HW 2: LPs, POMDPs, and Bandits

EECE 571N – Sequential Decision-Making (EECE 571N)

Instructor: Cyrus Neary

Due: **2025-10-20** at 23:59 PT

Instructions. Submit a single PDF to Canvas. Please include your name and student number at the top of that PDF. For all questions, show your work and clearly justify your steps and thinking. Please feel free to include any code as an attachment at the end of the PDF. State any assumptions. Unless otherwise specified, you may collaborate conceptually but must write up your own solutions independently.

Grading. Points for each part are indicated. The total number of achievable points is 100. Partial credit is available for incorrect answers with clear reasoning.

1. **LP solutions to MDPs and occupancy measure variables.** [20]

   (a) In the dual linear program formulation of MDPs, we introduce variables $x(s, a)$. Explain what $x_{s,a}$ represents in terms of trajectories under a stationary policy π and initial distribution α. Explain the relation between the variables $x_{s,a}$ and a corresponding stationary policy π.

   (b) In class, we saw that given a stationary policy π, the corresponding occupancy measure variables $x^{\pi}_{s,a}$ are defined as

   $$x^{\pi}_{s,a} = \sum_{s' \in S} \alpha(s') \sum_{t=0}^{\infty} \gamma^t Pr(S_t = s, A_t = a \mid S_0 = s', \pi).$$

   Use this definition to establish the relationship

   $$\sum_{s \in S} \sum_{a \in A} x^{\pi}_{s,a} R(s, a) = \sum_{s' \in S} \alpha(s') V^{\pi}(s').$$

   How can we interpret this statement? I.e., how does it connect the variables in the dual LP to the expected return of a policy?

2. **Belief updates in POMDPs.** In a POMDP, suppose your current belief is $b(s)$, you take action a, and you observe o. [10]

   (a) Write down the formula for the updated belief $b'(s')$. (Hint: see class notes)

   (b) In words, explain what the two terms (the transition probabilities and the observation probabilities) are contributing to this update.

3. **Confidence intervals for a one-armed bandit.** The remaining questions will all ask for you to implement certain ideas and algorithms in Python. Please use the starter code provided at `https://github.com/cyrusneary/UBC_EECE571N_fall_2025/blob/main/Homework-starter-code/HW2/bandits_starter_code.py`, and simply fill in the missing TODO comments to complete the assignment. Insert your figures as images to the relevant questions of your submission PDF document. [10]

(a) Write a class that implements a uniform random policy (i.e., the policy should always select among the environment arms uniformly at random). Your implementation should follow the provided policy class interface. More specifically, your `UniformPolicy` class should implement the `select_arm(self)` and `update(self, arm, reward)` methods which select an arm in the environment according to the policy's decision rule, and update the counts, running average reward, etc of the arm, respectively.

(b) Complete the implementation of the method `compute_radii(policy)`, which takes an instance of a policy class as input, and modifies the policy in place, updating its 'radii' attribute to reflect the CI radius of all of the arms $a \in A$, given the current time step t and the number of times arm a has been sampled in the past $N(a)$, using the following formula we explored in class.

$$\text{rad}_t(a) = \sqrt{\frac{2 * log(t)}{N(a)}}$$

(c) Use the starter code to implement an environment with a *SINGLE* Bernoulli random arm. Instantiate your uniform random policy class on this environment (since this is a one-armed environment, it should always select the same arm). Write a for loop that samples the arm and calls `policy.update(a, r)` each loop with the resulting reward value `r`. Use the provided method `plot_bandit_estimates(env, policy, title)` to generate plots of the confidence interval after $t = 100$, $t = 1000$, and $t = 10000$ steps. **Include those plots in your submission.**

4. **Creating a Bernoulli 9-armed bandit environment.** Use the provided code to create a bandit environment with 9 Bernoulli arms. The arms should have parameter $p = 0.1, 0.2, \ldots, 0.9$, respectively. [5]

5. **UCB1 algorithm implementation and analysis.** [10]

(a) Implement the `UCB1` algorithm that we discussed in class. Hint: You should again implement a `UCB1Policy` class that implements `select_arm(self)` and `update(self, arm, reward)`. Instantiate your uniform random policy class on the 9-arm environment from the previous question.

(b) Write a for loop over decision steps t that samples the arm using `policy.select_arm(self)` and calls `policy.update(a, r)` each loop with the resulting reward value `r`. Use the provided method `plot_bandit_estimates(env, policy, title)` to generate plots of the confidence interval after $t = 100$, $t = 1000$, and $t = 10000$ steps. **Include those plots in your submission.**

(c) Explain, in your own words, how UCB1 balances exploration and exploitation. Why do the upper bounds of the confidence intervals of all of the arms always appear to remain balanced with each other?

6. **Algorithm regret comparison.** [45]

(a) Implement the uniform exploration policy we saw in class with an per-arm exploration budget of $N = 500$. That is, the policy should try each arm 500 times. After this exploration budget (total of $k \times N = 9 \times 500$) has been exhausted, it should always select the arm with the highest estimated mean payout during every subsequent rount.

(b) Implement the ϵ-greedy algorithm with a parameter $\epsilon = 0.3$.

(c) Implement the successive elimination algorithm we saw in class.

(d) Implement the Thompson sampling algorithm we saw in class.

(e) Use the provided method `run_bandit(env, policy, horizon)` to run each of the implemented algorithms (including the previously implemented `UniformPolicy` and `UCB1` policies) for a horizon of $T = 10000$ steps. In a single plot, compare all of the cumulative regrets of the different algorithms. That is, plot cumulative regret on the y-axis, and the number of elapsed decision steps on the x-axis. **Include the plot in your submission**.

(f) Comment on the results. Why do the different algorithms have different shapes of cumulative regret plots?