دانشگاه صنعتی خواجه نصیرالدین طوسی
دانشکدهٔ مهندسی برق- گروه مهندسی کنترل

درس یادگیری ماشین

پاسخ مینی‌پروژهٔ چهارم

نام و نام خانوادگی	ایمان گندمی
شمارهٔ دانشجویی	۴۰۲۲۳۷۰۴

فهرست مطالب

لینک پوشهٔ گیت‌هاب:

https://github.com/ImanGandomi/MachineLearning2024/tree/main/ml_MiniProject_4

لینک گوگل گولب:

https://drive.google.com/file/d/1-tHIqd_tVjneoXtCRPTZ8l6QrclGb22U/view?usp=sharing

سوال دوم

قسمت الف.

Lunar Lander یک بازی است که در آن فرد با یک فرودگر ماه مانور می دهد تا سعی کند آن را با دقت روی یک سکوی فرود بیاورد. هدف توسعه یک عامل هوشمند است که بتواند یک ماژول ماه را به طور ایمن بر روی سطح ماه فرود بیاورد و در عین حال حداقل سوخت را مصرف کند. این محیط وظایف اصلی شبیه‌سازی، فیزیک، پاداش‌ها و کنترل بازی را انجام می‌دهد که به فرد اجازه می‌دهد تنها بر ساخت یک عامل تمرکز کند. اطلاعات وضعیت محیط با یک بردار 8 بعدی حاوی مقادیر پیوسته برای متغیرها، از جمله مختصات فرودگر، جهت گیری، سرعت ها و شاخص های تماس با زمین ارائه می شود.

مسئله و محیط بازی

ویژگی‌های کلیدی این محیط شامل فضای حالت، فضای عمل و سیستم پاداش است. آغاز حرکت کشتی فضایی از بالای صفحه با نیروی اولیه است و در سه حالت بازی به اتمام می‌رسد. کشتی فضایی سقوط کند، از صفحه خارج شود و هیچ حرکتی نکند.

فضای حالت شامل یک بردار دارای هشت متغیر پیوسته است.

- مختصات در محور x
- مختصات در محور y
- سرعت در محور x
- سرعت در محور y
- تماس پای چپ با زمین: بصورت یک عدد باینری که نشان‌دهنده در تماس بودن پای چپ با زمین است.
- تماس پای راست با زمین: بصورت یک عدد باینری که نشان‌دهنده در تماس بودن پای راست با زمین است.
- زاویه (θ): این مقدار زاویه فرودگر را نسبت به محور عمودی نشان می دهد.
- سرعت زاویه ای: این مقدار نشان می دهد که فرودگر با چه سرعتی در حال چرخش است.

فضای action شامل چهار عمل است:

- فعال‌کردن موتور راست
- فعال‌کردن موتور چپ
- فعال‌کردن موتور اصلی
- کاری نکردن

به عنوان مثال می‌توان یک برداری به صورت زیر تعریف کرد:

[x, y, x_dot, y_dot, θ, θ_dot, leg_contact_left, leg_contact_right] = [0.015, 1.232, -0.023, -0.459, 0.029, -0.005, 0, 0]

فضای پاداش بصورت زیر است:

- با تماس‌برقرار کردن هر پا با زمین، 10 امتیاز پاداش داده می‌شود.
- با فرود موفقیت آمیز بین 100 امتیاز پاداش داده می‌شود.
- با سقوط و تصادف، 100 جریمه می‌شود.
- با روشن بودن موتور اصلی 0.3 امتیاز جریمه می‌شود.
- با روشن‌بودن موتور جانبی، 0.3 امتیاز جریمه می‌شود.

ابتدا مطابق نوت‌بوک مورد نظر کتاب‌خانه‌ها و سایر موارد مورد نیاز نصب می‌شود

```
!pip install rarfile --quiet
!pip install stable-baselines3 > /dev/null
!pip install box2d-py > /dev/null
!pip install gym pyvirtualdisplay > /dev/null 2>&1
!sudo apt-get install -y xvfb python-opengl ffmpeg > /dev/null 2>&1

# install dependencies
!pip3 install gym --upgrade
!pip3 install pyglet
!pip3 install Box2D
!pip3 install box2d-py
!pip3 install gym[Box_2D]
!pip3 install gym[box2d]
```

مطابق این دستور اگر gpu در دسترس باشد، دیوایس را به gpu و در غیر این صورت به cpu می‌برد.

```
import torch

# if gpu is to be used
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
device
```

در مرحله نخست کتاب‌خانه‌های مورد نیاز برای قسمت اول را فراخوانی می‌کنیم.

```
from gym.wrappers import RecordVideo
import gym
import io
import os
import glob
import torch
from IPython.display import HTML
from IPython import display as ipythondisplay
from pyvirtualdisplay import Display
import random
import base64
import torch.optim as optim
import numpy as np
import matplotlib.pyplot as plt
from collections import namedtuple, deque
import torch.nn as nn
import torch.nn.functional as F
from stable_baselines3 import DQN
from stable_baselines3.common.results_plotter import ts2xy, load_results
from stable_baselines3.common.callbacks import EvalCallback
```

در این قسمت محیطی برای عامل RL تنظیم می‌شود. به این صورت که در state_size هشت حالت ممکنی که قبلا توضیح داده شد ذخیره می‌شود و در action_size چهار حرکت ممکن. ابعاد هم بصورت زیر قابل مشاهده است.

```
# enviroment
import gym
env = gym.make('LunarLander-v2')
#TODO: find observation size: 8
state_size = env.observation_space.shape[0]
#TODO: find action size: 4: 0- Do nothing 1- Fire left engine 2- Fire down engine 3- Fire right engine
action_size = env.action_space.n
state_size, action_size
```

```
(8, 4)
```

در دستور زیر یک صفحه نمایش مجازی با ابعاد داده شده تنظیم می‌شود.

```
# Set up virtual display
display = Display(visible=0, size=(1400, 900))
display.start()

"""
Utility functions to enable video recording of gym environment
and displaying it.
To enable video, just do "env = wrap_env(env)""
"""
```

در این قسمت تابعی تعریف شده است که امکان ضبط ویدیو از محیط بازی را فراهم کرده و آن را نمایش می‌دهد.

```
# Utility function to enable video recording of gym environment and displaying it
def show_video():
    mp4list = glob.glob('video/*.mp4')
    if len(mp4list) > 0:
        mp4 = mp4list[0]
        video = io.open(mp4, 'r+b').read()
        encoded = base64.b64encode(video)
        ipythondisplay.display(HTML(data='''<video alt="test" autoplay loop controls style="height: 400px;">
                    <source src="data:video/mp4;base64,{0}" type="video/mp4" />
                </video>'''.format(encoded.decode('ascii'))))
    else:
        print("Could not find video")
```

در کد زیر کلاسی با اسم Experience Replay تعریف شده است که در مباحث DQN و DDQN از اجزای کلیدی است. این قسمت با شکستن همبستگی بین تجربیات متوالی به تثبیت و بهبود آموزش یک عامل کمک می کند. Transition را در این قسمت به صورت ترکیبی از state, action, next-state, reward , done تعریف می‌کنیم. متد store-trans برای ذخیره سازی در حافظه استفاده می شود. متد sample برای نمونه گیری تصادفی در حافظه استفاده می شود.

```python
# experience replay
import random
from collections import namedtuple, deque

Transition = namedtuple('Transition', ('state', 'action', 'next_state', 'reward', 'done'))

class ExperienceReplay():
    def __init__(self, capacity):
        self.memory = deque([], maxlen=capacity)

    def store_trans(self, s, a, sp, r, done):
      # TODO: store new transition in memory
        transition = Transition(s, a, sp, r, done)
        self.memory.append(transition)

    def sample(self, batch_size):
      # TODO: take RANDOM sample from memory
        return random.sample(self.memory, batch_size)

    def __len__(self):
        return len(self.memory)
```

