)$$

The inputs $(x_1, x_2, x_3, ..., x_n)$ are fed to the equation. Where x is input; y is output; a and b are constant values. There is a diagram of multilayer feed forward artificial neural network is shown as following.

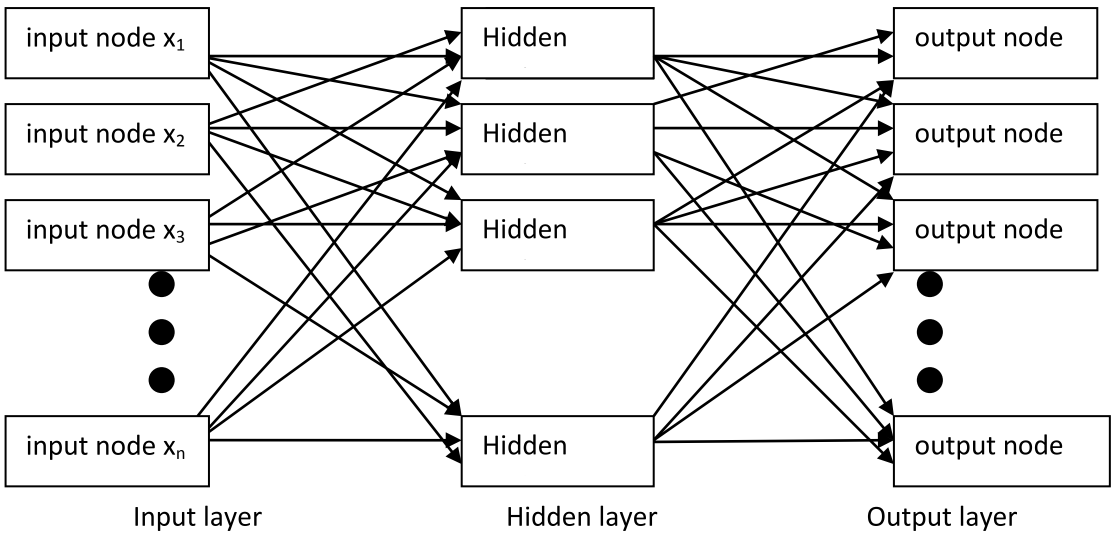


Figure 3.1 diagram of multilayer feed forward artificial neural network

In order to get output, input is needed for calculating with something in the middle way. As in equation (2), there are a and b to use. In ANN, a is "weight" and b is "bias". The data is formed to be layers because the data come from the same sources as levels of layers. The input nodes are in the input layer. The output nodes are also in the output layer. Furthermore, there are hidden nodes between input node and output node in Figure 3.1. Each hidden node has its a and its b. Each hidden node calculates its outputs separately and sends the outputs to output nodes. The hidden node can be more than one layer. If there are two or more than two hidden layers, the hidden node's outputs will be sent to the hidden nodes in the next hidden layers instead of the output nodes and the last hidden layer will send the hidden node's outputs to the output node. The output nodes have to calculate the outputs from the hidden nodes as their inputs. The output node also has its a and its b to calculate the outputs. And then the way of the arrows have only direction forward to the output. It is called a feed forward network – there is no backward error signal feed back to the previous layer. The error is also defined by mean of square error or sum square error:

Mean Square Error: $MSE = \frac{1}{N}\left((y'_1 - y_1)^2 + (y'_2 - y_2)^2 + (y'_3 - y_3)^2 + \ldots + (y'_N - y_N)^2\right)$

Sum Square Error: $SSE = (y'_1 - y_1)^2 + (y'_2 - y_2)^2 + (y'_3 - y_3)^2 + ... + (y'_N - y_N)^2$

Where N is total number of output (y); y' is the network output; y is the class of the output.

And then the error is used to adjust the weight by: $w_{ij}(t+1) = w_{ij}(t) - \varepsilon \frac{\partial E}{\partial w_{ij}}(t)$ where $w_{ij}(t)$ is old weight; $w_{ij}(t+1)$ is new weight; E is the error; ε is learning rate.

2.2. Decision Tree

Some of Machine Learning (ML) techniques are developed by using decision tree. Decision tree is the data structure which stores the data in the set – call node. The nodes have the connection that can track and link themselves like a tree. Decision tree has level of the tree that is defined by one node at the lowest level of the tree which is a root node and the nodes at the highest level of the tree are leaf nodes. The connections between the nodes call branches. The node that has a branch link to the node on the lower level of the tree call parent node. The node which has a parent call child node. The nodes have relation parent and child when they are linked together with a branch. The nodes have the grand child and grandparent relation when its child node has its child node and the other one is child of the child node of the child node. The node without any child node is also the leaf node. The nodes which are not the root node or the leaf node are the interval nodes. There are types of decision tree. If the tree has no more than two children in every node, it is called binary tree. If the tree has random children on some nodes, it is multiple tree.

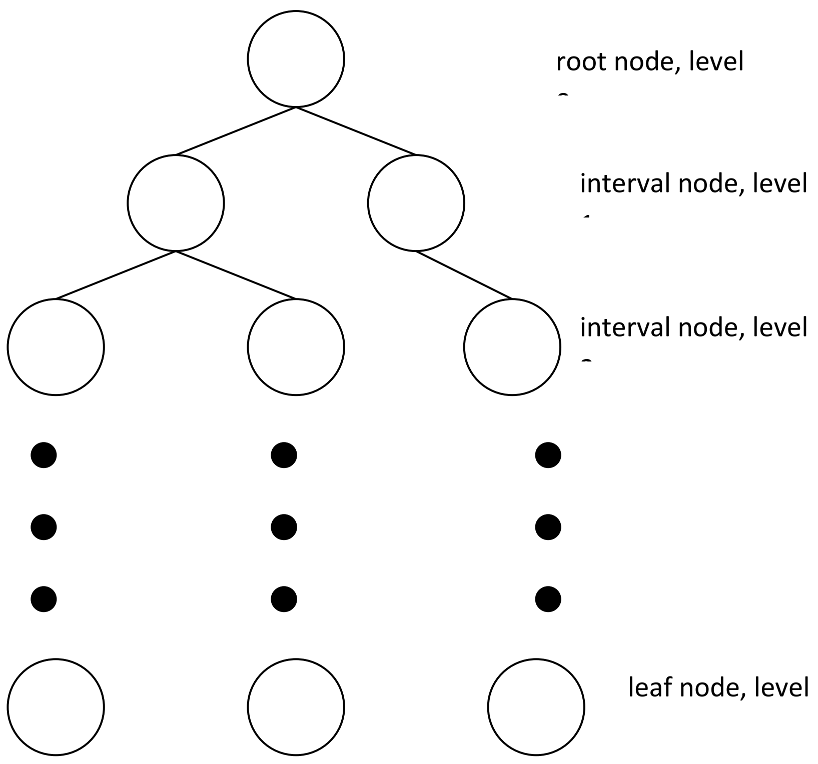


figure 3.2 diagram of decision tree

Where n is the highest level of the tree.

2.3 Reinforcement Learning

In this thesis, a lot of related works are used MLs which are reinforcement learning (RL). RL is a learning technique that uses a fitness function to adjust the main functions. The fitness function is the function that generates the reward. First, the main functions have to generate the output from the input that feed to them. Then the fitness function will give the reward according to the output. The reward is a score that can be positive or negative score. If the reward is a negative score, the main functions have to change in order to change the output and get a positive score. So when the main functions make more score, it means the learning algorithms have been trained the function successfully.

2.4. Greedy Algorithm

Greedy algorithm is the optimal selection. This algorithm is used to select the highest value from the data. However, in this thesis, there is greedy algorithm that is a different greedy algorithm. It is epsilon-greedy algorithm (ε-greedy algorithm). ε-greedy algorithm is reinforcement learning. ε-greedy algorithm selects the highest value from the data by probability $1-\varepsilon$. The ε can be changed by using the learning experience and fitness function.

CHAPTER III

RELATED WORKS

In computer game commercial nowadays, there are a lot of similar games with the same styles and contents. There are a lot of people who still buy new games which are similar to the old ones, so the computer game developers do not want to develop any new style game or create different content which cost a lot of money and have a risk for the popularity. People, who play the game as a player, are suffering from the business plan. Because the player wants to play new computer games but the new ones always are similar to the old ones. And then the game developer knows that the old style still make money, so there is no need to develop the new style; like a cycle.

In the past century, the most developing in computer game was game graphic and graphic art. A lot of games had series and a lot of them were developed mostly in the graphic. In addition to the computer market, the graphic card was developed so fast too. However, the graphic is now developed to the satisfy quality. It has some room to develop but it should not be the important content in the game anymore. A lot of players and game reviewers (the player who discuss about the content in the game; has to have a lot of experience on a lot of different games) always say that the computer game is good or bad, not come from the graphic but its Gameplay. Somehow graphic has to satisfy the player. Too low and outdate graphic game always has bad comments.

