Environment "Move Box" Documentation

Author: Shuo Jiang (jiang.shuo@husky.neu.edu)

Introduction

The document introduces an environment for multi agent study as two agents are trying to cooperatively move the box to the destination. The code is implemented in python and provided with simple interfaces. The environment is decoupled with learning method so reader can try any algorithms on.

This work is inspired by Coordinated Multi-Agent Object Transportation Problems (CMOTPs) problem in
Palmer, Gregory, et al. "Lenient multi-agent deep reinforcement learning." *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 2018.

This work extends the original domain into POMDP setting and fixed some bugs in the original domain.

Scenario Description

The task of the environment is explained as below

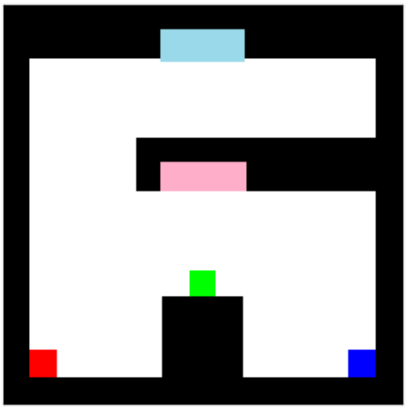

Figure 1

In the environment, two agents (agent 1 and agent2 marked using color red and blue) have to

find the way to be at the right or left side close to the box as green. When it is at the right or lift side of box, it will be glued to the box (to carry it). Only when both the agents are at the side of the box, they enter the joint carrying mode, that when they are taking the same action (go up/down/left/right), they will carry the box to the corresponding direction. If the two agents take different actions (i.e., one goes left one goes right), then the box will not move. The goal is to move the box to the light blue area in Figure 1, and each agent will be rewarded for 100, otherwise reward is 0. There is a sub optimal area in pink that the agents should avoid, when the box is pushed in the pink area, there will be 10. The joint carrying is as:

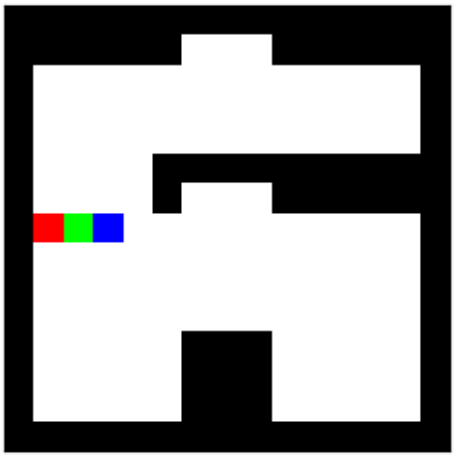

Figure 2

blocks in black are obstacles that agents could not step on and white ones are free spaces, also agents and box cannot stay at the same block. After reaching the goal area, environment will be reset to the initial state as in Figure 1.

The coordinate system is shown as

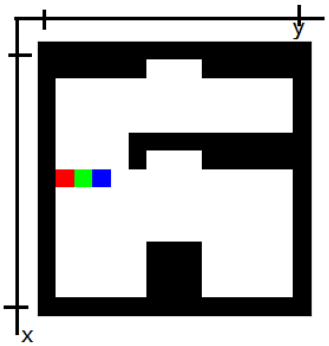

Figure 3

The area is 15*15 large, and box is initially at [10, 7], agent1 at [13, 1], agent 2 at [13, 13].

There are 4 possible actions for the two agents go up/down/left/right, using a integer as 0/1/2/3. When the agent moves, it will move to the free space of 1 cell near it. However, when there is a block or the other agent is in its way, the action won't work.

The observation is an image, with form (width = 15, height = 15, rgb_channel = 3) when it is fully observable, or (width = 3, height = 3, rgb_channel = 3) for each agent as the self-centered observation.

Class Organization

The environment is programmed in python 3.6 and has very simple interfaces. The file contains the environment is "env_MoveBox.py" and a class " EnvMoveBox" is defined.

The class has the following interfaces:

__init__(self)
reset(self)
obs_list = get_obs(self)
obs = get_global_obs(self)
state = get_state(self):
reward_list, = step(self, action_list)
plot_scene(self)
render(self)

__init__(self)
initialize the object, important variables are
agt1_pos
agt2_pos
box_pos

get _obs(self):
This function return a list of current visions of each agent asimages, with form (width = 3, height = 3, rgb_channel = 3). obs_list = [obs1, obs2]

get_global_obs(self)
This function returns global current vision as an image, with form (width = 15, height = 15, rgb_channel = 3) as shown in Figure 1.

get_state(self)
This function returns the normalized positions of agent1, agent2 and box, which is a numpy array of size [1, 9]

step(self, action_list):
The function updates the environment by feeding the actions for agent1 and agent2. The actions are denoted by integers 0/1/2/3. action_list = [action1, action2]. The return is an integer and done as if the episode finishes.

reset(self):
This function relocates the two agents and the box to the initial position

plot_scene(self):
This function plots the current state of the whole environment, and uses two subplots to show the views of each agent. This function can be useful in single step debug.

render(self)
This function plot scene in animation, call in each step

Example

Here is an example using test function, and it is in "test_MoveBox.py". The actions for agents are random chosen, click and run.

```
from env_MoveBox import EnvMoveBox
import random
import matplotlib.pyplot as plt

env = EnvMoveBox()
max_iter = 100000
for i in range(max_iter):
    print("iter= ", i)
    plt.figure(figsize=(3, 3))
    plt.imshow(env.get_global_obs())
    plt.xticks([])
    plt.yticks([])
    plt.show()
    # env.render()
    action1 = random.randint(0, 3)
    action2 = random.randint(0, 3)
    reward, done = env.step([action1, action2])
    print('reward', reward)
    if reward > 0:
        print('reset')
```