Last name:_____ First name:_____ SID#:_____

Collaborators:_____

# CMPUT 366Assignment 1: Step sizes & Bandits

Due: Tuesday Sept 18 by gradescope

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

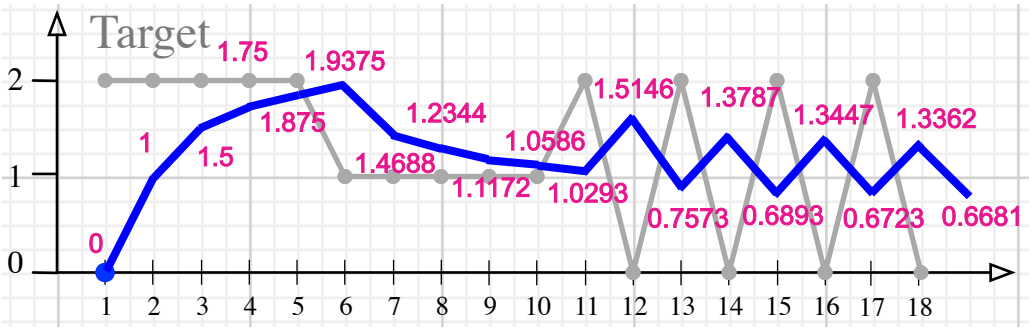
There are a total of 100 points on this assignment, plus 15 points available as extra credit!

**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.

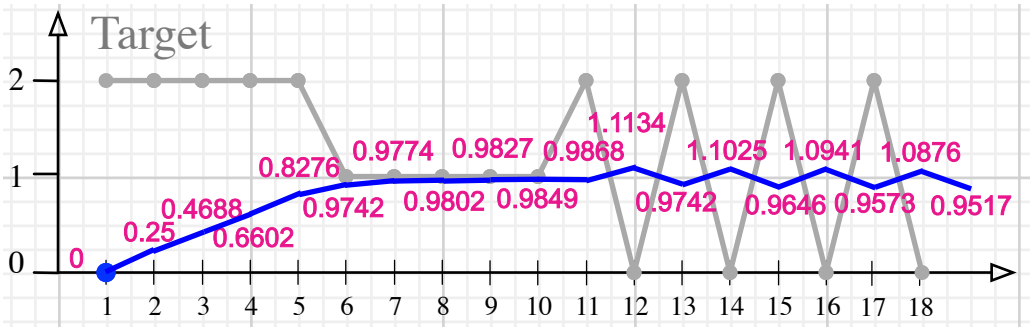
**Part 1.** [15 pts.]
Suppose the target is 2.0 for five steps, then 1 for five steps, and then alternates between 2.0 and 0 for 8 more steps, as shown by the grey line in the graph below. Suppose the initial estimate is 0, and that the step-size (in the equation) is 0.5. Your job is to apply Equation 2.5 iteratively to determine the estimates for time steps 1-19. 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:




$\alpha = 1/2$

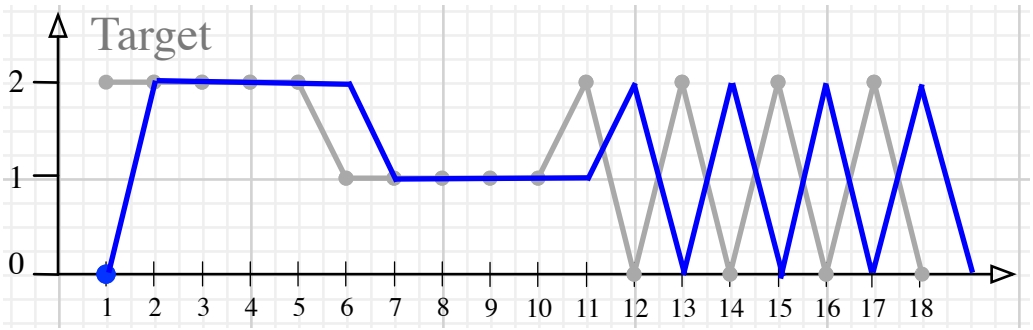
$$Q_{n+1} \doteq Q_n + \alpha\left[R_n - Q_n\right]$$

