International Conference on Innovative Research in Engineering, Management and Sciences
International Journal of Scientific Research in Computer Science, Engineering and Information Technology
© 2019 IJSRCSEIT | Volume 4 | Issue 9 | ISSN : 2456-3307

Game Playing Agent Using Artificial Neural Network

Deepti Rai*1
*1 Sr. Assistant Professor, Department of Computer Science and Engineering, New Horizon College of Engineering, Bangalore, Karnataka, India
deeptiraj@newhorizonindia.edu1

ABSTRACT

The project focuses to train a game playing agent to learn the game. AI comprises of the neural systems ANN where the neural system produces the controls for playing the game. Based on the Reinforcement Learning technique selections are done subjecting on the information which is collected from the environment. Here, Q-Learning is used where the agent decides the actions based on conditions. Here, the interface Unity SDK is used to build the game.

Keywords : Machine Learning, Artificial Neural Network (ANN), Reinforcement Learning, Q-Learning

I. INTRODUCTION

This paper outlines the introduction of the project Game Playing Agent Using Artificial Neural Network. Neural Network is the basic part of Machine Learning. This game consists of an environment which has two players i.e. an Agent and a Human Player. Here, Agent is the Neural Network which understands to learn the game on its own.

Agent can be grouped into two particular stages i.e. Training and Testing. The Agent will start learning when the user begins the procedure in the command prompt. In Training, in order to obtain the experience an Agent will learn the game by learning through itself. Initially, the score generated will be zero. The repetition is settled to any value.

Every repetition the mean award is estimated. When the Agent starts learning through its experience the award will be increasing to its maxima. When the game ends the award generated will be maximum. After the Agent is trained, the trained values are obtained from a file. The values which are trained are transferred to a model called tensor-flow. During Testing, the Agent is tested on gaming platform Unity SDK depending on the performance of the game play.

Using Reinforcement Learning technique, the Agent will play the game repeatedly and for every game played a rewarding system will reward the action. A Human Player plays the game on their own and as much as possible tries to improve the score. Humans can use the controls using the keys. Every game played by the human is used in the training set. Simulated game will be used to They send the control of the game to the controllers where the controls are generated and provide it to the environment.

II. ARCHITECTURE OF THE PROPOSED MODEL

The architecture diagram shows how the components of system are related to each other. The Figure 1 describes the architecture of the model. The

game can be played by a human player and a trained agent. For the human player to play the game the controls are sent through the keyboard.

The agent is first trained using random values initially. Based on those values certain actions are performed in the game and the reward is calculated and updated in the Q-table. The trained model is saved in a file and and the same file is loaded before the agent plays the game.

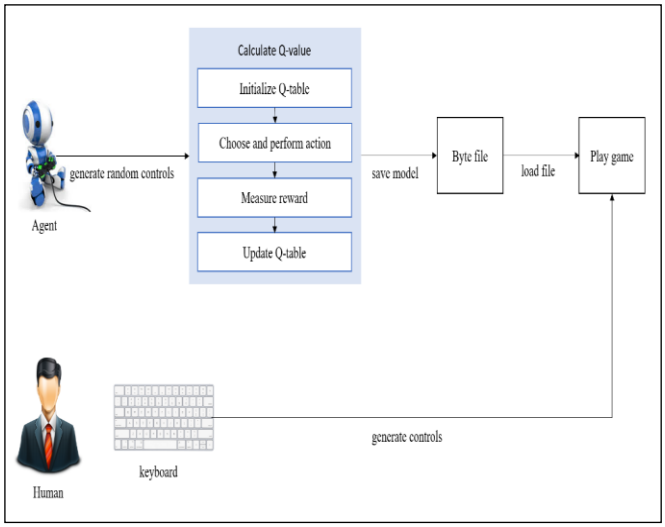

Figure 1: Architecture Diagram

The dataflow diagram in Figure 1 gives a representation of the flow of statistics throughout the system. The game has two phases - training phase and testing phase. In the training phase, random controls are generated to change the state of the game. Feedbacks are received in terms of reward for every action performed when the controls are generated. These values are updated in the Q-table. Q-table basically is made of lookup values for

This is an iterative process, after multiple times of learning to play the game and the model is saved in the file. During the training phase, the saved file is loaded before beginning the game. The trained values are used to play the game by the agent and improve the rewards.

