Project progress report

Changqi Chen , Michael Douglass, Mark Liu

ABSTRACT

Our project for the class is based around the Malmo platform for Minecraft. We want an AI agent to perform steps within Minecraft that a normal human player would perform. That includes mining wood, creating crafting tables, creating types of pickaxes (wood, stone, iron), and finding materials to craft with (stone, iron, diamonds). The AI has to perform these tasks in specific orders (E.G mining wood, creating crafting tables, creating wood pickaxe). Also each major milestone of creating a pickaxe has subtasks of finding the materials to create the pickaxe with.
So far what we have been able to accomplish is the beginning task needed for the rest of the tasks. Our AI is able to find and mine wooden logs, the basis of creating pickaxes and crafting tables. It also is able to avoid lava spots so as to not continually die over and over from falling into it. The AI also avoids hitting dirt blocks over time and only hitting the wooden logs due to the reward system laid out for the AI.

Keywords: Malmo, Minecraft, Pickaxe, Mining

INTRODUCTION

Our project is about the Malmo Platform for Minecraft. We intend to create an AI that can do various tasks within minecraft that a normal player might do during their time playing the game. This includes mining wooden blocks, stone, iron ore, and diamonds. This also includes creating items in the game like crafting tables, sticks and pickaxes of various materials like wood and stone. The AI uses the wood blocks to create new items like the crafting table, sticks, and wooden pickaxe. The AI would use the wooden pickaxe to ultimately obtain a stone pickaxe. Then the AI would then use the stone pickaxe to obtain an iron pickaxe which then could obtain a diamond.

We have a couple of problems that we are addressing. The first is how the environment around our AI should be created to maximize the learning that takes place. The second is stopping our agent from endlessly digging downwards and basically leaving the area of the environment. To combat these we created a specific environment with wooden logs sparsely put around it, with lava surrounding and below it so that our agent will learn to not go out of the environment. The problem was interesting for us in that, had we not found any way to change the environment for our AI to better suit our needs we would have had a very hard time completing our tasks in any sort of timely fashion. By creating the exact environment we need, we are able to test our AI and have the AI learn. Below is a picture of our environment.

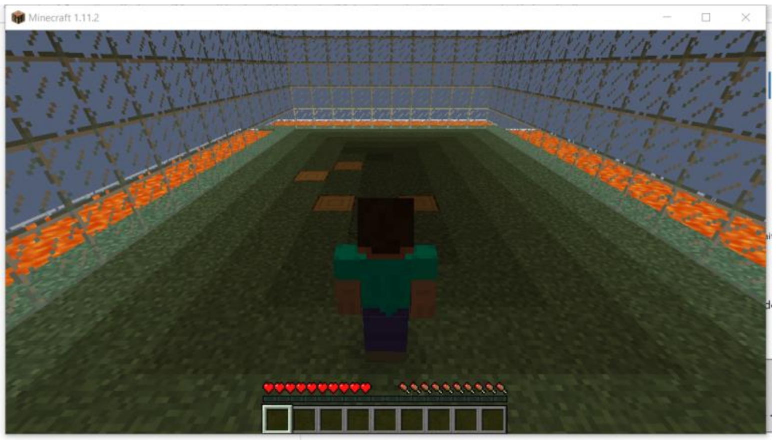

Figure 1. The environment our AI uses.

Our experiments so far have been at 300 trials with a 30 second cap, just letting the AI go into the environment and try to get the wood. It has successfully been able to get the wood blocks needed to pass the test multiple times throughout our testing. The AI has also been able to distinguish between wood blocks and dirt due to the reward system ultimately laying off of the dirt blocks. The AI also learner to not go into the lava as a result of Tabular Q agent. In the future we plan to adjust and test different learning rates from 0.1 to 1 to better see which is the best learning rate for our AI. We also plan to increase the trials from 300 to 500 for our tests.

