Reinforcement Learning to solve Inverted Pendulum Problem

Md Raisul Kibria

INTRODUCTION

A classic control problem called the inverted pendulum includes a pole connected with a hinge/joint over a cart which can freely move on a singular axis. Due to gravitational pull, the pole will always end up falling in either side. The goal of the problem is to move around the cart to balance the pole preventing it from falling over a target step in time. The state space includes two parameters – the angle of the pole (, and the angular velocity The angle of the pole can be at a continuous range of radians, and the angular velocity can range between rad/sec. The overall problem can be simplified to a few parameters, and two equations that represent the system and evolution of the state space. These parameters are the masses of the cart (M, kg) and the pole (m, kg), the length of the pole (d, m) and the time step ( The internal equations for the system are:

(1)

(2)

Where, , and is action applied at time , which is applying a force in Newtons on the cart towards either side or remaining stationary which is then transferred to the pole attached to it. For this specific implementation, the given parameters which are used across the experiments are:

The goal is to prevent the pendulum to fall during a 10 sec. simulation (100 steps).

The problem can be addressed through an intelligent agent that uses reinforcement learning algorithms to learn a good policy such that the agent can complete its mission (achieve the target of preventing falling). As part of the task requirement, the problem has been adapted for the Monte-Carlo for control and the temporal differencing equivalent – the SARSA algorithm. Both variants of the SARSA algorithm – SARSA (0) and SARSA () has been implemented and different parameters has been explored to analyze the effects on the agent. Finally, a comparative analysis has been presented among the experimented algorithms with a discussion. For easier structuring, the sections follow the given tasks in order.

EXPERIMENTATION

Task 1:

inverted\_pendulum.py:

Models the environment of the problem following the standards of other libraries like the OpenAI Gym. The class InvertedPendulum initiates the internal parameter and provides the following methods:

* Reset: The method clears out all data of state of the environment recorded in any previous runs. This includes clearing out the list of the angular states of the pendulum and setting the angular velocity to 0. If *exploring\_starts* is set, the agent is enabled to explore from randomized initial angular positions. The initial position is sampled from , so mostly from a vertical position.
* Step: The step method models the internal functionalities of the pendulum by applying the input action through equation (1) and (2) to the systems. The step function also determines whether the system has reached a terminal step (the pendulum has fall on either side) and uses a flag *done* to record that. Along with the new state, the method also calculates the reward for the input action. In this case, the ideal function for calculating the reward is the cosine of the angle, as in the specified range, it rewards the vertically aligned position with the maximum reward (Fig. 1).

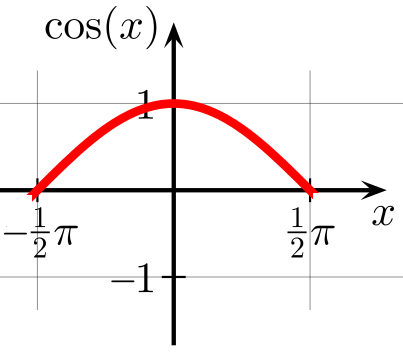


Fig. 1: Cosine of angular position as Reward function

* Render: Creates an animation across the recorded angular states of the system. The visualization is good way to observe how the system behaves in the given iteration (Example: Fig. 2).

random\_agent\_inverted\_pendulum.py: