COMP9417
CART POLE BALANCING
ASSIGNMENT 2 REPORT

Prepared By:

XUEFENG LI (Z5085453)
HAO WU (Z3444724)

Approved By:

Date: *04/06/2017*

TABLE OF CONTENTS

1. INTRODUCTION

1.1 PURPOSE

The Cart Pole is an classic topic in the realm of Machine Learning. It is a Cart moving fritionlessly along a 1-dimension line, having a pole attached by an un-actuated joint to it.

Player can only apply command to force the Cart moving to left or right. The purpose is to keep the pole in good balance so it will not fall, while keeping the Cart only move in a certain short range so it will not move to far away.

1.2 DEFINITIONS, ACRONYMS AND ABBREVIATIONS

The following abbreviations are used in this document:

Abbreviation	Description
DDQN	Double Deep Q- Network

1.3 REFERENCED DOCUMENTS

The following documents shall be referenced throughout this document and relate to this project:

Title	Issue Date	Author	Publisher	Issue Number
Deep Reinforcement Learning with Double Q-learning	N/A	Hado van Hasselt Arthur Guez David Silver	N/A	N/A
Playing Atari with Deep Reinforcement Learning	Dec 2013	Volodymyr Mnih. Koray Kavukcuoglu. David Silver. Alex Graves Ioannis Antonoglou. Daan Wierstra	University of Toronto	N/A
Machine Learning Series Youtube Videos	N/A	Morvan Zhou	Youtube	N/A
Learning Tensorflow (Early Release)	June 2017	Tom Hope Yehezkel S. Resheff Itay Lieder	O'Reilly	1
Hands-On Machine Learning with Scikit-Learn and TensorFlow	March 2017	Aurélien Géron	O'Reilly	1

2. SOFTWARE ENVIRONMENT

The software is created using Python 3.5. The required libraries are:

- Numpy 1.12.1
- Pandas 0.20.1
- Tensorflow 1.1.0
- Gym 0.9.1

The programming environment (IDE) is JetBrain PyCharm 2017.1.2.

2.1 GYM

We use gym as the environment so we can focus on the Machine Learning itself instead of spending much time to set up the physical model. Compared to the real world Cart, the model in gym only accept 0 or 1 as action indicating push the Cart to left or right – it is a descrete input rather than analogue force in real world.

Meanwhile, the gym library returns feedback for each action. They are:

- Observation: an 4-value array indicating the Cart's position, speed, angle, and angular speed
- Reward: It is always 1 – as long as the game is not over yet.
- Done: It is always False, until game over
- State: Not used in Cart-Pole game

In our learning model, we will use the Observation as the input (which is an array of 4 elements), and the Action as the output (which is an array of 2 elements). The target is to use the Neural Network to calculate the best Action given the circumstance of Observation at any moment.

2.2 TENSORFLOW

We use TensorFlow 1.1.0 to build and train the Neural Network. It provides a number of built-in Optimizer to be used in different application.

Tensorflow has very special running method. The programmer needs to build up the model (in this case DQN) before actually running it. We define Variables for the parameters to be learnt, and define Placeholder for the input / output of the Network. We also define the loss function which needs to be optimised. Tensorflow also provides a dozen of Optimisation mothod.

We use RMSPropOptimizer in this solution. RMSProOptimizer implements the RMSProp algorithm which was created by Geoff Hinton (http://sebastianruder.com/optimizing-gradient-descent/index.html#rmsprop).

3. IMPLEMENTATION

3.1 ALGORITHMS – 1 (Q-TABLE W/O NEURAL NETWORK)

Firstly we tried to solve the game using a Q-Table instead of neural network. The Q-Table is a table with 5 attributes (x, x', theta, theta',action), and the value of each slot is the culmutative reward for the specified action. Since the attributes of the environment state are continuous, we assigned each state to the corresponding discrete butket according to their value.

After each step, update the value in Q-Table according to the following function.

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha\big(r_{t+1} + \gamma \cdot \max Q(s_{t+1}, a) - Q(s_t, a_t)\big)$$

where α is the learning rate and γ is the discount factor.