در این قسمت یک شبکهٔ Q-Deep تعریف شده است که یک شبکه عصبی است که برای تقریب تابع Q-value مورد استفاده قرار می‌گیرد. در این ساختار پاداش‌های مورد انتظار آینده برای یک جفت حالت-عمل تخمین زده می‌شود. ورودی برای این شبکه حالت محیط است.

```python
# DQN
import torch.nn as nn
import torch.nn.functional as F

# Deep Q-Network
class DeepQNetwork(nn.Module):
    def __init__(self, state_size, action_size):
        super(DeepQNetwork, self).__init__()
        # TODO: define the architecture
        # NOTE: input=observation/state, output=action
        self.net = nn.Sequential(
            nn.Linear(state_size, 512),
            nn.ReLU(),
            nn.LayerNorm(512),
            nn.Dropout(0.1),
            nn.Linear(512, 512),
            nn.ReLU(),
            nn.LayerNorm(512),
            nn.Dropout(0.1),
            nn.Linear(512, 512),
            nn.ReLU(),
            nn.Linear(512, action_size)
        )

    def forward(self, x):
        # TODO: forward propagation
        # NOTE: use ReLu for activation function in all layers
        # NOTE: last layer has no activation function (predict action)
        # ReLU is created in init, no need here
        return self.net(x)
```

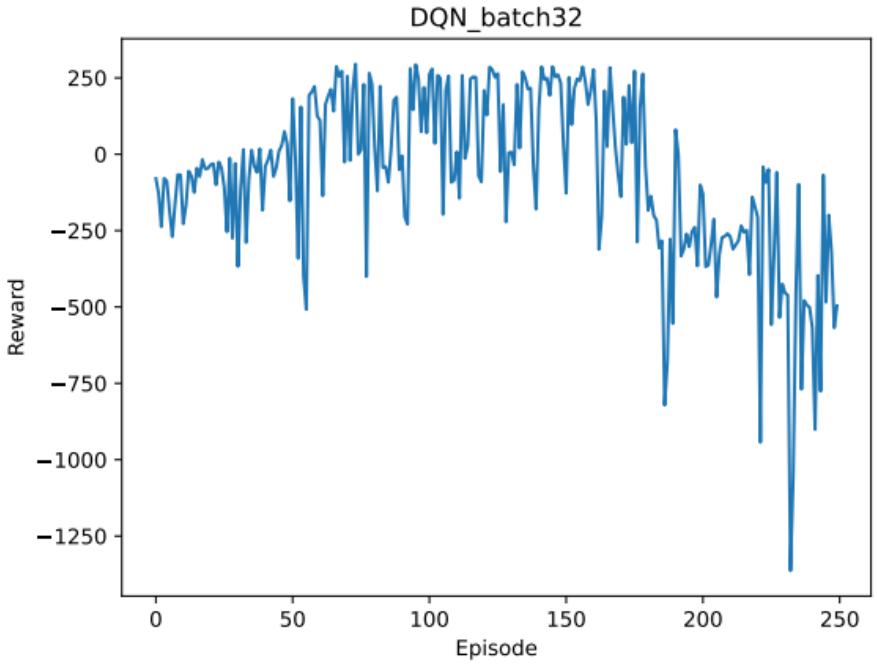
قسمت ب

در این قسمت هایپرپارامترهای مورد نیاز را تعریف میکنیم. تعداد اپیزود را برابر 250 و بچ سایز را برای این مرحله برابر 32 در نظر میگیریم. مطابق خواسته مسئله که بچ‌سایزهای برابر 32 و 64 و 128 و به ازای اپیزودهای 50 و 100 و 150 و 200 و 250 خواسته در ادامه عمل میکنیم و از آوردن توضیحات بدیهی برای هر یک جلوگیری میکنیم.