The important content in the computer game is Gameplay. Gameplay is the way that player plays the game; include the style of game such as First Person Shooting (FPS) game, Real Time Strategy (RTS) game, Turn-based Strategy game, Puzzle game, Fighting game, Sport game, Racing game and Role-playing game (RPG). In these different types of Gameplay, there are different ways to add or improve the contents of the game. The Gameplay is the important content because it is how the player plays the game. If the Gameplay focus on the wrong way, it can make the player bored and confused. For example, the RPG with a slow pace Gameplay is very bored; even, it has a fantastic graphic and wonderful story, it is still too bored to play. Because Gameplay is what player has to do for the most of the time in the game, playing with the bad Gameplay is not acceptable at all.

RPG is the story-based game. It has a long story which is told to the player along with the Gameplay. Tropical RPG always has a story about a hero or a group of heroes who have to save the world. The hero might be a kid or someone who have amnesia, so the person does not know anything about the world. Then the story is going to tell about the world to the hero and the player knows everything as the hero was told. The Gameplay always is turn-based strategy which the player choose the character's action in order to beat the other player. If it is single player game, the other player is AI. In this thesis, RPG is called Computer Role-playing game (CRPG) to avoid misunderstanding because Role-playing game in education and psychology is mean the game which people have to act as their roles and then the researchers record what the people do in the situation. There are some types of Gameplay in CRPG. The developers can mix CRPG with every other Gameplay. So instead of turn-based strategy, it can be fighting, FPS, RTS or the other types. Because CRPG has quite a few types of Gameplay, some literature researches about CRPG have to be reviewed.

Nowadays every computer game's contents have been developed so far but the contents which are still able to developed, are the adaptable contents. According to a research of R. Lopes el at. (2011), there are five adaptable contents – game world, Gameplay mechanics, NPC and AI, game narratives and game scenarios (include quests). These contents had been integrated with ML in some different ways. ML are proposed in a lot of researches which use ML for adapt the contents in the computer game. And to understand the research in this area, these following researches about CRPG have been reviewed in this section.

3.1. Dyna-H

CRPG was tested with different ML. M. Santos et al. (2012) using path finding in CRPG to demonstrate Dyna-H algorithm. In an environment where the path has a lot of obstructions, AI has to find the shortest path, by using ML. In this experiment, Dyna-H, Dyna-Q and Q-learning are the examined techniques.

Dyna architecture is a reinforcement learning which uses the experience to improve the learning model – the model is the classification of the learning. It looks like normal reinforcement learning; however, it has the direct learning that has the condition when the learning experience is not good enough to create a new model yet, so it can skip to create the poor model and learn from more experience before it can create the good model.

In Santos' work, Dyna architecture has already present that it can combine with Q-learning or heuristic planning. When it combines with Q-learning, it is Dyna-Q – presented by R. Sutton el at. 1991. First, Q-learning algorithm is explained and then Dyna-Q. Q-learning algorithm start with Q-table. It has state which is a situation where the AI are in and action which is the option to choose in the situation. There are always more than one action to choose and every action will bring the AI into a new state. In the beginning of the learning, there are all of possible states and actions in the Q-table. Every actions have to have their own state – $Q(s_x, a_x)$ where s is state, a is action and x is assumed to define any state or any action in the Q-table. And every action has the initial score equal to 0. In addition, there are some fitness functions to update the scores.

Algorithm 3.1.1 : Q-learning

1) Initiate Q-table which all states and actions have scores equal to 0 : $Q(s,a)=0$.

2) Use ε-greedy to randomly select the starting state.

3) In the AI's state, select the action in the state with highest score from the Q-table. If there is more than one, select randomly.

4) Do the selected action and use the fitness functions to update the Q-table.

5) Repeat 3) and 4) until the AI's state is the ending state.

6) Repeat 2) to 5), if the learning is going on to the next round.

The ending state has to be notified in the learning condition. Normally in the path finding, the ending state is the exit or the destination and the action is moving up, down, left or right. However P. Patel el at. (2011) presented using Q-table in First Person Shooting game (FPS game) which uses death of the character as the ending state.

There are the Q-table combine with Dyna architecture – Dyna-Q. The algorithm does not allow the learning to skip to create the model; however, the model is used to select the next state; allows the ML to predict the reward of the action on the previous state before select the next action.

Algorithms 3.1.2 : Dyna-Q

1) Initiate Q-table which all states and actions have scores equal to 0 : $Q(s,a)=0$. And Model which all states and actions have scores equal to 0 : $M(s,a)=0$.

2) Use ε -greedy to randomly select the starting state.

3) In the AI's state, select the action in the state with highest score from the Q-table. If there is more than one, select randomly.

4) Do the selected action and use the fitness functions to update the Q-table. And update the Model. $M(s,a)=reward$, where $reward$ is the score from the fitness functions.

5) Select the previous state randomly by using ε -greedy.

6) Select the previous action randomly by using ε -greedy.

7) Use the model to calculate the reward of the action that selected in 6) and update the score in Q-table.

8) Repeat 5) to 7) until N time; where N is the select number that is selected by user.

9) Repeat 2) to 8), if the learning is going on to the next round.

In the last, for Dyna-H, the heuristic function (H) is calculated to find the worst action. By finding the longest distance in the game, H can find the worst action by these equations.

$$H(s,a) = \| s' - goal \|^2 \qquad (1)$$

$$h_a(a,H) = \arg\max H(s,a) \qquad (2)$$

where $H(s,a)$ is heuristic function of action a at state s; s' is the next state after do action a; goal is the last state; $h_a(a,H)$ is the worst action in all states of $H(s,a)$.

H can find the worst action because $\| s' - goal \|$ has the highest value when s' is the state that has the longest distance from the goal. If the action is movement that moves away from the goal, the action is the worst. Algorithm of Dyna-H is stated as following:

Algorithms 3.1.3 : Dyna-H

1) Initiate Q-table which all states and actions have scores equal to 0 : $Q(s,a) = 0$. And Model which all states and actions have scores equal to 0 : $M(s,a) = 0$.

2) Use ε-greedy to randomly select the starting state.

3) In the AI's state, select the action in the state with the highest score from the Q-table. If there is more than one, select randomly.

4) Do the selected action and use the fitness functions to update the Q-table. And update the Model. $M(s,a) = reward$, where $reward$ is the score from the fitness function.

5) Select the worst action by H.

$$H(s,a) = \| s' - goal \|^2 \qquad (1)$$

$$h_a(a,H) = \arg\max H(s,a) \qquad (2)$$

where $H(s,a)$ is heuristic function of action a at state s; s' is the next state after do action a; goal is the last state; $h_a(a,H)$ is the worst action in all states of $H(s,a)$.

6) If the action that selected in 3) is not the worst action that selected in 5) then select state and action randomly by using ε -greedy.

7) Use the model to calculate the reward of the action that selected in 6) and update the score in Q-table.

8) Repeat 5) to 7) until N time; where N is the select number that is selected by user.

9) Repeat 2) to 8), if the learning is going on to the next round.

The results of Dyna-H are better than either Q-learning or Dyna-Q. Because of the model, Dyna-Q can get more scores and can update them into Q-table, so Dyna-Q has learned more than Q-learning in the same total amount of the learning rounds. And then Dyna-H also learns more than Dyna-Q in the same amoun of rounds too. However, in each round, Dyna-H spends more resources on the process than Dyna-Q and Dyna-Q is more than Q-table. So it depends on how the game is decided to use the ML. If the resources are low, Q-table is a good ML but if the resources are enough, using Dyna-H or Dyna-Q to use the resource should be better. Q-learning generates a random action to calculate the updating Q-table and then it selects the best score action from the table. In contrast, Dyna-H calculates all possible actions to calculate the table. When compare with Q-learning which always chooses the same path that it knows best, it is difficult for it to choose a new path. In these results, Dyna-H can select the best action in less learning rounds than Q-learning and Dyna-Q. After that it has much better results than the others. As mentioned in the previous chapter, some CRPGs have the battle field for the Gameplay mechanic. So CRPG should use the MLs to improve their AI's path finding.

3.2. ALeRT

M. Cutumisu et al. (2008) presented Action-dependent Learning Rates with Trends (ALeRT) algorithm which was adapted from State-Action-Reward-State-Action (Sarsa) algorithm. Sarsa is an algorithm that is similar to Q-learning. For Q-learning, the first action is selected randomly and then it is not recorded that it is the first action.

However, for Sarsa, the first action is still the same that is selected randomly but it is recorded and then in every round, the first action is this action. There is the algorithm of Sarsa.

Algorithms 3.2.1 : Sarsa Algorithm

1) Initiate Q-table which all states and actions have scores equal to 0 : $Q(s,a)=0$.

2) Use ε -greedy to randomly select the first state.

3) Use ε -greedy to randomly select the first action.

4) In the AI's state, select the action in the state with highest score from the Q-table. If there is more than one, select randomly.

5) Do the selected action and use the fitness functions to update the Q-table.

6) Repeat 4) and 5) until the AI's state is the ending state.

7) Repeat 4) to 6), if the learning is going on to the next round.

Using Sarsa can fix the first selecting action. The action is not selected randomly but selected based on selected strategies, so the experiments can be fixed into categories of the selecting actions.

