**Міністерство Освіти І НАУКИ України**

**Національний університет "Львівська політехніка"**

Інститут **ІКНІ**

Кафедра **СШІ**

**ЗВІТ**

До лабораторної роботи №4

**З дисципліни:**

**«Машинне навчання»**

**Виконав:**

ст. гр. КН-310

Шиманський П.С.

**Прийняла:**

Якимишин Х.М.

Львів - 2020

**Мета:** засвоїти основні відомості при роботі з навчанням з підкріпленням. Навчитись використовувати його на практиці.

**Хід роботи:**

1. Ознайомився з OpenAI gym документацією <https://gym.openai.com/docs/>
2. Ознайомитись з віртуальним середовищем CartPole-v1
3. Реалізувати алгоритм Q-learning та навчити агента балансувати карт

Потрібно буде написати Q-Leaning алгоритм щоб навчити нашу підставку балансувати палицею. Для початку, створмо функцію дискретизації:

n\_bins = ( 6 , 12 )

lower\_bounds = [ env.observation\_space.low[2], -math.radians(50) ]

upper\_bounds = [ env.observation\_space.high[2], math.radians(50) ]

def discretizer( \_ , \_\_ , angle, pole\_velocity ) -> Tuple[int,...]:

est = KBinsDiscretizer(n\_bins=n\_bins, encode='ordinal', strategy='uniform')

est.fit([lower\_bounds, upper\_bounds ])

return tuple(map(int,est.transform([[angle, pole\_velocity]])[0]))

Потім ініціалізуємо таблицю Q значень і заповнимо її нулями.

Q\_table = np.zeros(n\_bins + (env.action\_space.n,))

Q\_table.shape

1. Зобразити у вигляді відео результат - балансуючу cart pole понад 100 фреймів

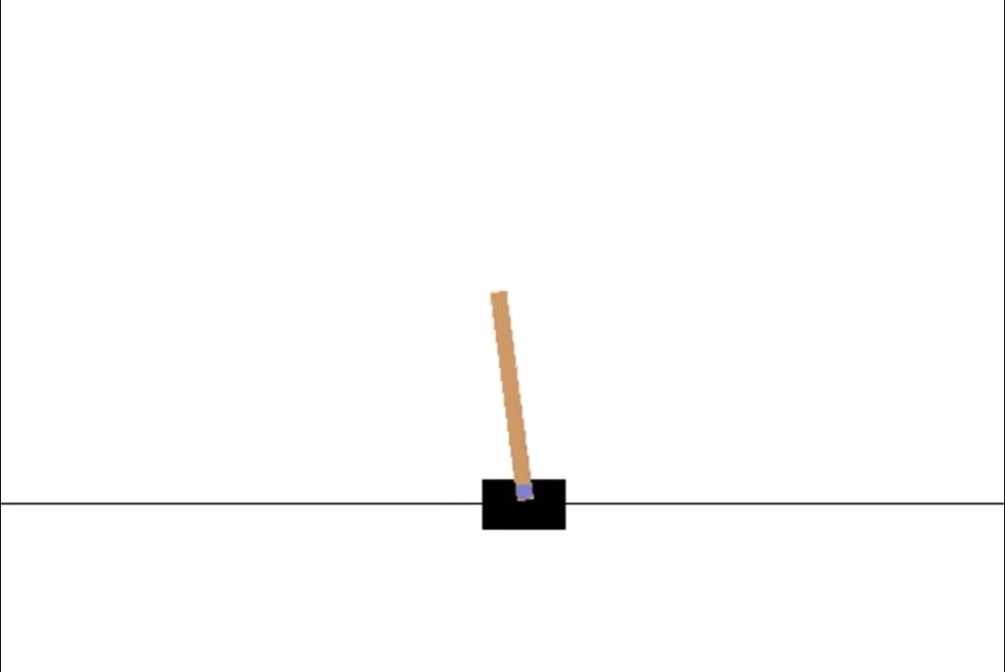


Рис.1. Результат роботи

На початку досліду об’єкт не може доволі довго втримувати рівновагу. Але після певного проміжку часу, він навчається і починає набагато краще балансувати. Тоді приблизно після 200 ітерацій об’єкт почне довго перебувати у стані спокою.

**Висновок:**

На даній лабораторній роботі було проведено ознайомлення з Q-Learning алгоритмом навчання. Даний алгоритм застосовують у різних середовищах з метою навчити агента діяти у певних обставинах. Було написано Q-learning алгоритм який навчив підставку рухатись вперед та назад щоб балансувати палицею і не дати їй впасти.

Посилання на відео: <https://www.youtube.com/watch?v=med248x6hzE&feature=youtu.be>

Код програми:

import numpy as np

import time, math, random

import gym

from sklearn.preprocessing import KBinsDiscretizer

from typing import Tuple

env = gym.make('CartPole-v1')

env.reset()

observation = env.reset()

?env.env

policy = lambda \_,\_\_,\_\_\_, tip\_velocity : int( tip\_velocity > 0 )

n\_bins = ( 6 , 12 )

lower\_bounds = [ env.observation\_space.low[2], -math.radians(50) ]

upper\_bounds = [ env.observation\_space.high[2], math.radians(50) ]

def discretizer( \_ , \_\_ , angle, pole\_velocity ) -> Tuple[int,...]:

est = KBinsDiscretizer(n\_bins=n\_bins, encode='ordinal', strategy='uniform')

est.fit([lower\_bounds, upper\_bounds ])

return tuple(map(int,est.transform([[angle, pole\_velocity]])[0]))

Q\_table = np.zeros(n\_bins + (env.action\_space.n,))

Q\_table.shape

def policy( state : tuple ):

"""Choosing action based on epsilon-greedy policy"""

return np.argmax(Q\_table[state])

def update\_q\_value( reward : float , new\_state : tuple , discount\_factor = 1 ) -> float:

"""Temperal diffrence for updating Q-value of state-action pair"""

future\_optimal\_value = np.max(Q\_table[new\_state])

learned\_value = reward + discount\_factor \* future\_optimal\_value

return learned\_value

def learning\_rate(n : int , min\_rate = 0.01 ) -> float :

"""Decaying learning rate"""

return max(min\_rate, min(1.0, 1.0 - math.log10((n + 1) / 25)))

def exploration\_rate(n : int, min\_rate = 0.1 ) -> float :

"""Decaying exploration rate"""

return max(min\_rate, min(1, 1.0 - math.log10((n + 1) / 25)))

n\_episodes = 10000

for e in range(n\_episodes):

current\_state, done = discretizer(\*env.reset()), False

while done == False:

action = policy(current\_state) # exploit

if np.random.random() < exploration\_rate(e) :

action = env.action\_space.sample()

obs, reward, done, \_ = env.step(action)

new\_state = discretizer(\*obs)

lr = learning\_rate(e)

learnt\_value = update\_q\_value(reward , new\_state )

old\_value = Q\_table[current\_state][action]

Q\_table[current\_state][action] = (1-lr) \* old\_value + lr \* learnt\_value

current\_state = new\_state

env.render()