딥러닝이 적용된 게임 밸런스에 관한 연구

게임 기획 방법론의 관점으로

Game Elements Balancing using Deep Learning in Artificial Neural Network

전준현, Joonhyun Jeon*

요약 게임플레이어는 게임에서 수많은 적들을 만나고 싸우게 되는데, 이 때 너무나 손쉽게 이기거나 진다면 게임의 재미는 반감될 것이다. 그 반대로 너무 어렵게 이긴다거나 지는 것도 게임을 지루하게 만드는 요인으로 작용한다. 따라서 상대방과의 전투나 경쟁에서 아슬아슬하게 승리하는 긴장감을 주기 위해서는 게임의 밸런스가 잘 맞아야 한다. 그만큼 게임 밸런싱 작업은 게임의 재미와 가장 직접적으로 영향을 미치는 요소로 작용한다. 그리고 게임 밸런스만큼 중요한 것이 있는데, 그것은 플레이어에게 적절한 난이도의 상대를 계속 만나게 하는 것이다. 본 연구에서는 이러한 문제를 해결하려는 방법으로써 게임 밸런스에 딥러닝을 적용하여 지능 캐릭터가 플레이어를 통해 학습하고 스스로 플레이어의 난이도에 따라 자신의 난이도를 조절할 수 있도록 고안하였다. 이것이 활성화되면 게임 기획자나 개발자에게는 그만큼의 비용을 절약하는 동시에 플레이어에게는 항상 흥미로운 상대를 제공할 수 있는 획기적인 방법이 될 것이다.

Abstract Game balance settings are crucial to game design. Game balancing must take into account a large amount of numerical values, configuration data, and the relationship between elements. Once released and served, a game – even for a balanced game – often requires calibration according to the game player's preference. To achieve sustainability, game balance needs adjustment while allowing for small changes. In fact, from the producers' standpoint, game balance issue is a critical success factor in game production. Therefore, they often invest much time and capital in game design. However, if such a costly game cannot provide players with an appropriate level of difficulty, the game is more likely to fail. On the contrary, if the game successfully identifies the game players' propensity and performs self-balancing to provide appropriate difficulty levels, this will significantly reduce the likelihood of game failure, while at the same time increasing the lifecycle of the game. Accordingly, if a novel technology for game balancing is developed using artificial intelligence (AI) that offers personalized, intelligent, and customized service to individual game players, it would bring significant changes to the game production system.

핵심어: Game artificial intelligence, Deep Learning AI, Automated game balance, Game character balancing, NPC balancing

*주저자 : Konkuk University Human ICT Professor; e-mail: naturaljeon@konkuk.ac.kr
■ 접수일 : 2018년 7월 5일 / 심사일 : 2018년 7월 25일 / 게재확정일 : 2018년 8월 9일

1. Introduction

'Game Balance' is a method to determine the relationships and perspectives of the elements that constitute a game, and also implement them in computer programs. Game balance is closely related to the development of computer technology; thus, it is limited by the computer's performance or processing capacity. Computer games are recognized as one of the creative arts competing with the media industry - movies, novels, TV shows, or music. Importantly, the biggest difference (between game balance and creative arts) is that game development is a software development process. Computer games are a convergent software program where media arts that are visually provided along with gameplay are integrated with musical elements. Notwithstanding that computer games use cinematic techniques or insert storytelling for its dramatic effect, they are basically computer software consisting of computer programming languages such as C++ or Direct3D. For this reason, computer games are confined to the computing environment. In other words, they are only possible within the range of values that computers can represent[11].

In the past, balancing was subject to many constraints owing to system limitations. Still, game balancing relies on the conventional techniques even today, when we enjoy remarkably advanced computer technologies and systems. There are two main reasons why the current game balancing has not been fully developed. First, relevant knowledge or methods are not shared or disclosed because game design, considered confidential business information, cannot easily be publicized. In turn, it impedes rigorous research on the topic. Second, the characteristics of game production environment are such that game designers are required to immediately work on a project and participate without sufficient preparation time for research to attempt to achieve game balance. Therefore, game balancing sticks to the old methods, except in cases where the conventional methods of game balancing are not applicable.

For the configuration of an artificial neural network, this study uses "Weka", a data mining program that allows AI characters to calculate the players' combat patterns and determine their optimized behaviors and settings through the application of Multilayer perceptron.

The method proposed in this study was designed to self-adjust game difficulty levels by identifying the player's propensity (combat pattern, player character settings, etc.) across all game contents using AI characters. However, the detailed settings may vary depending on the genre characteristics or diversity of a game, which makes it difficult to apply in some games. This is because online games may run into unexpected circumstances (breaking into a lobby for games with AI characters or attacking a player character, for example) because of interactions between players, and player experientiality is established in diverse ways for standalone and multiplayer games. Therefore, the scope of this study will be limited to the games that use AI characters.

In addition, to improve the clarity and reliability of the study, the default values of commercialized games are used for the data.

2. Literature Review and Game Balance

In recent times, game engines and game programming technologies have made huge strides. New games are being released almost every week. However, not every game becomes popular among players. Even if a game contains all the necessary gameplay elements, the game might be disregarded by the public, unless it effectively provides a balance between the components. For this reason, further consideration should be given to the definition of a 'well-balanced game.'