Then about ALeRT, it is added the eligibility trace of the state-action pair ($e(s,a)$) and the trace decay parameter (λ). The $e(s,a)$ is the value that give the positive or negative reward. The λ is a constant that updates the $e(s,a)$. And it uses Delta-Bar-Delta (presented by R. Sutton el at. 1992) to detect the changing environment. The learning rate is separated for each action and use the win or learning fast (WoLF presented by Bowling and Veloso 2001) to decrease the learning rate if the action make the character win the match to slow the learning or increase the learning rate. If the action make the character loss the match to speed the learning up because when the character losses, it should learning more quickly to change its strategies but when it wins, it does not need to learning to change its strategies as when it losses.

Algorithms 3.2.2 : ALeRT

1) Initiate Q-table with all state and action scores are 0. $Q(s,a)=0$.

2) Initiate the eligibility trace. $e(s,a)=0$.

3) Initiate the action value. $\theta(s,a)=0$.

4) Use ε-greedy to randomly select the first state.

5) Use ε-greedy to randomly select the first action.

6) Update $e(s,a)$ for every action in this state by $e(s_t,a_t)=e(s_{t-1},a_{t-1})+(1\div i_t)$: i_t is total number of available action at step t.

7) In the AI's state, select the action in the state with highest score from the Q-table. If there is more than one, select randomly.

8) Update error (δ) by $\delta(s_t,a_t)=r-\sum_{1}^{i_t}\theta(s,a)$: r is reward of (s_t,a_t).

9) Do the selected action and use the fitness functions to update the Q-table.

10) Update error (δ) by $\delta(s_t,a_t)=\delta(s_t,a_t)+\gamma * Q(s_t,a_t)$: In this experiment, decreasing factor (γ) of each algorithm is 1.

11) Update action value (θ) by $\theta(s_t,a_t)=\theta(s_t,a_t)+\alpha_a * \delta * e(s_t,a_t)$

12) Update the eligibility trace ($e(s,a)$) by $e(s_t,a_t)=\gamma * \lambda * e(s_t,a_t)$: In this experiment, trace decay parameter (λ) of ALeRT is 0.

13) Calculate a new changing learning rate ($\Delta\alpha$) by $\Delta\alpha=\frac{\alpha_{\max}-\alpha_{\min}}{\alpha_{step}}$: $\alpha_{\max},\alpha_{\min}$ are the highest learning rate value and the lowest learning rate value in the previous actions which are in range of α_{step} which is fixed to 20 in this experiment.

14) Increase learning rate (α) if $\alpha_{mean} * \alpha(s_t,a_t)>0 \wedge |\alpha_{mean}-average\alpha_{mean}|> f * \sigma\alpha_{mean}$, or decrease learning rate (α) if $\alpha_{mean} * \alpha(s_t,a_t)\leq 0$, or else use the current learning rate : α_{mean} is Delta-Bar which is average value of learning rate in range of α_{step} (20 previous steps), $average\alpha_{mean}$ is average of Delta-Bar, f is a constant factor which its value is not reviewed by M. Cutumisu et al. (2008), $\sigma\alpha_{mean}$ is a standard deviation of Delta-Bar.

15) Repeat 6) to 14) until the AI's state is the ending state.

16) Calculate a new ε (probability of greedy algorithm) by these conditions,

if the AI win the match, increase the ε : $\varepsilon = \varepsilon + \dfrac{\varepsilon_{\max} - \varepsilon_{\min}}{\varepsilon_{step}}$

, if the AI loss the match, decrease the ε : $\varepsilon = \varepsilon - \dfrac{\varepsilon_{\max} - \varepsilon_{\min}}{\varepsilon_{step}}$

: $\varepsilon_{\max}, \varepsilon_{\min}$ are the highest value of ε and the lowest value of ε in ε_{step} – the $\varepsilon_{\max}, \varepsilon_{\min}$ are limited to some value that is not reviewed by M. Cutumisu et al. (2008). The ε starts with the $\varepsilon_{\max}$ in this experiment. ε_{step} is the range of the record which is fixed to 15 in this experiment.

17) Repeat 6) to 16) , if the learning is going on to the next round.

In M. Cutumisu et al. (2008), the P. Spronck el at. (2006)'s dynamic script called M1, was used to compare with ALeRT. M1 is used in some computer games. It can change the rule-based of the AI game by using low resources in the computation process.

Algorithms 3.2.3 : M1

1) Use ε -greedy to randomly select the state.

2) Use ε -greedy to randomly select the action.

3) Do the selected action and use the fitness functions to update the Q-table.

4) Repeat 1) to 3) , if the learning is going on to the next round.

The last AI uses the optimal rule-based script which is the best strategy in the control environment. The authors have to predict the situation where the rule-based can make the best strategies in every situation. The optimal script is not reviewed by M. Cutumisu et al. (2008). However as Szita et al. (2009) stated, the rule-based script is too complicate to be a best strategy. In the M. Cutumisu et al. (2008)'s work, the optimal script is used to compare in the position of the best strategy – the worst that optimal script can do, is having equal win rate with the other algorithms when they have learned to their full potential.

ALeRT has the changing learning rate values due to changing of environment. So in the experiment, the fighting match of Never Winter Night (CRPG, own by BioWare) was tested with Sarsa, ALeRT, M1 and optimal script. The condition is the environment of the match will change when the characters change their equipments for melee fighter, range fighter and healer. The game is setup with two characters with two different MLs. The two of them have to fight until one of them die, so the survivor will be the winner. In the match between ALeRT and M1, at the beginning, both of them have the 50% win rate. However after the equipment changing ALeRT's win rate increases to 80% because it can adapt to the new environment faster. In the match between ALeRT and optimal script, ALeRT can increase win rate to 50% against optimal script in 500 matches but the other algorithms cannot do that.

In their experiment, M. Cutumisu et al. (2008) showed that ALeRT is a fast learning algorithm and it can still learning fast when there are changing in the environment by modifying the separated learning rate for each action, the changing probability of greedy algorithm and the changing of learning speed when win or loss the match.

3.3. Cross-Entropy Method

Szita et al. (2009) wants to solve the problems of static AI which are complicated, weak, predictable and static. The static AI always has weaknesses because it is difficult to program the AI to do the right thing in every situation and then the human player can predict or guess the weaknesses by using some playing experience. By using some strategies, the player can exploit the weakness of the static AI and dominate the game. With the static AI which cannot change the strategy at all, the player can find and use the strongest strategy against the AI to the end game which makes the game boring because for the rest of the game, the player has to do the same strategy – also when the player play the second time too. To eliminate the weaknesses, the programmers have to program the AI very carefully and the code must cover all situations, so there are too many conditions which are needed to be added to the code. After that when there are the problems, changing the codes must be very difficult tasks because the codes always have to be very complicated.

Too solve the problem, Szita et al. (2009) used cross-entropy method (CEM) that was presented by Rubinstein in a CRPG – Minigate environment. CEM is used to select the best strategy and also shift to the different strategies to get various strategies. CEM is one of reinforcement learning. Three random actions will be generated as a set of strategies to measure the performance of the strategies. The different orders of actions provide different strategies.

Algorithms 3.3.1 : Cross-Entropy Method (CEM)

1) Initiate probability of every action : $p = p_1 + p_2 + p_3 + ... + p_n$ where p_1 to p_n are the same value.

2) Use ε-greedy to randomly select three actions and the set of the three actions is fixed at order of the actions which are used in these orders by the AI character.

3) Do the action and update the action's score by using fitness function. In Szita's experiment, there are two sub fitness functions which are strength and diversity. If the AI wins the match, it will get more score from the strength function and if the AI uses the actions what are different from the previous match, it will get more score from the diversity function.

4) Do 2) and 3) until N matches: N is number of matches before the probability will be updated; N is assigned by the author.

5) Update the probability by: if p_i is in the top $\rho * N$ of set of action list, $p'_i = p_t /(\rho * N)$ and then $p_i = (1-\alpha) * p_i + \alpha * p_i$.

6) Do 2) to 5) again, if the learning is going on to the next round.

In their experiment, authors want CEM to select strategies for AI. The first is choosing the strongest strategy. The other is choosing diverse strategy. So, the strategy has to be strongest but not always be the same strategy. At the end of learning, the best set of strategies was chosen as a macro strategy. Such macros are brought to use with the dynamic script (see algorithms 3.2.3) to create random event. The dynamic script is one of reinforcement learning and usually implements in computer game. In the experiment, dynamic script is used to compare with CEM in the strong and diversity ways. Both CEM with dynamic script AI and dynamic script AI have to play with the four difficulties of the static AI and the results of the matches are recorded in how the strong and diversity ways they are.

The results showed that using CEM with dynamic script can choose stronger and more diverse strategies than using only dynamic script. Because the actions conflict with the others, some actions are good but when combine with the wrong actions, they do not work well. So using the macros can fix the good combination. About the diversity, CEM with the diverse fitness function also create the micros with the good diversity among themselves but dynamic which can only select the action randomly cannot tell the different between the selected action and the previous action to select them differently.