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




$\alpha = 1/8$

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




$\alpha = 1$

**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 = 0.8

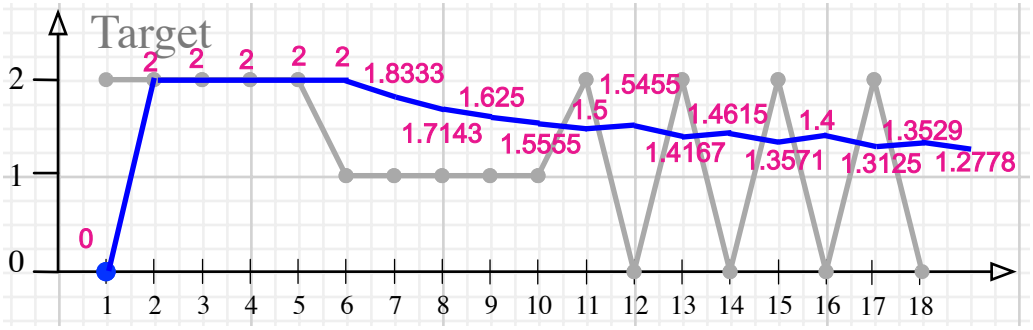
When target alternates between 0 and 4, Q will stay around 2, and the absolute error will be 2. Other step size will greater than 2.

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

When target stay the same, alpha = 1 will capture the change and follow it immediately, and no further absolute error.

**Part 5.** [ 15 pts.] Repeat with a step size of 1/($t$-1) for t≥2. (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?

At beginning change should be catch fast because the target remain constant for a long time. For the later on, the target becomes alternates, where we want to stay at the average to get the minimum absolute error. Therefore, we need step-size bigger at the beginning, and step-size smaller when t increase. That is exactly what 1/(t-1) do.
This compute the sample average.

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

Lets say target alternates at the beginning and stable later on, the 1/(t-1) step-size cannot fit in well because first it cannot minimize the error at the beginning, and cannot converge to the target fast later on. So it is not always the best choice.

**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 $\varepsilon$-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 $\varepsilon$ 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?

| Step | Q(1) | Q(2) | Q(3) | Q(4) | Q(5) | Q(6) | | Action | Reward |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | | 2 | -2 |
| 2 | 0 | -2 | 0 | 0 | 0 | 0 | | 1 | 5 |
| 3 | 5 | -2 | 0 | 0 | 0 | 0 | | ~~3~~ | 3 |
| 4 | 5 | -2 | 3 | 0 | 0 | 0 | | 1 | 4 |
| 5 | 4.5 | -2 | 3 | 0 | 0 | 0 | | ~~1~~ | 3 |
| 6 | 4.5 | -2 | 3 | 3 | 0 | 0 | | ~~2~~ | -1 |

Step 1, 2, 4 are possible greedy action or random action.
Step 3, 5, 6 are definitely random action

Orange background indicate the maximum value currently. If action is selected with orange background, then it is possible a greedy move since it is selecting the maximum potential reward.

**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 **Python3** 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)

**Exercise 2.4** If the step-size parameters, $\alpha_n$, are not constant, then the estimate $Q_n$ is a weighted average of previously received rewards with a weighting different from that given by (2.6). What is the weighting on each prior reward for the general case, analogous to (2.6), in terms of the sequence of step-size parameters? □