In gameplay, one of the important elements that determine whether a game is well-balanced and well-organized involves "whether the player's ability serves as a direct influence over the player's achievement." In general, players who possess more ability must be able to achieve more success than players with less ability. Unexpected circumstances, in other words, stochastic events such as luck may occur. However, failure or success caused by recurrent events, unfortunate or fortunate, regardless of the player's ability, should not continue to occur. The results must fully reflect one's own ability[9]. Game balance has been adjusted through a repeating process of gameplay and modification. It was more common to exhaust time toward game development before releasing a game. To minimize the trial-and-error process, major game developers often run the Quality Assurances (QA) department. After the game release, necessary modifications are made to balance the game in the form of a patch. The reason there is no such thing as scientific and stereotypical rules or formulae for game balancing is that balancing a game is extremely difficult and it is a complex process. Game balancing involves many pairs of independent geometric variables. Thus, 'how to optimize in multidimensional space' is the key issue for game balancing. Game balancing does not have a 'formal' rule or method that controls such optimization. Instead, it often refers to a minimal number of abstract mathematical models or scenarios or uses modified versions. Optimization of multidimensional space has already been performed in various ways, among other scientific disciplines, and such techniques can be adopted by the game industry. These methods, however, cannot resolve all the problems facing game balancing. Playing games and fixing problems at first hand, of course, would be the most effective method. Unfortunately, it consumes too much time and resources, and errors are highly likely to occur nonetheless. Furthermore, game balancing is a complex concept that cannot be strictly defined by a clear-cut proposition such as "A is B." For example, the answers to questions – against what it balances: either balancing the game itself or balancing between players or some other ways of balancing – are rather subjective by nature. For these reasons, the term 'balancing' in games often reminds us of the adjustment associated with player characters (or Avatar) or a character's combat-related skills or default settings[12]. However, there are many games without characters such as puzzle or flight-simulated games. As such, owing to a wide variety of genres and formats of games, combat-related balance is insufficient to represent the whole game balance. There are many other elements to be determined, besides combat. In his book "Man, Play and Games", Roger Caillois describes four elements that constitute play: competition, chance, simulation, and vertigo. This confirms that competing against opponents is a critical factor in games. In addition, the unique structure of computer games allows such elements of competition to be better reflected in games.

Popular traditional games usually require a counterpart to compete with. By contrast, computer games provide a counterpart so that game users can enjoy solo play, where the counterpart provided is called an AI character or Non-Player Character (NPC). The success or failure of a game can be determined by how much tension and excitement this AI character can offer. Therefore, it is important to build an AI character who improves as a player goes up the levels so that the player remain interested in playing the game, and indeed, actual game production has heavily focused on this aspect. As if to prove this point, many studies have been conducted mainly on character-related topics. Such research includes analysis of monsters and their AI that are closely related to a character's battle experience. The findings of previous studies are as follows:

Oh Byeol defines game balance as fairness towards all players of different abilities in the Player versus Player (PvP) game. AI and Excel were used for statistical analysis that repeats testing, and a method of game balancing was described based on the resulting values[1].

Jeon Jun-hyeon and Jeong Ui-jun proposed a method to evaluate game balance by combining the character's Status value, combat skill, and profession into a numerical value[2].

Hyeon Hye-jeong and Kim Tae-sik conducted a more detailed analysis of combat situation and proposed a method to modify character settings by using combat length, strike range, possible attacks, and character abilities[3].

Son Hyeong-ryul and Lim Chang-ju described game balancing between characters by analyzing and comparing the value of each unit according to the attack types of a character based on battle scenarios[4].

Lim Chang-ju and Jin Shin proposed a method that adjusts skill parameters to attach value to skills the massively multiplayer online role-playing game (MMORPG) players use and to balance out among characters[5].

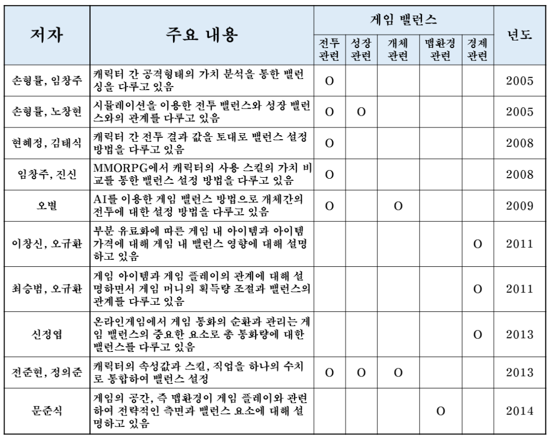
Shin Jeong-yeop conducted an analysis of in-game currency control model that takes a macroscopic approach to game balance and level design based on the in-game currency flow and the total currency in circulation[6].

In their study, Lee Chang-shin and Oh Gyu-hwan suggest that the price of an item is one of the vital elements for game balance, where the game items are regarded as a vital factor capable of destroying the balance of a game, ignoring the player's capability and expertise[7].

Choi Seung-beom and Oh Gyu-hwan covered the relationship between the control of in-game money earned and the game balance, while describing the relationship between game items and gameplay[8].

Moon Jun-sik describes the strategic aspects and applications of map environment - the space where a game takes play - in gameplay and explores the importance of game balance[9].

The findings of previous studies are summarized in Figure 1.



저자	주요 내용	게임 밸런스					년도
		전투 관련	성장 관련	개체 관련	맵환경 관련	경제 관련	
손형률, 임창주	캐릭터 간 공격형태의 가치 분석을 통한 밸런싱을 다루고 있음	O					2005
손형률, 노창현	시뮬레이션을 이용한 전투 밸런스와 성장 밸런스와의 관계를 다루고 있음	O	O				2005
현혜정, 김태식	캐릭터 간 전투 결과 값을 토대로 밸런스 설정 방법을 다루고 있음	O					2008
임창주, 진신	MMORPG에서 캐릭터의 사용 스킬의 가치 비교를 통한 밸런스 설정 방법을 다루고 있음	O					2008
오별	AI를 이용한 게임 밸런스 방법으로 개체간의 전투에 대한 설정 방법을 다루고 있음	O		O			2009
이창신, 오규환	부분 유료화에 따른 게임 내 아이템과 아이템 가격에 대해 게임 내 밸런스 영향에 대해 설명하고 있음					O	2011
최승범, 오규환	게임 아이템과 게임 플레이의 관계에 대해 설명하면서 게임 머니의 획득량 조절과 밸런스의 관계를 다루고 있음					O	2011
신정엽	온라인게임에서 게임 통화의 순환과 관리는 게임 밸런스의 중요한 요소로 총 통화량에 대한 밸런스를 다루고 있음					O	2013
전준현, 정의준	캐릭터의 속성값과 스킬, 직업을 하나의 수치로 통합하여 밸런스 설정	O	O	O			2013
문준식	게임의 공간, 즉 맵환경이 게임 플레이와 관련하여 전략적인 측면과 밸런스 요소에 대해 설명하고 있음				O		2014