3.4. Very Fast Decision Tree

After Szita proposed using CEM with Minigate environment, Vallim et al. (2010) also used Minigate environment to generate their different style patterns and selected very fast decision tree (VFDT) which created by Domingos et al. (2000), as the learning technique to classify the character's action which are created by the rule-based scripts. The rule-based scripts make the character always acts the same thing in the same situation and creates the patterns in the game. VFDT is used to classify the patterns. The character's actions are created by the four sets of rule-based scripts. Each set make the character do different actions in the same situation, so the four sets are created different patterns. These patterns are also called the character's strategies because the patterns give the character's decision to choose actions. In this experiment, the four sets of rule-based script create four types of character's strategies which are fighter's offensive tactic, fighter's defensive tactic, wizard's offensive tactic and wizard's defensive tactic. The rule-based scripts are shown as follows:

The fighter’s offensive tactic:

- if (HP < 50) and (PotionHealing = yes) then DrinkPotionH
- if (HP $\geq$ 50) then AttackClosestEnemy

The wizard’s offensive tactic:

- if (HP < 50) and (PotionHealing = yes) then DrinkPotionH
- if (HP $\geq$ 50) and (Level3SpellDG = yes) and (TypeClosestEnemy = wizard) then CastLevel3DGSpellCenterEnemy
- if (HP $\geq$ 50) and (Level2SpellDG = yes) then CastLevel2DGSpellClosestEnemy
- if (HP $\geq$ 50) and (Level1SpellDG = yes) then CastLevel1DGSpellWeakestEnemy
- if (HP $\geq$ 50) then AttackClosestEnemy

The defensive fighter’s tactic:

- if (RoundNumber $\leq$1) and (PotionFR = yes) then DrinkPotionFR
- if (HP < 50) and (PotionHealing = yes) then DrinkPotionH
- if (HP $\geq$ 50) then AttackClosestEnemy

The defensive wizard’s tactic:

- if (HP < 50) and (PotionHealing = yes) then DrinkPotionH
- if (HP $\geq$ 50) and (Level2SpellDF = yes) then CastLevel2DFSpellCenterEnemy
- if (HP $\geq$ 50) and (Level3SpellDF = yes) then CastLevel3DFSpell
- if (HP $\geq$ 50) and (Level1SpellDF = yes) then CastLevel1DFSpellClosestEnemy
- if (HP $\geq$ 50) then AttackClosestEnemy

Key words:

- ■ HP = percentage of own hit point
- ■ PotionHealing = potion of healing (a parameter for detecting potions of healing) *Potion of healing is used only in this rule-based script. Potion of healing will be called “potion of hit point” later.
- ■ DrinkPotionH = choose action “drink potion of healing”

- AttackClosestEnemy = find the closest enemy and choose action “malee attack” target the closet enemy.
- Level3SpellDG = level 3 damage spell (a parameter for detecting level 3 damage spell)
- Level2SpellDG = level 2 damage spell (a parameter for detecting level 2 damage spell)
- Level1SpellDG = level 1 damage spell (a parameter for detecting level 1 damage spell)
- TypeClosestEnemy = type of the closest enemy
- CastLevel3DGSpellCenterEnemy = find the center of enemies and choose the action “cast level 3 damage spell” target the center of enemies
- CastLevel2DGSpellClosestEnemy = find the closest enemy and choose the action “cast level 2 damage spell” target the closest enemy
- CastLevel1DGSpellWeakestEnemy = find the weakest enemy and choose the action “cast level 1 damage spell” target the weakest enemy
- RoundNumber = turn (a parameter for natural how many times the characters do their actions – start from 1 to 20)
- PotionFR = potion of fire resistance (a parameter for detecting potions of fire resistance)
- DrinkPotionFR = choose action “drink potion of fire resistance”
- Level3SpellDF = level 3 damage spell (a parameter for detecting level 3 defensive spell)
- Level2SpellDF = level 2 damage spell (a parameter for detecting level 2 defensive spell)
- Level1SpellDF = level 1 damage spell (a parameter for detecting level 1 defensive spell)
- CastLevel3DFSpell = choose the action “cast level 3 defensive spell”
- CastLevel2DFSpellCenterEnemy = find the center of enemies and choose the action “cast level 2 defensive spell” target the center of enemies

- CastLevel1DFSpellClosestEnemy = find the closest enemy and choose the action "cast level 1 defensive spell" target the closest enemy

Then these four types of strategies are used to create four types of databases. By using the strategies in the game, the situation and the action of the characters were recorded. A lot of playing matches were simulated by Minigate environment to generate large databases. In the records, the situations are defined as attributes and the actions are classes. The databases were designed as follows:

- Offensive fighter database has seven attributes and two classes. The attributes are hit point, closet enemy distance, influence status, location status, potion of hit point, potion of fire resistance, and potion of free action. The classes are attacking, and drinking potion of hit point.
- Defensive fighter database has eight attributes and three classes. The attributes are turn, hit point, closet enemy distance, influence status, location status, potion of hit point, potion of fire resistance, and potion of free action. The classes are attacking, drinking of potion hit point, and drinking of fire resistance potion.
- Offensive wizard database has 11 attributes and five classes. The attributes are hit point, closet enemy distance, closet enemy type, influence status, location status, potion of hit point, potion of fire resistance, potion of free action, offensive spell level 1, offensive spell level 2, and offensive spell level 3. The classes are attacking, drinking potion hit point, casting offensive spell level 1, casting offensive spell level 2, and casting offensive spell level 3.
- Defensive wizard database has 11 attributes and five classes. The attributes are hit point, closet enemy distance, closet enemy type, influence status, location status, potion of hit point, potion of fire resistance, potion of free action, defensive spell level 1, defensive spell level 2, and defensive spell level 3. The classes are attacking, drinking potion hit point, casting defensive spell level 1, casting defensive spell level 2, and casting defensive spell level 3.

definition of attributes

- Turn (1-20): number of playing rounds. Turn = 1 means that the character has passed 1 round. There is no record from an initial turn. It is recorded as natural number.
- Hit point (1-100): percentage of hit point. If the character receives damage from the enemy, it decreases and if it is 0, the character die, so the hit point is never 0 in the record. It is natural number, not float number.
- Closet enemy distance (1-20): distance of a closet enemy on the field. It is in-game distance. it is recorded as natural number, not float number.
- Closet enemy type (0, 1): a fighter type or a wizard type. For a fighter enemy, it is 0 and for a wizard, it is 1.
- Influence status (0, 1): mark of spell effect. For a character with no spell effect, it is 0 and for a character with spell effect, it is 1.
- Location status (0, 1): inside the area of spell effect. For a character in a normal area, it is 0 and for a character in a spell area, it is 1.
- Potion of hit point (0, 1): having potion of hit point in inventory. If the character has the item, it is 1, or else 0.*
- Potion of fire resistance (0, 1): having potion of fire resistance in inventory. If the character has the item, it is 1, or else 0.*
- Potion of free action (0, 1): having potion of free action in inventory. If the character has the item, it is 1, or else 0.*
- Offensive spell level 1 (0, 1): able to cast offensive spell level 1. If the character has the spell, it is 1, or else 0.*
- Offensive spell level 2 (0, 1): able to cast offensive spell level 2. If the character has the spell, it is 1, or else 0.*
- Offensive spell level 3 (0, 1): able to cast offensive spell level 3. If the character has the spell, it is 1, or else 0.*

- Defensive spell level 1 (0, 1): able to cast offensive spell level 1. If the character has the spell, it is 1, or else 0.*
- Defensive spell level 2 (0, 1): able to cast offensive spell level 2. If the character has the spell, it is 1, or else 0.*
- Defensive spell level 3 (0, 1): able to cast offensive spell level 3. If the character has the spell, it is 1, or else 0.*

* In the script, there are random generated rules which random character's items and spells.

VFDT are decision tree that can decide the highest entropy or information gain of all attributes in the patterns which are used as training set to create root node and use the decided attribute to separate the patterns into groups of the patterns with the same class at the leaves node. However, there is a threshold which is criteria to create a new level of tree. In the case of under the threshold, the creation of the new level will be terminated. Therefore, Naïve Bayes is used to solve this problem to select the class of the pattern at the current leaf node.

Algorithms 3.4.1: VFDT

1) Initiate the root node every data start at.

2) Calculate the entropy and the information gain in every leaf node by: $Entropy(t) = -\sum_{i=1}^{n} p(i \mid t)\log_2 p(i \mid t)$ where $Entropy(t)$ is Entropy of an attribute; $p(i \mid t)$ is probability of a value (i) in an attribute (t); n is total number of values (i) in an attribute (t).

$\Delta = Entropy(parent) - \sum_{j=1}^{k} \frac{N(leaf)_j}{N(parent)} Entropy(leaf)_j$ where Δ is the Information Gain; $Entropy(parant)$ is the entropy of the parent node; $Entropy(leaf)_j$ is the entropy of the leaf node which uses attribute j to split; $N(parant)$ is a total number of data in parent node before splitting; $N(leaf)_j$ is a total number of data in this leaf node after splitting by attribute j; k is total of attributes in the leaf node.

