eplg9tdan

March 30, 2023

:#Programming Assignment - 2 : Actor-Critic One-step

[1]:

```
'''
Installing packages for rendering the game on Colab
'''

!pip install gym pyvirtualdisplay > /dev/null 2>&1
!apt-get install -y xvfb python-opengl ffmpeg > /dev/null 2>&1
!apt-get update > /dev/null 2>&1
!apt-get install cmake > /dev/null 2>&1
!pip install --upgrade setuptools 2>&1
!pip install ez_setup > /dev/null 2>&1
!pip install gym[atari] > /dev/null 2>&1
!pip install git+https://github.com/tensorflow/docs > /dev/null 2>&1
!pip install gym[classic_control]
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-
wheels/public/simple/
Requirement already satisfied: setuptools in /usr/local/lib/python3.9/dist-
packages (67.6.1)
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-
wheels/public/simple/
Requirement already satisfied: gym[classic_control] in
/usr/local/lib/python3.9/dist-packages (0.25.2)
Requirement already satisfied: numpy>=1.18.0 in /usr/local/lib/python3.9/dist-
packages (from gym[classic_control]) (1.22.4)
Requirement already satisfied: importlib-metadata>=4.8.0 in
/usr/local/lib/python3.9/dist-packages (from gym[classic_control]) (6.1.0)
Requirement already satisfied: cloudpickle>=1.2.0 in
/usr/local/lib/python3.9/dist-packages (from gym[classic_control]) (2.2.1)
Requirement already satisfied: gym-notices>=0.0.4 in
/usr/local/lib/python3.9/dist-packages (from gym[classic_control]) (0.0.8)
Collecting pygame==2.1.0
  Downloading
pygame-2.1.0-cp39-cp39-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (18.3 MB)
                           18.3/18.3 MB
33.6 MB/s eta 0:00:00
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.9/dist-
```

```
packages (from importlib-metadata>=4.8.0->gym[classic_control]) (3.15.0)
Installing collected packages: pygame
  Attempting uninstall: pygame
    Found existing installation: pygame 2.3.0
    Uninstalling pygame-2.3.0:
      Successfully uninstalled pygame-2.3.0
Successfully installed pygame-2.1.0
```

[ ]:
```
tf.config.list_physical_devices('GPU')
```

[ ]:
```
[PhysicalDevice(name='/physical_device:GPU:0', device_type='GPU')]
```

[2]:
```
'''
A bunch of imports, you don't have to worry about these
'''


import numpy as np
import random
import torch
import torch.nn as nn
import torch.nn.functional as F
from collections import namedtuple, deque
import torch.optim as optim
import datetime
import gym
from gym.wrappers.record_video import RecordVideo
import glob
import io
import base64
import matplotlib.pyplot as plt
from IPython.display import HTML
from pyvirtualdisplay import Display
import tensorflow as tf
from IPython import display as ipythondisplay
from PIL import Image
import tensorflow_probability as tfp
```

0.1 Part 2: One-Step Actor-Critic Algorithm

Actor-Critic methods learn both a policy $\pi(a|s;\theta)$ and a state-value function $v(s;w)$ simultaneously. The policy is referred to as the actor that suggests actions given a state. The estimated value function is referred to as the critic. It evaluates actions taken by the actor based on the given policy. In this exercise, both functions are approximated by feedforward neural networks.

- The policy network is parametrized by θ - it takes a state s as input and outputs the probabilities $\pi(a|s;\theta)\ \forall\ a$
- The value network is parametrized by w - it takes a state s as input and outputs a scalar value associated with the state, i.e., $v(s;w)$

- The single step TD error can be defined as follows:

$$\delta_t = R_{t+1} + \gamma v(s_{t+1}; w) - v(s_t; w)$$

- The loss function to be minimized at every step ($L_{tot}^{(t)}$) is a summation of two terms, as follows:

$$L_{tot}^{(t)} = L_{actor}^{(t)} + L_{critic}^{(t)}$$

where,

$$L_{actor}^{(t)} = -\log \pi(a_t|s_t; \theta)\delta_t$$

$$L_{critic}^{(t)} = \delta_t^2$$

- **NOTE: Here, weights of the first two hidden layers are shared by the policy and the value network**
  - First two hidden layer sizes: [1024, 512]
  - Output size of policy network: 2 (Softmax activation)
  - Output size of value network: 1 (Linear activation)

0.1.1 Initializing Actor-Critic Network

[3]:

```
class ActorCriticModel(tf.keras.Model):
    """
    Defining policy and value networkss
    """
    def __init__(self, action_size, n_hidden1=1024, n_hidden2=512):
        super(ActorCriticModel, self).__init__()

        #Hidden Layer 1
        self.fc1 = tf.keras.layers.Dense(n_hidden1, activation='relu')
        #Hidden Layer 2
        self.fc2 = tf.keras.layers.Dense(n_hidden2, activation='relu')

        #Output Layer for policy
        self.pi_out = tf.keras.layers.Dense(action_size, activation='softmax')
        #Output Layer for state-value
        self.v_out = tf.keras.layers.Dense(1)

    def call(self, state):
        """
        Computes policy distribution and state-value for a given state
        """
        layer1 = self.fc1(state)
        layer2 = self.fc2(layer1)

        pi = self.pi_out(layer2)
        v = self.v_out(layer2)

        return pi, v
```

```
/usr/local/lib/python3.9/dist-packages/ipykernel/ipkernel.py:283:
DeprecationWarning: `should_run_async` will not call `transform_cell`
automatically in the future. Please pass the result to `transformed_cell`
argument and any exception that happen during thetransform in
`preprocessing_exc_tuple` in IPython 7.17 and above.
  and should_run_async(code)
```

