ECE 472 Robotic and Computer Vision

Prof. K. Dana

Project 2: Deep Reinforcement Learning

1. *Write a Jupyter notebook in Google Co-lab to implement the Pytorch tutorial cart-pole*

   Attached Separately

2. Give an overview of the DQN algorithm in written forms (1-3 paragraphs). Include definitions of key terms and how they relate to the cart-pole problem (state, action, environment, reward)

DQN (Deep Q-Network) is the utilization of Deep Learning Networks to achieve Q-learning. Q-learning is the function or a table that maps a reward for each state and the corresponding action for that given state and building a policy that will lead to maximization of the rewards. Thus, it is an optimization problem of which action for a given state will lead to the highest reward. However, in real world we do not know of this function/table. Thus, we utilize neural networks as they are a fantastic way to approximate a function. We can use the neural networks with a lots of training data to approach the real function. With the function it just then matters of picking the action that will result in the highest reward, when the state of the agent is passed to this neural network function.

In our case of cart and pole the agent is the cart and pole and the state is picture of the environment as we are using CNN (convolutional neural networks). Our last layer of CNN is a fully connected layer of size 2 as the amount of action our agent can perform is two. Thus, when we input a state to our CNN the last layer will approximate the reward for that corresponding action, and we just pick the one that lead to maximum reward. However, we are using the exploration and exploitation approach, so few time we will move our agent randomly to achieve better performance. Once we take the action, we will transition to the next state, and pass that state to the network, and this keeps going until the environment is restarted or the agent dies. We keep doing this until we find the best weight for the given amount of training.

3. *Choose performance metrics for your implementation and plot these as a function of the amount of training for your agent. Compare these plots for a random agent, and an agent trained for a small amount of time and a large amount of time. (Choose training times to show significant differences in the plots).*

When we look at the plot Random Agent, Agent trained for 200 episodes, and Agent trained for 1000 episodes. We see that the random agent survives for about 25 steps on average. However, when we switch to the agent trained for 200 episodes, we see that on average it stays alive for about 150 steps on average. At last, when we look at the agent trained for 1000 episodes, we see that the average steps taken by the agent is 470 steps approximately. Not only that we can also tell by the visualization of each agent that the amount of time spent training will slowly improve the agent's performance. I can conclude this, because for the visualization of agent trained for 200 episodes, we can observe that as soon the environment starts the agent tries to balance the pole but, is still lacking at controlling the pole when it swings very wide or fast. On the other hand when, we look at the video of the agent trained for 1000 episodes, we see that it has learned a lot more and can control the pole as soon as the environment starts.

4. *Explore open-AI gym to find three other problems that can be solved with RL. Compose a list of three other environments. Describe the state, action, environment and reward for each of these three environments.*

Cliff Walking (Environment)

This is a game where the agent starts at the bottom left of the screen and must reach the goal at the bottom right of the screen. However, there is a cliff between the straight paths. The agent must learn to traverse the terrain of a 4x12 grid, without steeping on the cliff

Action

The agent has the following four actions

0. Move Up
1. Move Right
2. Move Down
3. Move Left

State

The state of the agent is just the position of the agent in the grid, which is returned as a flattened index of the grid

Reward

The agent is rewarded -100 for stepping into the cliff, and since we want to reach our goal in the least amount of time, we give -1 for every step taken by the agent.

Taxi (Environment)

This is a game where the agent (a taxi) starts at one of the designated spots and tries to pick up the passenger who is at another designated spot, and drops the passenger off at the desired place

Action

The agent has the following six actions

0. Move South
1. Move North
2. Move East
3. Move West
4. Pickup Passenger
5. Drop Off Passenger

State

There in total 500 states of this environment, and the state is indexed as on these states in a numerical fashion. However, we can use the given function decode to decode the numerical state into a list of [taxi_row, taxi_col, passenger_location, destination]

The passenger location and destination are further numerically encoded by the following

0. R(ed)
1. G(reen)
2. Y(ellow)
3. B(lue)
4. In taxi (only for passenger)

Reward

The agent is rewarded +20 for dropping the passenger at the correct destination, -10 for using drop off and pick-up at wrong places, and -1 for every step taken as we want to minimize the amount of time taken to drop-off the passenger.

Lunar Lander Discrete Version (Environment)

This is a game where the agent is a lander which starts at the middle of the screen with a random force applied to it, and the goal is to land on the launch pad which is always located at (0,0)

Action

The agent has four actions he can take the following actions

0. Do Nothing
1. Fire Left Engine
2. Fire Main Engine
3. Fire Right Engine

State

The state of the agent given by an 8-dimensional vector which hold the coordinates (x and y) of the launcher, the velocity components of the launcher along the x and y axis, its angle, its angular velocity, and booleans for its two leg to indicate whether they are touching the ground or not.

Reward

The agent is rewarded for -100 for crashing, but on other it is rewarded +100 for coming to a gentle rest. Additionally, +10 points are awarded if the legs of the launcher are touching the ground. Firing the main engine will result in a penalty of -0.3 and -0.03 for firing the directional engines. However, if the launcher manages to move toward the launcher pad from the top of the screen and come to a gentle reset it will be rewarded anywhere from 100 to 140 points. At last, if the agent solves the environment, it will be rewarded additional +200 points.