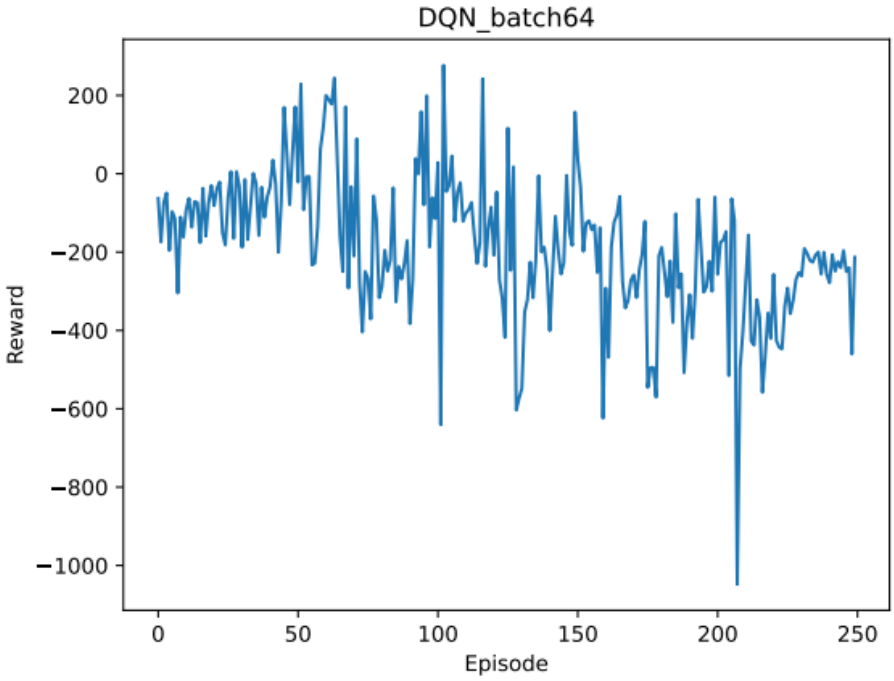
```
n_episodes = 250
eps = 1.0
eps_decay_rate = 0.97
eps_end = 0.01
BATCH_SIZE = 32
```

در این حالت آموزش را در یک حلقه for انجام می‌دهیم. (چون کدها از قبل آماده و در اختیار بوده است از آوردن توضیحات بدیهی شامل نحوه عملکرد حلقه for خودداری شده است)

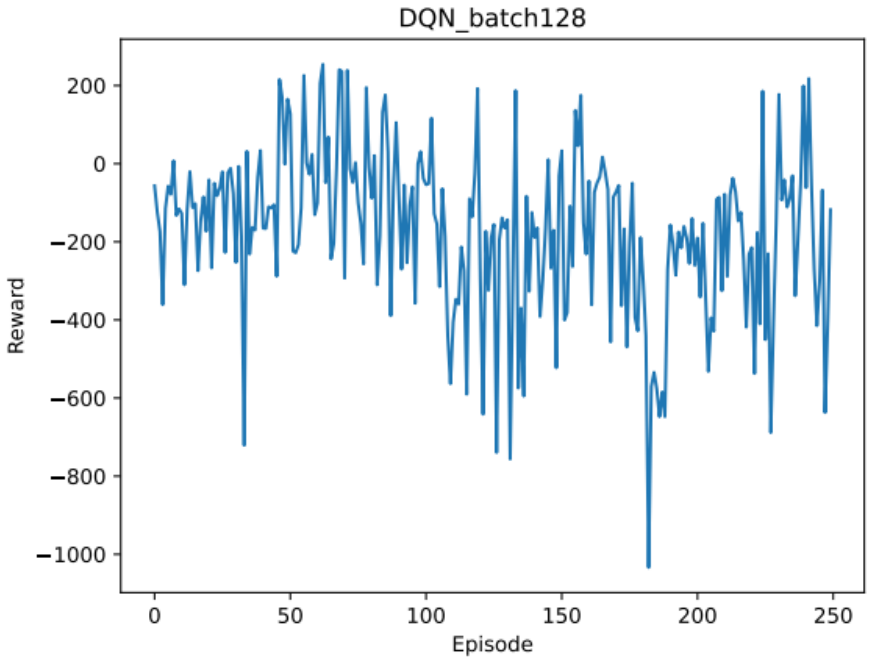
با پیاده کردن کدهای مربوطه برای بچ‌سایزهای مختلف نمودار پاداش بر حسب تعداد دوره آموزش به صورت زیر است:



نمودار پاداش برای batch_size=32



نمودار پاداش برای64=batch_size



نمودار پاداش برای batch_size=128

با توجه به پلات‌ها برخی از نتایج را می‌توان از عملکرد مدل ارزیابی کرد.