0.1.2 Agent Class

###**Task 2a:** Write code to compute δ_t inside the Agent.learn() function

[4]:

```
class Agent:
    """
    Agent class
    """
    def __init__(self, action_size, lr=0.001, gamma=0.99, seed = 85):
        self.gamma = gamma
        self.ac_model = ActorCriticModel(action_size=action_size)
        self.ac_model.compile(tf.keras.optimizers.Adam(learning_rate=lr))
        np.random.seed(seed)

    def sample_action(self, state):
        """
        Given a state, compute the policy distribution over all actions and␣
 ↪sample one action
        """
        pi,_ = self.ac_model(state)

        action_probabilities = tfp.distributions.Categorical(probs=pi)
        sample = action_probabilities.sample()

        return int(sample.numpy()[0])

    def actor_loss(self, action, pi, delta):
        """
        Compute Actor Loss
        """
        return -tf.math.log(pi[0,action]) * delta

    def critic_loss(self,delta):
        """
        Critic loss aims to minimize TD error
        """
        return delta**2

    @tf.function
    def learn(self, state, action, reward, next_state, done):
        """
        For a given transition (s,a,s',r) update the paramters by computing the
        gradient of the total loss
        """
        with tf.GradientTape(persistent=True) as tape:
            pi, V_s = self.ac_model(state)
            _, V_s_next = self.ac_model(next_state)

            V_s = tf.squeeze(V_s)
            V_s_next = tf.squeeze(V_s_next)


            #### TO DO: Write the equation for delta (TD error)
            ## Write code below
            delta = reward + self.gamma * V_s_next - V_s## Complete this
            loss_a = self.actor_loss(action, pi, delta)
            loss_c =self.critic_loss(delta)
            loss_total = loss_a + loss_c

        gradient = tape.gradient(loss_total, self.ac_model.trainable_variables)
        self.ac_model.optimizer.apply_gradients(zip(gradient, self.ac_model.
 ↪trainable_variables))
```

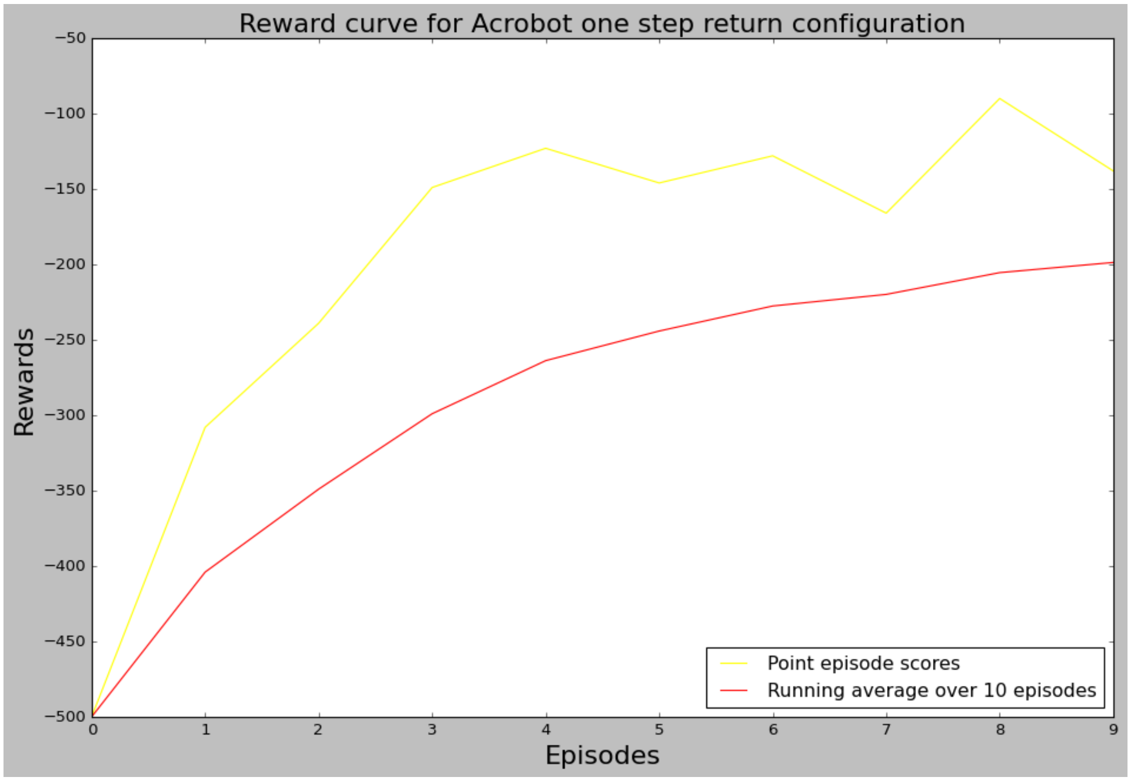
[6]:

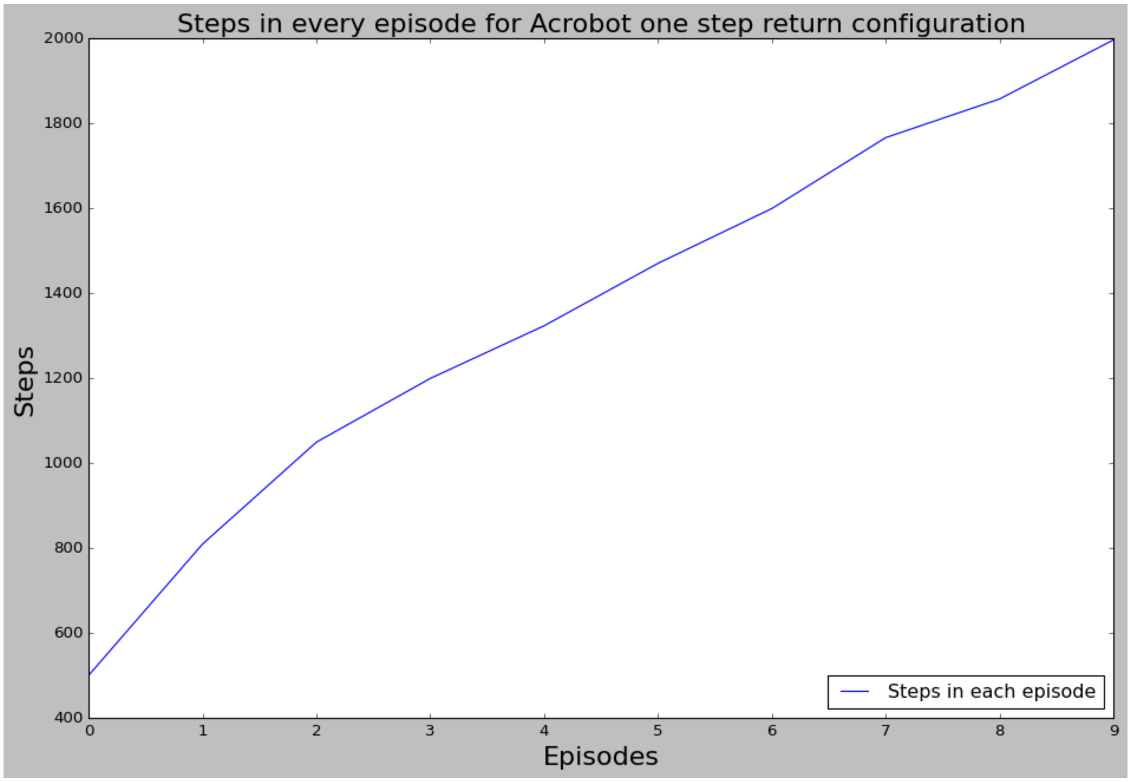
```
env = gym.make('Acrobot-v1')

#Initializing Agent
agent = Agent(lr=1e-4, action_size=env.action_space.n)
#Number of episodes
episodes = 1800
tf.compat.v1.reset_default_graph()

reward_list = []
average_reward_list = []

begin_time = datetime.datetime.now()

steps_his=[]
i=0

for ep in range(1, episodes + 1):
    state = env.reset().reshape(1,-1)
    done = False
    ep_rew = 0
    while not done:
        action = agent.sample_action(state) ##Sample Action
        next_state, reward, done, info = env.step(action) ##Take action
        next_state = next_state.reshape(1,-1)
        ep_rew += reward  ##Updating episode reward
        agent.learn(state, action, reward, next_state, done) ##Update Parameters
        state = next_state ##Updating State
        i+=1
    reward_list.append(ep_rew)
    steps_his.append(i)
    average_reward_list.append(np.mean(reward_list[-10:]))

    if ep % 10 == 0:
        avg_rew = np.mean(reward_list[-10:])
        print('Episode ', ep, 'Reward %f' % ep_rew, 'Average Reward %f' %␣
 ↪avg_rew)

    if ep % 100:
        avg_100 =  np.mean(reward_list[-100:])
        if avg_100 > -200:
            print('Stopped at Episode ',ep-100)
            break

time_taken = datetime.datetime.now() - begin_time
print(time_taken)
```

```
/usr/local/lib/python3.9/dist-packages/ipykernel/ipkernel.py:283:
DeprecationWarning: `should_run_async` will not call `transform_cell`
automatically in the future. Please pass the result to `transformed_cell`
argument and any exception that happen during thetransform in
`preprocessing_exc_tuple` in IPython 7.17 and above.
  and should_run_async(code)
/usr/local/lib/python3.9/dist-packages/gym/core.py:317: DeprecationWarning:
WARN: Initializing wrapper in old step API which returns one bool instead

of two. It is recommended to set `new_step_api=True` to use new step API. This

will be the default behaviour in future.
  deprecation(
/usr/local/lib/python3.9/dist-
packages/gym/wrappers/step_api_compatibility.py:39: DeprecationWarning:
WARN: Initializing environment in old step API which returns one bool

instead of two. It is recommended to set `new_step_api=True` to use new step

API. This will be the default behaviour in future.
  deprecation(

Episode  10 Reward -138.000000 Average Reward -198.700000
Stopped at Episode  -90
0:00:25.487957
```

[7]:

```
plt.style.use('classic')
plt.figure(figsize=(14,9))
plt.plot(np.arange(len(reward_list)),reward_list,label='Point episode␣
 ↪scores',color='yellow')
plt.plot(np.arange(len(average_reward_list)),average_reward_list,label='Running␣
 ↪average over 10 episodes',color='red')
plt.xlabel('Episodes',fontsize=20)
plt.ylabel('Rewards',fontsize=20)
plt.title('Reward curve for Acrobot one step return configuration',fontsize=20)
plt.legend(loc='lower right')
plt.figure(figsize=(14,9))
plt.plot(np.arange(len(steps_his)),steps_his,label='Steps in each episode')
plt.xlabel('Episodes',fontsize=20)
plt.ylabel('Steps',fontsize=20)
plt.title('Steps in every episode for Acrobot one step return␣
 ↪configuration',fontsize=20)
plt.legend(loc='lower right')
plt.show()
```





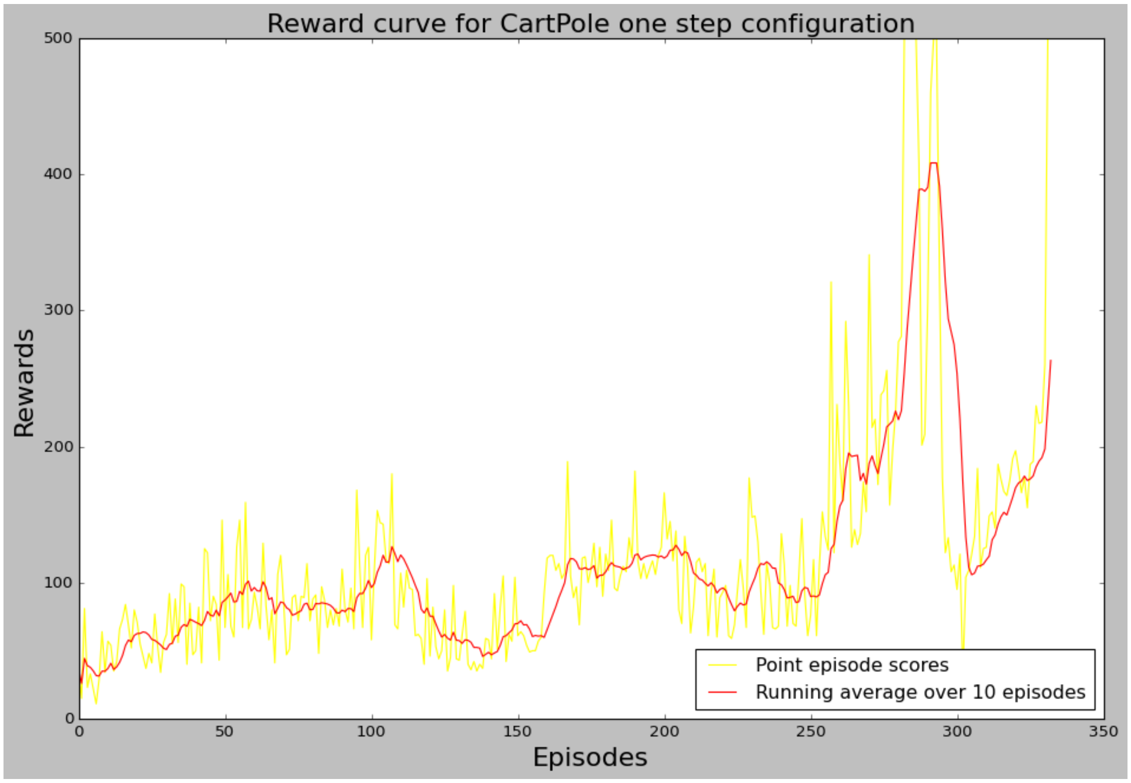
[8]:
```
env = gym.make('CartPole-v1')

#Initializing Agent
agent = Agent(lr=1e-4, action_size=env.action_space.n)
#Number of episodes
episodes = 1800
tf.compat.v1.reset_default_graph()

reward_list = []
average_reward_list = []
begin_time = datetime.datetime.now()

steps_his=[]
i=0

for ep in range(1, episodes + 1):
    state = env.reset().reshape(1,-1)
    done = False
    ep_rew = 0
    i=0
    while not done:
        action = agent.sample_action(state) ##Sample Action
        next_state, reward, done, info = env.step(action) ##Take action
        next_state = next_state.reshape(1,-1)
        ep_rew += reward  ##Updating episode reward
        agent.learn(state, action, reward, next_state, done) ##Update Parameters
        state = next_state ##Updating State
        i+=1
    reward_list.append(ep_rew)
    steps_his.append(i)
    average_reward_list.append(np.mean(reward_list[-10:]))

    if ep % 10 == 0:
        avg_rew = np.mean(reward_list[-10:])
        print('Episode ', ep, 'Reward %f' % ep_rew, 'Average Reward %f' %␣
 ↪avg_rew)

    if ep % 100:
        avg_100 =  np.mean(reward_list[-100:])
        if avg_100 > 195.0:
            print('Stopped at Episode ',ep-100)
            break

time_taken = datetime.datetime.now() - begin_time
print(time_taken)
```

