ECE ILLINOIS
Department of Electrical and Computer Engineering

Assignment 4: Reinforcement Learning

Corresponding TA: Safa Messaoud

Overview

Release Date: 11/20/17

Due Date: 12/12/17

In this assignment you will:

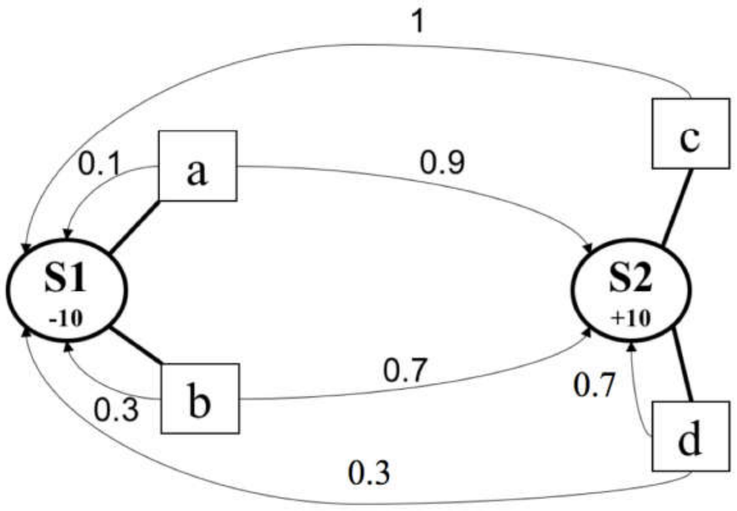
- Learn the basics of MDP, value iteration, policy iteration and Q-Learning.
- Learn to train a reinforcement learning agent using Q-Learning on the Pong game.
- Learn to train a reinforcement learning agent using the policy gradient algorithm on the Pong game.

Written Section

Note: Answers without sufficient justification will not recieve any points.

Exercise:

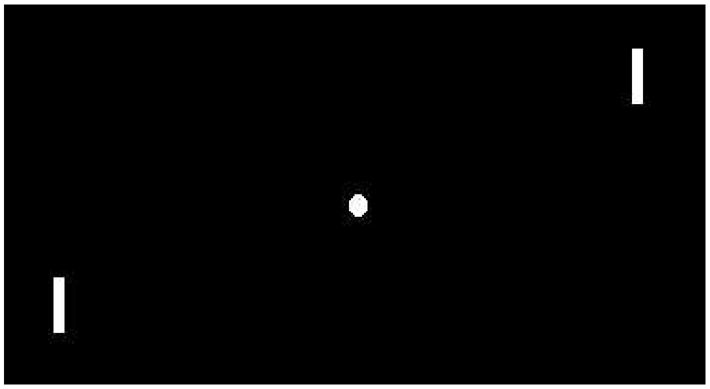
Consider the MDP with two states $S1$ and $S2$ represented in the figure below. In $S1$, the agent can perform either action a or action b. In $S2$, the agent chooses between actions c and action d. $S1$ and $S2$ are associated with rewards $R1 = -10$ and $R2 = 10$, respectively.



1. Apply 3 steps of the policy iteration algorithm with $\gamma = 0.9$. Assume that the initial policy has actions a in $S1$ and d in $S2$. Report the new policy after every iteration and its associated value function.
2. Apply 3 steps of the value iteration algorithm. Assume that the value function associated with every state is initialized to zero. Report a table with the value function computed at every iteration. What is the best policy obtained after these 3 iterations?
3. Assume that the transition probabilities are unknown, and that the agent's first 3 transitions are given by the tuples $(S1, a, S2)$, $(S2, d, S2)$ and $(S2, c, S1)$. Apply the Q-learning algorithm for three iterations with $\gamma = 0.9$ and $\alpha = 0.1$. Report a table with the Q-function computed at every iteration. What is the best policy obtained after these 3 iterations?

Programming Section

In this part, you will train an AI agent to play the Pong game with two different methods, namely Q-learning and policy gradient. On the low level, the game works as follows: we receive the last 4 image frames which constitute the state of the game and we get to decide if we want to move the paddle to the left, to the right or not to move it (3 actions). After every single choice, the game simulator executes the action and gives us a reward: either a +1 reward if the ball went past the opponent, a -1 reward if we missed the ball and 0 otherwise. Our goal is to move the paddle so that we get lots of reward.



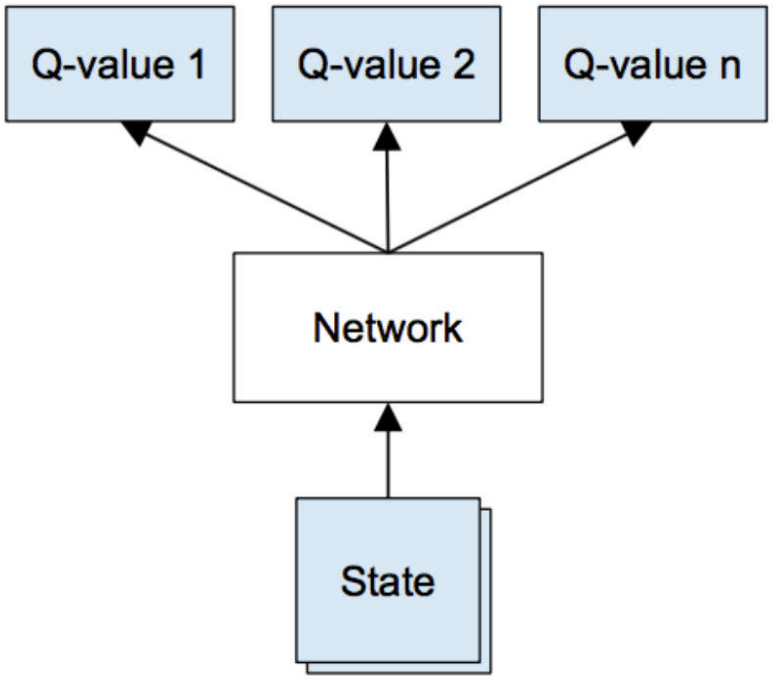
Part 1: Setup

This homework requires: python3, Tensorflow and pygame. Reuse the virtual environment from MP1.

Part 2: Q learning

This exercise requires you to use Q-Learning to train a convolutional neural networks for playing the Pong game. We consider the deep Q-netweork in the figure below. Follow the instructions in the starter code to fill in the missing functions. Include a learning curve plot showing the performance of your implementation. The x-axis should correspond to number of time steps and the y-axis should show the reward. Your agent should be performing well after 4-5m steps.

Relevant File: `q_learning.py`



Part 3: Policy Gradient

This exercise requires you to use Policy Gradient to train a convolutional neural networks for playing the Pong game. In this case, the network produces a probability distribution $p_\theta(x)$ over the set of possible actions x. After the end of every episode, we label the actions that were taken in this episode with the obtained final reward (+1 if we win and -1 if we lose). Then, we compute the discounted reward $f(x)$ for every action. The resulting score is used to compute the policy gradients with respect to the parameters of the convolutional network, i.e. $\nabla_\theta E_x[f(x)]$. Follow the instructions in the starter code to fill in the missing functions. Include a learning curve plot showing the performance of your implementation. The x-axis should correspond to number of time steps and the y-axis should show the reward. Your agent should be performing well after 4-5m steps.

Relevant File: `policy_gradient.py`

Part 4: Submission

Submitting your code is simply commiting you code. This can be done with the follow command:

```
svn commit -m "Some meaningful comment here."
```