IRSTI 06.58.45

M. Urmanov[1], M. Alimanova[2]
[1,2]Suleyman Demirel University, Kaskelen, Kazakhstan

TRAINING A SINGLE MACHINE LEARNING AGENT USING REINFORCEMENT LEARNING AND IMITATION LEARNING METHODS IN UNITY ENVIRONMENT

Abstract. This paper provides a research of Unity plugin that helps to develop Machine Learning Agents within Unity engine environment. This work introduces training a single Machine Learning Agent using both Reinforcement Learning and Imitation Learning methods, comparing the results and effectiveness.

Keywords: Computer Science, Game Development, Artificial Intelligence, Machine Learning, Unity.

***

Аңдатпа. Бұл мақала Unity үшін плагинді зерттеу ұсынады, ол қозғалтқыш ортасының ішінде машина оқыту агенттерін дамытуға көмектеседі. Бұл жұмыс екі әдісті пайдалана отырып, машиналық оқытудың жеке агентін оқыту, Нығайту және Имитацияны оқыту, нәтижелер мен тиімділікті салыстыруды ұсынады.

Түйін сөздер: Информатика, Ойын Дамыту, Жасанды Интеллект, Машинамен Оқыту, Unity.

***

Аннотация. Эта статья предоставляет исследование плагина для Unity, который помогает разрабатывать Агентов Машинного Обучения внутри среды движка Unity. Эта работа предоставляет тренировку единичного Агента Машинного Обучения, используя оба метода, Обучения с Подкреплением и Обучения Имитацией, сравнение результатов и эффективность.

Ключевые слова: Информатика, Разработка Игр, Искусственный Интеллект, Машинное Обучение, Unity.

Introduction

Since the making of computer games, the artificial intelligence development improvement issue, which would make games all the more fascinating to play, has consistently been important. Regularly artificial intelligence was sub-par compared to players in abilities, so computer games turned out to be unreasonably straightforward for them. For this situation, the game developers, as a rule, turned to different stunts to change the powers of a human and a computer rival. In cases when we are discussing a racing simulation, at that point, the alleged "catchup" word is used. The attributes of a car controlled by a computer are artificially exaggerated. Subsequently, paying little heed to how handy the player is, the computer can play with him as an equivalent. Another case - RTS genre games, where the computer is helped by additional resources. But such solutions seem to be a cheat and reject the player. That is the reason online games are so mainstream, since playing with a genuine individual is significantly more interesting. In this manner, the issue is the amazingly low or absurd degree of insight of computer rivals, which is the second rate compared to the human. With the help of Machine Learning, it will take care of this issue and significantly grow the abilities of AI in games. One of the most popular researches on this subject is the paper of DeepMind Technologies workers. Using Q-Learning [1], they figured out how to actualize a calculation fit for playing straightforward Atari 2600 computer games without knowing anything about them, aside from the pixels on the screen [2]. In the event that more data is given to the neural system (such as, the directions of game objects), the use of AI grows. Getting data about the condition of the game world is a genuinely basic task if the neural network is designed for a game with AI, which suggests the presence of game sources, rather than the work with Atari games. In this work, the development and training ML agent, which is able to control character in a three-dimensional Unity environment, will be considered. The agent’s tasks include maneuvering to avoid enemy traditional AI and getting to the aim area.

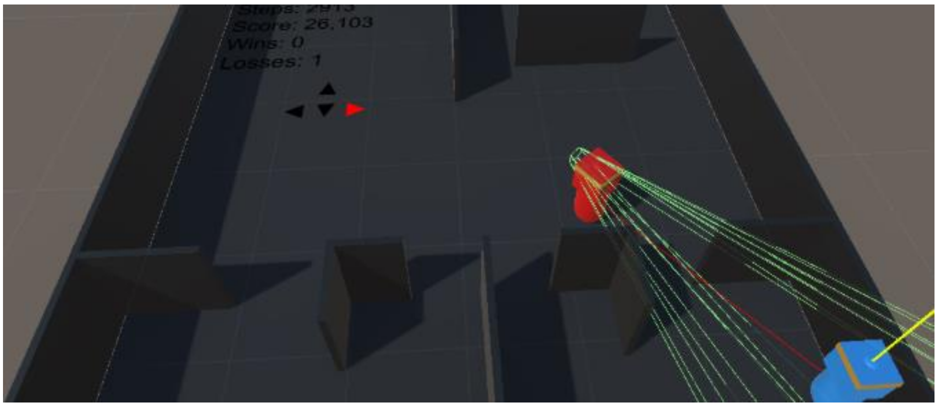

Fig. 1: The ML Agent is in the field of view and line of sight of the AI, and is being pursued.

Background

The Unity game engine was chosen as a tool for the development of a learning environment. To train agents The Unity Machine Learning Agents Toolkit (ML-Agents) [3] was used. For learning, two methods will be used one after another: Reinforcement Learning and Imitation Learning.

Reinforcement Learning

The principle thought of Reinforcement Learning is that the t-agent exists in a specific S-environment. Whenever the agent may process an action (or more than one action) from the arrangement of A-actions. Because of this action, the environment changes its state and the agent gets the r-reward[4]. In light of this cooperation with the environment, the agent must pick the ideal solution that boosts its reward. Reinforcement Learning is particularly useful for taking care of issues related to a decision between long-term and short-term profits. It has been effectively applied in different fields, for example, robotics, media communications, lift management. Likewise, Reinforcement Learning is a decent method to create AI in games. On account of games, the game character goes about as an agent and his general surroundings and his enemies go about as an environment. Each time the character plays out an action that approximates him to win, he gets a fortifying reward. For instance, the car agent on a dashing track gets a reward after some time if the separation to the finish line is decreased. This works in the same logic otherwise - when performing ineffective actions, the agent gets a punishment. All together for the agent to perform effective actions, it is fundamental for him to get a variety of data describing the condition of the environment. The measure of this data ought to be sufficient to guarantee that the operator gets all the vital data about the environment, yet not be unreasonably huge for the agent to train all the more effectively. Additionally, it is important to normalize the data, so the estimations of the signals showing up to the agent were inside the range of [0; 1] or [-1; 1]. There are examples of the input signal for a car like speed and position on the track. A list of action signals would be resulted out of the agent's work. Just as input signals, they require normalization. For example, the input signal for a car could be [0; 1] for the gas pedal and [-1; 1] for the steering.

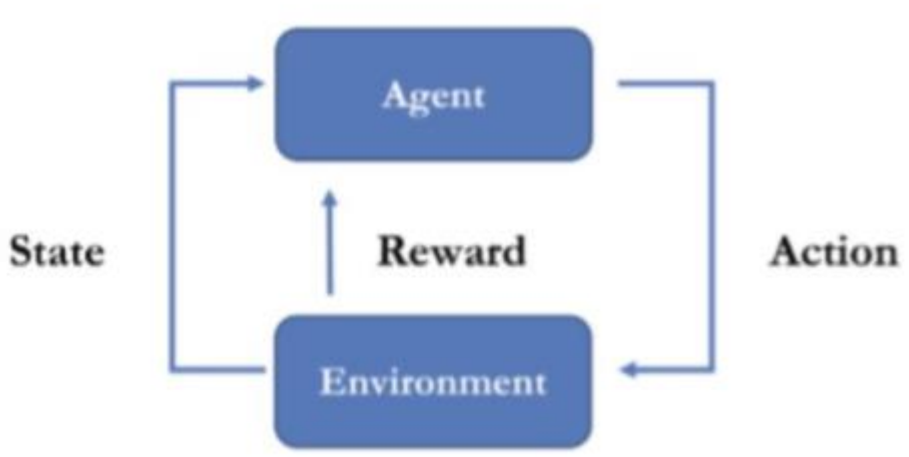

Figure 2: The Reinforcement Learning cycle

Imitation Learning

Unlike Reinforcement Learning, which processes with a reward/punishment system, Imitation Learning uses a framework dependent on the cooperation between a Teacher agent that executes the task and a Student agent that imitates the teacher. This is helpful in cases where you don't need your AI to have machine-like flawlessness, yet you need your agent to act like a real human being [5]. Imitation Learning Support was introduced in ML-Agents v0.3 Beta. This tool component is a ground-breaking feature that facilitates the development of a complex AI using fewer resources. All things considered, the procedure of AI development proceeds like this: there are two agents, one is a Teacher and another is a Student. Another neural network, a real person or a deterministic algorithm may go as a Teacher. The most effective outcomes are accomplished if the Teacher is a human being. Next, the learning procedure starts. The Teacher plays for some time. The planning shifts relying upon the task difficulty. For easier tasks, it takes around 4-6 minutes. For complex tasks, it is required about 2 hours. The learning is that while the Teacher agent plays, and the Student watches his activities and tries to imitate its Teacher.

Hide and Escape Scenarios

In the first two versions of the environment scenario, the goal for the agent was to avoid being captured by the traditional AI. The reward was calculated by how many seconds the agent would manage to avoid the traditional

AI. And in order to enhance learning speed, there were nine parallel environments with nine agents that trained simultaneously. After performing 3 million attempts in the first scenario progress of avoiding continuality occurred. But it was inconsistent and by the time there were still cases when the agent would be captured too fast. After testing some values of parameters in the curriculum, increasing buffer size twice, there were improvements in the learning, and results were more effective than in the first scenario, but still, it had flaws and inconsistent reward curve after the same 3 million attempts. Then a new approach of training was decided to perform. There was a problem that it never is possible to avoid the traditional AI indefinitely in a closed area like our environment. And even if the agent could eventually train to avoid indefinitely it meant that there was no win condition for the environment. It was decided that an agent should have an endpoint and the win condition. In this scenario, the reward was calculated by 10 minus how many seconds it took for the agent to reach the aim area. The expected result was achieved after the same 3 million attempts the agent could reach the aim area in less than a second [6].