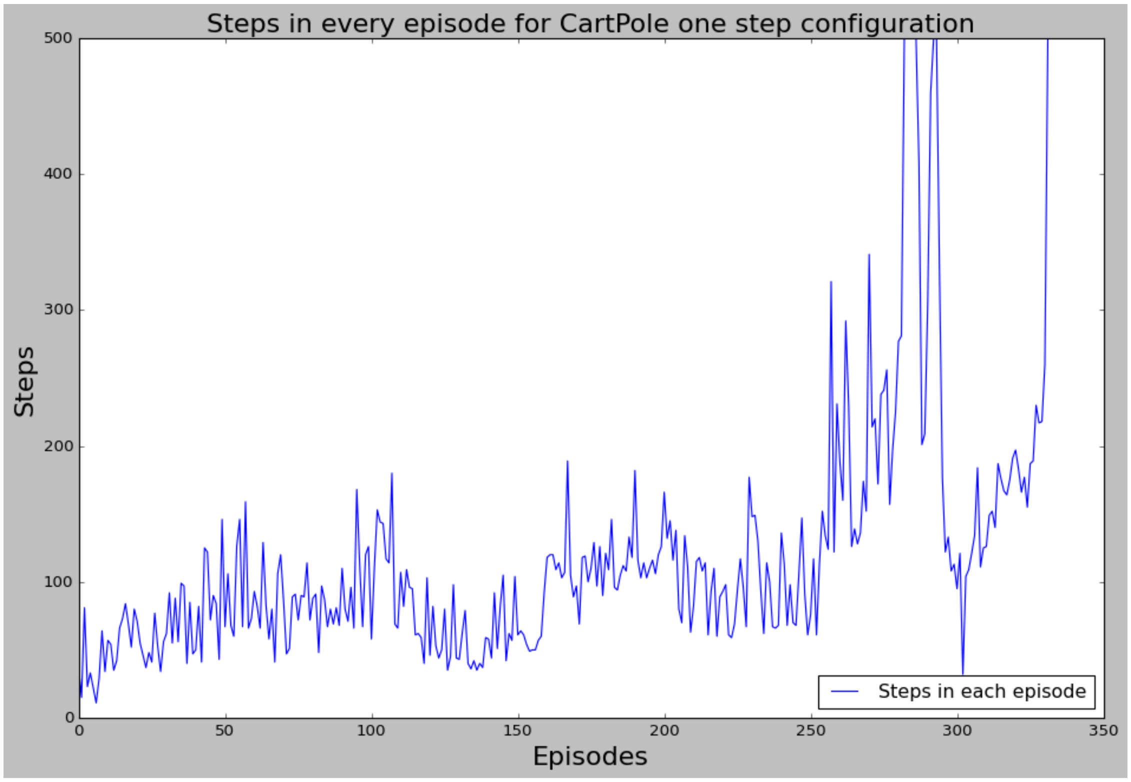
```
/usr/local/lib/python3.9/dist-packages/ipykernel/ipkernel.py:283:
DeprecationWarning: `should_run_async` will not call `transform_cell`
automatically in the future. Please pass the result to `transformed_cell`
argument and any exception that happen during thetransform in
`preprocessing_exc_tuple` in IPython 7.17 and above.
  and should_run_async(code)
/usr/local/lib/python3.9/dist-packages/gym/core.py:317: DeprecationWarning:
WARN: Initializing wrapper in old step API which returns one bool instead
of two. It is recommended to set `new_step_api=True` to use new step API. This
will be the default behaviour in future.
  deprecation(
/usr/local/lib/python3.9/dist-
packages/gym/wrappers/step_api_compatibility.py:39: DeprecationWarning:
WARN: Initializing environment in old step API which returns one bool
instead of two. It is recommended to set `new_step_api=True` to use new step
API. This will be the default behaviour in future.
  deprecation(
```

```
Episode  10 Reward 34.000000 Average Reward 34.900000
Episode  20 Reward 80.000000 Average Reward 61.200000
Episode  30 Reward 56.000000 Average Reward 51.800000
Episode  40 Reward 47.000000 Average Reward 72.100000
Episode  50 Reward 146.000000 Average Reward 85.500000
Episode  60 Reward 73.000000 Average Reward 93.800000
Episode  70 Reward 120.000000 Average Reward 85.800000
Episode  80 Reward 72.000000 Average Reward 80.100000
Episode  90 Reward 68.000000 Average Reward 77.600000
Episode  100 Reward 126.000000 Average Reward 101.600000
Episode  110 Reward 66.000000 Average Reward 115.400000
Episode  120 Reward 103.000000 Average Reward 81.400000
Episode  130 Reward 44.000000 Average Reward 57.700000
Episode  140 Reward 59.000000 Average Reward 47.300000
Episode  150 Reward 104.000000 Average Reward 69.700000
Episode  160 Reward 92.000000 Average Reward 59.800000
Episode  170 Reward 89.000000 Average Reward 117.400000
Episode  180 Reward 90.000000 Average Reward 105.500000
Episode  190 Reward 118.000000 Average Reward 114.200000
Episode  200 Reward 126.000000 Average Reward 119.600000
Episode  210 Reward 63.000000 Average Reward 115.500000
Episode  220 Reward 89.000000 Average Reward 95.000000
Episode  230 Reward 177.000000 Average Reward 93.100000
Episode  240 Reward 68.000000 Average Reward 99.700000
Episode  250 Reward 61.000000 Average Reward 95.800000
Episode  260 Reward 231.000000 Average Reward 145.100000
Episode  270 Reward 152.000000 Average Reward 172.400000
Episode  280 Reward 225.000000 Average Reward 226.100000
Episode  290 Reward 209.000000 Average Reward 387.600000
Episode  300 Reward 113.000000 Average Reward 274.900000
Episode  310 Reward 125.000000 Average Reward 113.600000
Episode  320 Reward 191.000000 Average Reward 162.700000
Episode  330 Reward 218.000000 Average Reward 191.900000
Stopped at Episode  233
0:05:53.312282
```

