Edward Johnson

Analyzing the differences between human and machine approaches to solving problems

- **Describing the steps a human being would take to solve this maze.**

The approach that a human would likely take in order to solve a maze puzzle would include a series of several steps. The first of these steps would be to analyze where they are currently at, or their starting position in the maze. The next step would be to determine what ending position is desired, or what position the player would need to be at in order to consider the puzzle as being successfully solved. Once these two points have been determined, the player would then begin tracing their way through the maze avoiding closed pathways that will not lead to the finish point.

- **Describing the steps my intelligent agent takes to solve this pathfinding problem.**

The steps that the intelligent agent is taking during this pathfinding problem starts out with the initialization of the environmental data and the training parameters for the model. The agent is initialized at a random starting position from one of the available cells on the game map. This particular agent implements a neural network to calculate and approximate the Q-value of actions in the system over time. At each step throughout the training process, the agent is observing the state of the environment that it is in. For this model, that primarily consists of the position of the agent on the board. To select actions to take in the environment, the agent will rely on choosing actions that are associated with the highest Q-value, as these should lead to the greatest reward as the model learns and updates the policy that it uses for taking actions. Initially, as we will have an epsilon value that is close to 1.0, the model will favor exploratory actions over those of immediate reward. These actions will be selected randomly from the available actions, and the purpose of doing this is so that the model can learn data about the environment. As the model trains and develops a policy for taking actions that is based on achieving maximum perceived rewards, the epsilon value can be decreased. The value is decreased because once the model has determined an accurate strategy to solve the system, we want to limit the degree to which it is taking random actions that will not directly or immediately bring the highest rewarding Q-value reward.

- **Similarities and differences between these two approaches.**

To a certain degree, the initial approach of a both human, and an agent that is using Q-learning based machine learning algorithms is to analyze what they know about the system, and then to begin taking actions based on that knowledge. The key difference is in how the two different intelligence gather data about the system. With the intelligent agent, it must take actions and observe changes in state, while associating those various states with values that assess their quality. For humans, we rely much more on heuristics, on our intuition, and on the ability to abstractly view the problem. Our approach is not typically nearly as analytical as that of the approach that a machine learning algorithm would use. Humans can much more easily generalize our problem solving abilities, an even if we have never seen this maze before, or a maze at all, if we understand the concept of a puzzle, it is likely that our brain would quickly allow us to create a solution to the problem with minimal correction needed. The machine learning approach is very much a trial and error based approach, and the initial actions that the model is taking before it has built a strategy for the system are quite random, as the goal initially is not to immediately solve the system, but to gain information about the various states that can be held within it.

Both of these approaches are goal oriented, but the human's goal is to reach the end of the maze, and while in a way that is similar to the end goal of the intelligent agent, the real goal that it has is to maximize the reward value that it gains through selecting actions with the highest quality values.

Assessing the purpose of the intelligent agent in pathfinding

- **The difference between exploitation and exploration. Speculations of the ideal proportion of exploitation and exploration for this pathfinding problem.**

In the context of this intelligent agent and in pathfinding, exploration is is the stage where the agent is actively seeking out to learn new information about the nature of the system that it is operating within. It is attempting to gather data about the system that it may be able to then use to develop a policy that it can then attempt to solve the characteristics of the system with. "Exploration consists of not taking the decision that seems to be optimal, betting on the fact that observed data are not sufficient to truly identify the best option," (The Exploration-exploitation Trade-off: Intuitions and Strategies | by Joseph Rocca | in Towards Data Science - Freedium).