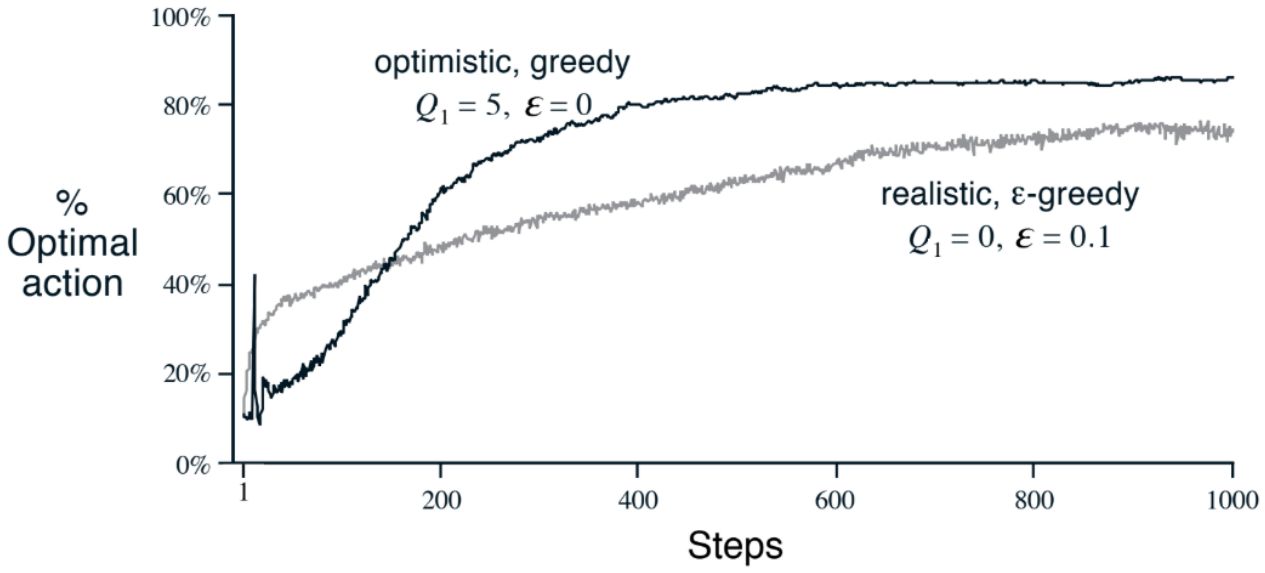
$$
\begin{aligned}
Q_{n+1} &= Q_n + \alpha\Big[R_n - Q_n\Big] \\
&= \alpha R_n + (1-\alpha)Q_n \\
&= \alpha R_n + (1-\alpha)\left[\alpha R_{n-1} + (1-\alpha)Q_{n-1}\right] \\
&= \alpha R_n + (1-\alpha)\alpha R_{n-1} + (1-\alpha)^2 Q_{n-1} \\
&= \alpha R_n + (1-\alpha)\alpha R_{n-1} + (1-\alpha)^2\alpha R_{n-2} + \\
&\qquad \cdots + (1-\alpha)^{n-1}\alpha R_1 + (1-\alpha)^n Q_1 \\
&= (1-\alpha)^n Q_1 + \sum_{i=1}^{n} \alpha(1-\alpha)^{n-i} R_i. \qquad (2.6)
\end{aligned}
$$

*Solution.* With the same update rule $Q_{n+1} := Q_n + \alpha_n[R_n - Q_n] = \alpha_n R_n + (1-\alpha_n)Q_n$, we see that

$$
\begin{aligned}
Q_{n+1} &= \alpha_n R_n + (1-\alpha_n)Q_n \\
&= \alpha_n R_n + (1-\alpha_n)\alpha_{n-1}R_{n-1} + (1-\alpha_n)(1-\alpha_{n-1})Q_{n-1} \\
&= \alpha_n R_n + (1-\alpha_n)\alpha_{n-1}R_{n-1} + (1-\alpha_n)(1-\alpha_{n-1})\alpha_{n-2}R_{n-2} \\
&\quad + (1-\alpha_n)(1-\alpha_{n-1})(1-\alpha_{n-2})Q_{n-2} \\
&\ \ \vdots \\
&= Q_1 \prod_{i=1}^{n}(1-\alpha_i) + \sum_{i=1}^{n} R_i \alpha_i \prod_{j=i+1}^{n}(1-\alpha_j)
\end{aligned}
$$

and so the coefficient of each $R_i$ is $\alpha_i \prod_{j=i+1}^{n}(1-\alpha_j)$.

**Exercise 2.6:** *Mysterious Spikes* The results shown in Figure 2.3 should be quite reliable because they are averages over 2000 individual, randomly chosen 10-armed bandit tasks. Why, then, are there oscillations and spikes in the early part of the curve for the optimistic method? In other words, what might make this method perform particularly better or worse, on average, on particular early steps? □




As we can see that the first spike is around the step 11, that is because it is too optimistic at the beginning, and therefore the program will select each different action once. After that 10 steps, the algorithm will do a greedy move since it is pure greedy. Therefore the step 11 will have a high possibility to select the optimal action.

But because after the first round (10 steps), the estimates are still too optimistic, so therefore the same thing happed again. The program select each action once in the next 10 steps. After that, the 21st step will also do a greedy move which gave the second spike because it is again likely the best action in all actions.