[9]:

```
plt.style.use('classic')
plt.figure(figsize=(14,9))
plt.plot(np.arange(len(reward_list)),reward_list,label='Point episode␣
  ↪scores',color='yellow')
plt.plot(np.arange(len(average_reward_list)),average_reward_list,label='Running␣
  ↪average over 10 episodes',color='red')
plt.xlabel('Episodes',fontsize=20)
plt.ylabel('Rewards',fontsize=20)
plt.title('Reward curve for CartPole one step configuration',fontsize=20)
plt.legend(loc='lower right')
plt.figure(figsize=(14,9))
plt.plot(np.arange(len(steps_his)),steps_his,label='Steps in each episode')
plt.xlabel('Episodes',fontsize=20)
plt.ylabel('Steps',fontsize=20)
plt.title('Steps in every episode for CartPole one step␣
 ↪configuration',fontsize=20)
plt.legend(loc='lower right')
plt.show()
```





[10]:

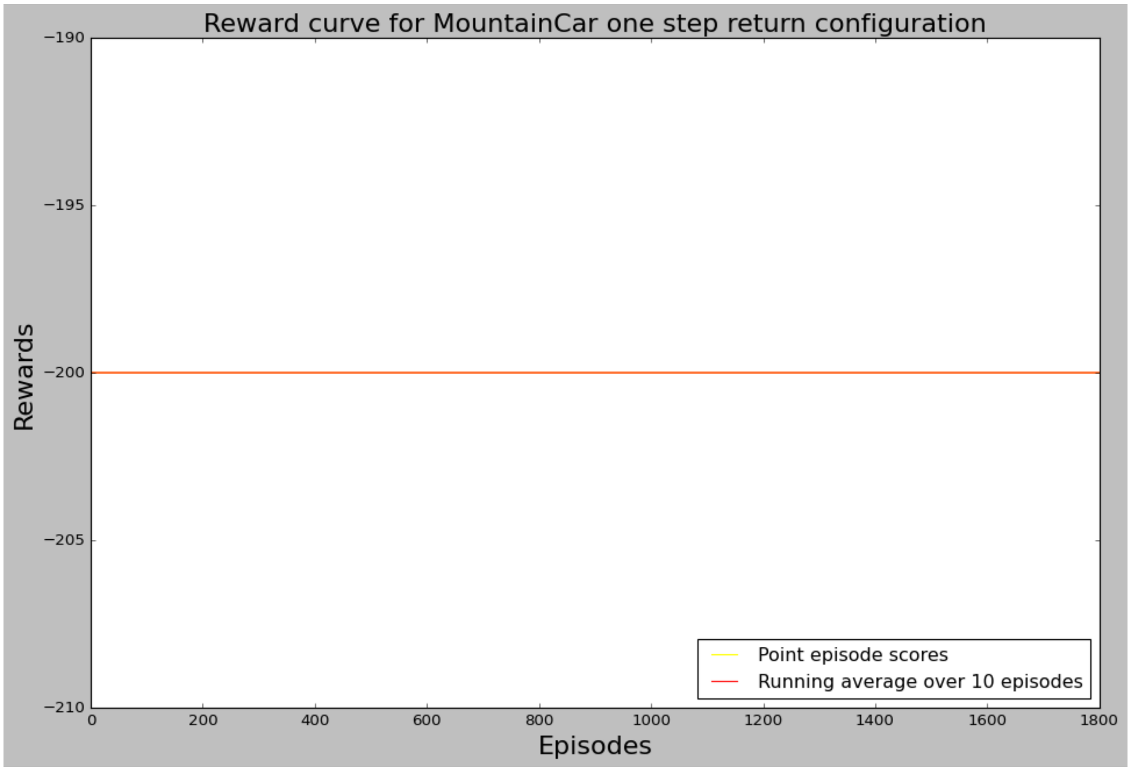
```
env = gym.make('MountainCar-v0')

#Initializing Agent
agent = Agent(lr=1e-4, action_size=env.action_space.n)
#Number of episodes
episodes = 1800
tf.compat.v1.reset_default_graph()

reward_list = []
average_reward_list = []
begin_time = datetime.datetime.now()

steps_his=[]
i=0

for ep in range(1, episodes + 1):
    state = env.reset().reshape(1,-1)
    done = False
    ep_rew = 0
    i=0
    while not done:
        action = agent.sample_action(state) ##Sample Action
        next_state, reward, done, info = env.step(action) ##Take action
        next_state = next_state.reshape(1,-1)
        ep_rew += reward  ##Updating episode reward
        agent.learn(state, action, reward, next_state, done) ##Update Parameters
        state = next_state ##Updating State
        i+=1
    reward_list.append(ep_rew)
    steps_his.append(i)
    average_reward_list.append(np.mean(reward_list[-10:]))

    if ep % 10 == 0:
        avg_rew = np.mean(reward_list[-10:])
        print('Episode ', ep, 'Reward %f' % ep_rew, 'Average Reward %f' %␣
 ↪avg_rew)

    if ep % 100:
        avg_100 =  np.mean(reward_list[-100:])
        if avg_100 > -150:
            print('Stopped at Episode ',ep-100)
            break

time_taken = datetime.datetime.now() - begin_time
print(time_taken)
```

