Winter 2020

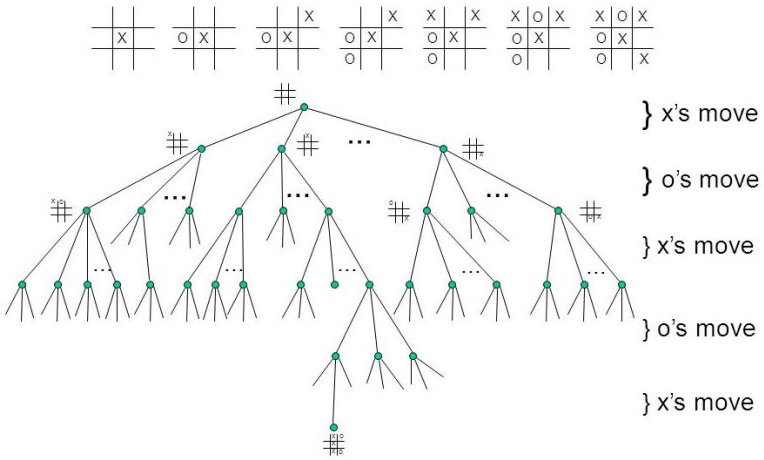
CS 446/546

Advanced Topics in Machine Learning

Programming Assignment #2

Due Date: Thursday, 3/12

In this homework you will write code to implement tic-tac-toe using techniques from Reinforcement Learning.



Your Assignment:

Part 1: Write a simple simulator for tic-tac-toe that encodes (and displays) a standard 3x3 grid, with a check for licit moves and "goal state" (i.e. did someone win/draw).

Implement a form of Q-learning to train a tic-tac-toe playing "agent". One straightforward option for Q-learning is to use a Q-matrix, in which the rows correspond to states and the columns correspond to actions (you have the option of using a more sophisticated method than a Q-matrix if you wish, such as a Q-network). The Q-matrix is commonly initialized to all zeros at the beginning of a run (although you may use a different initialization strategy if you prefer – please note this in your write-up).

At each time step *t* during an episode, your code should do the following:

- Observe the current state s_t
- Choose an action a_t, using ε-greedy action selection (I recommend training *on-policy*)
- Perform the action
- Observe the new state s_{t+1}
- Update $Q(s_t, a_t) = Q(s_t, a_t) + \eta(r_t + \gamma max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t))$
- Receive reward r_t (at the conclusion of a game, include "reward" for outcomes, including: win, lose or draw)

For choosing actions with ε-greedy action selection, initialize ε (for example: $\varepsilon = 0.1$), and decrease it by a constant value Δ every m epochs until it reaches 1 (I encourage you to experiment with different choices for Δ and m, as well as the initial value for ε).

After each training epoch, test your agent on 10 games against a baseline random opponent. Record the total score of your agent against the baseline out of these 10 games (+1 for a win, +.5 for draw, 0 for loss). After training is completed, print a plot of the training progress for your agent with the epoch number on the horizontal axis and the (total score) / 10 against the baseline opponent on the vertical axis.

When you have completed training, play 10 games against your agent and report these results.

(Optional) Parts 2-3 are optional

Part 2: Experiment with Learning Rate. Choose 4 different values for the learning rate, η, approximately evenly spaced in the range [0,1], keeping the other parameters set as in Part 1. For each value, give a performance plot as described above. Discuss how changing the learning rate changes these results.

Part 3: Q-network. Use a Q-network in place of a Q-matrix; describe how you train and architected your network. Provide a performance plot.

Here is what you need to turn in:

Your spell-checked, double-spaced report with the information requested above. Also, your (briefly) commented code with instructions how to run it. Your report should include a detailed summary of your approach and results.

How to turn it in:

- Send these items in electronic format to me (arhodes@pdx.edu) on the due date. No hard copy please!
- The report should be in pdf format and the code should be in its original format (e.g., .py, .m, etc.)