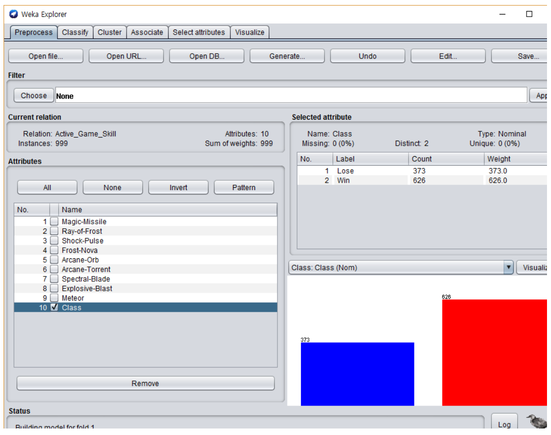
Figure 1. Previous research on game balance

If a game is defined as a competition that obeys fair rules, 'game balance' can be regarded as a state of equilibrium and harmony that has been reached through fairness in a game. Game balance, in the narrow sense, can be defined as fairness among in-game elements (the relationship between game characters, the values and settings between items, and the application of game objects). In a broader sense, a well-balanced game can be defined as a game in which the ideas reflecting the designer's intentions are arranged in a fair and harmonious manner.

3. A Study of Deep Learning Applications in Game Balance

This study will use the Weka framework that supports multilayer perceptrons and trains them with backpropagation for the purpose of full-scale neural network configuration.

To load a prepared dataset of combat skills in the Weka Explorer, click the Open file button in the menu bar. The data are excerpts from the Mage skills, one of the most played classes in RPGs. There are nine most frequently used combat skills, assigned to keypad 1 through 9, providing quick access to the skills. Taking into account the system status, this study analyzes the Mage abilities, mainly with the frequently used skills. Information about the nine skills can be loaded in the Skill.CSV data file as shown in Figure 2.



```
Filter: None

Current relation
Relation: Active_Game_Skill          Attributes: 10
Instances: 999                 Sum of weights: 999

Attributes
No.  Name
 1   Magic-Missile
 2   Ray-of-Frost
 3   Shock-Pulse
 4   Frost-Nova
 5   Arcane-Orb
 6   Arcane-Torrent
 7   Spectral-Blade
 8   Explosive-Blast
 9   Meteor
10 ✓ Class

Selected attribute
Name: Class                            Type: Nominal
Missing: 0 (0%)      Distinct: 2       Unique: 0 (0%)
No.  Label   Count   Weight
 1   Lose    373     373.0
 2   Win     626     626.0

Class: Class (Nom)

Status
Building model for fold 1
```

Figure 2. Dataset loading screen

Figure 2 shows that nine skills have been loaded, including "Magic-Missile" assigned to the corresponding key No. 1 through to "Meteor" for No. 9. The Class involves two labels: Lose and Win. The Win count is 373 and the Lose count is 626, indicating a higher weight on the Win.

The weights for all nine skills are exhibited in Figure 3. The three pillars, as shown in Figure 3, represent the Intelligent character, and Player 1 and Player 2, indicate the higher active weights placed on the higher counts. To put it plainly, these are the most frequently used skills among RPG players, and it is more advantageous to use a high weighted skill for victory.

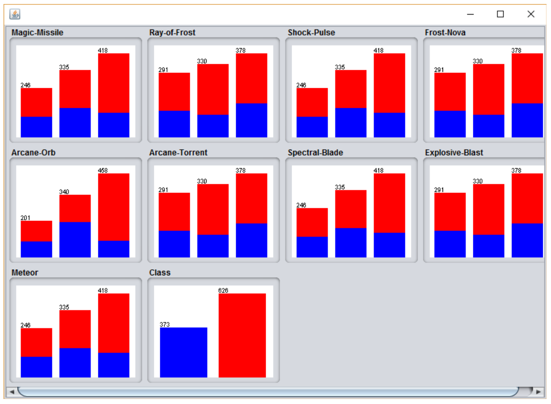


Figure 3. Active skill weights

To apply an artificial neural network to the nine skills, select "Classify" from the menu at the top of the Weka Explorer window, then set the Filter of "Classifier" to "Multiplayer Perceptron." The program will then create all active skills as input nodes and automatically generate the hidden layer nodes that connect Win and Lose of the output nodes. This allows the program to compute the weights for each skill. The weight is a probability that affects Win or Lose, an output node that indicates that selecting a skill with higher weight increases the chances of winning. Figure 4 shows the multilayer neural network configuration that contains a total of 27 nodes, where Intelligent character, and Player 1 and Player 2 have been setup to calculate the active weight of each skill.

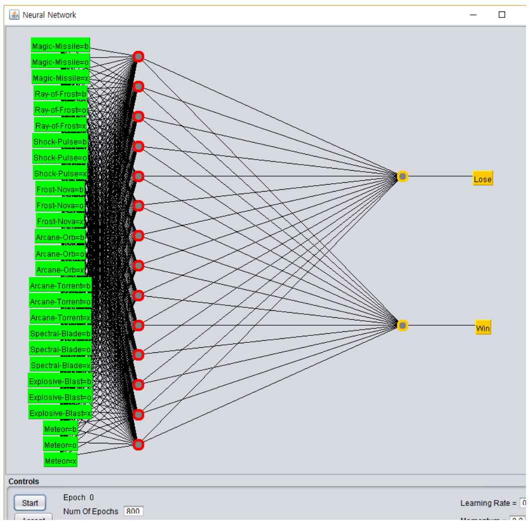
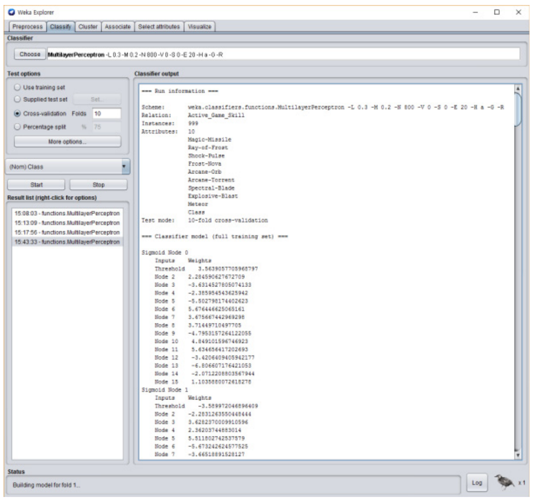