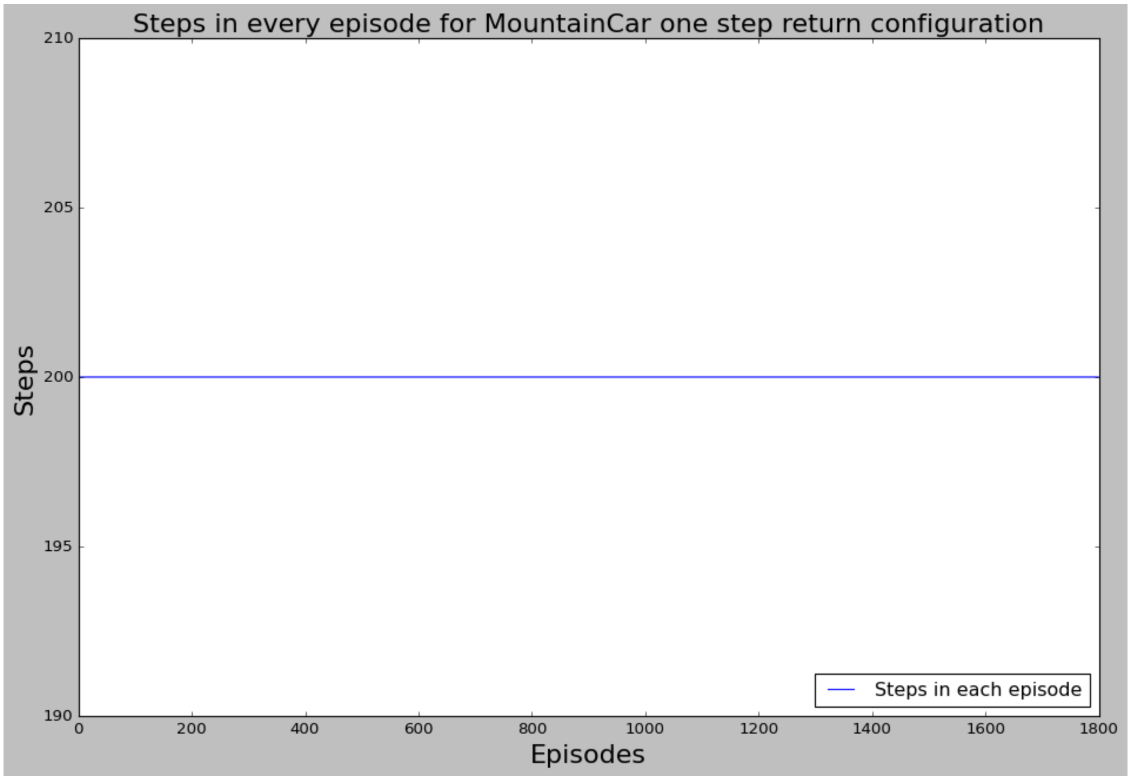
```
/usr/local/lib/python3.9/dist-packages/ipykernel/ipkernel.py:283:
DeprecationWarning: `should_run_async` will not call `transform_cell`
automatically in the future. Please pass the result to `transformed_cell`
argument and any exception that happen during thetransform in
`preprocessing_exc_tuple` in IPython 7.17 and above.
  and should_run_async(code)
/usr/local/lib/python3.9/dist-packages/gym/core.py:317: DeprecationWarning:
WARN: Initializing wrapper in old step API which returns one bool instead

of two. It is recommended to set `new_step_api=True` to use new step API. This

will be the default behaviour in future.
  deprecation(
/usr/local/lib/python3.9/dist-
packages/gym/wrappers/step_api_compatibility.py:39: DeprecationWarning:
WARN: Initializing environment in old step API which returns one bool

instead of two. It is recommended to set `new_step_api=True` to use new step

API. This will be the default behaviour in future.
  deprecation(
```

```
Episode  10 Reward -200.000000 Average Reward -200.000000
Episode  20 Reward -200.000000 Average Reward -200.000000
Episode  30 Reward -200.000000 Average Reward -200.000000
Episode  40 Reward -200.000000 Average Reward -200.000000
Episode  50 Reward -200.000000 Average Reward -200.000000
Episode  60 Reward -200.000000 Average Reward -200.000000
Episode  70 Reward -200.000000 Average Reward -200.000000
Episode  80 Reward -200.000000 Average Reward -200.000000
Episode  90 Reward -200.000000 Average Reward -200.000000
Episode  100 Reward -200.000000 Average Reward -200.000000
Episode  110 Reward -200.000000 Average Reward -200.000000
Episode  120 Reward -200.000000 Average Reward -200.000000
Episode  130 Reward -200.000000 Average Reward -200.000000
Episode  140 Reward -200.000000 Average Reward -200.000000
Episode  150 Reward -200.000000 Average Reward -200.000000
Episode  160 Reward -200.000000 Average Reward -200.000000
Episode  170 Reward -200.000000 Average Reward -200.000000
Episode  180 Reward -200.000000 Average Reward -200.000000
Episode  190 Reward -200.000000 Average Reward -200.000000
Episode  200 Reward -200.000000 Average Reward -200.000000
Episode  210 Reward -200.000000 Average Reward -200.000000
Episode  220 Reward -200.000000 Average Reward -200.000000
Episode  230 Reward -200.000000 Average Reward -200.000000
Episode  240 Reward -200.000000 Average Reward -200.000000
Episode  250 Reward -200.000000 Average Reward -200.000000
Episode  260 Reward -200.000000 Average Reward -200.000000
Episode  270 Reward -200.000000 Average Reward -200.000000
Episode  280 Reward -200.000000 Average Reward -200.000000
Episode  290 Reward -200.000000 Average Reward -200.000000
Episode  300 Reward -200.000000 Average Reward -200.000000
Episode  310 Reward -200.000000 Average Reward -200.000000
Episode  320 Reward -200.000000 Average Reward -200.000000
Episode  330 Reward -200.000000 Average Reward -200.000000
Episode  340 Reward -200.000000 Average Reward -200.000000
Episode  350 Reward -200.000000 Average Reward -200.000000
Episode  360 Reward -200.000000 Average Reward -200.000000
Episode  370 Reward -200.000000 Average Reward -200.000000
Episode  380 Reward -200.000000 Average Reward -200.000000
Episode  390 Reward -200.000000 Average Reward -200.000000
Episode  400 Reward -200.000000 Average Reward -200.000000
Episode  410 Reward -200.000000 Average Reward -200.000000
Episode  420 Reward -200.000000 Average Reward -200.000000
Episode  430 Reward -200.000000 Average Reward -200.000000
Episode  440 Reward -200.000000 Average Reward -200.000000
Episode  450 Reward -200.000000 Average Reward -200.000000
Episode  460 Reward -200.000000 Average Reward -200.000000
Episode  470 Reward -200.000000 Average Reward -200.000000
Episode  480 Reward -200.000000 Average Reward -200.000000
Episode  490 Reward -200.000000 Average Reward -200.000000
Episode  500 Reward -200.000000 Average Reward -200.000000
Episode  510 Reward -200.000000 Average Reward -200.000000
Episode  520 Reward -200.000000 Average Reward -200.000000
Episode  530 Reward -200.000000 Average Reward -200.000000
Episode  540 Reward -200.000000 Average Reward -200.000000
Episode  550 Reward -200.000000 Average Reward -200.000000
Episode  560 Reward -200.000000 Average Reward -200.000000
Episode  570 Reward -200.000000 Average Reward -200.000000
Episode  580 Reward -200.000000 Average Reward -200.000000
Episode  590 Reward -200.000000 Average Reward -200.000000
Episode  600 Reward -200.000000 Average Reward -200.000000
Episode  610 Reward -200.000000 Average Reward -200.000000
Episode  620 Reward -200.000000 Average Reward -200.000000
Episode  630 Reward -200.000000 Average Reward -200.000000
Episode  640 Reward -200.000000 Average Reward -200.000000
Episode  650 Reward -200.000000 Average Reward -200.000000
Episode  660 Reward -200.000000 Average Reward -200.000000
Episode  670 Reward -200.000000 Average Reward -200.000000
Episode  680 Reward -200.000000 Average Reward -200.000000
Episode  690 Reward -200.000000 Average Reward -200.000000
Episode  700 Reward -200.000000 Average Reward -200.000000
Episode  710 Reward -200.000000 Average Reward -200.000000
Episode  720 Reward -200.000000 Average Reward -200.000000
Episode  730 Reward -200.000000 Average Reward -200.000000
Episode  740 Reward -200.000000 Average Reward -200.000000
Episode  750 Reward -200.000000 Average Reward -200.000000
Episode  760 Reward -200.000000 Average Reward -200.000000
Episode  770 Reward -200.000000 Average Reward -200.000000
Episode  780 Reward -200.000000 Average Reward -200.000000
Episode  790 Reward -200.000000 Average Reward -200.000000
Episode  800 Reward -200.000000 Average Reward -200.000000
Episode  810 Reward -200.000000 Average Reward -200.000000
Episode  820 Reward -200.000000 Average Reward -200.000000
