Shaun Ryan

Professor Gray

CS-370: Current/Emerging Trends in CS

12/11/2022

7-3 Project Two

A maze, such as the one modeled in my game, is a simple problem with numerous ways to approach it. Humans have been creating and solving mazes for centuries and have yet to settle on a single methodology for solving that is superior to any other. These run the gamut of purely methodical, such as keeping one shoulder to the wall or taking only lefts at forks, to purely random with no logic behind any decision. In reality, most humans would use a combination of these two extremes. Remembering landmarks from previous attempts would be a major strategy for a human optimizing a run. As far as the random decisions, “intuition” would be a more informed form of random decision making. It takes into account memory of previous moves and a certain amount of spatial awareness. Taking two successive lefts would clue most people into the fact that they have reversed direction, even without being able to see further than adjacent spaces.

The intelligent agent solves the pathfinding problem in a similar, but not completely identical manner. The decision-making process can be exploratory, or random, which enables the agent to find new paths that may turn out to be more efficient than previously explored paths. More often, the process is exploitative, which uses knowledge of previous actions in similar situations to determine the most advantageous action to perform. This “decision-making process” is the implementation of a neural network which is trained on the experience of the agent and refines after each episode.

Both approaches are similar in that they use a mixture of random decisions and ones based on experience. Each would lead to a more optimized path with the increase in the number of times the maze was run, know as epochs in the intelligent agent’s vernacular. Where the two processes differ is the concept of rewards. In reinforcement learning, a system of rewards is implemented to act as an objective metric for the agent’s choices. Each action that does not result in the agent finding the treasure incurs a small negative award. Invalid actions, such as going outside the bounds of the board or into “walls” incurs a large negative award. Finding the treasure results in a positive award. By attempting to maximize the potential award, it incentivized the agent to solve the maze in as few steps as possible.

We already touched on the idea of exploration vs exploitation. The agent’s pathfinding should be largely exploitative, at that is the technique which utilizes the neural network and becomes more proficient with each subsequent episode. Exploration still holds an important role, especially early in the process where the neural network has not had much opportunity to train and gives random prediction and can cause poor Q-learning performance. (Gulli & Pal, 2017) The balance between exploration and exploitation is controlled by the “exploration factor”, also referred to as epsilon. The higher epsilon, the higher the chance of a particular action being exploration. Initially epsilon was set to 0.1 (or 10% chance) and scaled down to 0.05 (5% chance) once the win rate reached 90%. I changed this to a scaling factor where epsilon starts at 0.15 and scales after each epoch to 0.15 \* (1 – win rate). This improved the results of the agent dramatically. Early on, when the model has not had the opportunity to train, random exploration is more advantageous. As the training progresses, the Q-matrix becomes populated, and the algorithm converges on an optimal solution, excessive exploration can become detrimental to the progression of the model. (Beysolow, 2019) Therefore, it is advantageous to decrease the exploration factor as time goes on.

In this game deep Q-learning was implemented utilizing a neural network. A loop was created which would iterate for the total number of epochs, or until it is determined that the model has a hundred percent success rate. Within this loop, the environment is reset, and another loop is implemented where the agent “runs the maze”. The inner loop will repeat until the agent wins by reaching the treasure or loses by having the reward score drop below the set minimum value.

The first step in the game loop is determining an action, whether exploration which chooses an available action at random, or by exploitation which uses the model to predict the q-values of each possible action based on the current environment state. Next, the action is applied to the maze object, and the environment state, game status, and reward are updated. This is where it is checked whether or not the agent has lost or won the game. After that, all of the information for the current episode are stored in a GameExperience object which is used to calculate the target Q-values. These are passed to the neural network model for training. From here the loop will start over, choosing the next action. This process of taking an action based on predictions from a model, evaluating the reward, and using the new data to train the model is the basis of the deep Q-learning implementation in this game.

References

Beysolow, T. (2019). Applied Reinforcement Learning with Python. Apress

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.