گزارش تمرین ۵ ام هوش مصنوعی دکتر رهبان

امیر حسین باقری ۹۸۱۰۵۶۲۱

فهرست مطالب

۱ algorithm

در اینجا از روش epsilon decay استفاده می‌کنیم. در این روش مقدار اپسیلون را پس از هر اپیزود کم می‌کنیم تا explore محیط کمتر شود.
برای update مقادیر Q-table مطابق زیر نیز عمل می‌کنیم.

$$Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \alpha(r_{t+1} + \gamma \max_a Q_t(s_{t+1}, a) - Q_t(s_t, a_t))$$

همچنین برای گسسته سازی فضا بازه 0.07- تا 0.07 که برای سرعت است را به۱۵ قسمت و بازه 1.2- تا 0.6 که برای بازه مختصات است را به ۱۹ قسمت تقسیم می‌کنیم.

۲ code

در انتهای هر بخش مانند **train** و یا تست نتایج را اعم از جدول **Q** ها و نتیجه ریوارد ها و غیره را در قالب یک npy ذخیره می‌کنیم.

```
class DQAgent:
    def __init__(self, env):
        self.env = env
        num_states = (env.observation_space.high - env.
    observation_space.low) * np.array([10, 100])
```

```
        num_states = np.round(num_states, 0).astype(int) + 1
        self.Q = np.zeros((num_states[0], num_states[1],env.
    action_space.n))
        self.min = env.observation_space.low
    def save(self, path,start,end):
        np.save(path, self.Q)
        file = open('training_time.txt', 'w')
        file.write("start time is : "+str(start)+"\n"+"end time is : "+
    str(end)+"\n"+"total time is : "+str(end - start))
        file.close()
    def load(self, path):
        self.Q = np.load(path)
    def test(self):
      reward = 0
      env = self.env
      Q = self.Q
      obs = env.reset()
      done = False
      while done != True:
        state = self.get_state(obs)
        action = np.argmax(Q[state[0], state[1]])
        obs, r, done, info = env.step(action)
        reward +=r
        if done and obs[0] >= 0.5:
          break
      return reward
    def get_state(self,obs):
      return np.round((obs - self.min) * np.array([10, 100])).astype(
    int)
    def train(self, learning, discount, epsilon, minE, episodes):
        reduction = (epsilon - minE) / episodes
        env = self.env
        Q = self.Q
        start = datetime.now()
        for e in range(episodes) :
            done = False
            obs = env.reset()
            current = self.get_state(obs)
            while done != True:
                i,j = current[0],current[1]
                action = np.argmax(Q[i, j]) if np.random.random() >
    epsilon else np.random.randint(0, 3)
                obs, reward, done, info = env.step(action)
                next_state = self.get_state(obs)
                i_p,j_p = next_state[0],next_state[1]
                if done and obs[0] >= 0.5:
                    Q[i,j, action] = reward
                else:
                    Q[i,j, action] += learning * (reward + discount *
    np.max(Q[i_p,j_p]) -Q[i,j, action])
                current = next_state
                epsilon -= reduction
        env.close()
        self.Q = Q
        end = datetime.now()
        self.save("policy",start,end)
        return start,end
```

Q-table ها یک جدول ۳*۱۹*۱۵ هستند
بخش train
نیز بر حسب توضیحات داده شده در مقدمه کار می‌کند.

۳ results

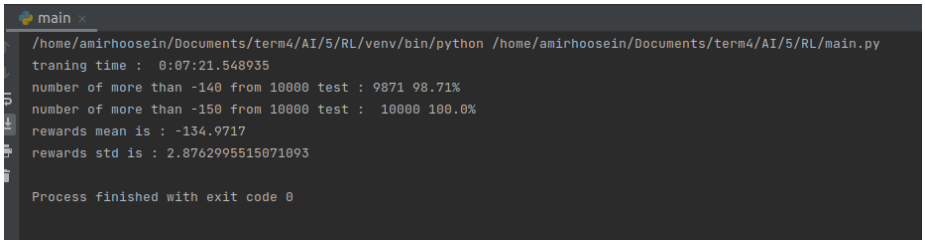
دقت کنید که نتایج حاصل از کد ران شده در دستگاه با نتایج حاصل از کد ران شده در کولب مقدار بسیار اندکی تفاوت بکنند.

۱.۳ نتیجه تایم train

start time is : 2021-07-15 03:02:40.025386
end time is : 2021-07-15 03:10:01.574321
total time is : 0:07:21.548935

۲.۳ میانگین و انحراف معیار ۱۰۰۰۰ بار ران تست

number of more than -140 from 10000 test : 9871 98.71%
number of more than -150 from 10000 test : 10000 100.0%
rewards mean is : -134.9717
rewards std is : 2.8762995515071093



۳.۳ نمودار نتایج برای ۱۰۰۰۰ بار تست