Games and Computation Homework #9: Monte Carlo Reinforcement Learning

Answer these questions within the HW #9 Moodle quiz:

MENACE

For this exercise, you will design a Machine Educable N-Approach Computational Engine to learn Approach N in the same tradition as Donald Michie's MENACE (Machine Educable Naughts And Crosses Engine). First, read about Michie's MENACE and play against it in the webpage demo linked in our course Readings. The main difference between Approach N and Naught and Crosses (a.k.a. Tic-Tac-Toe) is that Approach N is a game of chance whereas Naught and Crosses is a combinatorial game. This introduces a complication: If we eliminate the last "bead" of an optimal action because of bad luck, we can never play that move again. How might you apply Michie's main idea to Approach N without elimination of optimal plays? Please fully describe your design of MENACE for Approach N below such that someone else could implement your approach:

The Game of Turn Tail

In the game of Turn Tail, you are given a certain number of dollar coins and begin with no winnings. (Unless otherwise stated, assume you begin each game with 10 coins.) Each turn, you may choose to keep all of your winnings and end the game, or flip all of your coins at once to potentially win more. If all of your flipped coins come up tails, you lose all of your winnings and the game is over. If any of your flipped coins come up heads, you win as many dollars as the total number of coins you flip. Then, you take all coins that came up tails, and these "turn tail and run", i.e. they are removed from the game for your next turn. Finally, your winnings can never exceed some maximum winnings limit of dollars. (Unless otherwise stated, assume your winnings limit is 50.) If at any point your winnings would exceed this limit, your winnings are capped at the limit and the game is over.

Import the package fys-187-4-turn-tail.zip into your Eclipse workspace, open TurnTailReinforcementLearning.java, and run it. Play several games until you are familiar with Turn Tail.

Dynamic Programming Game Value Computation

Expected Winnings with 10 Coins

What are your expected (average) future winnings when you begin the game with 10 coins and no winnings?

_____

Turn Tail Splitting

If you were given the option to "split" the game, i.e. pay the same price to play, divide your coins into two halves, play two separate games with the two halves, and collect the winnings of both games, should you split? _____

Turn Tail Doubling

If you were given the option to "double" your game, i.e. pay twice the price to play the game with twice the coins, should you double? _____

Turn Tail 1 Coin Hold Value

What would be the minimum winnings for which a player would hold with 1 coin? _____

Turn Tail 2 Coins Hold Value

What would be the minimum winnings for which a player would hold with 2 coins? _____

Turn Tail 3 Coins Hold Value

What would be the minimum winnings for which a player would hold with 3 coins? _____

Raising the Winnings Limit

If the winnings limit was raised to 100, what would be the new expected (average) future winnings when you begin the game with 10 coins and no winnings? _____

Lowering the Winnings Limit

If the winnings limit was lowered to 25, what would be the new expected (average) future winnings when you begin the game with 10 coins and no winnings? _____

Monte Carlo Reinforcement Learning with Exploring Starts

Exploring Starts Expected Winnings with 10 Coins

Running Monte Carlo Reinforcement Learning with Exploring Starts for 1,000,000 iterations, what is the estimated expected (average) future winnings when you begin the game with 10 coins and no winnings? _____

Monte Carlo Reinforcement Learning with On-Policy UCB1 Control

On-Policy UCB1 Control Expected Winnings with 10 Coins

The "exploration boost factor" is an amount that we multiply the exploration term of UCB1 in order to promote more exploration. Running Monte Carlo Reinforcement Learning with On-Policy UCB1 Control for 1,000,000 iterations with an exploration boost factor of 1, what is the estimated expected (average) future winnings when you begin the game with 10 coins and no winnings? _____

On-Policy Control Sub-Optimal HOLD Values

How would you best explain the many HOLD values in the policy *for low winnings* where you know that a FLIP action is obviously optimal?

- ☐ The algorithm hasn't had enough iterations to experience these states at all.
- ☐ The algorithm hasn't had enough iterations to experience these states sufficiently to have the correct policy.
- ☐ The algorithm will not gain enough experiences of these states because of the low exploration boost factor.
- ☐ The algorithm will never experience most of these states, as most are unreachable from the initial state.

On-Policy UCB1 Control Convergence

Should the algorithm eventually converge to the optimal policy for all reachable states with infinite computation?

- ☐ Yes
- ☐ No

On-Policy UCB1 Control with 10-fold Exploration Expected Winnings with 10 Coins

The "exploration boost factor" is an amount that we multiply the exploration term of UCB1 in order to promote more exploration. Running Monte Carlo Reinforcement Learning with On-Policy UCB1 Control for 1,000,000 iterations with a higher exploration boost factor of 10, what is the estimated expected (average) future winnings when you begin the game with 10 coins and no winnings? _____

Monte Carlo Reinforcement Learning On-Policy epsilon-Greedy Control

On-Policy 0.1-Greedy Control Expected Winnings with 10 Coins

Running Monte Carlo Reinforcement Learning On-Policy epsilon-Greedy Control for 1,000,000 iterations with epsilon = 0.1, what is the estimated expected (average) future winnings when you begin the game with 10 coins and no winnings?

_____

On-Policy epsilon-Greedy Control Convergence

Should the algorithm eventually converge to the optimal policy for all reachable states with infinite computation for epsilon = 0.1? (Hint: Experiment with larger numbers of iterations. What do you observe when comparing results of lesser iterations?)

- ☐ The algorithm will not converge because of random variance of Monte Carlo simulation.
- ☐ The algorithm will not converge because of continued constant exploration affecting play policy.
- ☐ The algorithm will converge because random variance of Monte Carlo simulation does not affect large averages.
- ☐ The algorithm will converge because suboptimal exploration is lost from the average over time.

On-Policy 0.01-Greedy Control Expected Winnings with 10 Coins

Running Monte Carlo Reinforcement Learning On-Policy epsilon-Greedy Control for 1,000,000 iterations with epsilon = 0.01, what is the estimated expected (average) future winnings when you begin the game with 10 coins and no winnings? _____