Figure 4. Multilayer neural network configuration for active skills

In Figure 4, the trainingTime is set to 800 times and values of "hidden Layers" are set to 'a' so that the program can create as many hidden layers as it needs. When you click "Start," then the program will automatically set the input nodes for all active skills and use the automatically created Hidden Layer to generate the values of the sigmoid function, as shown in Figure 5, for each label: Win and Lose.



MultilayerPerceptron -L 0.3 -M 0.2 -N 800 -V 0 -S 0 -E 20 -H a -G -R

Cross-validation Folds 10

15:08:03 - functions.MultilayerPerceptron
15:13:09 - functions.MultilayerPerceptron
15:17:56 - functions.MultilayerPerceptron
15:43:33 - functions.MultilayerPerceptron

```
=== Run information ===

Scheme:       weka.classifiers.functions.MultilayerPerceptron -L 0.3 -M 0.2 -N 800 -V 0 -S 0 -E 20 -H a -G -R
Relation:     Active_Game_Skill
Instances:    999
Attributes:   10
              Magic-Missile
              Ray-of-Frost
              Shock-Pulse
              Frost-Nova
              Arcane-Orb
              Arcane-Torrent
              Spectral-Blade
              Explosive-Blast
              Meteor
              Class
Test mode:    10-fold cross-validation

=== Classifier model (full training set) ===

Sigmoid Node 0
    Inputs    Weights
    Threshold    3.5639057705968797
    Node 2    2.284590627672709
    Node 3    -3.6314527805074133
    Node 4    -2.385954543625942
    Node 5    -5.502798174402623
    Node 6    5.676446625065161
    Node 7    3.675667442969298
    Node 8    3.71449710497705
    Node 9    -4.7953157264122055
    Node 10    4.849101596746923
    Node 11    5.634656417202693
    Node 12    -3.4206409405942177
    Node 13    -6.806607176421053
    Node 14    -2.0712208803567944
    Node 15    1.1035880072618278
Sigmoid Node 1
    Inputs    Weights
    Threshold    -3.589972046896409
    Node 2    -2.2831263550448444
    Node 3    3.6282370009910596
    Node 4    2.36203744883014
    Node 5    5.511802742537579
    Node 6    -5.673242624577525
    Node 7    -3.66518891528127
```

Building model for fold 1...

Figure 5. Sigmoid output resulting from active skill values

Figure 5 displays the results of executing the model, as shown below.

=== Run information ===

Time taken to build model: 211.24 seconds

The results show that sigmoid node 0 and 1 represent "Lose" and "Win," respectively, and the nodes between 2 and 15, which are the hidden layer nodes, indicate the weight values of each node. Such weights provide information about what value to choose for reward, such as Q-Value. Here, selecting the higher values means a more likelihood for reward. For example, Node 13 of the sigmoid node 1 that represents the Win node has the highest value of 6.81092749182954, which implies that Node 13 is most likely to win. At Node 13, Frost-Nova=x has the highest value of 3.649. In conclusion, Player 2 (coded as xin the program) is more likely to win the battle when Frost-Nova is the first skill used in combat. If Intelligent character can compute Q-Values in the same way as above and select higher values through the Monte Carlo tree search, then the optimal skill pattern can be created.

The resulting data can also be visualized in Weka. Select "Classify" from the menu available at the top of the window and set the filter to C4.5 Algorithm for data visualization.

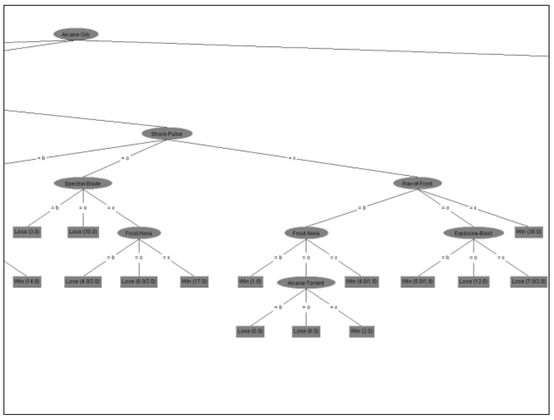


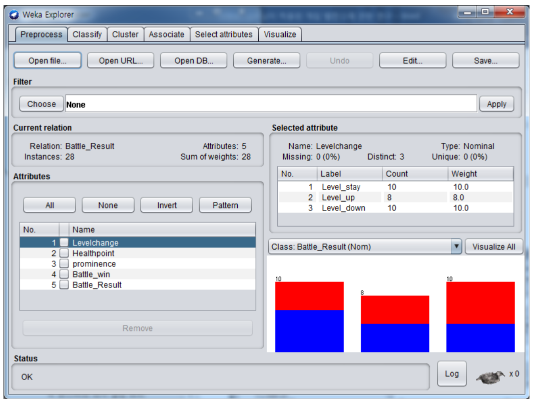
Figure 6. Visualization based on active skill pattern

Weka's visualization tools offer options that allow users to view the optimized order and value of the skills at a glance, as shown in Figure 6, along with the weight assigned to each skill.

In RPGs, the types of skills and attack power used by a player vary depending on character roles, but in a limited amount of time, the pattern of skill use can be determined according to the role of the opponent.