3.2 ALGORITHMS – 2 (WITH NEURAL NETWORK)

3.2.1 Neural Networks

We use two same 4-layer Neural Netwoks in our solution. The input layer has four nodes, mapped to the 4 features of the Observation (Position, Speed, Angle, and Angular Speed). The output layer has two nodes, shich stores the Q value of the two possible Actions.

There are 2 hidden layers, each of which has 10 nodes. So it is a 4-10-10-2 Network.

3.2.2 Algorithms Procedure

We are using DQN (Deep Q-Network) as the model to solve the Cart-Pole problem. The basic steps are:

1. Build two same Neural Networks, which are initialised with same weights and bias. The first is called Evaluation Network, the second is called Target Network.
2. Read the Observation as the input of the Evaulation Network to get the output called Evaluation Q, which represents the predicted Rewards in regards to different Actions (0 or 1).
3. Pick the Action with higher Evaluation Q, then apply it on the Cart Pole environment to get the next Observation.
4. Use the next Observation as the input of the target Network to get the output called Next Q, which represents the predicted Rewards given the output of the Evaluation Network
5. Use Bellman Equation to calculate the Target Q from the Next Q:

   Target Q = Reward of this Step + gemma * Next Q

   In this case, we need to manually calculate the reward of this step, as the Reward returned from gym environment is not very helpful (It is always 1 until game over, so can't indicate which action is more rewarding).
6. Calculate the square difference between Evaluation Q and Target Q, and use it as the cost function.
7. Optimise the weights and bias of Evaluation Q Neural Network while keep the Target Network unchanged. The Target Network's weights and bias will only be changed periodically so the Target Q will not follow the change of Evaluation Q.

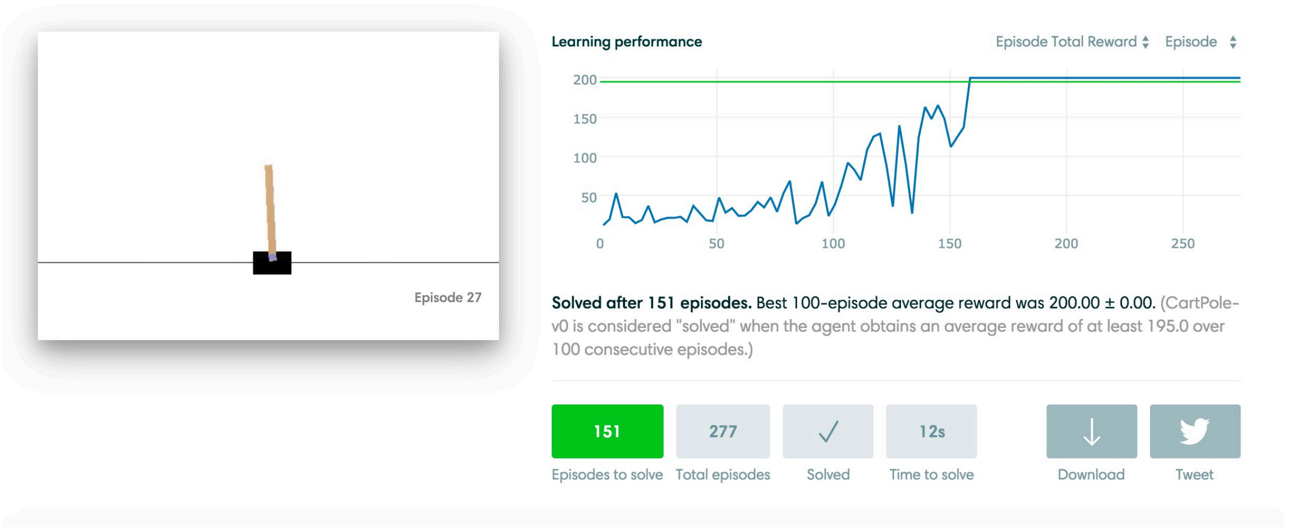
In addition, the Experience Replay is also used – the lastest 500 transactions are 'remembered' (Transaction means applying Action on Observation to get the Next Observation plus the Reward). When training the Neural Network a random sets of Transations in memory will be called out to train the Evaluation Network.