In the context of the same system, exploitation is the point at which the agent is attempting to perform / take actions within the rules of the system that it believes will bring about the greatest degree of reward. "Exploitation consists of taking the decision assumed to be optimal with respect to the data observed so far", (The Exploration-exploitation Trade-off: Intuitions and Strategies | by Joseph Rocca | in Towards Data Science - Freedium). The ideal balance of exploration versus exploitation in a reinforcement based learning problem such as with the treasure puzzle we solved depends on the amount of time that it takes for the agent to train itself on the system. Initially, the exploration will be of higher value to the agent as it is building a policy and then learning to reinforce future decisions based on that policy. The policy and agent can be adjusted by modifying the epsilon value of the agent. A value between 0.9 and 0.99 will be optimal in most cases. As epsilon approaches a value of 1.0, the system will more heavily favor decisions that provide a long term reward to the agent, and decreasing the value will favor decisions that provide an immediate reward to the agent, at the cost of long term.

Lower values would signify a higher degree of exploitation, and higher values would signify a higher degree of exploration. I think that with an initial value between 0.95 and 0.99, the system would be most effective in learning the long term strategies for solving a maze based puzzle system such as with the pirate treasure problem. "The reason epsilon should be probability one for selecting a random control starting out training is that we want the agent to explore many different controls across the state space (exploration). The reason we want epsilon small near the end of the training horizon is that we want the agent to exploit what it has learned," (Strategies for Decaying Epsilon in Epsilon-Greedy | by Caleb M. Bowyer, Ph.D. Candidate - Freedium, n.d.).

- **How reinforcement learning can help the agent (the pirate) determine the path to the goal (the treasure).**

Reinforcement learning greatly helps when solving a problem such as the one presented by the pirate treasure maze scenario. Reinforcement learning allows the agent to learn from the reward and penalty values that it associates with actions. As the agent takes actions in the system it can develop a policy that is based on providing it with the highest level of reward possible from its possible actions. It can assign and calculate the probability that certain actions will provide it with rewards, and then take the actions that generally lead to a higher and higher level of reward. The agent learns what actions are valuable in specific circumstances, through trial and error, and then can create a meaningful strategy to solve the problem. In this particular model, we use Q-learning which is a form of reinforcement learning. The model gathers data about the environment until it reaches a defined threshold, at which point it uses that data along with the predicted and observed outcomes to update its learning policy for the system. The outcome of this style of learning is that the Q-values trend and converge towards their optimal values as time progresses. "Through repeated iterations, the Q-learning algorithm gradually updates the Q-values based on the observed rewards and transitions. Eventually, the Q-values converge to their optimal values, reflecting the best action to take in each state. This convergence guarantees that the agent has learned an optimal policy, a strategy that maximizes the cumulative reward in the given environment," (Understanding Q-Learning: A Powerful Reinforcement Learning Technique | by Navneet Singh - Freedium).

Evaluating the use of algorithms to solve complex problems

- **How I implemented deep Q-learning using neural networks for this game.**

For this game specifically, I implemented deep Q-learning by designing and by designing an agent that can interact with the system and then make accurate predictions about the outcomes of those interactions. The agent does this by learning the value, or Quality in this context, of those actions. "Q-learning algorithm involves an agent, a set of states and a set of actions per state. It uses Q-values and randomness at some rate to decide which action to take. Q-values are initialized and updated according to the reward and possible future rewards of an action taken," (Finding Shortest Path Using Q-Learning Algorithm | by Doğacan Düğmeci | in Towards Data Science - Freedium). In the context of Q-learning, the quality of these actions is what is the Q-value. Through interactions with the environment of the system, the neural network learns to approximate and assign Q-values to actions and states. The agent continues to interact with the system, during the neural network training steps, and it stores the outcomes of those interactions as what are known as experiences. These experiences contain valuable data regarding the values of the actions taken within the system.