Next, the AI character should be able to level up itself, after recognizing that the player's level has improved. This is because higher-level attack skills cause more damage no matter how perfect the attack pattern is. Therefore, the AI characters should be designed in such a way that they can decide whether to level up or stay, based on the combat results after the battle ends.

Data use should be based on the skill data configuration available in 〈Appendix〉.



Filter: None

Current relation

Relation: Battle_Result   Attributes: 5
Instances: 28   Sum of weights: 28

Attributes

No.	Name
1	Levelchange
2	Healthpoint
3	prominence
4	Battle_win
5	Battle_Result

Selected attribute

Name: Levelchange   Type: Nominal
Missing: 0 (0%)   Distinct: 3   Unique: 0 (0%)

No.	Label	Count	Weight
1	Level_stay	10	10.0
2	Level_up	8	8.0
3	Level_down	10	10.0

Class: Battle_Result (Nom)

Visualize All

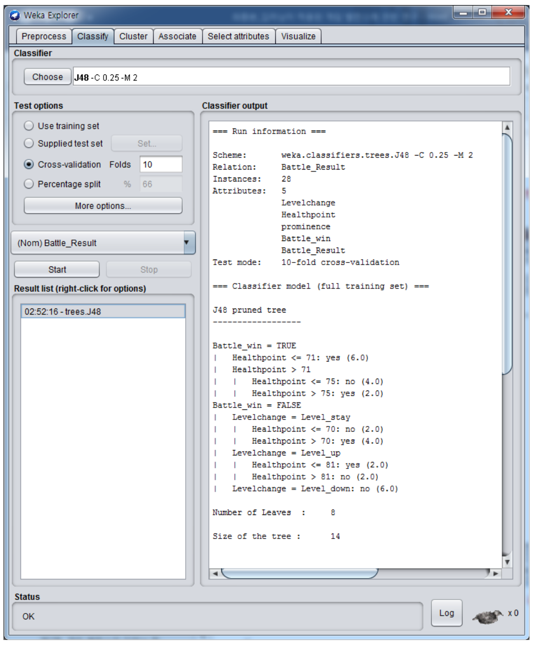
10
8
10

Status
OK

Figure 7. Preprocessing for AI character's auto level-up

Attributes consist of five elements: Levelchange, Healthpoint, Prominence, Battle_win, and Battle_Result. For data processing, the battle Log of the players is loaded in the CSV file, then it is converted into an arff file, a Weka file format.

Weka reads the data model to preprocess the data. In this way, information based on the data model can be obtained, as shown in Figure 7 where Levelchange has the highest value. In addition, for the selected attribute, Levelchange, the following three labels are produced as important elements: Level_stay, Level-up, and Level-down. To apply the decision tree to this data model, select J48 of Classifier from the Classify tab on top of the window.



Classifier: J48 -C 0.25 -M 2

Test options
- Use training set
- Supplied test set
- [✓ Cross-validation] Folds 10
- Percentage split % 66

(Nom) Battle_Result

Result list (right-click for options)
02:52:16 - trees.J48

Classifier output

```
=== Run information ===

Scheme:       weka.classifiers.trees.J48 -C 0.25 -M 2
Relation:     Battle_Result
Instances:    28
Attributes:   5
              Levelchange
              Healthpoint
              prominence
              Battle_win
              Battle_Result
Test mode:    10-fold cross-validation

=== Classifier model (full training set) ===

J48 pruned tree
------------------

Battle_win = TRUE
|   Healthpoint <= 71: yes (6.0)
|   Healthpoint > 71
|   |   Healthpoint <= 75: no (4.0)
|   |   Healthpoint > 75: yes (2.0)
Battle_win = FALSE
|   Levelchange = Level_stay
|   |   Healthpoint <= 70: no (2.0)
|   |   Healthpoint > 70: yes (4.0)
|   Levelchange = Level_up
|   |   Healthpoint <= 81: yes (2.0)
|   |   Healthpoint > 81: no (2.0)
|   Levelchange = Level_down: no (6.0)

Number of Leaves  :     8

Size of the tree :      14
```

Status
OK

Figure 8. Resulting values of decision tree for auto level-up

Once the program starts, the Run information can be obtained from the Classifier output window as shown in Figure 8. Ten-fold-Cross-Validation was used for data validation. This can be visualized as a decision tree shown in Figure 9.

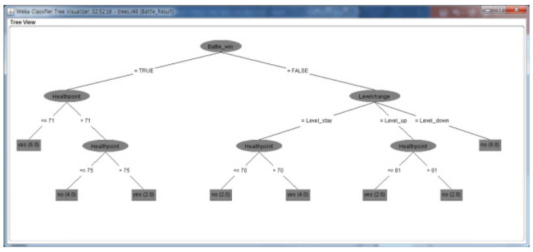


Figure 9. Decision tree resulting values based on battle results

The results in Figure 9 show that on the top tier, judgment will be made as to whether the battle was won or lost (Battle_win), based on battle results. If defeated, the AI character will decide to change its level (Levelchange), moving down along the line on the right hand (FALSE) side, but in the case of victory, it proceeds to the left (TRUE) side. When the battle is lost, judgment should be made as to whether the defeat was due to the attack pattern or level difference. Here, the Healthpoint serves as a criterion; if the character's healthpoint is greater than 70%, then the character maintains its current level. If greater than 80%, the character levels up. In sum, when engaged in PvP combat with other players, the AI character adjusts its own attack patterns and levels to consume 20% to 30% of the character's healthpoint.

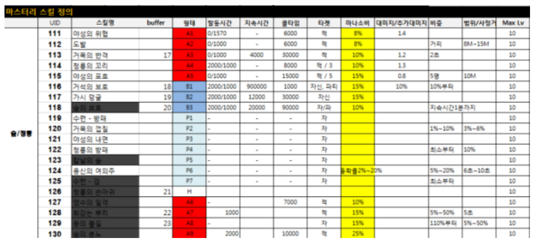
To apply the decision tree described above to a program, the source code must be extracted first.

