International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 8 Issue 05, May-2019

Self-Learning Games using Neural Networks

Lalithadevi. B, Sourabh Prakash Pati, Dhruv Patel, Nihir Sharma
Asst. Professor, Student, Student, Student
Department of Computer Science and Engineering
SRM Institute of Science and Technology,
Chennai, India

Abstract: **This paper focuses on, how we can inculcate the modern advancements in the field of machine learning into the video games to model the behavior of expert players into the game AI to give the players a very realistic and natural challenge. Customarily developers would program their game and behavior of the element/characters. There is no fun in playing the game once the pattern in which Artificial intelligence plays is figured out. In a player versus player or multiplayer game, your opponents are real people. These players are not bound by rules and they make mistakes. These mistakes or miscalculations provide a chance or for other players. The challenge and hence the actual fun which comes from such a multiplayer game is the real give and take that comes as a result of human interaction. It may be noted even from a quick literature review that the application of neural networks in mainstream commercial games is still quite non-existent and that the variety of neural networks used is extremely limited. The most widely used neural network is the error back-propagation algorithm since it is the most well-known. However, the use of artificial neural networks in digital games such as Othello, Mastermind, Backgammon and Checkers (Draughts), is not uncommon and has been popular and successful in many cases such as with Big Blue. However, the use of artificial neural networks in this genre of games mostly pays attention on strategy and these games often are slow paced. Modern digital games generally have more dynamic environments and the CPU has to deal with a ton of stuff other than just the AI. The standard rivals in a modern game make use of a series of hand coded rules and a finite-state machine. This type of system has many drawbacks including a huge amount of cumbersome work to manually program each rule and as a result a high level of predictability of the opponents. The mimicking of the player versus player experience of human vs. human combat generally involves a large amount of tuning and adjusting for game balance. Owing of the complexity of the problem, most single player games rather focus on plot and other game types. With the inclusion of different game play each time a game is played or at least more realistic or human like AI counterpart on the game playing against or along with them is sure to make games a lot more engaging. The goal is to make games where the AI makes the same mistakes like smashing into walls on turns or/and hit other vehicles to get ahead of them just as any real players would do, if it's a racing game. If it's a shooting game, the AI in the game knows what would be the best places to hide or take cover, which locations are good to do sniping, which weapon will be the best choice at long range combat or which weapon can deal the highest damage in near combat or when opponent is at mid-range. Recent advances in machine learning have made is possible for agents to very accurately learn new rules from a set of examples. We use these machine learning algorithms, especially deep learning to model an AI player in a modern game by sampling data obtained by recording the gameplay of an expert player. With the application of this system, the programmer would have a lot less work to do when hand coding the rules for combat and an addition advantage being that the learned behaviors are very likely to be more unpredictable and a lot life-like compared to any type of hard-wired finite state machine.**

IndexTerms – Artificial Intelligence, Video Games, Neural Networks, Machine Learning

I. INTRODUCTION

Almost all modern games especially in the genres of racing, first person shooters, open world and the likes are largely devoid of any realistic artificial intelligence. In most of the shooting games, the enemies simply walk towards the player while there guns are blazing, until they die. This may have its own merits as a game type in and of itself. However, there is big gap for other types of games that are more interactive and lifelike with opponents that provide a real challenge. Recently game developers began noticing this deficit and games such has UnReal©, Half-Life, © Call of Duty© and Halo© have popped up. These games have successfully used simple finite-state machines and expert-based systems to simulate the illusion of an intelligent enemy. As a result, these well-established algorithms have helped shape the game Artificial Intelligence (AI) and advancements in leaps and bounds in recent years. Even so, there is still a lot more to be done. The AI is never good enough for the experienced player. For the AI to be good enough, it must keep challenging the player and also adapt to the player's style of gameplay and also, if possible learn from the player. Presently there is no such behavior of the AI agents in the realm of the modern games. A good player learns the behavior of the enemy AI and begins exploiting its weaknesses. The exploitation of the flaws of the AI should be an aspect of the game, but the game must also keep challenging skilled players. While modeling such systems, one thing to keep in mind is to account for the different levels of skills of different players and incorporate difficulty auto adjustment systems into the choices and behavior patterns of the AI rivals. This paper focuses on using a deep learning system to model player behavior. However, one must keep in mind that any such system needs to be designed with enough overrides and hooks so as to ensure game balance. There are many different machine learning algorithms available to choose from, some of which excel in certain domains while others don't. It's always better to frame the problem in hand as much as possible and pick the suitable machine learning algorithm, rather than hastily picking one. After we determine the appropriate data set, we need to gather plenty of samples and go ahead and try some of the standard algorithms. It will be more effective to use the more recently considered algorithms like genetic algorithms or neural networks. This

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 8 Issue 05, May-2019

will help the developers to introduce the unpredictability factor to the AI players. This will certainly also help in modeling expert players with minimal specifications of rules. This in its very essence means that it will virtually be effortless to model complex behavior of a racer or sniper or any other class of gamer into the AI agent without needing any predetermined rules.

Observing the current trends in the world of gaming, there couldn't be a better time to introduce these current and advanced machine leaning algorithms into the gaming industry. The revenue from video games across the globe is estimated to be $81.5B solely in 2014 and over the years it has been steadily increasing without any signs of falling anytime soon. In terms of gameplay, this is a very competitive market with key factor being the quality of their AI routines. In order to gain a competitive advantage over the traditional hand coded expert systems, a games need to have a variety of unique and distinct behaviors of its AI agents. So, apart from being highly profitable to developers, it has now become feasible to run on modern hardware. Owing to the recent improvements in hardware and memory devices, it now cheap and viable to not only run these systems on Personal Computers, but also do it during the gameplay. Implementation of Neural Networks in PC games would seem too farfetched during the late nineties but now it has already been done and successfully so in games like "Creatures" and "Black and White".

II. RELATED WORK

Although not in mainstream genres like shooting, racing and open world games, a few developers have taken to the realm of self-learning games and have made great use of the modern machine learning algorithms to produce games that are a genre in themselves. Some of the work done is realm include the Stealth Horror thriller "Hello Neighbor" developed by Dynamic Pixels. This very modern game, released for Windows and Xbox in December 2017 and for play station, Nintendo and Android in July 2018 includes an advanced AI that learns for every move of the player. The goal of the game is to find the treasure that your neighbor is hiding in his basement. The game AI becomes smarter with each move of the player and places traps at more strategic locations to outplay the player or even finds shortcuts to catch the player if he tries to escape. The game boasts a procedural AI and Pixar-style visuals with a extremely tense gameplay. Another notable work in the field is the Sci-Fi horror game "Echo. In this game, the player is surrounded by copies of himself which are called "echoes" in the game. These "echoes" learn from the player. If the player shoots at the "echoes", they will learn how to use a gun. If the player stealthily gets behind their back, they too will learn to do so. The echoes can learn to do anything that the player does in the game. While most games provide abilities to overcome obstacles, echo in unique due to the fact that your abilities are your obstacles here. In traditional games, improvisation and creativity assets of the player solely, but when anything you do can be learned by your AI opponents, that advantage doesn't help out anymore. Another such work is undertaken by Microsoft Studios in the FORZA racing series. The series gathers data about how the players drive, and even have the capability to make unique individual "drivatars" that use deep learning in order to imitate the way the player plays. This gives a very realistic challenge from the fellow AI racers.

III. PROBLEM AND REQUIRED DATA

Machine learning is primarily concerned with building a computer program that has the ability to improve its own performance on a particular task by help of examples. A sub branch of this field, known as deep learning tries to mimic the natural process of learning in the human brain, by making use of artificial neural networks inspired by the network of neurons present in the human brain. There are various algorithms associated with machine learning. To choose the ones that will be best suited for the problem at hand, need the though study of the problem and the desired output set along with input set of the problem. A task may be thought of as a function to be learned. This function should take responsibility of accepting input and producing output based on that input. This can be better explained by means of an example. Consider a simple shooting game. At any instance the player must make a decision about whether he should move forwards or backwards, whether to open fire or to hide. The input to the algorithm would be a vector containing the information about how many opponents are in the vicinity of the player, how much ammo the player has and any other relevant information. Based on this input, the player needs to take a decision on what to do next. This may vaguely be thought of as a classification problem. Very often, the decisions made by the human players, at any instance of this kind of games is unexplainable except by the means of examples. So the problem that arises is that the input/output pairs can easily be determined by it is very difficult to determine a concise relationship between the input parameters and the desired output. The best that can be done is to sample the actions of the AI agent and based on its performance in the game, we can decide whether the decisions taken by it are examples of correct or incorrect decisions. Learning by means of such examples is what machine learning does. So by providing such input/output pairs to a deep learning algorithm, the internal structure of the neural network can be adjusted to capture the relationship between those input and output pairs. This can further help the machine learner to produce a function that approximates this implicit relationship between input/output pairs provided in the examples. In order to apply a machine learning algorithm to the task in hand it is first essential to determine the features that are relevant to the task. It is also essential to determine what we want the machine learning model to learn from the examples that we provide it with. It is fairly simple to take a well-defined game, add some control structure to the machine learning agent and begin training. But, unfortunately, the modern games, especially that of the popular genres, are neither simple nor well-defined. There is no particularly correct strategy to follow in such games. Not even the best players in such games would be able to completely quantify their tactics. The most they can to do is give tips on the basics of the games. Even then these tips or rules are not absolute.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 8 Issue 05, May-2019

There are many exceptions, to even any of those basic rules, which makes it very tedious to hard code any type of rule system. The very first step would be to figure out what is the input that our program has access to and what is the output that we are looking for. Although it may seem obvious that our learning system would account for each and every decision that out AI agent shall take at any instance, but it is not quite so. There will always be a need to allow some amount of control on the programmer's side. This is due to the exceptions that are likely to arise to the rules that our agent learns from the expert players. A shooting game agent may learn to avoid slipping off of edges but sometimes it may be strategically preferable to jump over it. So it's a good idea to decide before hand, what rules we need to hand code into the system and which ones we want it to learn by itself from the expert players. Although the concepts discussed in this paper are readily applicable to any game, for the sake of discussion I shall consider a hypothetical shooting game in further describing the problem. In a basic shooting game, there are a handful of quantifiable actions that are really the core of the game. For the sake of simplicity, let us consider only the basic actions like accelerating, decelerating, moving forwards and backwards, facing forwards or backwards and finally whether to jump or not. The general decisions taken by the AI agent can be divided into a sequence of moves or combinations of moves. To achieve a realistic and challenging character, developers can model the decisions of the agent based on those take by an actual player. This task now becomes a classic machine learning problem. The environment of the game needs to be represented by a rich set of features.

IV. SET OF FEATURES

We will consider only the basic moments and actions, namely, accelerate and decelerate direction to move, face direction, and whether or not to jump. These actions will constitute the output set of the data samples. Once we have defined our input and output sets, we must identify the useful and useless information that will help our agent in taking those decisions. In an open world battle royal type shooting games, a huge amount of information is available at the agent's dispatch, but most of it is superfluous and won't help our agent to take meaningful decisions or learning the strategies of the player. Perhaps the location of the enemies is the most important piece of information for the agent in such a game. Since this piece of information is central to the performance of our agent, let us examine it closely by dividing the game environment into various sectors with our agent at the center. The spatial data about the locations of the enemies can be determined by breaking up the world into four sectors around our agent as shown in figure 1.

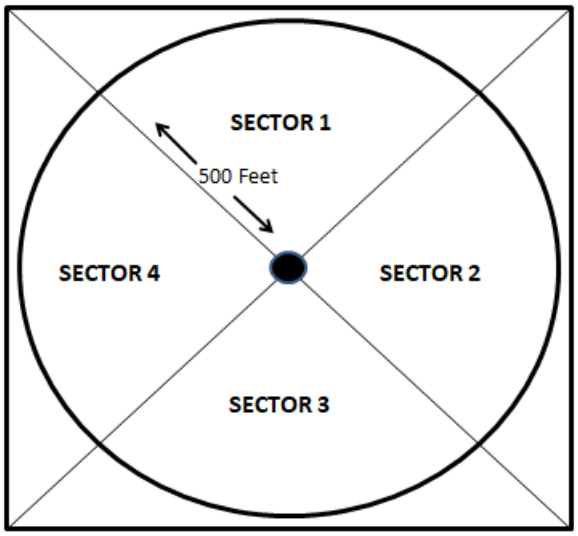

Fig 1: Game environment divided into sectors with player in middle

As the sectors are made around the player, the information about these sectors becomes relative to the player. For instance, the area in front of the player is represented by sector 1. The actual position in the world is not as important for the agent as information about its immediate surroundings. With this system, the data about the required objectives and enemy locations can be represented by sector co-ordinates. If objectives are due west of the player, then it needs to move in sector 3. Also the maximum radius of the sectors has been constrained to 500 feet which gives a better perception of the maximum viewing distance of the player. Another key information that our agent needs to keep track of is his health. The actions of our human player will vary significantly when his health is low compared to when it is high. We need our model to understand this difference.

V. DATA SET EXTRACTION

After the determination of the feature set, we can run the game and collect the required data samples. The output that comes from any set of features will be any of the basic decisions discussed earlier. The set of features is measured in each game frame (roughly 100 milliseconds intervals). Each sample produces a outcome which occurs about 50 milliseconds later. This outcome is nothing but the decision to do one or more of the following basic actions in that instance:

- To move front or back
- To face forwards or backwards
- Whether to jump or not to jump
- Whether to accelerate or not

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 8 Issue 05, May-2019

VI. LEARNING FROM THE SAMPLE DATA

We have collected the required data by sampling the decisions taken by the expert player in each frame. When we sample a game frame, the data like locations of enemies nearby or with within a fixed radius can be easily accesses. Then 3-D geometry (vector mathematics) can help us find the positions of those enemies relative to our AI agent. We extend this technique to every sector features or information. The movement and facing direction data can both be easily be learned by observing the same of our expert player. The output is computed based on input of the feature vector from the sample. The acceleration, movement, facing direction and jumping of the expert player is recorded. This is the part that will represent the decisions made by player in the feature vector. Doing this, we have obtained both the input features and the output features and hence have obtained a complete feature vector. Once we have saved this, the collection of samples will serve as our training and testing data for our deep learning algorithms.

VII. ANN

An artificial neural network (ANN) is a network of artificial neurons that learn by using a set of training data to regress through the examples obtaining more complex features with each hidden layer from the data and learning in a non-linear fashion. The actual algorithm that would be most effective to use depends on the features of the game to which it is being applied. But for this paper we will consider the most popular deep learning algorithm which is generally suitable for most of the cases, namely the back-propagation algorithm.

Back-Propagation Algorithm

Backpropagation(dataset, e_{ta}, N_{inp}, N_{out}, N_{hidden}) Data in the dataset is (x,t), e_{ta} be the rate of learning, N_{inp} be the size of input layer, N_{tout} of the output layer, and N_{hid} of the hidden layer

- Make a feed forward network with N_{inp} inputs, N_{out} outputs, and N_{hid} hidden nodes. Initialize each W_{ij} with some small random value
- Till error value is less than threshold, repeat:
- For each (**x**, t) in dataset:
- Take instance **x** as input and compute outputs o^u for each node (Forward propagation of activation)
- Propagate the errors backwards through the neural network
- For each output unit *k*, calculate the error term: $delta^k = o^k (1 - o^k) (t_- -o^k)$
- For each hidden unit *h*, calculate the error term: $delta^h = o^h (1 - o^h)(\sum k \subseteq outputs)$ of $W_{kh} delta^k$
- For each $W_{ij} = W_{ij} + delta\text{-}W_{ij}$, where $delta\text{-}W_{ij} = e_{ta} delta^j x^{ji}$

We can use the validation-set concept to avoid overfitting the neural network. A small part of the training data (say 10%) can be moved into a tuning set to validate the learned function. We can save the synaptic weights and the weights of the network after every five epochs and calculate the error on the validation set with these weights. When we get an error rate lower than that of the previous five epochs, we may stop the training and use the weights of the network from the previous validation step.

VIII. CLEANING THE DATA:

The types of games we are dealing with are very complex and hence sometimes some problems may arise while collecting the data. As we have sampled the data each 100 milliseconds, we are very likely to obtain many input sets that are exactly the same since nothing might change for a few frames. In such a case, we only record one of those samples. However, if at least one of the features has changed, we consider that sample as unique and it is recorded. This is essential in order to make sure that we do not miss any crucial feature. Any sample event which does not have any associated actions is discarded.

IX. CONCLUSION

In this paper we have explored some of the popular machine learning algorithms and have shown how they can be deployed to the mainstream games in order to give the player a much more life-like and realistic experience. We have shown that at least a subset of the behaviors of the AI can be learned from the players and hence provide a human-like challenge, using machine learning techniques. For this paper we have successfully teach the AI agent, the basic actions of a shooting game which include facing in the right direction, accelerating or decelerating in the right time and jumping when required. However, this is just the top of the iceberg. What we have discussed in this paper can easily be extended to other genres of mainstream gaming like racing, open world, battle royal, arcade, multiplayer, story based etc. Also in this paper we have only tried to model an expert player's gameplay into the AI agent. If required, we can even model an inexperienced player to serve as fodder for the other players. Similarly we can model players with different skill levels to provide a range of difficulty levels for the players. Apart from creating enemies for the players, this system can also be used to create AI agents that also serve as non-human buddies in co- operative games.

X. REFERENCES

[1] A survey on feature selection methods, by Chandrashekar, G., & Sahin, F. Int. J. on Computers & Electrical Engineering
[2] Large-Scale Video Classification with Convolutional Neural Networks, by Fei-Fei, L., Karpathy, A., Leung, T., Shetty, S., Sukthankar, R., & Toderici, G. (2014). IEEE Conference on Computer Vision and Pattern Recognition
[3] Machine Learning with R by Brett Lantz, 2013
[4] "Neural Networks for Animating Variations in Character Behaviours" by Wen Z, Gough N and Mehdi Q, 3rd International Conference on Intelligent Games and simulation.
[5] AI Game Programming Wisdom by Rabin S, Charles River Media, 2002
[6] MIT Open Courseware, MIT 6.006: Introduction to algorithms