The Enforcers: Reinforcement Learning in the Wumpus World Environment

Jordan Miller Guangyu Nie Abdul Rahman Zindani Aakarsh Reddy Duvvuru

***Abstract*—This is a team project for Arizona State University's Artificial Intelligence CSE course 571. The project was to create a reinforcement learning agent to be deployed inside the Wumpus world. Reinforcement learning allows the agent to explore its environment and learn while executing actions [1]. The agent can access prior knowledge about a state when faced with the same situation in a later episode. These situations are stored in a q-table. The agent can then play the Wumpus Game and perform the best actions for each state due to playing the game before and learning from prior experiences. We tested three separate models and compared the results to a logic agent in the same environment. The model that performed the best was the model where when an agent chose an action, 20% of the time the agent would not move. In all models, the reinforcement learning agent outperformed the logic agent.**

***Index Terms*—Wumpus, Wumpus world, artificial intelligence, reinforcement learning, logic agent**

I. Introduction

This project used reinforcement learning in the Wumpus world environment for the CSE 571 Artificial Intelligence course at Arizona State University. This was project number five listed on the project choices and the team had four members. The team choose this project to allow the team members to gain a new skill set for many of them were new to the topic area.

A. Wumpus World

The Wumpus world is a simple problem used in artificial intelligence to examine how a knowledge-based agent will react in an environment [2]. The goal of the game is for the agent to collect the gold and then climb out of the cave while avoiding the Wumpus agent. If the agent enters a location where a Wumpus is, the agent will perish and the game will end. However, if the agent senses that there is a Wumpus in a location before it enters it, the agent can shoot the Wumpus and the Wumpus will perish making that location safe for our agent. If the agent's sensors were wrong and there was no Wumpus in that location, then our agent loses points from the overall score. The agent must also avoid falling into pits while navigating the environment or else our agent will perish and the game ends.

The agent can navigate the environment with sensors. The sensors are able to detect if there is a pit in front of the agent, if there is a breeze, or stench. The breeze indicates there is gold nearby and the stretch indicates there is a Wumpus nearby. Due the way the environment is laid out, when the agent detects a breeze or stench that means that the gold or Wumpus can be in one of the adjacent squares to the north, south, west, or east. This is why shooting the arrow is essential for the agent to discover the location of the Wumpus.

B. Reinforcement Learning

Reinforcement learning allows the agent to explore its environment and learn while executing actions [1]. It is assumed that the agent does not know the environment prior to starting. The benefit of this technique is that it allows the agent in the environment to access prior knowledge about a state when faced with the same situation in a later episode [1]. These situations are stored in q-table. However, each time the agent learns in the environment there is a small probability that the action the agent is trying to take will not happen. This is accounted for in probabilistic action outcomes [1].

The agent is a q-learning agent that compares the expected utility values for each possible state from its location. The agent performs active learning that requires the agent to learn what actions to perform while playing the game. Q-learning agents are not able to look ahead which is why the table is so important to allow the agent to remember states and look up expected reward [1].

C. Our Approach

In this project, the Wumpus world was simplified to make the problem manageable. The agent in this project cannot detect breezes and stenches. The agent can detect gold, pits, and the Wumpus in this scenario. We developed a reinforcement learning algorithm based on q-learning that allowed the agent to learn the environment and save the outcome from performing an action in that state.

This was compared to the project three Wumpus world to see the improvement reinforcement learning had on the environment. We developed three separate models to compare what happens when the noise was implemented differently. These separate models include: having the agent not move when it thought it would, having the agent perform a random action, or occasionally when the optimal action was chosen a different action would take place.

II. Technical Approach

A. Environment

The environment was constructed to be simple. The basic environment consists of a 4x4 grid containing a Wumpus and 3 pits, one arrow, and gold. The locations of these, as well as the starting location of the agent are randomized for every run. This allows the agent to learn the environment and what

to do when faced with a different situation rather than just repeating the same scenario each time. The agent then checks if the state is a terminal state, and if it is, the agent will end the game. If the state is not a terminal state, the agent will continue to learn in the environment.

The agent can move either vertically or horizontally, one location at a time. When the agent approaches a location adjacent to either a Wumpus or a pit, the agent may detect that the respective obstacle is nearby. In addition, if the gold is nearby the agent can detect it. The agent can use this information to move around the obstacle. For any action that the agent makes, there is 20% chance of a different, random decision being made instead of the optimal action.

Additionally, if the agent detects a Wumpus, the agent can choose to shoot the Wumpus with an arrow. This action would make the location safe for the agent to travel to as long as a pit does not exist in the safe location. In order to kill the Wumpus successfully, the agent must be facing the proper direction when shooting the arrow. The agent must also have at least one remaining arrow in order to kill the Wumpus.

The environment for the Wumpus world within the game can be viewed as a grid. "W" represents the Wumpus, "0" represents the pits, "G" represents the gold, and the "X" is the agent's current location. All other locations are represented with a ".".

B. Rewards

During training, the rewards are set up as follows. There are two ways in which the agent could possibly be killed. The agent could either fall down a pit or be eaten by a Wumpus. For this to occur, the agent would have to travel to the location of a pit or a living Wumpus. The agent's death ends the game and results in a negative reward of 100 points.

When approaching a Wumpus, the agent may detect that the Wumpus is nearby and choose to shoot the Wumpus. The agent must aim an arrow towards the Wumpus in order to kill it. If the agent shoots an arrow and misses the Wumpus, the reward is -1 points. Alternatively, if the agent is successful in killing the Wumpus, the reward is +10 points.

Other than the death of the agent, the other way to end the game is to win by successfully finding the gold on the map and picking it up. When the agent picks up the gold it is rewarded +5 points. By finding and retrieving the gold by climbing out of the pit, the agent receives +100 points and successfully wins the game.

Moving in any direction or not moving at all will lead to a reward of -1 points for every turn (unless the above reward requirements are fulfilled). Additionally, if the agent runs into the walls the agent receives a -5 reward.

For the evaluation, the rewards were only moving, shooting the arrow, and exiting the cave with the gold.

C. Q-Learning

The actions stored inside the q-table are moving up, moving down, moving left, moving right, shooting the arrow up, shooting the arrow down, shooting the arrow to the right, and shooting the arrow to the left. While the agent still has arrows, the agent may choose to shoot, otherwise, the agent will not shoot.

The q-table is a matrix that stores information regarding which actions should be performed depending on the agent's state and sensory information [3]. As the agent progresses through the game, if the agent encounters a situation that is stored in the q-table, it can use this information to select the best course of action. Once this action has been made and the reward for the action is received, the q-table will update to reflect this received reward.

If the current state of the agent does not exist within the q-table, then the state must be added. This is done by checking all the different actions that the agent could perform in the current state and determining which would provide the best reward. This information, along with the received reward is stored in the q-table for the next time the agent encounters this situation.

D. Learning-Agent

The learning agent file was developed to look at states and add them to the q-learning table. If the state had not been visited, it was added to the table. The state is represented by (agent_position, if_gold_exists, initial_position) in each model. The initial position is essential to include because without this information, the agent does not know the way out of the maze and thinks any position could be the end. When this information was excluded, the agent sat idly not knowing what to do next after retrieving the gold.

When the game was being tested, the agent had access to these states that were saved and was able to choose the next state with the highest reward for each location.

E. Testing

The test file was designed to run the tests for the reinforcement learning algorithm. The agent moved through actions and environment learning states and their rewards. The agent learns until the values converge. Once the agent is done learning, it plays the game with the best moves at each location.

The team ran several tests to examine which conditions gave the best data. These tests were a random starting initial position with a 20% chance the agent does not move when given an action, a noisy action model where there is a 20% the optimal outcome of the agent will result in a different unexpected outcome, and finally an agent that has a 20% chance of acting randomly but the correct action was recorded in the q-table.

The evaluation was ran 1,000 times for the reinforcement learning agent to learn the environment. These three reinforcement learning models were compared against the logic agent in the same environment.

III. Results

Below are the results for all three models of the reinforcement learning compared with the logic agent in the same environment. We will present the statistics on how well each

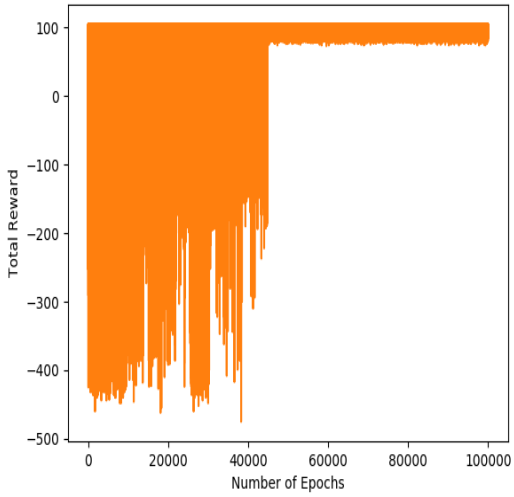

Fig. 1. Convergence after 100,000 epochs

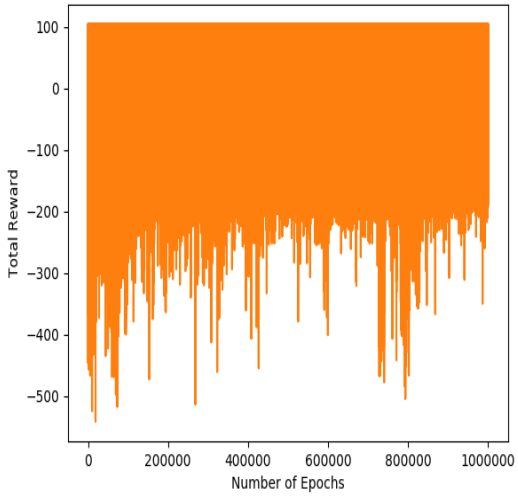

Fig. 2. No Convergence after 1,000,0000 epochs

model performed along with the logic agent in the same environment. The logic agent was not tested as many times as the reinforcement agent due to it not having the ability to remember the environment and learn from it. The convergence was plotted using scipy savgol_filter(Total_reward, 201, 3) and can be seen in the images provided. The rewards during training can be seen in the table below along with the action associated with each reward:

move	-1
shoot with arrow	-1
shoot without arrow	-5
get out of cave with gold	+100
bump into walls	-5
pick up gold	+5
bump into Wumpus	-100
fall into cave	-100

Rewards during the evaluation:

move	-1
shoot with arrow	-1
get out of cave with gold	+100

The environment for each model was laid out as follows:

+	–	–	–	–	+
—	.	.	.	.	—
—	0	0	.	.	—
—	W	.	0	.	—
—	.	G	.	.	—
+	–	–	–	–	+

A. *Stochastic Action - No Movement*

This model allowed the agent a 20% chance that when an action was executed, the agent would remain in place. This model was chosen to examine how the agent would perform when there was uncertainty in what the agent thought it was executing.

The logic agent was tested over 11 different initial positions. The scores for each position were as follows: 95, -2, 93, -8, -9, -2, -8, -9, -8, -9, and -9. The reinforcement learning agent was ran 1,000 times for the evaluation. When tested, it scored an average of 92/100. Convergence for this model can be seen in Figure 1.

This agent performed well due to not having to second guess which actions lead to which rewards. For the most part, the agent performed as it was expecting but only penalized -1 reward for not moving which does not affect planning much and introduce doubt when executing the optimal policy.

B. *Stochastic Action - Noisy Model*

This model allowed the agent a 20% chance of when the optimal action is chosen that the agent will choose a different outcome. This model significantly set back the training progress for the agent. It showed signs of converging, however, it took a very long time. This is because the 20% chance the agent thought it chose the optimal action, it received a sub-optimal result. This provided the agent with doubt in itself so the agent would stray from the optimal action.

For the logic agent, the agent only completed 2/11 of the games played. This agent took the chance to get the gold and this caused the agent to loose 9/11 times. The reinforcement learning agent was ran 1,000 times to evaluate the environment from a random initial starting position. When tested it averaged a score of 8/100.

Convergence for this model can be seen in Figure 2. This model did not perform well due to the agent doubting itself while it was on the optimal path and set back the learning of the agent significantly.

C. *Stochastic Action - No Noise*

This model allowed the agent a 20% chance to do a random action but recorded the correct value for that action inside the

q-table.

The logic agent was tested over 11 initial starting potions and did not find a value to converge. The reinforcement learning agent evaluated the environment 1,000 times and when tested it scored an average of 72/100.

This model performed the second best due to the agent still receiving the expected outcome from an action even if that action was not what the agent intended to take.

D. T-Test

A t-test is a statistic and hypothesis testing tool used to determine if there is a significant difference between the means of two groups [4]. The test involves taking samples from both groups and and assuming a null hypothesis that both samples have equal means. The mean difference, the standard deviation of each group, and the number of data values in each group are used to determine the t-value and degrees of freedom. These values are then compared against standard values in order to accept or reject the null hypothesis [4].

The reinforcement learning agent and the logic agent were compared for the stochastic action test with no movement in order to test whether the reinforcement agent was significantly better than the logic agent. The test should resulted in a t-value of 6.53 and 10 degrees of freedom. Based on a standard T-distribution table, these values imply that the two groups have statistically different means with a probability of over 99.95% and that reinforcement agent results in a better score than the logic agent. [5].

IV. Conclusion

It can be seen in each of the three models that the reinforcement learning agent still performed better than the logic agent, even when the reinforcement learning agent did not do well. Three models were chosen to allow comparison of the reinforcement learning agent in different stochastic environments. This provided insight on how uncertainty can impact an agents decision.

The environment was chosen to be static and we chose to only make the actions of the agent stochastic. When the environment was also initialized to be stochastic, after 1 million epochs the value had no converged.

The model that performed the best was having the agent 20% of the time not move when it chose an action to execute. This model performed the best because the agent did not have to doubt the logic it learned from previous epochs, but instead could account for not moving some of the time.

V. Team Effectiveness

All team members worked on this project. Writing of the paper was split between Jordan Miller and Abdul Zindani. Guangyu Nie developed the file 'Q-learning.py' and ran the tests for the project. Aakarash Reddy developed the files 'test.py' and 'learning-agent.py' in addition with adding parts to the 'env.py' file. Jordan Miller developed the inital 'env.py' file. All team members contributed to this project and communicated during the process to ensure this project was completed. We met several times via Zoom and communicated via email and through a group text. This allowed all members to work together and ensure we were on the same page during all parts of the project.

References

[1] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 3rd ed. Prentice Hall, 2010.

[2] "The wumpus world in artificial intelligence." [Online]. Available: https://www.javatpoint.com/the-wumpus-world-in-artificial-intelligence

[3] A. Violante, "Simple reinforcement learning: Q-learning," Jul 2019. [Online]. Available: https://towardsdatascience.com/simple-reinforcement-learning-q-learning-fcddc4b6fe56

[4] W. Kenton, "T-test definition," Mar 2020. [Online]. Available: https://www.investopedia.com/terms/t/t-test.asp

[5] "T-distribution table (one tail and two-tails)," Oct 2017. [Online]. Available: https://www.statisticshowto.com/tables/t-distribution-table/