Dustin Haugh

Project Three

* Analyze the differences between human and machine approaches to solving problems.
  + Describe the steps a human being would take to solve this maze.

As a human being I would look for open pathways through barriers, ignoring the fact it is on a grid, but noting the entire access around the barrier as a single path.

Then, I would check to make sure there was at least one single pathway from my starting point to the end point and choose any of the open squares that would lead me through the path without consideration of efficiency.

It would take very little mental effort as our minds already deal with this type of decision making every time we walk.

* + Describe the steps your intelligent agent is taking to solve this pathfinding problem.

The code in TreasureHuntGame is utilizing two main methods to navigate the maze. The first one is to generate a random action about 5% of the time, depending on the value of epsilon.

The next is to utilize a neural network that generates an action with an input state. The input state is the maze along with the location of the pirate. Utilizing back propagation, the neural network learns better and better movements through each iteration.

The pseudocode asks that we generate a random location for the Pirate on the board, but in the final check it places the Pirate at (0,0) and then traces out the movement it takes to the treasure.

The random location through each epoch is good because it allows many different states to be analyzed, thus forming a more complete neural network that will allow the Pirate to navigate from any point.

Also, with random starting locations and random actions taken it allows the pirate to explore and solve easier puzzles to begin training the neural network until it is finally able to solve and train more difficult states.

Eventually, the neural network has been trained to generate good actions at enough states to win 100% of the time.

* + What are the similarities and differences between these two approaches?

Both the neural network in the TreasureHuntGame and the mind of a human need to be developed before being able to navigate through obstacles.

If we took a human mind that has never been exposed to the tangible world before, it would likely go through a lot of trail and error to develop the skill to travel through the maze, just as the neural network does.

* Assess the purpose of the intelligent agent in pathfinding.

The neural network that generates an action in TreasureHuntGame is used to generate the best action based on the current state. It doesn’t work well at first, but after being developed it becomes very efficient at it. It also does not need to be trained again unless the maze changes.

* + What is the difference between exploitation and exploration? What is the ideal proportion of exploitation and exploration for this pathfinding problem? Explain your reasoning.

I’ve tried a few epsilon values. My results have varied but keeping it at 0.05 seems to work well enough. The epsilon value is the percent of random action that are taken by the agent. This function, combined with random starting locations, allows for enough variability in the state to generate a useful and complete neural network that has gathered data from most, if not all states available in the maze.

* + How can reinforcement learning help to determine the path to the goal (the treasure) by the agent (the pirate)?

The neural network in the TreasureHuntGame is collecting successful and failed state/action pairs. When successful state/action pairs are found, they are given a higher Q value, and vice versa with failed state/action pairs.

After development, the neural network is able to find successful actions in most if not all states on the board.

* Evaluate the use of algorithms to solve complex problems.
  + How did you implement deep Q-learning using neural networks for this game?

The pseudocode had most of the code for us to use already, so all I added was the basic functionality of the game loop and states.

However, the TresureHuntGame utilizes TensorFlow APIs to develop and train neural networks.

First, we generate inputs and targets from the experience object and then pass them on to three different APIs:

Model.fit() will be used to set up the neural network

Model.evaluate() will use the data to train the neural network (called an agent)

Model.predict() will infer the best course of action utilizing the neural network

The following code was used in my program to create and train the Karas neural network:

model.fit(inputs, targets, epochs=8, batch\_size=64, verbose=0)

loss = model.evaluate(inputs, targets, verbose=0)

I chose to set the batch\_size to 64 because we are utilizing an 8 by 8 maze (64 bits). It seemed to work very well so I kept it. I am frequently able to reach 100% win rate within 100 to 150 epochs.

Epochs were arbitrary, I did not have a specific reason for choosing 8, it just sounded reasonable.

I set verbose at 0 because it was not outputting correctly otherwise.

Tensorflow.org Dictionary

**epochs** = Number of epochs to train the model. An epoch is an iteration over the entire x and y data provided

**batch\_size** = Number of samples per gradient update

**verbose** = verbose=0 will show you nothing (so is silent)

verbose=1 will show you an animated progress bar

verbose=2 will just mention the number of epochs

Resources

*Tf.keras.model  :   tensorflow V2.11.0*. TensorFlow. (n.d.). Retrieved December 10, 2022, from https://www.tensorflow.org/api\_docs/python/tf/keras/Model#evaluate

*Training and evaluation with the built-in methods  :   Tensorflow Core*. TensorFlow. (n.d.). Retrieved December 10, 2022, from https://www.tensorflow.org/guide/keras/train\_and\_evaluate