Select the "Output source code" option from Test options in Weka to obtain the code available for the program. The application of this code to the program allows AI characters to learn and modify their attack patterns or adjust their own levels by making decisions based on the results of battle against other players.

4. Conclusions and Implications

Even in a complex RPG environment, it is possible to configure an environment such that the value of a player's decision and behavior pattern can be calculated by controlling a situation where variables associated with the player may occur. Indeed, for balancing MMORPGs, assuming that there is no possibility of movement, only attack and defense skills, depending on character roles, are taken into account, the possibility of movement does not have to be considered for game balancing in the environment where fair rules apply to all players. Therefore, each character skill would be the sole consideration. However, unlike other games, RPG has multiple skills and allows for a flexible skill arrangement varying with the player's UI settings. Therefore, the problem of 'how to compare game balance' still remains to be solved.

Balance comparison may be regarded by players as an alterable element which, however, is not. This is because balance comparison is, from a game producer's viewpoint, an internally fixed element designed to calculate the value of the optimized pattern and behavior such as Q-Values. As shown in Figure 10, each character skill has its own Unique ID value, and all skills consist of attack power, attack range, cast time, resource required for attacks, and cooldown time. Therefore, depending on the five inputs, the total amount of damage a player can inflict on an opponent or the defense (constitution-based) value will be measured and displayed.



마스터리 스킬 정의

	UID	스킬명	buffer	형태	발동시간	지속시간	쿨타임	타겟	마나소비	대미지/추가대미지	비중	범위/사정거	Max Lv
	111	야성의 위협		A1	0/1570	-	6000	적	8%	1.4			10
	112	도발		A2	0/1000	-	6000	적	8%		거리	8M~15M	10
	113	거목의 반격	17	A3	0/1000	4000	30000	적	10%	1.2	2초		10
	114	청룡의 꼬리		A4	2000/1000	-	8000	적 / 3	10%	1.3			10
	115	야성의 포효		A5	0/1000	-	15000	적 / 5	15%	0.8	5명	10M	10
	116	거석의 보호	18	B1	2000/1000	900000	1000	자신, 파티	15%	10%	10%부터		10
	117	가시 덩굴	19	B2	2000/1000	12000	30000	자신	15%				10
	118	숲의 보호	20	B3	2000/1000	20000	90000	자/파	10%		지속시간1분까지		10
	119	수련 - 방패		P1	-	-	-	자					10
숲/정령	120	거목의 껍질		P2	-	-	-	자			1%~10%	3%~6%	10
	121	야성의 내면		P3	-	-	-	자					10
	122	청룡의 방패		P4	-	-	-	자			최소부터	10%	10
	123	칼날의 숲		P5	-	-	-	자					10
	124	용신의 여의주		P6	-	-	-	자	등확률2%~20%		5%~20%	6초~10초	10
	125	수련 - 검		P7	-	-	-	자			최소부터		10
	126	청룡의 손아귀	21	H									10
	127	맹수의 일격		A6	-		7000	적	10%				10
	128	휘감는 뿌리	22	A7	1000			적	15%		5%~50%	5초	10
	129	용의 불길	23	A8	-			자	15%		110%부터	5%~50%	10
	130	숲의 분노		A9	2000		10000	적	25%				10

Figure 10. Worksheet for character skill settings balance

One of the most common ways to compare game balance in actual games is to convert each skill into DPS for comparison purposes. A higher DPS usually means more damage to an opponent per second. Therefore, if fighting a battle for the same amount of time, a player with a higher DPS will be more likely to win because the player can inflict more damage to the opponent. However, higher DPS does not necessarily guarantee victory. A character with a higher DPS, but lower stamina could die from a low DPS attack. As such, balancing the value of skills has been noted as a tricky issue owing to many possible outcomes for the attack and defense skills in RPGs. For these reasons, in "A Pilot Study of MMORPG Combat Balance Evaluation Model" by Jeon Joon-hyeon, et al., a character's 'constitution' was determined to be the most important value, and the values of all skills were converted into a single value related to the 'constitution' for comparison purposes. Their study has implications for game balance research because it combined the complex balance values into a single value for easy comparison based on the settings where the constitution attribute is considered preeminent.

Unfortunately, their study has limitations in that it requires manual settings by a game producer. If artificial intelligence can be put to use in such settings, then it will facilitate an objective comparison based on numeric values. In RPGs, it is pointless to compare only individual skills. This can be explained by the fact that the resulting values will vary depending on the combination of a character's skills, even if an individual's skill is well-balanced. Figure 12 shows a pattern of in-combat skills used by a Mage character in World of Warcraft.'This pattern is based on a combination of skills that can strike a fatal blow to the opponent while defending oneself. Fstart, Pyro, Conf, MI, IF, and ROP are abbreviations of the corresponding skill.

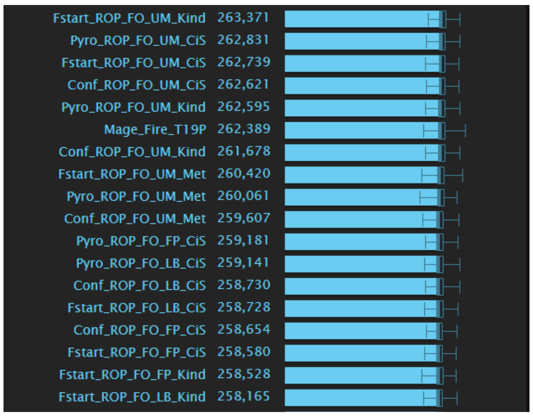


Figure 11. Mage skill pattern in World of Warcraft

Figure 11 shows various ways of combining the individual skills. A skill combo usually consists of the initial skill and the following one that maximizes the previous skill effect. For example, ROP is the most frequently used skill after Fstart (Fstart__ROP__), which is often followed by FO (__ROP__FO__). Such combination is indicative of optimal behavior for the next skill, following a certain skill.

