

CS 5804: Homework #6

Assigned: November 27, 2006

Date Due: December 11, 2006

1. (60 points) For your final assignment, you will implement Monte Carlo and temporal difference algorithms to play the dots and boxes game. More details of this game can be had from <http://mathworld.wolfram.com/DotsandBoxes.html>. The goal of the game is, when all squares are completed, to be the player who has scored the greater number of points (however, the game can also end in a draw).

Assume that the initial board given is an m -by- n rectangular lattice of points. You should formulate this game as a reinforcement learning problem, i.e., suitably define the states, actions, rewards, and value function (involving the discount factor). You should view the game from the perspective of one player so that the other player can be treated as part of the environment. It is your responsibility to figure out what the rewards and discount factor should be so that maximizing (optimizing) value corresponds to winning the game.

Write your code generically so that it is easy to change m and n . Start with small values of m and n initially to get the hang of the game and then proceed to higher values.

Design an epsilon-soft on-policy Monte Carlo control algorithm for playing this game, i.e., the goal is to find a policy π for winning the game. Assume that the reinforcement learning agent is one of the players (and who is learning the Q values for various state-action combinations). The other player will be simulated by your program; this can be just a black box that makes random moves or some other sophisticated program (it could even be the agent again!). Use the template given in class as a guide. Define states, actions, rewards, and values yourself.

Each time the policy is improved, evaluate its performance by pitting it against another player for, say, 100 new games (this is just to evaluate the performance and is not used by the learning algorithm). Plot a graph of number of games won versus each iteration of policy improvement.

Run your program on different choices of m and n . When the dust settles, inspect the final Q values and see if you can state, in English, the policy learnt by your algorithm (i.e., what is the rule for winning at this game?). For full credit, give a careful definition of how you formulated the RL problem (what are the states, actions, rewards, and how is the value function defined), a URL for your code, graph(s), and a set of observations.

2. (20 points) Inspect your Q-values and determine if it is beneficial to go first in this game for certain values of m and n .
3. (20 points) Implement the TD (temporal differences) algorithm and see if it improves convergence. You have to be careful here because this is a two-player game and only certain temporal differences have to be backed up for updating the Q function (see your book for more details). Pick some m and some n , and plot the curves for both Monte Carlo and TD. Which is better? For full credit, give a URL for your code, graph(s), a description of how you setup the algorithm, and a list of observations.