Assignment 3 - Cart Pole

July 12, 2021

1 Assignment 3

1.1 Bulut Fıçıcı

1.2 Tried to implement the Watkin's Q Lambda algorthym but continuous states were the main problem faced with.

1.3 The article below gave a Cart-Pole example with Q Lambda, but it includes discretization of the continuous state variables. However, pseudo-code in the book or the Chapter 9 does not contain discretization process.

http://karanmg.com/Computers/reinforcementLearning/finalProject/KaranComparisonOfSarsaWatkins.pdf

1.4 Tried different approaches but couldn't find a solution for the problem. Hence tried to gave you a full solution.

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.kernel_approximation import RBFSampler
from sklearn.preprocessing import PolynomialFeatures
import time
import gym
```

[2]:
```
env = gym.make("CartPole-v0")
# initial state
s=env.reset()
s
```

[2]:
```
array([-0.03334536,  0.00114822, -0.03873455, -0.03685818])
```

[3]:
```
env.reset()
done=False
while not done:
    a=env.action_space.sample() #random action
    s_next, r, done, info = env.step(a)
    time.sleep(0.05)
    env.render()
env.close()
```

[4]:
```
def merge_state_action(s, a):
    return np.concatenate((s, [a]))
```

[5]:
```
s=env.reset()
a=env.action_space.sample()
sa=merge_state_action(s, a)
sa
```

[5]:
```
array([-0.04282015, -0.00518802, -0.02220161, -0.03140426,  1.        ])
```

[6]:
```
def gather_samples(n_episodes=10000):
    samples = []
    for i in range(n_episodes):
        s = env.reset()
        done=False
        while not done:
            a=env.action_space.sample() #random action
            sa = merge_state_action(s, a)
            samples.append(sa)
            s, r, done, info = env.step(a)
    return samples
```

[7]:
```
samples=gather_samples(n_episodes=10000)
```

[8]:
```
rbf_feature = RBFSampler(n_components=100)
rbf_feature.fit(samples)
```

[8]:
```
RBFSampler()
```

[9]:
```
def predictV(s,a,w,kernel):
    sa = merge_state_action(s, a)
    if kernel=='poly':
        poly_features.fit([sa])
        x=poly_features.transform([sa])[0]
    elif kernel=='rbf':
        x = rbf_feature.transform([sa])[0]
    else:
        x=nystrom_featurizer.transform([sa])[0]
    return np.dot(x,w)
```

[10]:
```
def gradientV(s,a,kernel):
    sa = merge_state_action(s, a)
    if kernel=='poly':
        poly_features.fit([sa])
        x=poly_features.transform([sa])[0]
    elif kernel=='rbf':
        x = rbf_feature.transform([sa])[0]
    else:
```

```
        x=nystrom_featurizer.transform([sa])[0]
    return x
```

[11]:
```
def predictV_all_actions(s,w,kernel):
    values=[]
    for a in range(env.action_space.n):
        values.append(predictV(s,a,w,kernel))
    return values
```

[12]:
```
def epsilon_greedy(s, w, kernel, eps=0.1):
    values=[]
    p = np.random.random()
    if p < (1 - eps):
        values = predictV_all_actions(s,w,kernel)
        return np.argmax(values)
    else:
        return env.action_space.sample()
```

[13]:
```
def watch_agent(w,kernel):
    done = False
    episode_reward = 0
    s = env.reset()
    while not done:
        a = epsilon_greedy(s, w, kernel,eps=0)
        s, r, done, info = env.step(a)
        time.sleep(0.05)
        env.render()
        episode_reward += r
    print("Episode reward:", episode_reward)
```

[14]:
```
GAMMA=1
ALPHA = 0.1
w = np.zeros(100)
n_episodes = 500
for it in range(n_episodes):
    s=env.reset()
    done=False
    while not done:
        a = epsilon_greedy(s, w, kernel='rbf',eps=0.1)
        s2, r, done, info = env.step(a)
        if done:
            target = r
        else:
            values = predictV_all_actions(s2,w,kernel='rbf')
            target = r + GAMMA * np.max(values)
        g = gradientV(s,a,kernel='rbf')
        err=target - predictV(s,a,w,'rbf')
```

```
        w += ALPHA * err * g
        # update state
        s = s2
```

[15]:
```
watch_agent(w,kernel='rbf')
env.close()
```

```
Episode reward: 132.0
```