Reinforcement Learning in Minecraft

Background

Reinforcement Learning. Reinforcement Learning is a subfield of machine learning that teaches an agent how to choose an action from its action space, within a particular environment, in order to maximize rewards over time.

Minecraft. Minecraft is a 3D, first-person, open-world game revolving around gathering resources and creating structures and items. Because it is a sandbox environment, designers can devise a variety of goals and challenges for intelligent agents. Additionally, because it is an embodied domain and the agent's surroundings are varied and dynamic, it presents many of the same challenges found in real-world control tasks.

MineRL competition. MineRL 2020, hosted and supported again by the competition platform AIcrowd, is part of the competition track at this year's Conference on Neural Information Processing Systems (NeurIPS 2020). Last year, the competition was also included in the conference lineup. Over 1,000 participants registered, and more than 50 people attended the affiliated NeurIPS workshop, during which the top teams presented their creative approaches.

Introduction

MineRL is a research project started at Carnegie Mellon University aimed at developing various aspects of artificial intelligence within Minecraft. It is a large-scale, dataset of over 60 million state-action pairs of human demonstrations across a range of related tasks in Minecraft. To capture the diversity of gameplay and player interactions in Minecraft, MineRL includes six tasks with a variety of research challenges including open-world multi-agent interactions, long term planning, vision, control, and navigation, as well as explicit and implicit subtask hierarchies. We are able to implement these tasks as sequential decision-making environments in an existing Minecraft simulator. As a lot of people are familiar with Minecraft, I hope you can enjoy the variety of missions.

Installation

To start using the data and set of reinforcement learning environments comprising MineRL, you'll need to install the main python package, mineral. Please follow the steps below to avoid dependency problems.

Debian(Ubuntu!) :

```
conda create -n mineRL python = 3.7
conda activate mineRL
conda install tqdm
conda install scikit-learn
pip install --upgrade minerl
pip install pyglet==1.5.11
```

Windows:

```
conda create -n minerl python = 3.7 matplotlib = 3.0.3 pillow = 7.2.0 jupyter coloredlogs
numpy scipy scikit-learn pandas tqdm joblib requests lxml psutil dill future cloudpickle
typing

pip install minerl == 0.3.6
```

Mac:

```
brew tap AdoptOpenJDK/openjdk

sudo add-apt-repository ppa:openjdk-r/ppa
sudo apt-get update
sudo apt-get install openjdk-8-jdk

pip3 install --upgrade minerl
```

After installing all those packages, there is something we need to fix. Since the definition of gradle is old and the URL is wrong , open the following file with a text editor and edit it manually.

C: \ Users \ <username> \ anaconda3 \ envs \ minerl \ Lib \ site-packages \ minerl \ env \ Malmo \ Minecraft \ build.gradle

from

url = "http://repo.spongepowered.org/maven/”

to

url = "https://repo.spongepowered.org/maven/"

Hello World: Your First Agent

With the minerl package installed on your system you can now make your first agent in Minecraft! For this example we’ll choose the “MineRLNavigateDense-v0” environment. In this task, the agent is challenged with using a first-person perspective of a random Minecraft map and navigating to a target.

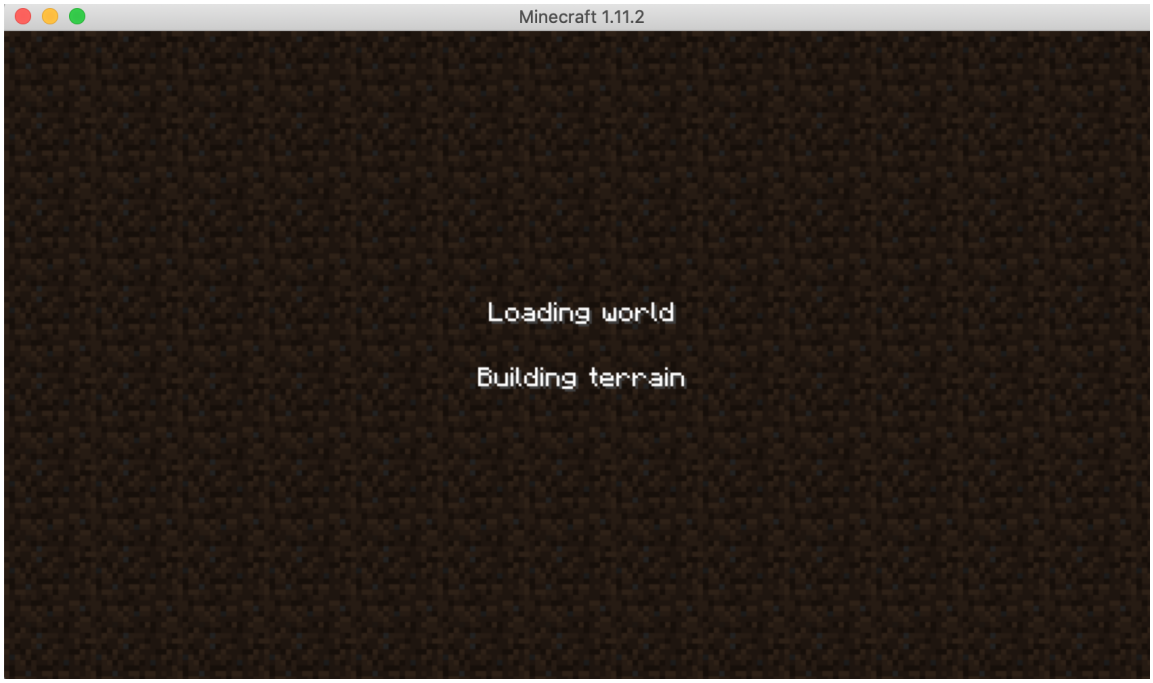
```
import minerl
import gym
import logging

logging.basicConfig(level=logging.DEBUG)
env = gym.make('MineRLNavigateDense-v0')
obs = env.reset()

done = False

while not done:
    action = env.action_space.sample()
    obs, reward, done, _ = env.step(action)
```

The agent will move randomly and keep running until it ends.


Minecraft 1.11.2
Loading world
Building terrain


Minecraft 1.11.2

Sample test

Below is a trained model written by pytorch. To test the model we need to install pytorch first.
https://pytorch.org/

```
import minerl
import gym
import logging
import numpy as np
import torch

def select_action(state, action, yaw, pitch):
    val = (model(state.long()).max(1).indices).long()

    binvec = dec2bin(val, 8).squeeze()

    action['forward'] = binvec[4]
    action['left'] = binvec[5]
    action['right'] = binvec[6]
    action['back'] = binvec[7]

    action['camera'] = [5*(binvec[0] - binvec[1]), 5*(binvec[2] - binvec[3])]
    return action

logging.basicConfig(level=logging.DEBUG)

env = gym.make('MineRLTreechop-v0')

obs  = env.reset()
done = False
net_reward = 0

num_episodes = 50

for i_ep in range(num_episodes):
    net_reward = 0
    yaw = 0
    pitch = 0

    done = False
    count = 0
    dur = 1000
    while not done:

        obs = torch.tensor(obs['pov'])
        obs = torch.transpose(obs, 0,2).unsqueeze(0).long()
        print(obs.shape)
        action = select_action(obs, env.action_space.noop(), yaw, pitch)
```

```
    yaw += 5*(binvec[0] - binvec[1])
    pitch += 5*(binvec[2] - binvec[3])

    obs, reward, done, info = env.step(action)
    net_reward += reward
    if count > dur:
        done = True
obs = env.reset()
print(net_reward)
```