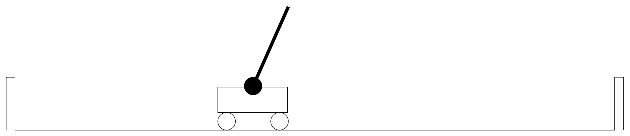
CSE 6369 - *Reinforcement Learning*

Homework 3- Spring 2014

Due Date: March 20 2014

Learning to Balance an Inverted Pendulum on a Cart

Consider the task where you have to learn how to successfully balance an inverted pendulum on a cart moving on a limited track (i.e. on a track that has end points that the cart can not cross). To balance the inverted pendulum the system can read the location and velocity of the cart on the track, the angle at which the inverted pendulum is leaning, and the angular velocity with which the pendulum is tilting. To influence the cart, it can apply a force on the cart in either direction.



This application has been used relatively frequently for basic algorithm testing in Reinforcement learning and its dynamic equations (for the 1D version) as well as the description of an Actor-Critic implementation for learning it can be found in *Anderson, Charles W. "Learning to control an inverted pendulum using neural networks." Control Systems Magazine, IEEE 9.3 (1989): 31-37*[1].

Simulation Code

For the following project components, code that implements the dynamic equations of the inverted balancing system and a corresponding graphical simulation in the original 1D version as well as in 2D and 3D (i.e. where the cart can move in a rectangular region of the 2D plane or in a cubic volume of 3D space, respectively) is provided in C. This code also provides a routine that performs the original state space discretization used in the original Anderson paper listed above. Instructions on how to use the code are provided in a separate document. If you prefer to implement the simulation yourself in a different system you can do so as long as the code can be run on a system I have access to.

Model-Free RL Algorithms

1. Using a 1D version of the task (i.e. the original task with a linear track), the state discretization used in the Anderson paper, and two available actions (force to the left and force to the right), implement the listed reinforcement learning algorithms. For each algorithm perform a number

[1]Note, the original paper had a typo that inverted the direction of gravity. Depending on which version of the paper you find, you might have to invert the gravity term.

of learning runs for different settings of learning rates and discount factors and plot the learning performance (in terms of the time the inverted pendulum is kept up until it falls) agains the number of learning trials (i.e. restarts of the cart in the center with a straight inverted pendulum). You can choose what exploration policy you want to use for learning.

a) Implement Q(0) learning for the 1D cart-pole system.

b) Implement SARSA(0) learning for the 1D cart-pole system.

c) Implement Actor-Critic learning for the 1D cart-pole system.

d) Briefly discuss your experiences with the different learning algorithms and the changes in learning rate, discount factor, and exploration rate.

2. Using the same scenario as in Part 1., add eligibility traces to the algorithm implementations.

   a) Implement Q(λ) learning for the 1D cart-pole system.

   b) Implement SARSA(λ) learning for the 1D cart-pole system.

   c) Implement Actor-Critic learning with eligibility traces (i.e. TD(λ) for the critic) for the 1D cart-pole system.

   d) Briefly discuss your experiences and the effect eligibility traces had on learning performance.

2D Inverted Pendulum Balancing

2. Expanding the task to a 2D balancing task where the cart can move in the X and the Y direction and where the inverted pendulum can tilt around 2 axes (forward and sideways), implement a learning system to learn to balance it. For the state representation you can either use the previous discretization for each of the two axes or you can choose your own. For actions you now have 4 action possibilities, each consisting of applying a constant force (positive or negative) in each one of the two directions.

   a) Implement a SARSA learner for the 2D task.

   b) Implement a Q learner for the 2D task.

   c) Briefly discuss your experiences.

3. *Extra Credit:* Implement a Reinforcement learner for the 2D task from Part 2 that does not use the standard discretization but instead uses a function approximator using tile coding and at least 3 tilings with SARSA to the problem.

   a) Implement a SARSA with tile coding for the 2D task.

   b) Try different tilings and discuss the differences in terms of learning performance.