BACKGROUND

So far most of our code comes from the Malmo tutorial 6 example provided by the Malmo installation. We also are using many normal python libraries including json, sys, os, time, logging, random and MalmoPython. We have a class for our Tabular Q agent as well that was pre-written in the tutorial. We also use TKinter to draw out our Q-table, again pre-written. The platforms we are using are Malmo and Minecraft for our AI agent.

For our Tabular Q agent almost all of our algorithms and functions are pre-written in the tutorial. DrawQ, run, UpdateQTable, UpdateQTableTerminatingState and act are nearly entirely from the Malmo tutorial except for a few changes here and there to better fit for our agent. We have made some modifications including how many trials we are running and for how long, but most of it is already written by the Malmo tutorial. Our XML file is mostly composed by us however. The underlying structure of the XML file was already established but the main contents was written by us.

PROBLEM STATEMENT

The first problem we had is how do we set our environment. If the environment is too simple, then the agent can achieve the goal without any reinforcement learning. So we set only four log blocks on our 9x15 world. Besides, we want to prevent our agent from keep digging down without any limit. So we set a lava floor two floors below the floor our agent starts. Once the agent touches the lava floor, it receives -100 reward and ends the trial.

We also set some other negative rewards: -5 for every movement and -1 for collecting dirt. The only positive reward we have right now is +1000 if the agent collects the log.

Our minimum milestone is to find and acquire two wooden logs by hitting them and collecting them. Our agent right now can already achieve this milestone but, most of the time, with a lot of redundant steps. Our next goal is to try to optimize our agents in several ways. Maybe by adding more rewards, both negative and positive or by adding codes to stop it from repeatedly doing the same movement, like keep turning in the same place. After that, our next milestone is to try to use the logs we acquire to make a crafting table, two sticks, and two wood planks. Then combine them to make a wooden pickaxe.

METHODS

The model class we use in our project to find the optimal policy is the tabular Q-learning. It takes action A on state S, notes the reward and the next state S', then it chooses the max Q-Value in state S' then use all these info to update Q(S, A), then moves to S' and executes epsilon greedy action which does not necessarily result in taking action that has the max Q-Value in state S'.(Sutton and Barto, 2018)

The formula for our one-step tabular Q-learning is shown in Figure 1. We set the learning rate alpha 0.1 and the reward discount factor gamma 0.8 at first.

$$Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma \max_a Q(S',a) - Q(S,A)\big]$$

Figure 2. Our Q-Learning Formula.

EXPERIMENTS

We are going to run our agents with different learning rates 0.1, 0.5 and 1 to see which learning rate can help our agent collect the logs best. We are going to run 500 trials for each learning rate. The environment and the reward discount factor remain the same for different learning rates.

RESULTS

After running the aforementioned experiments, we find that our agent performances best in learning rate 0.5. The average reward of our agent in the 0.5 learning rate is about 705 which is much higher when the learning rate is 0.1 and 1. Figure 3, which plots the learning curve of our agent (mean return per episode over 10) with learning rate 0.5, also shows that our agent performance better with more and more training.

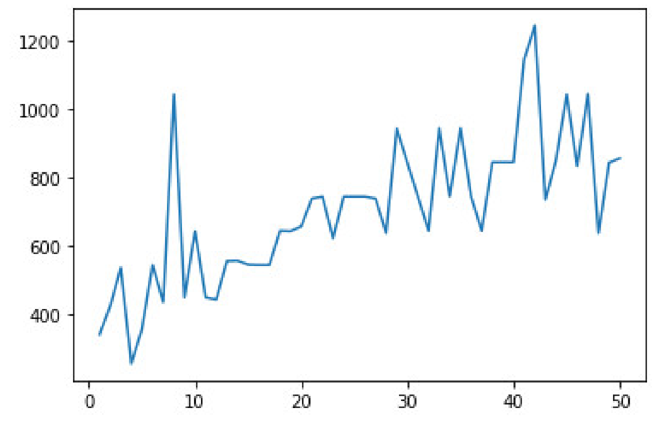

Figure 3. Our agent learning curve.

REFERENCES

Sutton, R. S. and Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.