Last name: Liang First name: Haoming SID#: 1430396

Collaborators:

CMPUT 366/609 Assignment 1: Step sizes & Bandits

Due: Tuesday Sept 19 by gradescope

Policy: Can be discussed in groups (acknowledge collaborators) but must be written up individually

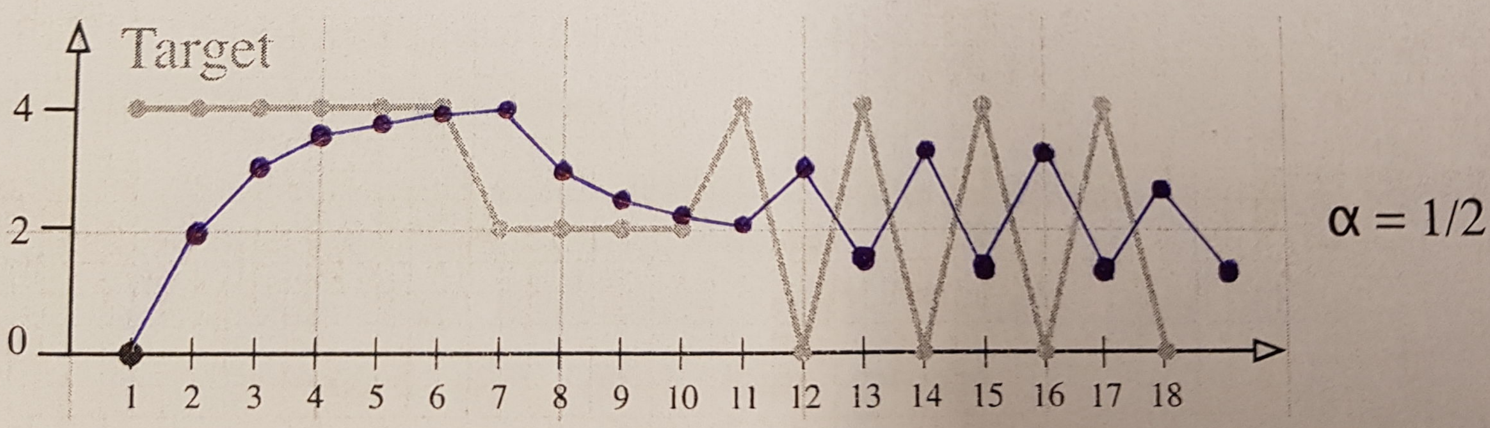
There are a total of 100 points on this assignment, plus 15 points available as extra credit!

Question 1 [50 points] Step-sizes. Plotting recency-weighted averages.

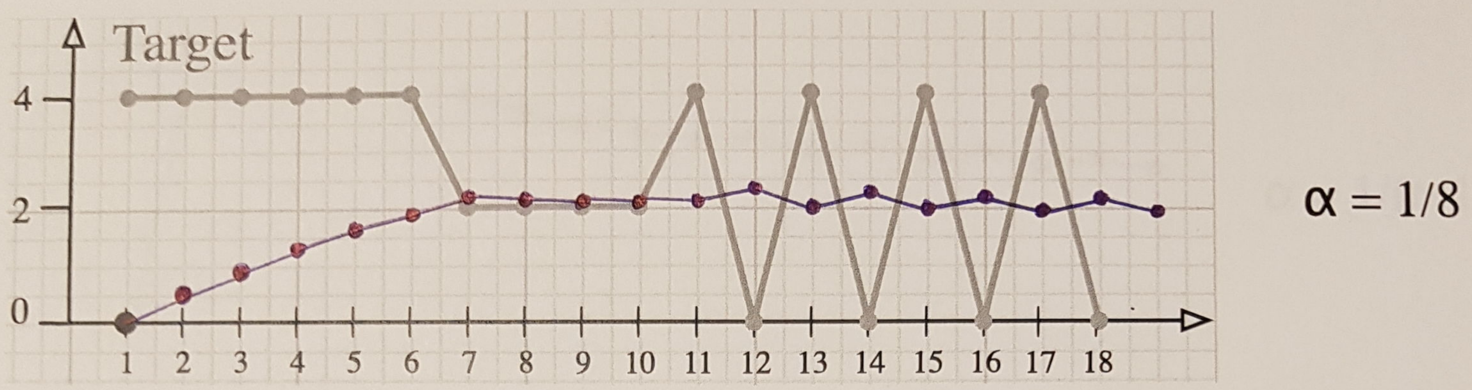
Equation 2.5 (from the SB textbook, 2nd edition) is a key update rule we will use throughout the course. This exercise will give you a better hands-on feel for how it works. This question has **five** parts.