Episode  830 Reward -200.000000 Average Reward -200.000000
Episode  840 Reward -200.000000 Average Reward -200.000000
Episode  850 Reward -200.000000 Average Reward -200.000000
Episode  860 Reward -200.000000 Average Reward -200.000000
Episode  870 Reward -200.000000 Average Reward -200.000000
Episode  880 Reward -200.000000 Average Reward -200.000000
Episode  890 Reward -200.000000 Average Reward -200.000000
Episode  900 Reward -200.000000 Average Reward -200.000000
Episode  910 Reward -200.000000 Average Reward -200.000000
Episode  920 Reward -200.000000 Average Reward -200.000000
Episode  930 Reward -200.000000 Average Reward -200.000000
Episode  940 Reward -200.000000 Average Reward -200.000000
Episode  950 Reward -200.000000 Average Reward -200.000000
Episode  960 Reward -200.000000 Average Reward -200.000000
Episode  970 Reward -200.000000 Average Reward -200.000000
Episode  980 Reward -200.000000 Average Reward -200.000000
Episode  990 Reward -200.000000 Average Reward -200.000000
Episode  1000 Reward -200.000000 Average Reward -200.000000
Episode  1010 Reward -200.000000 Average Reward -200.000000
Episode  1020 Reward -200.000000 Average Reward -200.000000
Episode  1030 Reward -200.000000 Average Reward -200.000000
Episode  1040 Reward -200.000000 Average Reward -200.000000
Episode  1050 Reward -200.000000 Average Reward -200.000000
Episode  1060 Reward -200.000000 Average Reward -200.000000
Episode  1070 Reward -200.000000 Average Reward -200.000000
Episode  1080 Reward -200.000000 Average Reward -200.000000
Episode  1090 Reward -200.000000 Average Reward -200.000000
Episode  1100 Reward -200.000000 Average Reward -200.000000
Episode  1110 Reward -200.000000 Average Reward -200.000000
Episode  1120 Reward -200.000000 Average Reward -200.000000
Episode  1130 Reward -200.000000 Average Reward -200.000000
Episode  1140 Reward -200.000000 Average Reward -200.000000
Episode  1150 Reward -200.000000 Average Reward -200.000000
Episode  1160 Reward -200.000000 Average Reward -200.000000
Episode  1170 Reward -200.000000 Average Reward -200.000000
Episode  1180 Reward -200.000000 Average Reward -200.000000
Episode  1190 Reward -200.000000 Average Reward -200.000000
Episode  1200 Reward -200.000000 Average Reward -200.000000
Episode  1210 Reward -200.000000 Average Reward -200.000000
Episode  1220 Reward -200.000000 Average Reward -200.000000
Episode  1230 Reward -200.000000 Average Reward -200.000000
Episode  1240 Reward -200.000000 Average Reward -200.000000
Episode  1250 Reward -200.000000 Average Reward -200.000000
Episode  1260 Reward -200.000000 Average Reward -200.000000
Episode  1270 Reward -200.000000 Average Reward -200.000000
Episode  1280 Reward -200.000000 Average Reward -200.000000
Episode  1290 Reward -200.000000 Average Reward -200.000000
Episode  1300 Reward -200.000000 Average Reward -200.000000
Episode  1310 Reward -200.000000 Average Reward -200.000000
Episode  1320 Reward -200.000000 Average Reward -200.000000
Episode  1330 Reward -200.000000 Average Reward -200.000000
Episode  1340 Reward -200.000000 Average Reward -200.000000
Episode  1350 Reward -200.000000 Average Reward -200.000000
Episode  1360 Reward -200.000000 Average Reward -200.000000
Episode  1370 Reward -200.000000 Average Reward -200.000000
Episode  1380 Reward -200.000000 Average Reward -200.000000
Episode  1390 Reward -200.000000 Average Reward -200.000000
Episode  1400 Reward -200.000000 Average Reward -200.000000
Episode  1410 Reward -200.000000 Average Reward -200.000000
Episode  1420 Reward -200.000000 Average Reward -200.000000
Episode  1430 Reward -200.000000 Average Reward -200.000000
Episode  1440 Reward -200.000000 Average Reward -200.000000
Episode  1450 Reward -200.000000 Average Reward -200.000000
Episode  1460 Reward -200.000000 Average Reward -200.000000
Episode  1470 Reward -200.000000 Average Reward -200.000000
Episode  1480 Reward -200.000000 Average Reward -200.000000
Episode  1490 Reward -200.000000 Average Reward -200.000000
Episode  1500 Reward -200.000000 Average Reward -200.000000
Episode  1510 Reward -200.000000 Average Reward -200.000000
Episode  1520 Reward -200.000000 Average Reward -200.000000
Episode  1530 Reward -200.000000 Average Reward -200.000000
Episode  1540 Reward -200.000000 Average Reward -200.000000
Episode  1550 Reward -200.000000 Average Reward -200.000000
Episode  1560 Reward -200.000000 Average Reward -200.000000
Episode  1570 Reward -200.000000 Average Reward -200.000000
Episode  1580 Reward -200.000000 Average Reward -200.000000
Episode  1590 Reward -200.000000 Average Reward -200.000000
Episode  1600 Reward -200.000000 Average Reward -200.000000
Episode  1610 Reward -200.000000 Average Reward -200.000000
Episode  1620 Reward -200.000000 Average Reward -200.000000
Episode  1630 Reward -200.000000 Average Reward -200.000000
Episode  1640 Reward -200.000000 Average Reward -200.000000
Episode  1650 Reward -200.000000 Average Reward -200.000000
Episode  1660 Reward -200.000000 Average Reward -200.000000
Episode  1670 Reward -200.000000 Average Reward -200.000000
Episode  1680 Reward -200.000000 Average Reward -200.000000
Episode  1690 Reward -200.000000 Average Reward -200.000000
Episode  1700 Reward -200.000000 Average Reward -200.000000
Episode  1710 Reward -200.000000 Average Reward -200.000000
Episode  1720 Reward -200.000000 Average Reward -200.000000
Episode  1730 Reward -200.000000 Average Reward -200.000000
Episode  1740 Reward -200.000000 Average Reward -200.000000
Episode  1750 Reward -200.000000 Average Reward -200.000000
Episode  1760 Reward -200.000000 Average Reward -200.000000
Episode  1770 Reward -200.000000 Average Reward -200.000000
Episode  1780 Reward -200.000000 Average Reward -200.000000
Episode  1790 Reward -200.000000 Average Reward -200.000000
Episode  1800 Reward -200.000000 Average Reward -200.000000
0:54:06.593545
```