3) Spit the leaf node by the highest entropy or the highest information gain if the condition in the Hoeffding function is true. Calculate Hoeffding function which is criteria to create a new level of tree by: $\varepsilon = \sqrt{R^2 \frac{\ln\left(\frac{1}{\delta}\right)}{2n}} < \Delta(a) - \Delta(b)$ or $\Delta(a) - \Delta(b) < \varepsilon = \sqrt{R^2 \frac{\ln\left(\frac{1}{\delta}\right)}{2n}} < \tau$ where ε is Hoeffding bound which is the criteria that is compared with the difference between the highest entropy and the second highest entropy; R is the range of a random variable r; δ is a value that equal to 1 minus probability of r (probability of $r = 1 - \delta$); $\Delta(a)$ is the highest entropy or the highest information gain in the leaf node; $\Delta(b)$ is the second highest entropy or the second highest information gain in the leaf node; τ is the threshold what is suggested by the authors – in their experiment, τ is equal to 0.
4) Every data in the leaf node has to be classified by Naïve Bayes.
5) Rebuild the tree when there are more N number of data come into the data set – N is the lowest number of data that the user suggest to rebuild the tree. In Vallim et al. (2010), N is equal to 1.

In their works, they have four types of characters which are offensive fighter, defensive fighter, offensive wizard and defensive wizard. First, the characters were tested separately in static strategy learning. All character's actions of each character type were used to create a decision tree by the VFDT algorithms. Then it was used to classify the test sets and the classification compared the output with the actual test sets to get the accuracy. Second, the offensive fighter and defensive fighter were tested in the changing strategy learning. The classification did the same with the static strategy learning but there is a different data set. The sequence of the data set had an importance role. The offensive fighter and the defensive fighter have a mixture in the data set with the sequence which offensive fighter's actions were randomly selected first until a thousand actions, defensive fighter's actions were played until a thousand and five hundreds actions and then these two types of actions were randomly selected one thousand and five hundreds each alternatively.

Their work was a main reference in this work. The character's actions in this work are given by them too. In order to improve the performance of the classification, the resilience backpropagation (RPROP) has tested with the character's actions.

3.5. Resilience Backpropagation (RPROP)

For RPROP related work, A. Mark et al. (2004) use Genetic Algorithm and RPROP to predict opponent's strategies and find counter strategies in Prisoner's Dilmma game and Rock Paper Scissors with Well game. Both games are not CRPG type but are competitive game between two players that the one who gets more score at the end of the game is the winner. Genetic Algorithm is the other reinforcement learning. By randomly generating playing situation and test with fitness function, it can select best opponent strategy and best counter strategy. Combined with playing history, all of playing situations and strategies as patterns are fed to RPROP to build the network and find the predicted action of opponent.

Algorithms 3.5.1 : Genetic Algorithm

1) Randomly initiate data set.

2) Randomly initial copies of data by using random weights – the random weights are changes which some copies are created complete randomly but some copies are created with some parts of the other data in their parts, like the crossing over in the biological chromosomes.

3) Select best four actions into the future of the opponent from 16 previous actions of the opponent.

4) Calculate of the actions to get the best action of the opponent in the next round.

5) Select own action and do the action to evaluate the score.

6) Use the new score to update the Genetic algorithm's random weight and RPROP network's weight.

RPROP is a multi-layered feed-forward network with the increasing and decreasing factors which are different in values – presented by M. Riedmiller el at. 1992. It is proposed to improve the original Backpropagation. RPROP can train a network fast because the idea that use of the high increasing factor can push the point forward to where the lowest error can be obtained. In the case that the error is not the lowest, the low decreasing factor can slowly find the lowest point. This is usually faster than using only learning rate of the Backpropagation to adjust weight values when the learning rate has to be low, so it makes the pushing forward steps slower than RPROP by

$$\Delta w_{ij} = \begin{cases} \Delta_0 * \eta^{+}, if \dfrac{\partial E}{\partial w_{ij}}(t-1) * \dfrac{\partial E}{\partial w_{ij}}(t) > 0 \\ \Delta_0 * \eta^{-}, if \dfrac{\partial E}{\partial w_{ij}}(t-1) * \dfrac{\partial E}{\partial w_{ij}}(t) < 0 \\ \Delta_0, else \end{cases}$$

Where Δ_0 is the original weight adjusted or the $w_{ij}(t+1)$ in the Multilayer Feed Forward Artificial Neural Network which is stated as follows (see Chapter 2.1.):

$w_{ij}(t+1) = w_{ij}(t) - \varepsilon \dfrac{\partial E}{\partial w_{ij}}(t)$ Where w_{ij}(t) is old weight; w_{ij}(t+1) is new weight; E is the error; ε is learning rate;

η^{+} is increasing factor and η^{-} is decreasing factor and $0 < \eta^{-} < 1 < \eta^{+}$.

In A. Mark's work, they focus only on two similar players, not apply on CRPG which has diverse characters.

CHAPTER IV

PROPOSED METHOD

This proposed method aims to eliminate the problem of VFDT which the problem is when the number of data increases, the accuracy increase slowly because VFDT uses information gain or entropy as mentioned in the previous section. By using RPROP, the increasing data will be considered as the input patterns and the data can improve the accuracy much more faster compared to VFDT. So this proposed method aims to use RPROP to add more accuracy in the classifying character's action.

4.1. Databases of Character's Actions

The databases of the character's actions have been received from Vallim et al. (2010). The data were generated by recording the patterns of a computer role-playing game – Minigate environment created by Pieter Spronck, 2007. There are four types of database – offensive fighter database, defensive fighter database, offensive wizard database, and defensive wizard database.

- **Offensive fighter database** has seven attributes and two classes. The attributes are hit point, closet enemy distance, influence status, location status, potion of hit point, potion of fire resistance, and potion of free action. The classes are attacking, and drinking potion of hit point.
- **Defensive fighter database** has eight attributes and three classes. The attributes are turn, hit point, closet enemy distance, influence status, location status, potion of hit point, potion of fire resistance, and potion of free action. The classes are attacking, drinking of potion hit point, and drinking of fire resistance potion.
- **Offensive wizard database** has 11 attributes and five classes. The attributes are hit point, closet enemy distance, closet enemy type, influence status, location status,

potion of hit point, potion of fire resistance, potion of free action, offensive spell level 1, offensive spell level 2, and offensive spell level 3. The classes are attacking, drinking potion hit point, casting offensive spell level 1, casting offensive spell level 2, and casting offensive spell level 3.

- **Defensive wizard database** has 11 attributes and five classes. The attributes are hit point, closet enemy distance, closet enemy type, influence status, location status, potion of hit point, potion of fire resistance, potion of free action, defensive spell level 1, defensive spell level 2, and defensive spell level 3. The classes are attacking, drinking potion hit point, casting defensive spell level 1, casting defensive spell level 2, and casting defensive spell level 3.

The definitions of the attributes

- Turn (1-20): number of playing round. Turn = 1 is mean the character has passed one round. There is no record from an initial turn. It is recorded as natural number.
- Hit point (1-100): percentage of hit point. If the character receives damage from the enemy, it decreases and if it is 0, the character die, so the hit point is never 0 in the record. It is natural number, not float number.
- Closet enemy distance (1-20): distance of a closet enemy on the field. It is in-game distance. It is record as natural number, not float number.
- Closet enemy type (0, 1): a fighter type or a wizard type. For a fighter enemy, it is 0 and for a wizard, it is 1.
- Influence status (0, 1): mark of spell effect. For a character with no spell effect, it is 0 and for a character with spell effect, it is 1.
- Location status (0, 1): inside the area of spell effect. For a character in a normal area, it is 0 and for a character in a spell area, it is 1.
- Potion of hit point (0, 1): having potion of hit point in inventory. If the character has the item, it is 1, or else 0.*

- Potion of fire resistance (0, 1): having potion of fire resistance in inventory. If the character has the item, it is 1, or else 0.*
- Potion of free action (0, 1): having potion of free action in inventory. If the character has the item, it is 1, or else 0.*
- Offensive spell level 1 (0, 1): able to cast offensive spell level 1. If the character has the spell, it is 1, or else 0.*
- Offensive spell level 2 (0, 1): able to cast offensive spell level 2. If the character has the spell, it is 1, or else 0.*
- Offensive spell level 3 (0, 1): able to cast offensive spell level 3. If the character has the spell, it is 1, or else 0.*
- Defensive spell level 1 (0, 1): able to cast offensive spell level 1. If the character has the spell, it is 1, or else 0.*
- Defensive spell level 2 (0, 1): able to cast offensive spell level 2. If the character has the spell, it is 1, or else 0.*
- Defensive spell level 3 (0, 1): able to cast offensive spell level 3. If the character has the spell, it is 1, or else 0.*

* In this rule, there are random generated character's items and spells.

Examples of the databases

Table 4.1.1 : Examples of offensive fighter database

HP	CED	IS	LS	Potion HP	Potion FR	Potion FA	Class
60	5	0	0	0	0	1	Attack_closest_enemy
63	2	0	1	0	1	0	Attack_closest_enemy
36	19	1	0	1	0	0	Drink_potion_A
89	8	0	1	1	1	1	Attack_closest_enemy
12	4	1	0	0	1	1	Attack_closest_enemy
90	13	1	0	1	1	1	Attack_closest_enemy
22	14	0	0	1	1	0	Drink_potion_A
25	18	0	1	1	1	1	Drink_potion_A
34	11	1	1	1	1	1	Drink_potion_A
75	20	1	0	1	1	1	Attack_closest_enemy

Table 4.1.2 : Examples of defensive fighter database

Turn	HP	CED	IS	LS	Potion HP	Potion FR	Potion FA	Class
8	34	12	1	0	1	1	0	Drink_potion_A
12	84	19	0	1	0	0	0	Attack_closest_enemy
8	20	3	0	0	1	1	1	Drink_potion_A
13	46	7	0	0	0	1	1	Attack_closest_enemy
12	98	2	0	1	1	1	1	Attack_closest_enemy
4	27	9	1	0	0	0	1	Attack_closest_enemy
5	75	14	0	0	0	0	1	Attack_closest_enemy
7	98	7	1	0	1	0	1	Attack_closest_enemy
13	22	2	0	0	1	1	0	Drink_potion_A
1	37	11	0	1	1	0	1	Drink_potion_B

Table 4.1.3 : Examples of offensive wizard database

HP	CED	CET	IS	LS	Potion HP	Potion FR	Potion FA	OFS L1	OFS L2	OFS L3	Class
23	7	1	0	0	1	0	1	1	1	0	Drink_potion_A
75	20	0	1	1	1	0	0	1	1	1	Cast_DSpell2_closestEnemy
1	14	0	0	1	1	1	1	0	1	0	Drink_potion_A
83	2	1	0	0	0	0	0	0	0	1	Cast_DSpell3_centerEnemy
73	9	1	0	0	0	1	0	1	1	1	Cast_DSpell3_centerEnemy
74	18	1	1	0	1	1	0	0	1	0	Cast_DSpell2_closestEnemy
25	15	0	1	1	1	0	1	0	0	1	Drink_potion_A
92	2	0	0	0	1	0	0	0	0	0	Attack_closest_enemy
98	9	0	0	0	0	1	1	1	1	0	Cast_DSpell2_closestEnemy
51	6	1	1	1	0	1	0	1	0	0	Cast_DSpell1_weakestEnemy

Table 4.1.4 : Examples of defensive wizard database

HP	CED	CET	IS	LS	Potion HP	Potion FR	Potion FA	OFS L1	OFS L2	OFS L3	Class
23	7	1	0	0	1	0	1	1	1	0	Drink_potion_A
75	20	0	1	1	1	0	0	1	1	1	Cast_DSpell2
1	14	0	0	1	1	1	1	0	1	0	Drink_potion_A
83	2	1	0	0	0	0	0	0	0	1	Cast_DSpell3
73	9	1	0	0	0	1	0	1	1	1	Cast_DSpell2
74	18	1	1	0	1	1	0	0	1	0	Cast_DSpell2
25	15	0	1	1	1	0	1	0	0	1	Drink_potion_A
92	2	0	0	0	1	0	0	0	0	0	Attack_closest_enemy
98	9	0	0	0	0	1	1	1	1	0	Cast_DSpell2
51	6	1	1	1	0	1	0	1	0	0	Cast_DSpell1

4.2. Learning Strategies

First, the static strategies need to be tested with the ML algorithm – RPROP. In order to classify the character's action, the character must have their situation pattern in the game been recorded. As presented in the previous topic, the databases are used as the character's situation pattern that can describe their situation in the game. And then the pattern is used to classify the character's action in the training of ML. After the ML train the network, the network will be able to classify the character's action from the character's situation pattern of the test sets. Finally the classes from the testing will be compared with the class from the original records of the test sets. The comparison is used to calculate the classification accuracy which shows the relation of the ML and the data set. If the percentages of the accuracy are high that mean the ML is suitable with the data set. In the testing run, the number of the data set was proved to have impact with the accuracy. So the test was repeated with the different number of the data in the training sets.

After the first test, the changing strategies are the next tests for the ML. If ML is suitable for the CRPG, it should be able to handle with changing strategies. The changing strategy is the natural way of the human player who is learning to use a new strategy. A new strategy makes a human player be better because there are more strategies to choose. Some strategies are for fun too but they are not the purpose. The human player also adapts a new strategy to counter the enemy's strategy which is made the AI have to learn the changing strategies. Without ML, it is very difficult to make the AI learn the changing strategies, so using ML is solution and this is why the changing strategies are tested.

4.2.1. Learning Static Strategies

In the research, the test runs show that when using the attributes as inputs, RPROP will give outputs what classes of the character's actions are. In these following steps, the network was configured and tested with the database:

1) There are four RPROP networks for four databases which contain configured parameters and function as follows.

- one input layer with number of attributes that equals to number of input nodes
- one hidden layer with five hidden nodes
- one output layer with number of classes that equals to number of output nodes
- mean square error as performance function
- Log-Sigmoid as transfer function
- Stoping criteria:
    - performance goal is 0
    - maximum number of epochs to train = 1,000
    - minimum performance gradient = 0.000001
- learning-rate = 0.01

• increasing factor = 1.2

• decreasing factor = 0.5

2) For each database from section 4.2., ten different sets of 1,000 patterns are used as ten training sets.

3) For each database, ten different sets of 500 patterns are used as ten test sets.

4) The character's action was changed into binary numbers corresponding to output classes:

• Offensive fighter. There are two output classes: attacking (1, 0) and drinking potion of hit point (0, 1)

• Defensive fighter. There are three output classes: attacking (1, 0, 0), drinking potion of hit point (0, 1, 0) ,and drinking potion of fire resistance(0, 0, 1)

• Offensive wizard. There are five output classes: attacking (1, 0, 0, 0, 0), drinking potion of hit point (0, 1, 0, 0, 0), casting offensive spell level 1 (0, 0, 1, 0, 0), casting offensive spell level 2 (0, 0, 0, 1, 0), and casting offensive spell level 3 (0, 0, 0, 0, 1).

• Defensive wizard. There are five output classes: attacking (1, 0, 0, 0, 0), drinking potion of hit point (0, 1, 0, 0, 0), casting defensive spell level 1 (0, 0, 1, 0, 0), casting defensive spell level 2 (0, 0, 0, 1, 0), and casting defensive spell level 3 (0, 0, 0, 0, 1).

5) There were total numbers of 5, 10, 15, 20, 25, 30, 40, 50, 100, 200, 300, 400, 500, and 1000 training set patterns.

6) Each training set in 2) with each total number of the patterns in 5) were used to train each network to test with the 10 different test sets in 3)

7) The overall accuracy of the network is calculated. From the output, the outputs always are values from 0 to 1, so the greatest value among all output nodes in the network will be considered as 1 and the others are 0. The output was compared to the actual class for each character's action in the test set. Then the accurate ones are taken into account and used to calculate percentage of accuracy.

8) The average percentage of accuracy between each pair of different training set and test set with the same total number of patterns in 5) is calculated.

4.2.2. Learning Changing Strategies

After that, in order to compare the changing strategies which have been tested with VFDT by Vallim et al., the changing strategies were also tested by using RPROP under the consideration as follows.

1) The RPROP was configured the same as the first testing in the previous subsection.

2) Because the database from Vallim et al. (2010) does not contain patterns for changing strategies experiment, the new datasets of patterns were created. There are new training sets which are created by mixing offensive and defensive fighter's actions and two types of test sets: offensive test set and defensive test set.

3) The new training sets were generated from a combination of offensive and defensive fighters's actions as follows.

a) Training by 500 offensive fighter's actions and testing by 50 offensive fighter's actions

b) Training by a) including 500 offensive fighter's actions (1,000 offensive fighter's actions) and testing by 50 defensive fighter's actions

c) Training by b) plus 500 defensive fighter's actions (1,000 offensive fighter's actions and 500 defensive fighter's actions) and testing by 50 defensive fighter's actions

d) Training by c) plus 500 defensive fighter's actions (1,000 offensive fighter's actions and 1,000 defensive fighter's actions) and testing by 50 defensive fighter's actions

e) Training by d) plus 500 defensive fighter's actions (1,000 offensive fighter's actions and 1,500 defensive fighter's actions) and testing by 50 offensive fighter's actions

f) Training by e) plus 500 offensive fighter's actions (1,500 offensive fighter's actions and 1,500 defensive fighter's actions) and testing by 50 offensive fighter's actions

g) Training by f) plus 500 offensive fighter's actions (2,000 offensive fighter's actions and 1,500 defensive fighter's actions) and testing by 50 offensive fighter's actions

h) Training by g) plus 500 offfensive fighter's actions (2,500 offensive fighter's actions and 1,500 defensive fighter's actions) and testing by 50 defensive fighter's actions

i) Training by h) plus 500 defensive fighter's actions (2,500 offensive fighter's actions and 2,000 defensive fighter's actions) and testing by 50 defensive fighter's actions

j) Training by i) plus 500 defensive fighter's actions (2,500 offensive fighter's actions and 2,500 defensive fighter's actions) and testing by 50 defensive fighter's actions

k) Training by j) plus 500 defensive fighter's actions (2,500 offensive fighter's actions and 3,000 defensive fighter's actions) and testing by 50 offensive fighter's actions

4) Each pair of training set and test set from a) to k) was tested five times.