The program will play the Cart Pole game for 10000 episodes. Each episode will finish when the game is over (due to the Pole falling over or the Cart moving out of the window), or when the Pole has been kept balance for more than 2000 transaction.

4. EXPERIMENTATION

We use the intregted testing environment of gym to verify the algorithm's implementation.
We have experimented three versions of our solution:

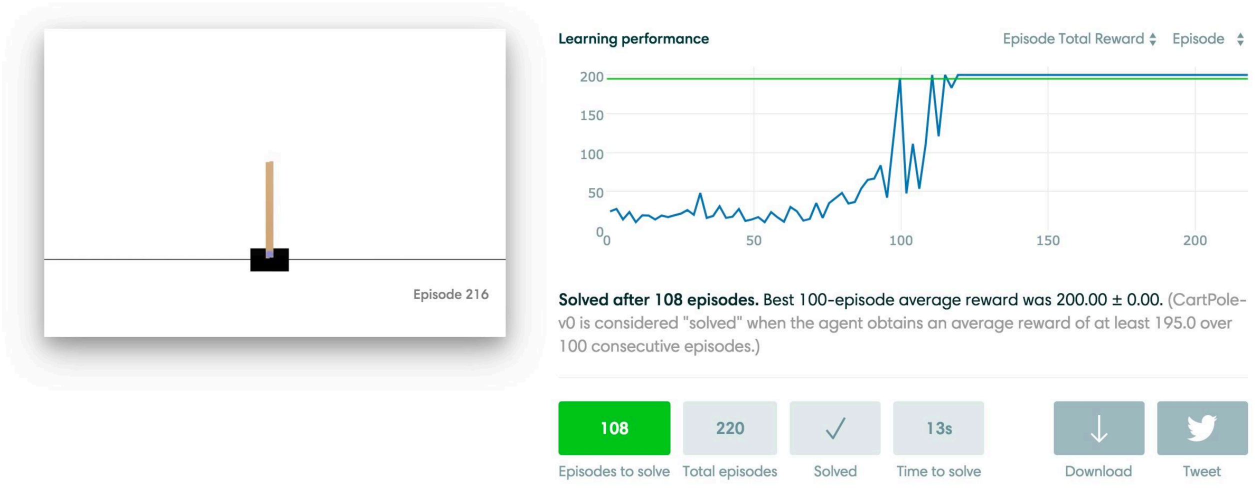
4.1 Q-TABLE WITHOUT NEURAL NETWORKL



Solved after 151 episodes. Best 100-episode average reward was 200.00 ± 0.00. (CartPole-v0 is considered "solved" when the agent obtains an average reward of at least 195.0 over 100 consecutive episodes.)

4.2 DDQN

The Neural Network is able to find the optimised parameters in 100+ episodes.



Solved after 108 episodes. Best 100-episode average reward was 200.00 ± 0.00. (CartPole-v0 is considered "solved" when the agent obtains an average reward of at least 195.0 over 100 consecutive episodes.)

Figure 1 – Experimenting Result when the Exploration Rate and Learning Rate attenuate over time

5. CONCLUSION

The DDQN solution resolves the Cart-Pole problem quite well. In most cases it only takes 100 ~ 200 steps to converge given the learning rate and exploration rate are fined tuned properly.

Initially we have also used Q-Table (without using Neural Network) to solve the problem. It can slove this problem as fast as DDQN, but it has a fatal disadvantage that the Q-Table can only be created manually with discrete values, this may leading to faiing to converge in some more complicated problems with continuous states. Comaring with Q-Table, DDQN can solve more complicated problems or continuous problems due to the advantages of Neural Network.

For sure that the parameters selection / and network architecture are important for the Neural Network. If the model is correct while using wrong parameters, the DQN will also take very long time to converge.

Picking the right model / choosing the right parameters require experience (if does not have much, then a lot of experiments). From the assignment we can see the power of combining Reinforcement Learning and Neural Network together so definitely worth spending time to study.