Q-learning on cartpole

LINKÖPINGS UNIVERSITET

Farnaz Adib Yaghmaie

Linkoping University, *Sweden*
farnaz.adib.yaghmaie@liu.se

April 6, 2021

A harbor

Photo credit: @http://rhm.rainbowco.com.cn/

The cartpole

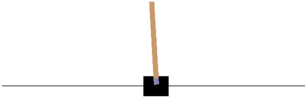

Photo credit: @https://gym.openai.com/

- **States:** 1. position of the cart on the track, 2. angle of the pole with the vertical, 3. cart velocity, and 4. rate of change of the angle.
- **Actions:** +1, -1
- **Reward:**

$$r_t = \begin{cases} 1, & \text{if the pendulum is upright} \\ 0, & \text{otherwise} \end{cases}$$

Episode ends when:

- The pole is more than 15 degrees from vertical or
- The cart moves more than 2.4 units from the center or
- The episode lasts for 200 steps.

Solvability Criterion: Getting average sum reward of 195.0 over 100 consecutive trials.

We build a (deep) network to take the state and generate Q for all actions
$Q(s, a) = network(state)$

```
network = keras.Sequential([
keras.layers.Dense(30, input_dim=n_s, activation='relu'),
keras.layers.Dense(30, activation='relu'),
keras.layers.Dense(30, activation='relu'),
keras.layers.Dense(n_a)])
```

and assign a mean squared error cost function for it

```
self.network.compile(loss='mean_squared_error',
        optimizer=keras.optimizers.Adam())
```

The policy π is the index which the output of the network is maximized.

```
policy = np.argmax(network(state))
```

1. Collect data
   - Observe s and select a

$$a = \begin{cases} \text{random action} & \text{if } r < \epsilon, \\ \arg\max_a Q(s,a) & \text{Otherwise.} \end{cases}$$

   - Apply a and observe r and the next state s'..
   - Add $s,\ a,\ r, s'$ to the history.
2. Update the parameter θ.
   - Define $Q_{target}(r_t, s_{t+1}) = r_t + \gamma \ \arg_a \max Q(s_{t+1}, a)$

```
for i in range(eps_length):
        if dones[i]:
                q_target[i, actions[i]] = rewards[i]
        else:
                q_target[i, actions[i]] = rewards[i]
                + Gamma * tf.math.reduce_max(network(next_states[i])).numpy()
```

   - Minimize the mean squared error

```
loss = self.network.train_on_batch(states, q_target)
```

Try the following:

- Run

  Crash_course_on_RL/q_on_cartpole_notebook.ipynb

  and verify to get the solution after ∼ 2885 episodes.

- Set

  'epsilon': 0.0 in agent_par

  and verify that the agent cannot solve the problem!

- Make sure you understand the code!

How the reward looks like during learning

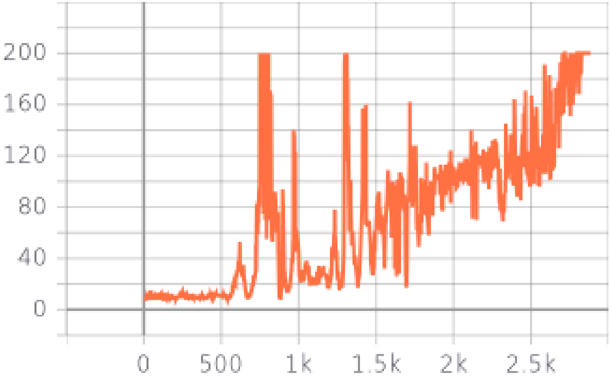

Figure: Total reward vs. no. of episodes

Replay *Q* learning

2885 episodes?? quite bad!

Replay *Q* can improve it!

- Build a memory and save data sequentially. When the memory is full, disregard the oldest data and add the new data
- Sample the memory instead of using the latest episode

Try the following:

- Run replay_q_on_cartpole_notebook.ipynb and verify to get the solution after ~ 475 episodes.
- Make sure you understand the code!

How the reward looks like during learning

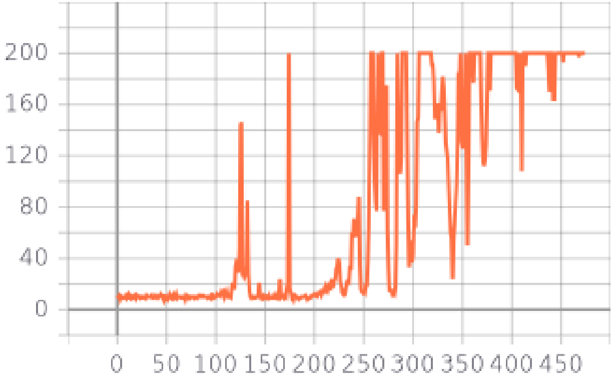

Figure: Total reward vs. no. of episodes

Email your questions to

farnaz.adib.yaghmaie@liu.se