Лабораторная работа №4.

Q-обучение

1. Подготовительные действия

```
import sys, os
# Этот код создает виртуальный дисплей для рисования игровых изображений.
# Это не будет иметь никакого эффекта, если на вашей машине есть монитор.
if type(os.environ.get("DISPLAY")) is not str or
len(os.environ.get("DISPLAY")) == 0:
    os.environ['DISPLAY'] = ':1'

import numpy as np
import matplotlib.pyplot as plt

from collections import defaultdict
import random
import math
import numpy as np
```

2. Класс QLearningAgent

Требуется дополнить код. Что должен выполнять код в каждой позиции <ВАШ КОД> указано в примечаниях к коду.

```
class QLearningAgent:
    def __init__(self, alpha, epsilon, discount, get_legal_actions):
        """
        Q-Learning Agent
        based on https://inst.eecs.berkeley.edu/~cs188/sp19/projects.html
        Переменные экземпляра, к которым у вас есть доступ
           - self.epsilon (исследование)
           - self.alpha (скорость обучения)
           - self.discount (дисконт, она же гамма)

         Функции, которые вы должны использовать
           - self.get_legal_actions(state) {состояние, хешируемое -> список
действий, каждое из которых хешируемое}
             который возвращает разрешенные действия для состояния
           - self.get_qvalue (состояние, действие)
             который возвращает Q (состояние, действие)
           - self.set_qvalue (состояние, действие, значение)
             который устанавливает Q (состояние, действие): = значение
         !!!Важно!!!
         Примечание: пожалуйста, избегайте прямого использования
self._qValues.
             Для этого есть специальный self.get_qvalue/set_qvalue.
        """

        self.get_legal_actions = get_legal_actions
        self._qvalues = defaultdict(lambda: defaultdict(lambda: 0))
        self.alpha = alpha
        self.epsilon = epsilon
        self.discount = discount

    def get_qvalue(self, state, action):
        """ Возвращает Q(state,action) """
        return self._qvalues[state][action]

    def set_qvalue(self, state, action, value):
        """ Устанавливает Qvalue для [state,action] в определенное значение
```

```
"""
        self._qvalues[state][action] = value

    #---------------------ВЫ НАЧИНАЕТЕ ДОБАВЛЯТЬ СВОЙ КОД С ЭТОГО МЕСТА------
---------------#

    def get_value(self, state):
        """
        Вычислите оценку вашего агента V (s), используя текущие значения q.
         V(s) = max_over_action Q(состояние,действие) по возможным действиям.
         Примечание: обратите внимание, что значения q могут быть
отрицательными.
        """
        possible_actions = self.get_legal_actions(state)

        # If there are no legal actions, return 0.0
        if len(possible_actions) == 0:
            return 0.0

        <ВАШ КОД>

        return value

    def update(self, state, action, reward, next_state):
        """
        Необходимо обновить значение Q-Value:
           Q(s,a) := (1 - alpha) * Q(s,a) + alpha * (r + gamma * V(s'))
        """

        # agent parameters
        gamma = self.discount
        learning_rate = self.alpha

        <ВАШ КОД>

        self.set_qvalue(state, action, <ВАШ КОД: Q-value> )

    def get_best_action(self, state):
        """
        Вычислите наилучшее действие для состояния (используя текущие
значения q).
        """
        possible_actions = self.get_legal_actions(state)

        # If there are no legal actions, return None
        if len(possible_actions) == 0:
            return None

        <ВАШ КОД>

        return best_action

    def get_action(self, state):
        """
        Вычислите действие, которое нужно предпринять в текущем состоянии,
включая исследование.
         С вероятностью self.epsilon мы должны предпринять случайное
действие.
             иначе - лучшее действие политики (self.get_best_action).

         Примечание. Чтобы выбрать случайным образом из списка, используйте
random.choice(list).
               Чтобы выбрать True или False с заданной вероятностью,
сгенерируйте универсальное число в [0, 1]
```

```
                и сравните с вашей вероятностью
        """

        # Pick Action
        possible_actions = self.get_legal_actions(state)
        action = None

        # If there are no legal actions, return None
        if len(possible_actions) == 0:
            return None

        # agent parameters:
        epsilon = self.epsilon

        <ВАШ КОД>

        return chosen_action
```

Попробуем на среде такси. Здесь мы используем агент *qlearning* на такси *env* от *openai gym*. Вам нужно будет вставить сюда несколько функций агента.

```
import gym
env = gym.make("Taxi-v3")

n_actions = env.action_space.n
```

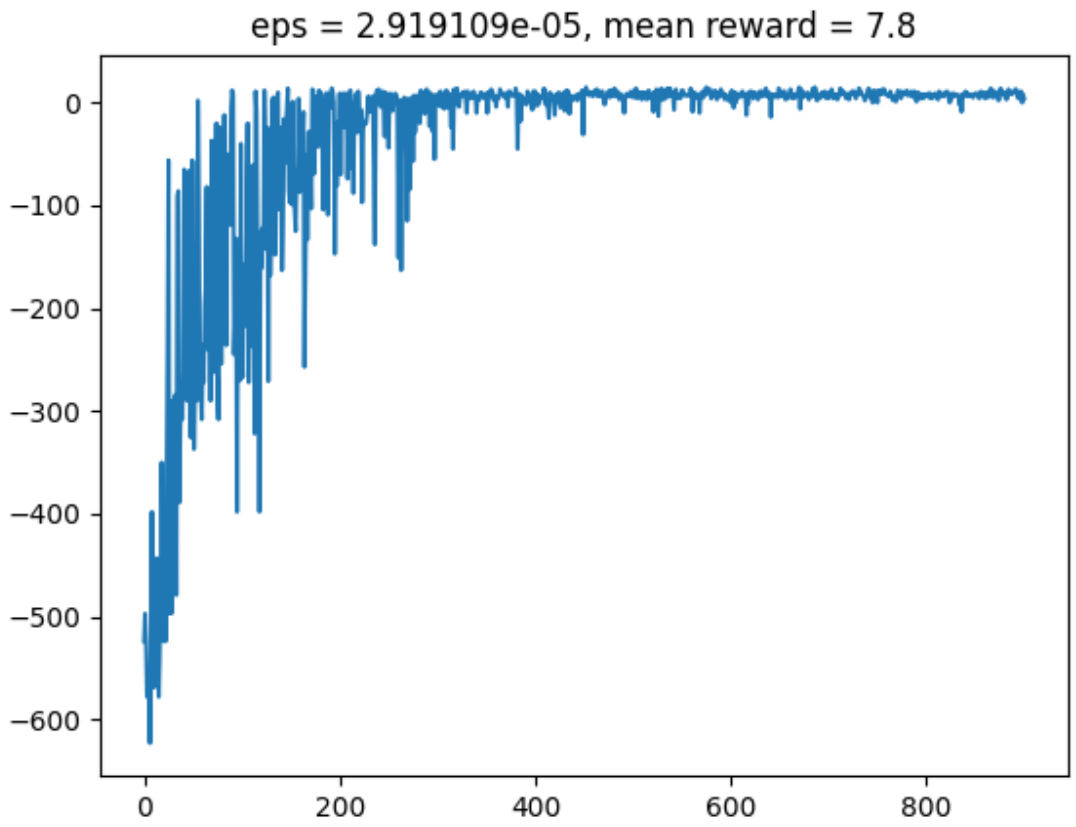
```
agent = QLearningAgent(
    alpha=0.5, epsilon=0.25, discount=0.99,
    get_legal_actions=lambda s: range(n_actions))
```

```
def play_and_train(env, agent, t_max=10**4):
    """
    Эта функция должна
     - запустить полную игру, действия заданы политикой e-greeding агента
     - обучать агента, используя agent.update(...) всякий раз, когда это
возможно
     - вернуть общую награду
    """
    total_reward = 0.0
    s = env.reset()

    for t in range(t_max):
        # get agent to pick action given state s.
        a = <ВАШ КОД>

        next_s, r, done, _ = env.step(a)

        # train (update) agent for state s
        <ВАШ КОД>

        s = next_s
        total_reward += r
        if done:
            break

    return total_reward
```

```
from IPython.display import clear_output
```

```
rewards = []
for i in range(1000):
    rewards.append(play_and_train(env, agent))
    agent.epsilon *= 0.99

    if i % 100 == 0:
        clear_output(True)
        plt.title('eps = {:e}, mean reward = {:.1f}'.format(agent.epsilon,
np.mean(rewards[-10:])))
        plt.plot(rewards)
        plt.show()
```

Вывод:



3. Бинаризованные пространства состояний

Используйте агент для эффективного обучения на CartPole-v0. Эта среда имеет непрерывный набор возможных состояний, поэтому вам придется каким-то образом сгруппировать их в бины. Самый простой способ — использовать round(x, n_digits) (или np.round) для округления действительного числа до заданного количества цифр. Сложность заключается в том, чтобы правильно подобрать n_digits для каждого состояния для эффективного обучения. Обратите внимание, что вам нужно преобразовывать состояние не в целые числа, а в кортежи любых значений.