Suppose there are about four candidate skills optimized for the second slot in a skill combo, if the values of the four skills such as Q-Values discussed in Chapter 3 can be computed individually and the NPC skill pattern optimized for the combination of such patterns can be determined, then the general pattern of players will change; accordingly, the NPC skill patterns will automatically change, allowing an artificial intelligence character to adjust its own difficulty level. In the study of Shin Yong-woo and Jeong Tae-Chung, similar to this concept, an attempt was made to predict a player's behavior by using sequential prediction and N-grams models. In this process, a Pattern can be recognized with sequential prediction that involves predicting the next value for a given Sequence of certain values, and the next value can be predicted through the N-grams model. In other words, when a set of skills determined by a player are denoted by 1 2 3 4 3 4 3 4, the counterattack can be made by considering the high probability of attack 4, following the 1 2 3 sequence because the number 4 is highly likely to come after 3. In this case, however, even though the optimal pattern can be identified through reinforcement learning, based on a simple probability, the value of the pattern cannot be observed and it is difficult to make a weighted selection for each behavior. Another limitation is that, considering the actual game production conditions, there are too many possible combinations of skills despite the predicted values obtained from the pattern. In fact, in the case of AlphaGo, the Monte Carlo tree search (MCTS) was used to overcome the limitations. Therefore, this study should also find the optimal pattern that allows the AI character to predict the player's behavior while making the player feel a sense of tension[13]. In addition, the following design strategy is proposed to ensure that the skill pattern of each character maintains a balance between DSP and constitution.

Step 1: Compute the total sum of DPS and constitution (output values) from the combination of skills for each character (input values) by using Q-Value algorithm.

Step 2: Enter the skill combo pattern intended by the designer as an input value for the input node, then enter the total sum generated from Step 1 into the output node. Calculate the value of skill combo pattern obtained from the hidden layer node to generate the numerical values.

Step 3: Based on the pattern obtained from Step 2, allow NPC to learn the optimal pattern by using the Monte Carlo tree search.

Step 4: Collect the combat Log (a record of gameplay) among players and compare with the pattern entered. Design it in such a way that it can vary with the player's pattern, based on reinforcement learning through backpropagation.

This study proposes a method of game balancing using deep learning techniques. The proposed method will save time and effort in the process where a player character is first set up and accordingly balanced as intended. In addition, the proposed method does not require a complete definition of the AI character's behaviors and conditions that uses finite-state machine (FSM) or flowcharts; rather, the AI character would adjust the difficulty level tailored to its player through self-learning from the player's behavior and decision making. This will always engage players with new competitors as if they encounter different opponents.

This study will also significantly save time and costs required for game balancing in the related industries. Much of game design costs are labor related. However, the permanency of labor is difficult to guarantee compared with the spending. Human beings can make mistakes and leave the company owing to sickness or unexpected events. On the other hand, programs make relatively less mistakes or have less of a factor that lowers the permanency compared to humans. In this sense, this study will deliver significant cost savings from the design sector while facilitating reinvestment in other sectors, leading to a higher success rate of new game introductions in the market. In addition, If the user's physical ability is numerically applied in a medical game field where a game level system is used. It could be used in various fields including the medical field.

In the academic domain, such attempts at artificial intelligence applications and innovations will serve as a starting point for further research on game balancing design. Moreover, the game engine using Q-Value and Deep Learning techniques will be commercialized so that everyone can compare the values of skills they intend to design.

참고문헌

[1] 오병. AI를 이용한 게임 밸런스 방법. 정보과학회지. 27(10). 한국정보과학회. pp. 25-28. 2009

[2] 전준현, 김동은, 정의준. MMORPG 전투밸런스 평가모델 기초연구. 한국컴퓨터게임학회논문지. 26(3). 한국컴퓨터게임학회. pp. 49-60. 2013

[3] 현혜정, 김태식. 게임 밸런싱을 위한 효과적인 캐릭터 조절 알고리즘. 한국콘텐츠학회논문지. 8(1). 한국콘텐츠학회. pp. 339-347. 2008.

[4] 손형률, 임창주. 전투시나리오를 기반으로 한 유니트 런싱: WAP 게임 D.N.A.의 적용 사례. 한국컴퓨터게임학회논문지. (7). 한국컴퓨터게임학회. pp. 73-78. 2005.

[5] 신정엽, 스마일 게이트. 온라인게임의 게임통화 관리모델 연구. 한국게임학회논문지. 13(5). 한국게임학회. pp. 5-18. 2013.

[6] 이창신, 오규환. 멀티 플레이 지원 게임의 온라인 게임 전환을 위한 디자인 이슈 연구. 한국컴퓨터게임학회논문지. 24(4). 한국컴퓨터게임학회. pp. 153-162. 2011.

[7] 최승범, 오규환. 온라인 게임에서 게임머니 판매모델을 위한 게임 디자인. 한국컴퓨터게임학회논문지. 24(2). 한국컴퓨터게임학회. pp. 127-136. 2011.

[8] 문준식. FPS 게임의 공간 디자인을 위한 VAE 모델 적용에 관한 연구. 디자인융복합논문지. 13(3). 디자인 융복합학회. pp. 77-90. 2014.

[9] Schell, J. The Art of Game Design: A Book of Lenses (2nd ed.). San Francisco, CA: CRC press. 2014.

[10] Rollings, A. and Adams, E. Andrew Rollings and Ernest Adams on game design (1st Ed.). Indianapolis, Ind: New Riders. 2003.

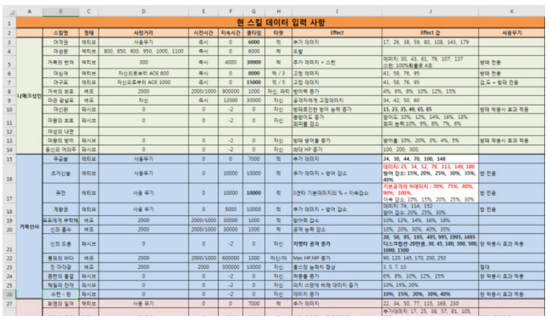
[11] Adams, E. Fundamentals of game design (3rd Ed.). Indianapolis, Ind: New Riders. 2013.