5) The average accuracy was calculated from a) to k) separately.

6) Do 1) to 5) again in opposite way by swapping each offensive set to defensive set and swapping each defensive set to offensive set.

Pattern of the charging strategies training

In step 3) of the changing strategies, there are patterns as follows:

Table 4.2 : Pattern of the changing strategies training

Step	Purpose of step
1) Initiate the network. Train 500 data of A training set and test by A test set. (A is the first strategy to start learning from. A can be offensive fighter strategy or defensive fighter strategy)	This is the simulation where the learning ML does normally training – it is trained and tested with the same type of strategy.
2) Add 500 data of A training set to training set. test by B test set. (B is the second strategy to learning. B is the opposite strategy of A. If A is offensive fighter strategy, B has to be defensive fighter strategy.)	In this step, it is the simulation of the situation when the opponent suddenly changes its strategy and the learning character must be catch off guard because it is only learning the opponent's previous strategy. The purpose of the step is creating the most errors from the classifying when the opponent changes its strategy. The errors will show how the accuracy will improve when the character learns the changing strategies.
3) Add 500 data of B training set. Test by B test set.	In this step, the data of B training set is added to the training set, so the learning character has some data of B to classify the strategy of B in the B test set. So this step should have more accuracy than step 2). However, in the training set, there are 1,000 of A and 500 of B, so the learning character will confuse. Then the learning character is more likely to use the data of A to classify and make some errors

	because the test set is the data set of B.
4) Add 500 data of B training set. Test by B test set.	This step is the same as the previous step but adding more data of B will make the learning character more likely to use data of B to classify, so it can get more accuracy. And from step 2) to 4), the accuracy will be improved because of adding data of B, but the purpose is comparing the improved accuracy between VFDT and RPROP to see which one is the best.
5) Repeat step 2) to step 4) but swap A to B and B to A.	In this step, the learning character's situation is repeat step 2) to 4) again but this time, the character will have some data which are the same type of strategy as the test set, so the learning will be improved its accuracy. about the purpose of this step, the accuracies are compared between VFDT and RPROP after the changing strategies occur continuously.

At step 5), there will be the accuracy which is the first result of the learning changing strategies and at step 6), the accuracy of RPROP is the second result of the learning changing strategies. These two results are different from using different data sets by swapping the time when using offensive training sets and defensive training sets.

CHAPTER V

RESULTS

In this section, there are results of learning static strategies and results of learning changing strategies. The results show the accuracy and compare the accuracy between VFDT and RPROP when use the same data sets.

5.1. Results of Learning Static Strategies

In this section, the results from VFDT in Vallim et al. (2010) and proposed method have been compared in Table 5.1.1 to Table 5.1.4 for the learning static strategies which are four different data types: offensive fighter, defensive fighter, offensive wizard, and defensive wizard. The four data types are different in their attributes and classes as mentioned in section III. Each data type was trained and tested separately.

Table 5.1.1 : Comparison between RPROP and VFDT for offensive fighter

total action in training set	VFDT accuracy (%)	RPROP accuracy (%)	VFDT SD	RPROP SD
5	76.58	67.13	1.75	8.6
10	80.06	76.45	4.33	7.99
15	83.9	78.07	7.25	8.83
20	86.76	83.41	6.02	8.23
25	88.74	86.89	4.75	8.06
30	89.04	90.79	4.03	4.42
40	91.28	92.81	3.88	4.13
50	92.5	94.28	2.29	3.3
100	96.2	97.49	1.34	1.73
200	96.16	98.79	1.27	1.12
300	96.04	99.22	1.2	0.84
400	96.74	99.63	1.57	0.52
500	96.8	99.65	1.47	0.5
1000	97.51	99.89	2.04	0.24

In Table 5.1.1, RPROP has higher accuracy than VFDT when the total of actions in training sets are at least 30.

Table 5.1.2 : Comparison between RPROP and VFDT for defensive fighter

total action in training set	VFDT accuracy (%)	RPROP accuracy (%)	VFDT SD	RPROP SD
5	71.9	56.56	10.8	9.91
10	72.8	65.7	10.65	9.53
15	75.72	70.9	6.2	7.72
20	79	72.94	8.36	6.3
25	80.38	77.18	5.89	6.15
30	84.14	79.11	5.18	6.57
40	88.66	83.3	3.1	4.83
50	89.98	84.94	4.74	4.56
100	92.86	88.86	3.08	2.86
200	94.86	92.94	3.59	2.27
300	95.64	94.45	2.04	1.85
400	96.1	95.85	1.47	1.74
500	96.44	<u>96.81</u>	1.59	1.84
1000	96.68	<u>97.36</u>	1.01	2.37

In Table 5.1.2, RPROP has more accuracy than VFDT when the total number of actions in training sets are at least 500. The total actions have to have more compared to the offensive fighter.

Table 5.1.3 : Comparison between RPROP and VFDT for offensive wizard

total action in training set	VFDT accuracy (%)	RPROP accuracy (%)	VFDT SD	RPROP SD
5	30.14	36.19	2.02	6.85
10	38.54	41.44	1.8	8.76
15	46.56	45.158	5.23	7.71
20	55.42	50.16	6.6	7.76
25	60.48	51.78	5.47	7.12
30	65.84	54.28	3.66	7.28
40	72.4	60.2	3.55	7.27
50	76.44	65.7	4.46	6.75
100	84.8	78.22	3.48	5.69
200	89.78	90.66	1.88	4.69
300	92.18	94.35	1.97	2.95
400	93.58	95.91	1.43	2.47
500	93.9	96.74	1.54	1.87
1000	95.64	97.64	1.25	1.67

In Table 5.1.3, RPROP has higher accuracy than VFDT when the total number of actions in training set are at least 200.

Table 5.1.4 : Comparison between RPROP and VFDT for defensive wizard

total action in training set	VFDT accuracy (%)	RPROP accuracy (%)	VFDT SD	RPROP SD
5	32.58	40.7	6.85	6.86
10	40.82	46.84	9.72	8.09
15	52.46	53.33	7.22	6.69
20	60.4	57.76	7.1	7.51
25	65.58	61.07	5.55	7.34
30	70.22	65.28	5.29	6.45
40	77.36	71.88	5.34	6.55
50	84.08	76.64	5.25	5.87
100	91.6	89.18	4.37	4.54
200	95.82	96.19	1.04	2.06
300	77.12	97.57	20.73	1.21
400	84.44	98.03	12.21	1.40
500	92.78	98.55	3.87	0.76
1000	94.52	99.1	2.9	0.97

In Table 5.1.4, RPROP has higher accuracy than VFDT when the total number of actions in training sets are at least 200.

At the beginning of the learning static strategies, RPROP yields accuracy lower than that of VFDT. However after the total number of training data increases to 100-500, the accuracies of RPROP are going to be better than those of VFDT, so if there are at least 500 data, RPROP can be used to classify the character’s action better than VFDT.

5.2. Result of Learning Changing Strategies

In Table 5.2.1 and Table 5.2.2 the results of changing strategies between offensive fighter and defensive fighter showed that the accuracies have been changed between increasing and decreasing. When the strategy has been changed, the accuracy becomes decrease and then conversely increases when more actions have been added more to the training set which is the pattern in section 4.3.2. There are two results. The first is the result of learning changing strategies when the training data set starts from pure offensive fighter’s actions and the second is the result which is started from pure defensive fighter’s actions.

Table 5.2.1 : Comparison between RPROP and VFDT for Changing Strategies Fighter by starting from Offensive Fighter

total action in training set	VFDT accuracy (%)	RPROP accuracy (%)
500	78	83.6
1000	64	64
1500	82	68.8
2000	94	70.4
2500	64	82
3000	78	84.4
3500	80	86
4000	74	71.2
4500	76	70
5000	88	70.8
5500	70	76.4

In table 5.2.1, the accuracy of RPROP is better every time when using offensive fighter test set but with the defensive fighter test set, the accuracy is worse.

Table 5.2.2 : Comparison between RPROP and VFDT for Changing Strategies Fighter by starting from Defensive Fighter

total action in training set	VFDT accuracy (%)	RPROP accuracy (%)
500	94	78
1000	60	<u>70.8</u>
1500	58	<u>80.58</u>
2000	84	<u>85.6</u>
2500	64	<u>66.64</u>
3000	70	<u>72.4</u>
3500	100	76.8
4000	70	<u>83.2</u>
4500	78	<u>84.4</u>
5000	78	<u>85.6</u>
5500	88	74.8

In table 5.2.2, when the training set starts from the pure defensive fighter training set, the accuracy corresponding to defensive fighter training set is better.

In learning changing strategies, the RPROP's accuracy is better when the test sets are the offensive fighter test sets. The defensive fighter test sets yields the lower accuracy. And then, in Table 5.1.2, when the training sets start from pure defensive fighter's action, the accuracy is better than that of Table 5.1.1 which starts from pure offensive fighter's actions.

