WuBenjaminAssignment11

May 15, 2022

1 CS156 (Introduction to AI), Spring 2022

2 Homework 11 submission

2.0.1 Roster Name: Benjamin Wu

2.0.2 Student ID: 013607880

2.0.3 Email address: benjamin.wu01@sjsu.edu

2.1 Solution

2.2 Import libraries, setup random seed

[ ]:
```
import numpy as np
import gym
```

[ ]:
```
np.random.seed(42)
```

2.3 References and sources

List all your references and sources here. This includes all sites/discussion boards/blogs/posts/etc. where you grabbed some code examples.

2.4 Code the solution

[ ]:
```
env = gym.make("FrozenLake-v1", is_slippery=False).env
env.seed(42)
env.reset()
env.render()

print("Action Space {}".format(env.action_space))
print("State Space {}".format(env.observation_space))
```

```
SFFF
FHFH
FFFH
HFFG
```

```
Action Space Discrete(4)
State Space Discrete(16)
```

[ ]:
```
qtable = np.zeros([env.observation_space.n, env.action_space.n]) #You could also make this dynamic if you don't know all games states upfront
discount = 0.9 # discount factor
learningrate = 0.9 # learning rate
epsilon = 0.2 # threshold of stochasticity
for episode in range(1,10001):
    done = False
    reward_total = 0
    state = env.reset()
    while done != True:
        explore_eploit = np.random.uniform(0, 1)
        if explore_eploit < epsilon:
            action = env.action_space.sample() # explore action space
        else:
            action = np.argmax(qtable[state]) # exploit learned values

        state_new, reward, done, info = env.step(action) #take the action
        qtable[state,action] += learningrate * (reward + discount * np.max(qtable[state_new,:]) - qtable[state,action]) #Update Q-marix using Bellman equation
        reward_total = reward_total + reward
        state = state_new
```

[ ]:
```
print(qtable)
```

```
[[0.531441   0.59049    0.59049    0.531441  ]
 [0.531441   0.         0.6561     0.59048888]
 [0.59049    0.729      0.58830519 0.65607312]
 [0.65603439 0.         0.53090956 0.4782969 ]
 [0.59049    0.6561     0.         0.531441  ]
 [0.         0.         0.         0.        ]
 [0.         0.81       0.         0.65583132]
 [0.         0.         0.         0.        ]
 [0.6561     0.         0.729      0.59049   ]
 [0.6561     0.81       0.81       0.        ]
 [0.729      0.9        0.         0.729     ]
 [0.         0.         0.         0.        ]
 [0.         0.         0.         0.        ]
 [0.         0.81       0.9        0.729     ]
 [0.81       0.9        1.         0.81      ]
 [0.         0.         0.         0.        ]]
```

[ ]:
```
reward_total=0
obs= env.reset()
```

```
env.render()
done=False
while done != True:
    action = np.argmax(qtable[obs])
    obs, reward, done, info = env.step(action) #take step using selected action
    reward_total = reward_total + reward
    env.render()
#Print the reward of these actions
print("Total reward is %r" % reward_total)
```

```
SFFF
FHFH
FFFH
HFFG
  (Down)
SFFF
FHFH
FFFH
HFFG
  (Down)
SFFF
FHFH
FFFH
HFFG
  (Right)
SFFF
FHFH
FFFH
HFFG
  (Down)
SFFF
FHFH
FFFH
HFFG
  (Right)
SFFF
FHFH
FFFH
HFFG
  (Right)
SFFF
FHFH
FFFH
HFFG
Total reward is 1.0
```