[12] Csikszentmihalyi, M. Flow: the psychology of optimal experience. New York, USA: Harper Perennial Modern Classics. 2008.

[13] Silver, D. and Huang, A. et al. Mastering the game of Go with deep neural networks and tree search. Nature. 529(7587). Nature Press. pp. 484-489. 2016

[14] Caillois, R. Les jeux et les hommes. Paris, FRA: Gallimard Education. 1992.

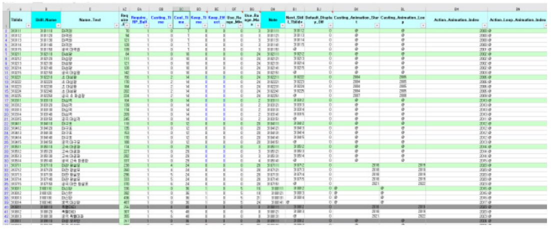
〈Appendix〉



현 스킬 데이터 입력 사항

	스킬명	형태	사정거리	시전시간	지속시간	쿨타임	타겟	Effect	Effect 값	사용무기
나메크성인	마격권	액티브	사용무기	즉시	0	6000	적	추가 데미지	17, 26, 38, 59, 80, 108, 143, 179	
	마성광	액티브	800, 850, 900, 950, 1000, 1100	즉시	0	6000	적	도발		
	거룩의 반격	액티브	300	즉시	4000	30000	적	추가 데미지 + 스턴	데미지: 30, 43, 61, 79, 107, 137 스턴: 100%확률로 4초	방패 전용
	마심력	액티브	자신으로부터 AOE 800	즉시	0	8000	적 / 3	고정 데미지	41, 58, 76, 95	방패 전용
	마구표	액티브	자신으로부터 AOE 1000	즉시	0	15000	적 / 5	고정 데미지	41, 58, 76, 95	검,도 + 방패 전용
	거석의 보호	버프	2000	2000/1000	900000	1000	자신, 파티	방어력 증가	4%, 6%, 8%, 10%, 12%, 15%	
	마관 팔살표	버프	자신	즉시	12000	30000	자신	공격자에게 고정데미지	34, 42, 50, 60	
	마신완	패시브	0	0	-2	0	자신	방패로인한 방어 능력 증가	15, 23, 35, 49, 65, 85	방패 착용시 효과 적용
	마왕의 보호	패시브	0	0	-2	0	자신	총방어도 증가 회피율 감소	방어도: 10%, 12%, 14%, 16%, 18% 회피 능력:10%, 9%, 8%, 7%, 6%	
	야성의 내면									
	마왕의 방어	패시브	0	0	-2	0	자신	방패 방어률 증가	방어률: 10%, 20%, 3%, 4%, 5%	방패 착용시 효과 적용
	용신의 여의주	패시브	0	0	-2	0	자신	최대 HP 증가	100, 200, 300,	
가북선사	무공쌍	액티브	사용무기	0	0	7000	적	추가 데미지	24, 30, 44, 70, 100, 148	
	초거신쌍	액티브	사용무기	0	10000	10000	적	추가 데미지 + 방어 감소	데미지: 25, 34, 52, 79, 111, 149, 180 방어 감소: 15%, 20%, 25%, 30%, 35%, 40%	창 전용
	류전	액티브	사용 무기	0	10000	10000	적	3연타 기본데미지의 % + 이속감소	기본공격의 %데미지 : 70%, 75%, 80%, 90%, 100%, 이속 감소: 10%, 15%, 20%, 25%, 30%	창 전용
	계왕권	액티브	사용 무기	0	5000	10000	적	추가 데미지 + 방어 감소	데미지: 74, 114, 152 방어 감소: 20%, 25%, 30%	창 전용
	묘묘에게 부탁해	버프	2000	2000/1000	30000	1000	적	방어력 감소	10%, 12%, 14%, 16%, 18%	
	신의 홍수	버프	2000	2000/1000	30000	1000	적	공격 능력 감소	10%, 20%, 30%, 40%, 35%	
	신의 도움	패시브	0	0	-2	0	자신	치명타 공격 증가	20, 50, 95, 195, 495, 995, 1995, 3495 디스크립션-20만큼. 30, 45, 100, 300, 500, 1000, 1500	창 착용시 효과 적용
	풍요의 바다	버프	2000	2000/1000	600000	1000	자신/아	Max HP,MP 증가	90, 120, 145, 170, 200, 250	
	진 마각광	버프	2000	2000	300000	10000	자신	몰스탯 능력치 향상	3, 5, 7, 10	절대
	혼란의 물결	패시브	0	0	-2	0	자신	적중률 증가	6%, 8%, 10%, 12%, 15%	창 착용시 효과 적용
	해일의 잔재	패시브	0	0	-2	0	자신	의지 스탯에 비례 데미지 증가	10%, 15%, 20%	
	수련 - 창	패시브	0	0	-2	0	자신	데미지 증가	10%, 15%, 20%, 30%, 40%	창 착용시 효과 적용
	화염의 일격	액티브	사용 무기	0	0	7000	적	추가 데미지	22, 34, 50, 77, 115, 169, 230	
									추가데미지: 17, 25, 38, 57, 81, 105,	

#1 스킬 공격 패턴에 사용한 스킬 데이터 사항



TblIdx	Skill_Name	Name_Text	Require	Require_HP_Ball	Casting_Time	Cool_Time	Keep_Time	Keep_Effect	Use_Range_Min	Use_Range_Max	Note	Next_Skill_TblIdx	Default_Display_Off	Casting_Animation_Start	Casting_Animation_Loop	Action_Animation_Index	Action_Loop_Animation_Index
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

#2 스킬 데이터의 서버 데이터