III. METHODOLOGY

Q-Learning

The strategy is used to maximize the values expected of the award, beginning from the present state. The objective of Q-learning is to get a procedure which advises an agent what moves to make under certain conditions. The strategy is used to maximize the values expected of the award, beginning from the present state. It is utilized to store the information in the tables.

The Reinforcement learning techniques is adopted out to the activities resulting, which gives feedback as a reward. In the game, Agent sends the controls to the controller which creates the controls for the game and offers it to environment. The environment changes the condition of the game and updates the q-table. The condition is sent back to the Agent with the q-value. Agent will then update the q-value in the Neural Network and the last outcome is sent to the file to store the trained data.

The Q-learning is mainly based on operating the q-table with the q-values. Whenever the Agent learns the game the agent will be in a remarkable state which represents the present condition of the agent. The agent is provided with the information about the various states in a game. Based on certain action the agent will go to the next state. Here the learning agent will try to take more number of awards rather than punishments. The environment will change the state and also refreshes the q-values until the life exists of the agent.

Pseudocode for Q-learning

1. Initializing the q value
2. Choosing from the q table
   a. Choosing a policy which is derived from Q
   b. Taking actions
3. Performing the actions
4. Rewards are collected
5. Q-values updated

In a q-table, the rows stipulate the activities and columns stipulates the conditions. Each Q-table score will be the greatest expected future reward that the agent will get in the event that it makes that move at that state. This is a repetitive procedure, as to make the Q-Table better at every point. The Q function takes two inputs that is actions and states. There is a

repetitive procedure of updating the values. As we investigate the earth, the Q-work gives us better and good approximations by updating the Q-values in the table.

Q-Learning is statistically based on reinforcement learning used to find the optimum value of action policy using a Q function approximator. The goal is to increase the values in the function. The Q table finds out the action for each state. The reinforcement learning is usually a time-consuming task as it requires several runs of the same program or task. After that the results are stored. These results are then utilized to better the future run of the program. The game under observation here would benefit from the same learning strategies and will be easily played by agents. The entire operation rests on the reward promised to the agent after every successful run of the trial run.

IV. RESULTS AND ANALYSIS

To analyze the utility of reinforcement learning, the trained values are fed into Tensor Flow, a google open source modeling platform for all machine learning libraries. Tensor flow uses data flow graphs to record the usage of different graphs. These then are altered or modified as per the data fed into it through data sets. After several runs, the data graph is as below for the number of repetitions and mean award.

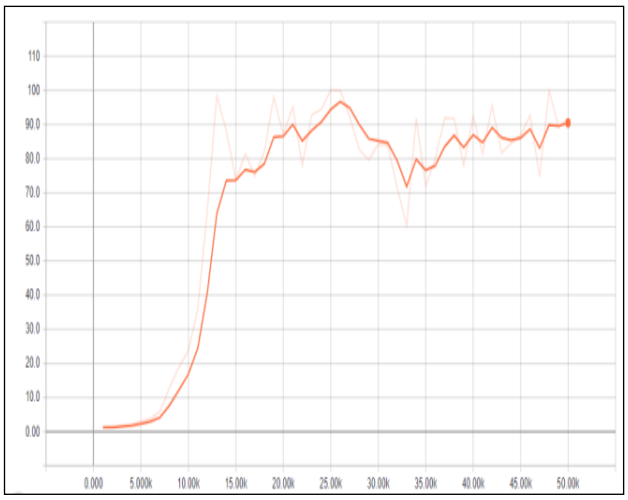

Figure 2: Cumulative Reward

X axis stipulates the number of repetitions and Y axis stipulates the mean award. It is noticed after the graph plotting that after every repetition the mean award will be increasing. This mean award towards the end shall be used to stipulate the game behaviour and will get the training data to show the game learning mechanism.

V. CONCLUSION

The proposed system trains the game playing agent to play the game using the trained value to improve the mean rewards in the game. For future work, we propose trained game playing agent for pre-existing games which has high rewards. Also, the below outputs can be looked into for improvement in the future.

1) The iterations can be speeded up depending on how fast the agent learns. Even a small improvement in iteration time can result in big impact on the reinforcement learning.

2) The game used for learning can be divided into better levels which can test the agent's ability to learn harder game movements.

VI. REFERENCES

[1] Matthew S. Emigh , Evan G. Kriminger, Austin J. Brockmeier, Jose C. Principe and Panos M. Pardalos, "Reinforcement Learning in Video Games using Nearest Neighbor Interpolation and Metric Learning", IEEE Transactions on Computational Intelligence and AI in Games, ISSN(e):1943-0698, Vol-08, Issue-01, March-2016, pp.56-

[2] Sebastian Risi and Julian Togelius, "Neuroevolution in Games: State of the Art and Open Challenges", IEEE Transactions on Computational Intelligence and AI in Games, ISSN(e):1943-0698, Vol-09, Issue-01, March-2017, pp.25-41

[3] Daniel Jallov, Sebastian Risi and Julian Togelius, "EvoCommander: A Novel Game Based on Evolving and Switching Between Artificial Brains", IEEE Transactions on Computational

Intelligence and AI in Games, ISSN(e):1943-0698, Vol-09, Issue-02, June-2017, pp.181-191

[4] Sam Snodgrass and Santiago Ontanon, “Learning to Generate Video Game Maps Using Markov Models”, IEEE Transactions on Computational Intelligence and AI in Games, ISSN(e):1943-0698, Vol-09, Issue-04, December-2017, pp.410-422

[5] Pawel Liskowski, Wojciech Jaskowski, Krzysztof Krawiec, ”Learning to Play Othello with Deep Neural Networks”, IEEE Transactions on Games, ISSN(e):2475-1510,pp.1-1.

[6] Peizhi Shi and Ke Chen, “Learning Constructive Primitives for Real-Time Dynamic Difficulty Adjustment in Super Mario Bros”, IEEE Transactions on Games,ISSN(e):2475-1510, Vol-10, Issue-02, June-2018, pp.155-169.

[7] Tristan Cazenave, “Residual Networks for Computer Go”, IEEE Transactions on Games, ISSN(e):2475-1510, Vol-10, Issue-01, March-2018, pp.107-110

[8] Xiaomin Lin, Peter A. Beling and Randy Cogill, “Multiagent Inverse Reinforcement Learning for Two-Person Zero-Sum Games”, IEEE Transactions on Games,ISSN(e):2475-1510, Vol-10, Issue-01, March-2018, pp.56-68.

[9] Hendrik Baier, Adam Sattaur, Edward J. Powley, Sam Devlin, Jeff Rollason and Peter I. Cowling, “Emulating Human Play in a Leading Mobile Card Game”, IEEE Transactions on Games, ISSN(e):2475-1510, pp.1-1

[10] Hao Wang, Hao-Tsung and Cheun-Tsai Sun, ”Thinking Style and Team Competition Game Performance and Enjoyment”, IEEE Transactions on Computational Intelligence and AI in Games, ISSN(e):1943-0698, Vol-07, Issue-03, September-2015,pp.243-254.

[11] Federico Chesani, Andrea Galassi, Marco Lippi and Paola Mello, “Can Deep Networks Learn to Play by the Rules? A Case Study on Nine Men’s Morris”, IEEE Transactions on Games, ISSN(e):2475-1510, pp.1-1.