Do all the plots in this question by hand. To make it easier for you, I'll include some graphing area and a start on the first plot here, so you should just be able to print these pages out and draw on them.

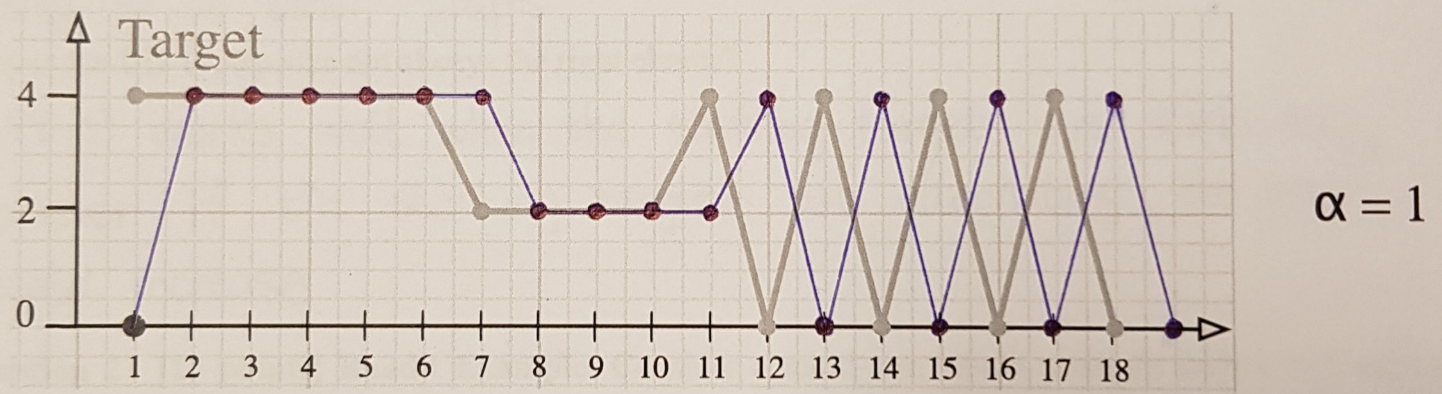
Part 1. [15 pts.]
Suppose the target is 4.0 for six steps, then 2 for four steps, and then alternates between 4.0 and 0 for the remaining time steps, as shown by the grey line in the graph below. Suppose the initial estimate is 0 ($Q_1 = 0$), and that the step-size (in the equation) is 0.5. Your job is to apply Equation 2.5 iteratively to determine the estimates for time steps 1-19 (one time-step past step 18). Plot them on the graph below, using a blue pen, connecting the estimate points by a blue line. The first estimate Q_1 is already marked below:



Part 2. [5 pts] Repeat the graphing/plotting portion of Part 1, this time with a step size of 1/8.



Part 3. [5 pts.] Repeat with a step size of 1.0.



Part 4. [10 pts.] Best step-size questions.

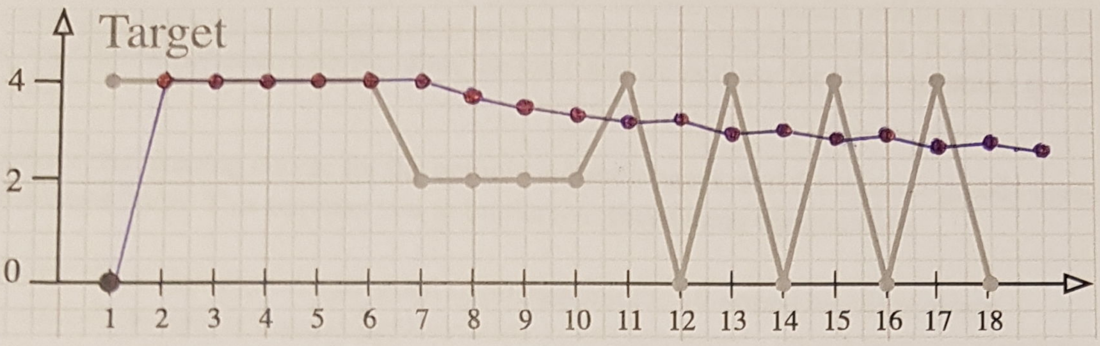
Which of these step sizes would produce estimates of smaller absolute error if the target continued alternating for a long time? Please explain your answer.

