Reinforcement project

1. Easy 21 game(35 points)

1.1 Problem introduction

The goal of this assignment is to apply reinforcement learning to solve a simple game, called Easy 21. This exercise is similar to the traditional Blackjack example. However, please note that the rules of the card game are different and non-standard.

1.2 Environment Introduction

The rule of game of Easy 21 is defined as follows:

- Each draw from the deck results in a value between 1 and 10 (uniformly distributed) with a color of red (probability 1/3) or black (probability 2/3).
- At the start of the game both the player and the dealer draw one black card.
- Each turn the player may either stick or hit. If the player hits then he/she draws another card from the deck. If the player sticks he/she receives no further cards.
- The values of the player's cards are added (black cards) or subtracted (red cards). If the player's sum exceeds 21, or becomes less than 1, then he/she goes “bust" and loses the game (reward -1).
- If the player sticks then the dealer starts taking turns. The dealer always sticks on any sum of 16 or greater, and hits otherwise. If the dealer goes “bust”, then the player wins (reward+1); Otherwise the outcome and reward is computed as follows: the player wins (reward+1) if player’s sum is larger than

the dealer's sum; the player loses (reward -1) if the player's sum is smaller than the dealer's sum; the reward is 0 if the two sums are the same.

- Assumption: The game is played with an infinite deck of cards (i.e. cards are sampled with replacement)

1.3 Problems

- **Implementation of Easy21 environment**

You should write an environment that implement the Easy21 game. Specifically, the environment should include a function step, which takes a state (dealer's first card, player's current sum) and an action (stick or hit) as inputs, and returns a next state (which may be terminal) and a reward.

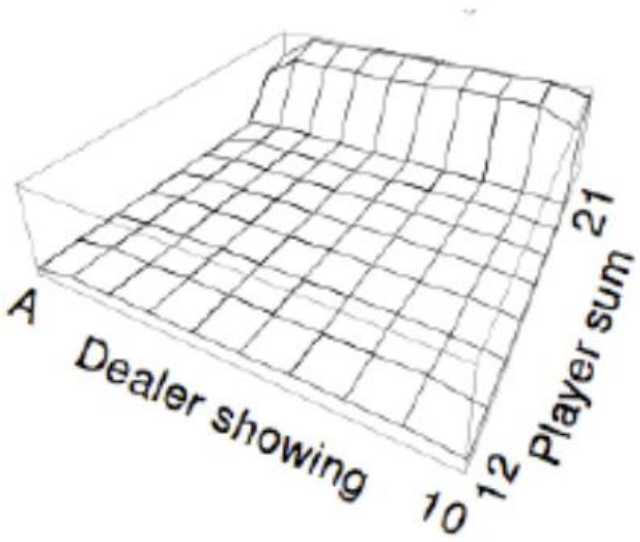
$$\text{next state, reward} = \text{step (state, action)}$$

You should treat the dealer's moves as part of the environment. In other words, i.e. calling step with a stick action will play out the dealer's cards and return the final reward and terminal state. There is no discounting ($\gamma = 1$). We will be using this environment for reinforcement learning.

- **Q-learning in Easy21**

Apply Q-learning to solve Easy21. Try different learning rates α, exploration parameter ε and episode numbers. Plot the learning curve of the return against episode number. Can you find the optimal policy in this game? Plot the optimal the state-value function using similar axes to the following figure taken from Sutton and Barto's Blackjack example.

$$V^*(s) = \max_a Q^*(s, a)$$



- **Policy iteration in Easy21**

Apply policy iteration method to solve Easy21. Try different parameters of policy iteration method. Plot the learning curve of the return against episode number. Compare the performance and characteristic between Q-learning method and policy iteration method.

- **Some additional requirements in the reports**

Your reports should also contain the introduction of q-learning methods and policy iteration methods that you used in this problem.

2. Reinforcement learning in Quanser Robot platform(65 points)

2.1 Problem Introduction

Investigates the applications of two deep reinforcement learning algorithms including TRPO which is model-free method and model predictive control

(MPC) which is model-based method on quanser robot

Platforms. Our inference codes: https://github.com/michaelliyunhao/RL-project

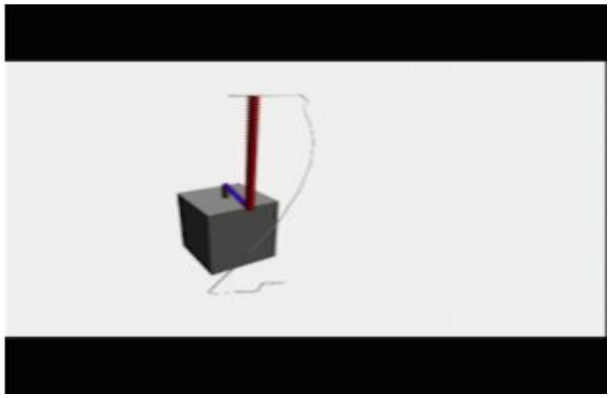
2.2 Environment Introduction

For the installation of the Quanser robot simulation environment, please see https://git.ias.informatik.tu-darmstadt.de/quanser/clients.

2.3 Problems

- **Implementation of different Quanser robot environment**

In this project, you are required to use 4 quanser robot environments as the evaluation platforms: Qube, Ball Balancer, CartPoleSwing. Install the 4 quanser robot environments and render the environments. Give a detail introduction of each environment, including the structure of the model, the action space, observation space parameters, episode reward.



- **Implementation of TRPO method on Quanser robot environment**

Try to implement the TRPO methods on Qube, Ball Balancer, CartPoleSwing platforms and solve the problem of each environment. (PS: the method may not work on some environments, don't worry). Evaluate the TRPO algorithm with many trials of hyperparameter mainly including learning rate, batch size etc. Plot the curve of training loss and episode reward for different hyperparameters and

try to infer some conclusions of the TRPO methods.

- **Implementation of MPC method on Quanser robot environment**

Try to implement the MPC methods on Qube, Ball Balancer, CartPoleSwing platforms and solve the problem of each environment. (PS: the method may not work on some environments, don't worry). Modify the MPC algorithm with your own optimization algorithm, To optimize the MPC controller, our reference code uses the Artificial Bee Colony (ABC) optimization algorithm, in this part, you are required to try different methods such as the original random shooting method in the paper or many dynamics programming methods. Evaluate the MPC algorithm with many trials of hyperparameter mainly including learning rate, batch size , optimization methods etc. Plot the curve for different hyperparameters and try to infer some conclusions of the MPC methods.

- **Some additional requirements in the reports**

Your reports should also contain the introduction of TRPO methods and MPC methods that you used in this problem, and give a comparison of the TRPO and MPC methods.