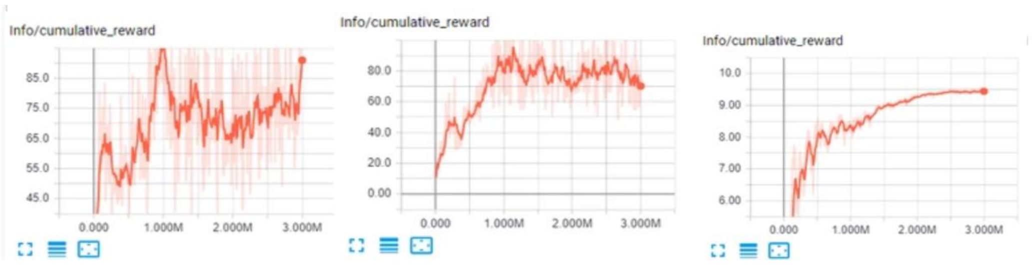

Figure 3: Reward graph of Hide, Hide2 and Escape scenarios

Human Escape Scenario

After we reached sufficient results by using Reinforcement Learning it was time for the next step which is applying Imitation Learning to training ML agent in the same environment. Although the controls for the environment were super simple it was very time consuming to record enough data for upcoming calculations. Also, a flaw occurred in the behavior of a human being since the aim area is randomly placed in the environment, including the character and traditional AI, after each attempt. It took some time for a real person to detect its aim area and the position of the character. While the agent would know the environment state in the first frame of the attempt. And because of this, the best result that the agent would achieve after applying Reinforcement Learning was a little less than 2 seconds, which is three times more than the result of Escape scenario. But in this scenario, the agent manages to behave like a real person and if this result is expected by the developers, then one second is not that important comparing to the realism.

Comparing the methods

Comparing two methods of learning is difficult since they achieve different results and behaviors. And the intersection results may only occur for early training which may be random and not satisfactory. It is obvious that Imitation Learning approach consumed more time to perform due to recording realtime human behavior which also consumes human resources more. If the developers seek the most effective results then Reinforcement Learning method will be more preferable. And if their intentions are to develop an AI that is close to human behavior, then Imitation Learning method is the best choice. However in both approaches there were some cases when even after 3 million steps of learning, the agent was able to reach the point only when the traditional AI is gone in a very long distance. And as the traditional AI patrols the environment randomly, there were cases, when the agent was waiting unnecessarily for a long time. That happened because the agent does not consider barriers in his path like walls and also he does not consider the direction of the traditional AI.

Conclusion

This study explores the opportunities and benefits of single use of Reinforcement Leaning and simultaneous use of Reinforcement Learning and Imitation Learning in artificial intelligence development for video games. Tools for creating the Learning Environment and learning AI agents have been considered. Practical recommendations, allowing to optimize the parameters and characteristics of the neural network and to conduct more effective training, were given. In the final result, a video game agent, which controls the character, effectively uses the available game mechanics and whose behavior is similar to a human, was created and trained. And also a video game agent who shows the most effective result was developed and trained.

References

1 Volodymyr, M., Puigdomenech, A.B., Mehdi, M., Graves, A., Timothy, P., Lillicrap, T.H., Silver, D., Kavukcuoglu, K. Asynchronous methods for deep reinforcement learning. arXiv preprint arXiv:1602.01783, 2016.

2 Bellemare, M.G., Naddaf, Y., Veness, J., Bowling, M. The Arcade Learning Environment: An evaluation platform for general agents. *J. Artif. Intell. Res.*, 47 (2013): pp. 253–279.

3 Juliani, A., Berges, V., Vckay, E., Gao, Y., Henry, H., Mattar, M., Lange, D. (2018). Unity: A General Platform for Intelligent Agents. arXiv preprint arXiv:1809.02627. URL: https://github.com/Unity-Technologies/ml-agents.
4 Gupta, A., Devin, C., Liu, Y.X., Abbeel, P., Levine, S. Learning invariant feature spaces to transfer skills with reinforcement learning. *In Int. Conf. on Learning Representations (ICLR),* (2017): pp. 1-14.
5 Englert, P., Paraschos, A., Peters, J., Deisenroth, M.P. Model-based Imitation Learning by Probabilistic Trajectory Matching. *Proceedings of the International Conference on Robotics and Automation,* (2013): pp.1-6.
6 Urmanov, M., Alimanova, M., Nurkey, A. Training Unity Machine Learning Agents using reinforcement learning method, *2019 15th International Conference on Electronics, Computer and Computation (ICECCO),* Abuja, Nigeria, (2019): pp. 1-4.