$\alpha = 1/8$ since the alternating part is less alternating compared with other two graphs

Which of these step sizes would produce estimates of smaller absolute error if the target remained constant for a long time? Please explain your answer.

$\alpha = 1$ since the estimate remained constant when the target is constant

Part 5. [15 pts.] Repeat with a step size of 1/(*t*-1) (i.e., the first step size you will use is 1, the second is 1/2, the third is 1/3, etc.).



$\alpha = 1/(t-1)$

Based on all of these graphs, why is the 1/(t-1) step size appealing?

Because the estimated values ~~is~~ are closest to the targets compared to other graphs

Why is the 1/(t-1) step size not always the right choice?

1/(t-1) works well when the starting targets are stationary, however if the starting targets are alternating, the error would increase

Question 2 [10 points] Bandit Example. Consider a multi-arm bandit problem with $k = 5$ actions, denoted 1, 2, 3, 4, and 5. Consider applying to this problem a bandit algorithm using ε-greedy action selection, sample-average action-value estimates, and initial estimates of $Q_1(a) = 0$ for all a. Suppose the initial sequence of actions and rewards is $A_1 = 2$, $R_1 = -2$, $A_2 = 1$, $R_2 = 5$, $A_3 = 3$, $R_3 = 3$, $A_4 = 1$, $R_4 = 4$, $A_5 = 4$, $R_5 = 3$, $A_6 = 2$, $R_6 = -1$. On some of these time steps the ε case may have occurred causing an action to be selected at random. On which time steps did this definitely occur? On which time steps could this possibly have occurred?

A_1 is definitely random since the reward is unknown and must explore
A_2 and A_3 is also definitely random because still exploring
A_4 is possibly random since A_2 produces greedy reward
A_5 is definitely random since this action is never selected
A_6 is definitely random since A_1 is not a greedy choice

Question 3. Bandit task Programming. [40 pts.]

This programing exercise will give you hands-on feel for how bandit problems are implemented, and how incremental learning algorithms select actions based on observed rewards. In addition, this exercise will be your first experience with RL-glue, the interface we will use for all programing questions in this course.

Recreate the learning curves for the optimistic bandit agent, and the epsilon-greedy agent in Figure 2.3 of Sutton and Barto. This requires you to implement **three** main components:

1) A RL-Glue Environment program implementing the 10-armed bandit problem
2) A RL-Glue Agent program implementing an epsilon-greedy bandit learning algorithm. Use the incremental update rule (Equation 2.5), with two different parameter settings:
    - alpha = 0.1, epsilon = 0, and $Q_1 = 5$
    - alpha = 0.1, epsilon = 0.1, and $Q_1 = 0$
3) A RL-Glue Experiment program implementing the experiment to generate the data for your plot. Compute the % Optimal action per time-step, averaged over 2000 runs

All code must be written in **Python2** to be compatible with the RL-Glue interface provided to the class. It is not acceptable to implement your own interface.

Please submit:
1) your plot [10 pts.]
2) all your code (including any graphing code used to generate your plot) [30 pts.]

Bonus Programing Question. [5 pts.]
Implement the UCB agent described in chapter two and evaluate it on the bandit environment from Question 3. Can you get the UCB agent to outperform the epsilon-greedy agent? Feel free to modify the parameters of the epsilon-greedy agent (alpha, epsilon, and the initial Q estimates) in order to better understand the relative strengths of both algorithms. Describe how we would go about determining and reporting on which agent is better for this task.

Bonus Question. [5 points extra credit]
Exercise 2.4 from Sutton and Barto (*Reward weighting for general step sizes*)

Bonus Question. [5 pts.]
Exercise 2.6 from Sutton and Barto (*Mysterious Spikes*. Use your implementation from Question 3 to better understand what is happening in Figure 2.3)