In the model that I implemented, we begin with the setting of the parameters of the agent, the layers of the neural network, the optimizer of the network, and then initializes the environment. The training is performed for the amount of cycles, or epochs, that are designated in the first steps of the training function call. During each of these epochs, the agent randomly selects a starting position from within the available free cells on the maze map. This allows the agent to properly approximate the Q-value of actions taken in the system, so that it is able to effectively generalize and create a policy that maximizes the value return that it expects to receive when selecting valid actions to take on the map. The model is being updated as it trains so that the weights better reflect policy decisions that will maximize on earned reward. The wins and losses are recorded to the win_history for each episode. Each episode of the training stage is added to a circular memory buffer for optimal memory management. This buffer continues appending saved episodes until reaching the max_memory value that was defined in the initial stages of the training function. If the agent reaches this point during the training and there is still training to be done, then the agent begins overwriting the values of the circular memory buffer, starting with the first position, index 0, in the list. After an experience is added to the buffer using this second method, the buffer index is incremented. When the length of the buffer reaches or exceeds the data_size that was defined in the initial steps of the training function, the model has reached the point where it will then process the data it has gathered. The training function then uses the game experience object to batch update the agent. This batch update is the process of taking a batch of data that is generated through accumulated experiences in the environment. These experiences include actions taken, various agent states, and rewards gained from various states. The model then makes Q-value predictions about the input states that it has been given, as well as the calculates the targets. The targets are calculated using the Bellman equation, which is instrumental in Q-learning. "The Bellman equation decomposes the value function into two parts, the immediate reward plus the discounted future values," (The

Bellman Equation | by Jordi TORRES.AI | in Towards Data Science - Freedium). "The Bellman equation is important because it gives us the ability to describe the value of a state s, with the value of the s' state,"(The Bellman Equation | by Jordi TORRES.AI | in Towards Data Science - Freedium). Finally, the loss value is calculated by finding the difference in the predicted Q-values versus the target Q-values. The weights for the model, and the future policy that it uses to make decisions can then be optimized using this most recent batch of data.

Sources:

- The exploration-exploitation trade-off: intuitions and strategies | by Joseph Rocca | in Towards Data Science - Freedium. https://freedium.cfd/https://towardsdatascience.com/the-exploration-exploitation-dilemma-f5622fbe1e82
- Understanding Q-Learning: a Powerful Reinforcement Learning Technique | By Navneet Singh - Freedium. (n.d.). https://freedium.cfd/https://medium.com/@navneetskahlon/understanding-q-learning-a-powerful-reinforcement-learning-technique-29a3da36f611
- Finding Shortest Path using Q-Learning Algorithm | by Doğacan Düğmeci | in Towards Data Science - Freedium. (n.d.). https://freedium.cfd/https://towardsdatascience.com/finding-shortest-path-using-q-learning-algorithm-1c1f39e89505
- Dubon, T. (2023, October 11). Reinforcement Learning Mechanisms - Tannia Dubon - medium. Medium. https://medium.com/@tdubon6/reinforcement-learning-mechanisms-23811c0a47f6
- Deriving policy gradients and Implementing REINFORCE | By Chris Yoon - Freedium. (n.d.). https://freedium.cfd/https://medium.com/@thechrisyoon/deriving-policy-gradients-and-implementing-reinforce-f887949bd63

- Beysolow, T. (2019). Applied Reinforcement Learning with Python. In Apress eBooks. https://doi.org/10.1007/978-1-4842-5127-0
- Reinforcement Learning Explained Visually (Part 4): Q Learning, step-by-step | by Ketan Doshi | in Towards Data Science – Freedium. https://freedium.cfd/https://towardsdatascience.com/reinforcement-learning-explained-visually-part-4-q-learning-step-by-step-b65efb731d3e
- The Bellman Equation | by Jordi TORRES.AI | In Towards Data Science - Freedium. https://freedium.cfd/https://towardsdatascience.com/the-bellman-equation-59258a0d3fa7
- Strategies for Decaying Epsilon in Epsilon-Greedy | by Caleb M. Bowyer, Ph.D. Candidate - Freedium. (n.d.). https://freedium.cfd/https://medium.com/@CalebMBowyer/strategies-for-decaying-epsilon-in-epsilon-greedy-9b500ad9171d