7-3 Project Two Submission

Joel De Alba

Southern New Hampshire University

Professor Tim Alexander

08 / 19 / 23

In this design defense, I'll give an overview of the steps I took to create a deep Q-learning intelligent agent that could solve the maze-based treasure hunt game's pathfinding challenge. The agent's objective is to find the treasure by navigating through the maze while avoiding obstacles and maximizing its pathfinding approach. I'll outline the procedures required to solve the issue, contrast human and automated methods, assess the algorithm, and go through how reinforcement learning was applied.

For the most part, humans use their cognitive abilities and senses to find their way through mazes. To determine what to do next, they combine memory, spatial reasoning, and pattern recognition. The deep Q-learning agent, on the other hand, uses a combination of neural networks and reinforcement learning algorithms to learn the best possible decision-making strategy.

Human-Based Method Steps

1. Study the maze's design.

2. Identify possible routes and obstructions.

3. Visualize several routes and base decisions on past experiences.

4. Utilize heuristic judgements to navigate the maze, changing tactics as necessary.

Deep Q-Learning Steps

1. Create a Q-learning neural network model from scratch.

2. Create a maze environment and look at the starting condition.

3. Decide on a course of action utilizing the exploration-exploitation (epsilon-greedy) strategy.

4. Perform the chosen action, then look at the result and your new state.

5. Using the Bellman equation, update the Q-value of the selected action.

6. Continue for a certain number of episodes.

7. To overcome the problem of correlated data, train the model using experience replay.

8. Assess the model's performance and adjust as necessary.

The intelligent agent and humans both employ trial-and-error to find their way around the maze. Humans frequently rely on intuition, while the agent employs a data-driven methodology to choose the best course of action. Both strategies entail making decisions based on observations of the surrounding environment. The goal of the intelligent agent is to successfully navigate the maze and locate treasure while dodging dangers. To select choices that optimize the predicted cumulative benefit, it employs a learnt policy. The agent's goal is to strike a balance between using taught tactics and exploring new avenues in order to maximize reward.

Exploitation is the process of choosing behaviors that, considering the agent's current knowledge, believes will result in the greatest immediate payoff. Exploration entails attempting various tactics in order to learn new facts and deepen one's awareness of the surroundings. The intricacy of the problem and the learning stage determine the optimal balance between exploitation and exploration. To find the best techniques, more exploration is needed early on, while more exploitation may be needed afterwards.

A potent model for teaching agents to learn from their interactions with the environment is reinforcement learning. By obtaining incentives for taking the right actions, reinforcement learning in this situation enables the agent to learn the best course through the maze. The agent gradually refines its plan through trial and error to maximize cumulative rewards, ultimately guiding it to the prize.

Neural network-based deep Q-learning has shown to be successful in resolving challenging issues like maze navigation. The agent is suited for pathfinding strategy optimization because of its capacity for learning from experiences and changing its behavior in response to incentives. By reducing the correlation between successive samples, experience replay increases learning's stability and effectiveness.

Several dense layers of a neural network architecture were used to build the deep Q-learning technique. In order to interact with the world, the agent chooses actions and gains rewards. Using the Bellman equation, the Q-values of actions are updated during training. To strike a balance between exploitation and exploration, the agent employs an epsilon-greedy exploration strategy. Experience replay is used to stabilize learning and avoid convergence problems.

In conclusion, the pathfinding issue in the maze-based treasure hunt game is efficiently and effectively resolved by the deep Q-learning intelligent agent's design. The agent uses neural networks and reinforcement learning to find the treasure by navigating the maze in the most efficient way possible. This method illustrates how machine learning techniques can be used to resolve challenging issues in a variety of fields.

Sources

Karagiannakos, S. (2018, November 17). The idea behind Actor-Critics and how A2C and A3C improve them | AI Summer. AI Summer. <https://theaisummer.com/Actor_critics/>

Juliani, A. (2018, June 21). Simple Reinforcement Learning with Tensorflow Part 8: Asynchronous Actor-Critic Agents (A3C). Medium. <https://medium.com/emergent-future/simple-reinforcement-learning-with-tensorflow-part-8-asynchronous-actor-critic-agents-a3c-c88f72a5e9f2>

Emergent tool use from multi-agent interaction. (n.d.). <https://openai.com/research/emergent-tool-use>

OpenAI. (2019b, September 17). Multi-Agent hide and seek [Video]. YouTube. <https://www.youtube.com/watch?v=kopoLzvh5jY>

Part 2: Kinds of RL Algorithms — Spinning Up  documentation. (n.d.). <https://spinningup.openai.com/en/latest/spinningup/rl_intro2.html#a-taxonomy-of-rl-algorithms>

Yoon, C. (2021, December 7). Deriving policy gradients and implementing REINFORCE. Medium. <https://medium.com/@thechrisyoon/deriving-policy-gradients-and-implementing-reinforce-f887949bd63>

Gulli, A., & Pal, S. (2017). *Deep learning with keras : Get to grips with the basics of keras to implement fast and efficient deep-learning models*. Packt Publishing, Limited.

Beysolow, I. T. (2019). *Applied reinforcement learning with python : With openai gym, tensorflow, and keras*. Apress L. P..

Lamba, A. (2018b, September 3). An introduction to Q-Learning: reinforcement learning. Medium. <https://medium.com/free-code-camp/an-introduction-to-q-learning-reinforcement-learning-14ac0b4493cc>

Düğmeci, D. (2022, July 6). Finding Shortest Path using Q-Learning Algorithm - Towards Data Science. Medium. <https://towardsdatascience.com/finding-shortest-path-using-q-learning-algorithm-1c1f39e89505>

Surma, G. (2021, October 13). Cartpole - Introduction to Reinforcement Learning (DQN - Deep Q-Learning). Medium. <https://gsurma.medium.com/cartpole-introduction-to-reinforcement-learning-ed0eb5b58288>

Manrique Escobar, C. A., Pappalardo, C. M., & Guida, D. (2020). A parametric study of a deep reinforcement learning control system applied to the swing-up problem of the cart-pole. Applied Sciences, 10(24), 9013.

Kumar, S. (2020). Balancing a CartPole System with Reinforcement Learning--A Tutorial. arXiv preprint arXiv:2006.04938.

Van Hasselt, H., Guez, A., & Silver, D. (2016, March). Deep reinforcement learning with double q-learning. In Proceedings of the AAAI conference on artificial intelligence (Vol. 30, No. 1).

Watkins, C. J., & Dayan, P. (1992). Q-learning. Machine learning, 8, 279-292.

Dearden, R., Friedman, N., & Russell, S. (1998). Bayesian Q-learning. Aaai/iaai, 1998, 761-768.

Aleksandar Haber. (2023, February 5). Introduction to OpenAI Gym (Gymnasium): Cart-Pole Environment - Reinforcement Learning Tutorial [Video]. YouTube. <https://www.youtube.com/watch?v=2sp_eucoX2I>

Admin. (2023, January 31). Cart Pole Control environment in OpenAI Gym (Gymnasium)- Introduction to OpenAI Gym – fusion of engineering, control, coding, machine learning, and science. <https://aleksandarhaber.com/cart-pole-control-environment-in-openai-gym-gymnasium-introduction-to-openai-gym/>

Admin. (2023b, February 10). Deep Q Networks (DQN) in Python from scratch by using OpenAI GyM and TensorFlow- Reinforcement Learning tutorial – fusion of engineering, control, coding, machine learning, and science. <https://aleksandarhaber.com/deep-q-networks-dqn-in-python-from-scratch-by-using-openai-gym-and-tensorflow-reinforcement-learning-tutorial/>