[11]:

```python
plt.style.use('classic')
plt.figure(figsize=(14,9))
plt.plot(np.arange(len(reward_list)),reward_list,label='Point episode scores',color='yellow')
plt.plot(np.arange(len(average_reward_list)),average_reward_list,label='Running average over 10 episodes',color='red')
plt.xlabel('Episodes',fontsize=20)
plt.ylabel('Rewards',fontsize=20)
plt.title('Reward curve for MountainCar one step return configuration',fontsize=20)
plt.legend(loc='lower right')
plt.figure(figsize=(14,9))
plt.plot(np.arange(len(steps_his)),steps_his,label='Steps in each episode')
plt.xlabel('Episodes',fontsize=20)
plt.ylabel('Steps',fontsize=20)
plt.title('Steps in every episode for MountainCar one step return␣
 ↪configuration',fontsize=20)
plt.legend(loc='lower right')
plt.show()
```





0.1.3 Code for rendering (source)

[ ]:

```
# Render an episode and save as a GIF file

display = Display(visible=0, size=(400, 300))
display.start()


def render_episode(env: gym.Env, model: tf.keras.Model, max_steps: int):
  screen = env.render(mode='rgb_array')
  im = Image.fromarray(screen)

  images = [im]

  state = tf.constant(env.reset(), dtype=tf.float32)
  for i in range(1, max_steps + 1):
    state = tf.expand_dims(state, 0)
    action_probs, _ = model(state)
    action = np.argmax(np.squeeze(action_probs))
    state, _, done, _ = env.step(action)
    state = tf.constant(state, dtype=tf.float32)
    # Render screen every 10 steps
    if i % 10 == 0:
      screen = env.render(mode='rgb_array')
      images.append(Image.fromarray(screen))

    if done:
      break

  return images


# Save GIF image
images = render_episode(env, agent.ac_model, 200)
image_file = 'cartpole-v1.gif'
# loop=0: loop forever, duration=1: play each frame for 1ms
images[0].save(
    image_file, save_all=True, append_images=images[1:], loop=0, duration=1)
```

```
/usr/local/lib/python3.8/dist-packages/gym/core.py:43: DeprecationWarning:
WARN: The argument mode in render method is deprecated; use render_mode

during environment initialization instead.

See here for more information: https://www.gymlibrary.ml/content/api/
  deprecation(
```

[ ]:

```
import tensorflow_docs.vis.embed as embed
embed.embed_file(image_file)
```

[ ]:

```
<IPython.core.display.HTML object>
```