CHAPTER VI

DISCUSSIONS

This section has to follow the previous section, so there are discussion of learning static strategies and discussion of learning changing strategies.

6.1. Discussion of Learning Static Strategies

From the results, RPROP performs better than VFDT when the number of data increases and the accuracy of VFDT starts to increase slowly. In the small number of the actions, RPROP cannot show a good performance; however, practically, the small number always occurs only in the initial state of the learning system. In the CRPG, the number of character's actions always increases rapidly. For example, in Minigate environment, there are eight characters in the game and each character has its own action every turn. By average, the game ends in 10 turns, which means there are 80 character's actions in one game. And from the result, RPROP gives very good results if the number of training character's actions is not less than 300 which are only about four to seven game records, so it is suitable to classify the character's action.

The reason that RPROP is better than VFDT when the total number of data in training set increases is the useage of the pattern of character situation as the input. For an example, in table 6, the offensive fighter has 60 percentage of hit point, closest enemy distance of five units, no influence status, no local status, no potion of hit point, no potion of fire resistance and some potion of free action, so the inputs are $x_1 = 60$, $x_2 = 5$, $x_3 = 0$, $x_4 = 0$, $x_5 = 0$, $x_6 = 0$, $x_7 = 1$. These inputs are used to train the network. They are different inputs for every character situations.

Table 6.1 : Examples of offensive fighter database

HP	CED	IS	LS	Potion HP	Potion FR	Potion FA	Class
60	5	0	0	0	0	1	Attack_closest_enemy

For VFDT case, it has to calculate all character situations into the information gain or entropy for each of the attribute that is a value ranging from 0 to 1. And then VFDT uses the value to create the tree. In the way, the process does not use the primary data. So some data can be different but the value of the information gain or entropy are the same. Because of that, when the number of data is higher, the more possible that will happen is higher too. In addition, the secondary data which are the information gain of entropy only used to select the attribute that will use to create the next level of the tree. It does not do anything different if it does not change the sequence. In an example, if entropy of hit point is the highest, even it decreases but still is the highest; in this case, it is still do the same effect for the classification.

In learning static strategy, the defensive fighter test has a problem that attacking class and drinking potion of fire resistance class have very similar attributes. From human's view, drinking potion of fire resistance only occurs in the first turn when the character has a potion of fire resistance to drink but in the other hand, RPROP does not focus on the turn attribute enough while the other attributes are very similar. This might be caused by a very few number of drinking potion of fire resistance class in the training set. In the common way, this action rarely occurs in the game. However, the result should be better by using more balanced class database.

6.2. Discussion of Learning Changing Strategies

In the same reason as mentioned in learning static strategies, the defensive fighter's problem decreases accuracy in the changed strategy when using defensive fighter test set. The classification cannot distinguish between attacking class and drinking potion of fire resistance class in the test set, so the accuracy was quite low when compared to when the testing uses the offensive fighter test set. The accuracy should be improved by using balanced class training sets. However, the other idea is about the drinking potion of fire resistance action which should not be able to classify by human player. Because the defensive fighters always drink the potion of fire resistance in the first turn of the match if they have the potion of resistance in their inventories and no human player can judge the strategies of the opponents before seeing any previous actions. In addition, there is no way to see their inventories at all. So there is no way for the human player to classify this type of opponent action. Then by comparing the ML in the way that even human player cannot be able to do is not useful and this way might lead to the over power AI player which always ruin the balance of strategies in the every types of computer games. However, the experiments still have to maintain the reference format in order to compare the results in respect.

CHAPTER VII

CONCLUSION

This research is one attempt to find an adaptive AI in CRPG by using RPROP to improve the accuracy of classifying character's action. The computer game nowadays has been developed to have various styles. At the start, this research is for improving computer game content. The computer game contents which can be developed are graphic, sound, story, world and gameplay. The important content in this research is gameplay. It is the content that rules the game. Gameplay has a lot of types which are fighting, racing, sports, FPS, RTS, and RPG. Gameplay can be developed by different ways which are adding new strategy, mixing between two or more types of gameplay (for example, mixing FPS and RPG become Action RPG) or creating a new type of Gameplay. This research focuses on adding new strategy into Gameplay. Because the different types of Gameplay are not always compatible to developing way, RPG was the one that is focused on this research. RPG player usually must play when the game allows the player to play and then wait while the game allows the others to play until the next time when the game allows the player to play again – this style is also called turn-based game. The moment when the game allows the player to play is call the player's turn. The player must play and change to the other player turn. The turn-based game leaves a duration for the player to plan and play their strategy. This duration can be long for a while, so if it is the AI player's turn, it can also take its time a little bit to use its complicated computation – it still should not take a long time because the human player does not want to wait but it has more time when compared to some types of gameplay which use fast pace or real time interactive. On the other hand, RPG has grinding which force the player to play with the same enemies multiple times which are boring, so it is an inspiration to add new strategies to the enemies and it is also useful to have a lot of playing records in the game. This is the inspiration that an adaptive strategy AI is needed. The RPROP is chosen in this implementation because of its speed of learning and accuracy. The results show that RPROP is good to be used as character’s action classifier. In addition, to improve the AI in computer game and to provide more content in the gameplay, this research way must continue to make computer game become more entertaining.

REFERENCE

[1] M. C. Machado, E. P. C. Fantini and L. Chaimowicz, Player Modeling: Towards a Common Taxonomy, *The 16th International Conference on Computer Games*, pp. 50-57, 2011.

[2] I. Szita, M. Ponsen and P. Spronck, Effective and diverse adaptive game AI, *IEEE Transaction Computational Intelligence and AI in Game*, vol. 1, no. 1, pp.16-27, 2009.

[3] S. Yildirim and Sindre Berg Stene, A Survey on the Need and Use of AI in Game Agents, *Society for Computer Simulation International*, pp. 225-237, 2008.

[4] J. Furnkranz, Recent Advances in Machine Learning and Game Playing, *J. OGAI*, vol. 26, no. 2, 2007.

[5] M. Riedmiller and H. Braun, RPROP – a Fast Adaptive Learning Algorithm, *Proceeding of ISCIS VII*, 1992.

[6] M. Santos, J. A. Martín H., V. López and G. Botella, Dyna-H: A Heuristic Planning Reinforcement Learning Algorithm Applied to Role-playing Game Strategy Decision Systems, *Knowledge-base Systems*, vol. 32, pp. 28-36, 2012.

[7] M. Cutumisu, D. Szafron, M. Bowling and R. S. Sutton, Agent Learning using Action-dependent Learning Rates in Computer Role-playing Games, *Proceeding of the 4th Artificial Intelligence and Interactive Digital Entertainment Conference*, pp. 22-29, 2008.

[8] R. S. Sutton and A. G. Barto, Temporal-Difference Learning, *Reinforcement Learning: An Introduction*. Cambridge, Mass: MIT Press, 1998.

[9] P. Spronck, M. Ponsen, I. Sprinkhuizen-Kuyper, and E. Postma, Adaptive Game AI with Dynamic Scripting, *Machine Learning*, vol 63, no. 3, pp. 217-248, 2006.

[10] R. Rubinstein, The Cross-Entropy Method for Combinatorial and Continuous Optimization, _Methodology and Computer in Applied Probability_, vol. 1, pp. 127-190, 1999.

[11] R. M. M. Vallim and J. Gama, Data Stream Mining Algorithms for Building Decision Models in a Computer RolePlaying Game Simulation, _Brazilian Symposium on Games and Digital Entertainment_, pp. 108-116, 2010.

[12] P. Domingos and G. Hulten, Mining High-Speed Data Streams, _Proceeding of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining_, pp. 71-80, 2000.

[13] A. Mark, B. Sendhoff and H. Wersing, A Decision Making Framework for Game Playing Using Evolutionary Optimization and Learning, _Evolution Computation_, pp. 373-380, 2004.

[14] R. S. Sutton, Dyna, an integrated architecture for learning, planning, and reacting. _SIGART Bulletin_, vol. 2 issue 4, pp. 160–163, 1991.

[15] C. J. C. H. Watkins and P. Dayan, _Machine Learning_, Technical note Q-learning. vol. 8, issue 3-4, pp. 279, 1992.

[16] P. G. Patel, N. Carver and S. Rahimi, Tuning Computer Gaming Agents using Q-Learning, _Computer Science and Information Systems_, pp. 581-588, 2011.

[17] R. S. Sutton, Adapting Bias by Gradient Descent: An Incremental Version of Delta-Bar-Delta, _Proceedings of the 10th National Conference on AI_, pp. 171-176, 1992.

[18] P. Spronck, Minigate Environment, http://ilk.uvt.nl/~pspronck/minigate.html, 2007.

[19] P. Tan, M. Steinbach and V. Kumar, _Introduction to Data Mining_, International Edition, Addison Wesley, Pearson, 2006.

[20] S. Haykin, _Neural Network a Comprehensive foundation_, Second Edition, Prentice Hall, Pearson, 1999.

BIOGRAPHY

Piyachai Eamsukawat was born at Khonkean, Thailand. He received a bachelor degree of Biology Science from Chulalongkorn University. Now he is pursuing a Master Degree in Computer Science from Chulalongkorn University.