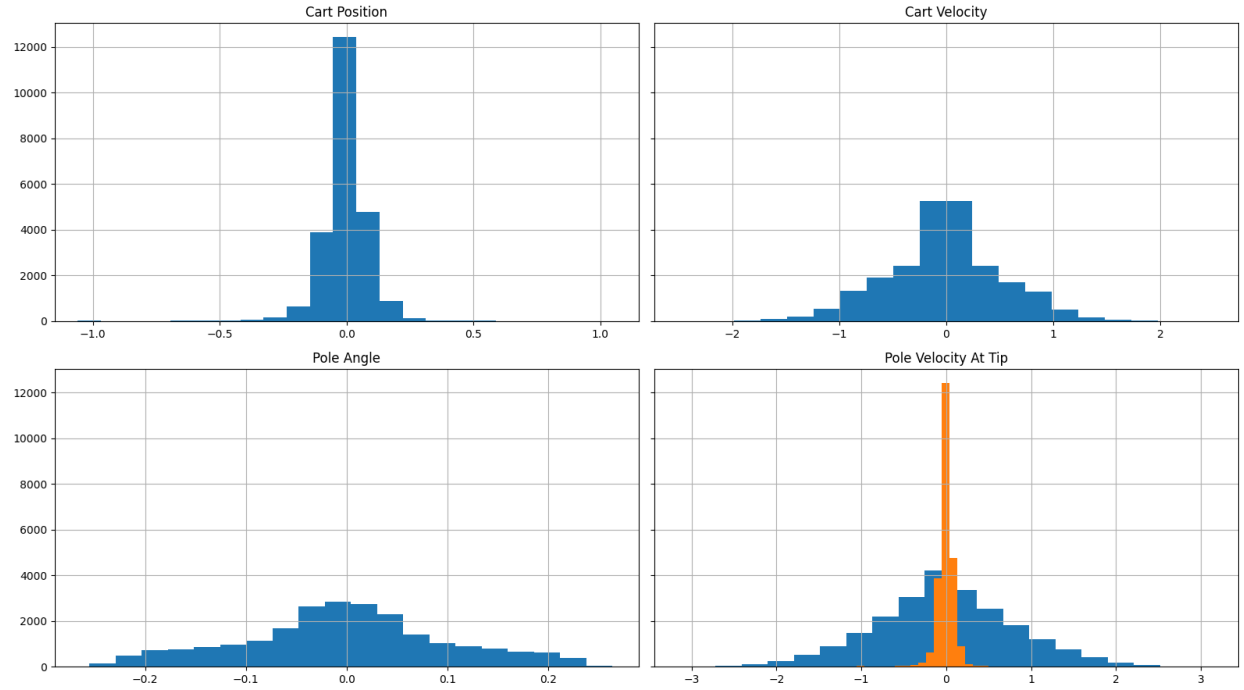
```
def make_env():
    return gym.make('CartPole-v0').env  # .env unwraps the TimeLimit wrapper

env = make_env()
n_actions = env.action_space.n
```

```
print("first state: %s" % (env.reset()))
plt.imshow(env.render('rgb_array'))
```

Нам нужно оценить распределения наблюдений. Для этого мы сыграем несколько игр и запишем все состояния.

```
seen_observations = []
for _ in range(1000):
    seen_observations.append(env.reset())
    done = False
    while not done:
        s, r, done, _ = env.step(env.action_space.sample())
        seen_observations.append(s)

seen_observations = np.array(seen_observations)

for obs_i in range(env.observation_space.shape[0]):
    plt.hist(seen_observations[:, obs_i], bins=20)
    plt.show()
```

Вывод:



4. Бинаризованная среда

Требуется дополнить код. Что должен выполнять код в каждой позиции <ВАШ КОД> указано в примечаниях к коду.

```
from gym.core import ObservationWrapper


class Binarizer(ObservationWrapper):
    def observation(self, state):
        # Hint: Используйте round(x, n_digits).
        # Вы можете выбрать разные n_digits для каждого измерения..
        state = <ВАШ КОД: округлите состояние до нескольких значимых цифр>

        return tuple(state)
```

```
env2 = Binarizer(gym.make("CartPole-v0")).env
```

```
seen_observations = []
for _ in range(1000):
    seen_observations.append(env.reset())
    done = False
    while not done:
        s, r, done, _ = env.step(env.action_space.sample())
        seen_observations.append(s)
        if done:
            break

seen_observations = np.array(seen_observations)

for obs_i in range(env.observation_space.shape[0]):
    plt.hist(seen_observations[:, obs_i], bins=20)
    plt.show()
```

Задания к лабораторной работе

1. Написать свой код согласно заданиям для всех фрагментов, помеченных как: <ВАШ КОД:>

```
<ВАШ КОД: >
```

2. Запустить полученную программу, получить ожидаемые выходные данные.

Требование к отчету

1. В качестве отчета принимается Python файл с кодом.
2. Код должен быть в достаточной мере прокомментирован.
3. Отчет предоставляется в виде архива zip, формат имени архива: <номер группы>_<Фамилия_Имя>_LR<номер работы>.zip