Braydon Woodward

CS-370

October 19, 2024

**Design Defense: Pathfinding Intelligent Pirate Agent**

In this project, I implemented an intelligent agent capable of navigating a maze using deep Q-learning. This design defense outlines the approach taken to solve this problem, evaluates the algorithm, and highlights key insights into the development process.

**Human vs. Machine Problem-Solving Approaches**

Humans solve problems with an intuitive mix of experience, trial-and-error, and deduction. For a maze, they would visually examine it, develop a plan, and adjust if necessary. In contrast, an intelligent agent follows an algorithm and relies on reward structures. Our agent learns an optimal policy based on reinforcement signals, exploring new actions or exploiting learned paths to reach the goal. Over multiple attempts, it improves efficiency through incremental learning.

While both humans and machines require exploration and backtracking, the agent lacks human intuition and must depend entirely on rewards to guide it. Humans can plan ahead and adapt as well as generalize information to learn from a short experience dataset. The agent improves iteratively by updating Q-values over many attempts.

**Pathfinding Approach: Human vs. Intelligent Agent**

Humans may use a technique like always turning right or left, relying on spatial awareness and predicting obstacles. In contrast, the agent begins with no knowledge of the maze and learns through interaction. The agent assigns Q-values to actions and updates them based on the rewards received.

**Purpose of the Intelligent Agent in Pathfinding**

The goal of the intelligent agent is to autonomously find an efficient path from the start to the treasure. The agent balances exploration (trying new paths) and exploitation (using known paths). In our maze, the strategy of high initial exploration, followed by increasing exploitation is effective. This ensures the agent understands the environment early on and then focuses on optimal paths.

**Reinforcement Learning and Pathfinding**

Using deep Q-learning, the agent learns from its experiences. Positive rewards are given for moving towards the goal, while negative rewards discourage unhelpful actions. By storing these experiences and updating Q-values, the agent improves its strategy over time, eventually finding efficient paths to the treasure.

**Evaluation of Deep Q-Learning Implementation**

The deep Q-learning model uses a neural network to predict Q-values for possible actions based on the current maze state. Training is enhanced by a replay buffer that stores past experiences, allowing learning from multiple episodes and reducing correlations between states. This improves stability and speed of convergence. The model uses stochastic gradient descent optimizer ‘adam’ to minimize the error between predicted and target Q-values.

**Implementing Epsilon Decay for Dynamic Exploration Rate**

To adapt exploration dynamically, we implemented an epsilon decay mechanism. Initially, the agent explores extensively with a high epsilon value of 1.0. After each epoch, epsilon is decayed by 0.995 until reaching a minimum value of 0.1. This approach allows the agent to start with broad exploration and gradually shift towards exploitation, as described by Lim, Hsu, and Lee (2021). Their research emphasized that balancing exploration and exploitation is key to finding the optimal solution in an uncertain environment, and our implementation of epsilon decay follows this principle to maintain adaptability while maximizing efficiency (Lim et al., 2021).

Steps for implementing dynamic exploration rate:

1. **Define Parameters**: Set epsilon, epsilon\_min, and epsilon\_decay values.
2. **Epsilon-Greedy Strategy**: Decide between exploration or exploitation based on the current epsilon.
3. **Decay Epsilon**: At the end of each epoch, multiply epsilon by epsilon\_decay until reaching epsilon\_min.

This strategy lets the agent explore effectively early on and exploit optimal actions later, resulting in a refined and efficient solution. It’s exciting to see how these strategies lead to an agent that learns smarter and faster with every step!

**References**

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

Gulli, A., & Pal, S. (2017). *Deep learning with keras : Implementing deep learning models and neural networks with the power of python*. Packt Publishing, Limited.

Lim, Z. W., Hsu, D., & Lee, W. S. (2021). *Shortest Path under Uncertainty: Exploration versus Exploitation*. Association for Uncertainty in Artificial Intelligence. Retrieved from https://auai.org/uai2017/proceedings/papers/26.pdf