

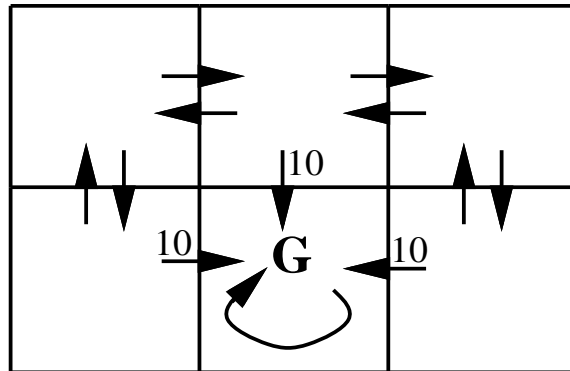
CS 410/510
Machine Learning
Winter, 2006

Homework 8:
Reinforcement Learning

Due Thursday, March 16.

1. (Adapted From T. M. Mitchell, *Machine Learning*, problem 13.2.) Consider the deterministic grid world shown below with the absorbing goal state **G**. Here the immediate rewards are 10 for the labeled transitions and 0 for all unlabeled transitions. Let the discounting factor $\gamma = 0.8$.

- (a) Give the V^* value for every state in this grid world.
- (b) Give the $Q(s, a)$ value for every transition in this grid world.
- (c) Show an optimal policy π^* —i.e., the optimal action from each state.
- (d) Now consider applying the Q -learning algorithm to this grid world, assuming the table of \hat{Q} values is initialized to zeros. Assume the agent begins in the bottom left grid square and then travels clockwise around the perimeter of the grid until it reaches the absorbing goal state, completing the first training episode. Describe which \hat{Q} values are modified as a result of this episode, and give their revised values. Answer the question again assuming the agent now performs a second identical episode. Answer it for a third episode.



2. Textbook, problem 13.3.

3. The game of Nim (also called Tactix) is played by the following rules: Starting with one or more piles (heaps) of one or more pieces each, players alternate by taking all or some of the pieces in a single heap. The player taking the last piece or stack of pieces is the winner.

For example, the picture below illustrates a starting position with four heaps, with one, three, five, and seven pieces respectively. Players alternate in removing a number of pieces from a chosen heap. The number of pieces chosen to remove must be greater than zero, and less than or equal to the number of pieces in the heap.



Suppose you wanted to write a Q -learning program that learned to play Nim with four piles, each of which could initially contain between one and seven pieces.

Let a state of the world be represented by the four-tuple (n_1, n_2, n_3, n_4) , where n_i is the number of pieces in the i th heap.

- (a) Define a possible reward function $r(s, a)$ for this game.
- (b) Suppose you represent the \hat{Q} function as a table. How many entries would it have, and why? (Recall that you have four heaps, with at most seven pieces in each heap. Allow the \hat{Q} function table to have enough entries to cover all the possible states and actions.)
- (c). Suppose you have initialized \hat{Q} function table to all zeros. Suppose the initial state of the game is $(1, 3, 5, 7)$, and your program is going to go first. Using your reward function above and $\gamma = 0.8$, simulate a learning session of one game, as follows:

- Repeat until the game is over:
 1. Choose an action (i.e., a move), and update \hat{Q} table your program would create using the Q -learning algorithm (give only the entries that change, if any do).
 2. Choose a move for the opponent, and update the state.

Give the final non-zero entries in your \hat{Q} table.

4. Optional Extra Credit Problem! If you are feeling really ambitious, implement the Nim Q -learner in any language of your choice, and demonstrate how the \hat{Q} -table changes as the learner plays against an opponent.