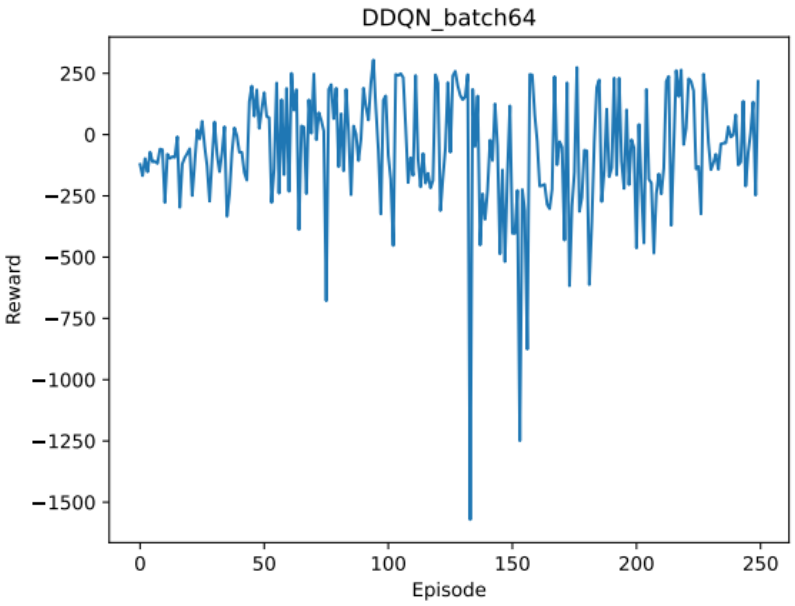
به طور کلی میزان پاداش در هر سه نمودار تا یک جایگاه مشخصی دارای روند صعودی بوده است و پس از آن نزولی شده است. اما بلخره این روند صعودی در یک بازه نشانه‌ای از آموزش مدل است و تا قبل از نزولی شدن مطلوب است. اندازه دسته 32 بهترین عملکرد را از نظر حداکثر پاداش بالاتر و یادگیری نسبتاً پایدار نشان می دهد. این نشان می دهد که اندازه دسته کوچکتر ممکن است برای این مشکل خاص مؤثرتر باشد. برای دسته برابر با 64 و 128 پاداش ها روند افزایشی را نشان می دهند، اما بهبود کلی در مقایسه با اندازه دسته 32 کندتر به نظر می رسد.

برای رسیدن به بهبودهای بیشتر میتوان موردهایی مثل تنظیم دقیق نرخ یادگیری و آزمایش با معماری های مختلف شبکه را درنظر گرفت.

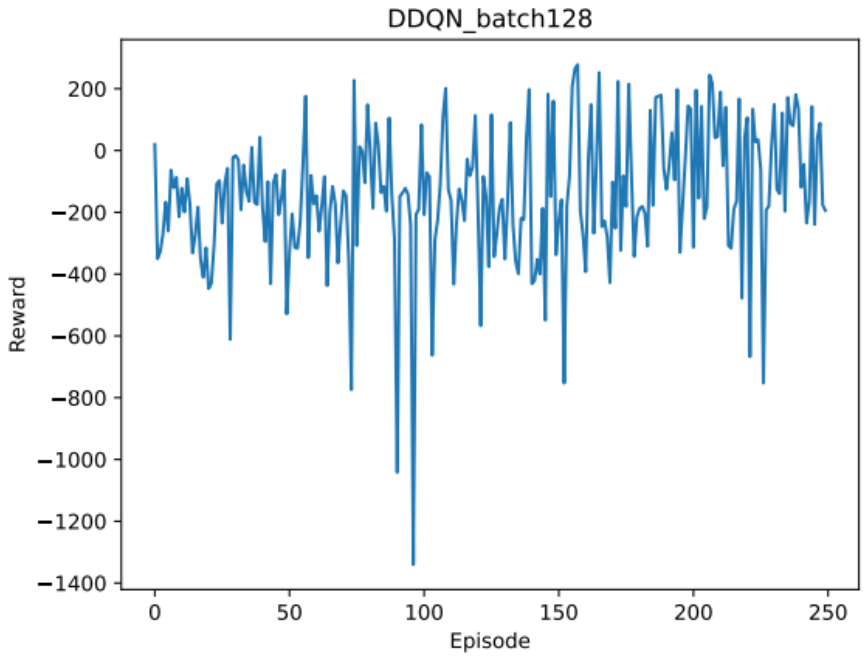
قسمت ج

در این قسمت با استفاده از کلاس تعریف شده و در درسترس DDQNAgent کار آموزش را انجام می‌دهیم. تفاوت بین این روش و روش قبلی در روش به روزرسانی Q-values است.

در این مرحله نیز در به ازای بچ‌سایزهای 64 و 128 و به ازای اپیزودهای 50 و 100 و 150 و 200 و 250 کار آموزش را جلو میبریم. نمودارهای پاداش بر حسب دورهٔ آموزش به صورت زیر است.



نمودار پاداش برای batch_size=64



نمودار پاداش برای batch_size=128

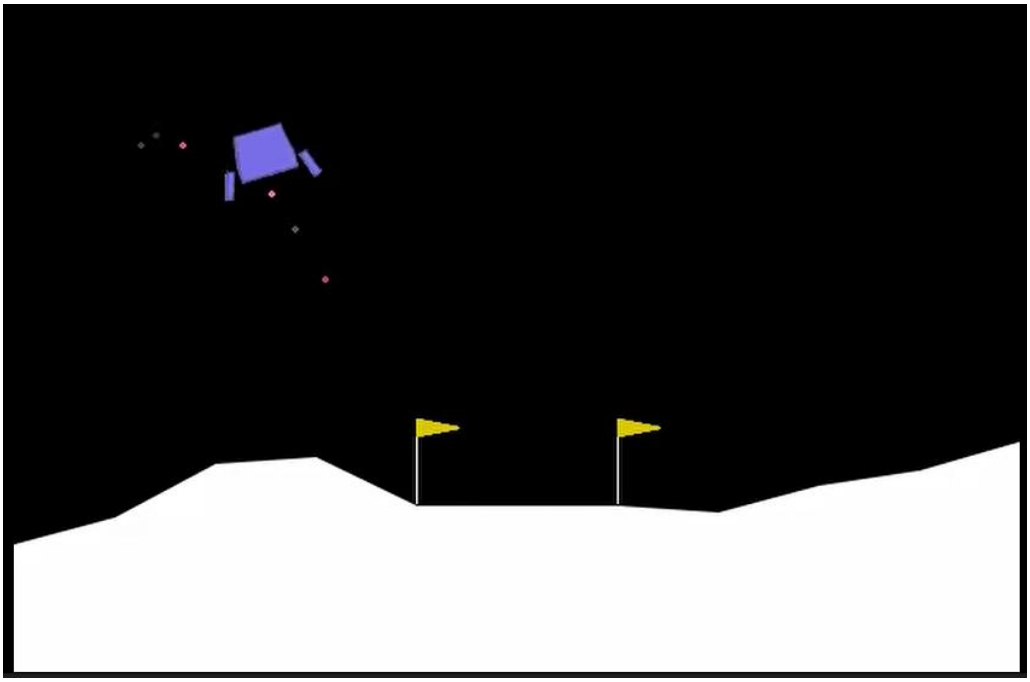
در این حالت با توجه به نمودارهای آورده شده عملکرد در اندازه بچ‌سایز برابر 64 کمی مناسب‌تر است اما ناپایدار بوده و سیر صعودی کاملا مشهود نیست. نوسان بسیار زیادی داشته و در حالت اندازهٔ 64 ماکزیموم پاداش بیشتر از حالت با اندازهٔ بچ 128 است.

در حالت کلی در همهٔ آموزش‌های روند آموزش تا اپیاک مشخصی مناسب بوده و روند صعودی داشته است. اما پس از آن روند نویزی‌تر و نزولی شده است. بنابراین در همهٔ آموزش‌ها با تعداد بچ‌های متفاوت در اپیاک های وسط و نه در انتهای آموزش، آموزش بهتری و پاداش مناسب‌تری داریم و احتمالا سفینه فضایی بهتر توانسته است فرود بیاید.

یک نمونه فرود مناسب از طریق این لینک قابل مشاهده است.

همچنین عملکرد نهایی آموزش‌ها بصورت ویدیویی در بچ‌سایزهای و دوره‌های آموزشی مختلف از طریق این لینک در دسترس است.

لینک دسترسی به گیت‌هاب مینی‌پروژه



یک فریم از خروجی ویدیویی فرود سفینه فضایی