Homework #4 - Reinforcement Learning and Ethics

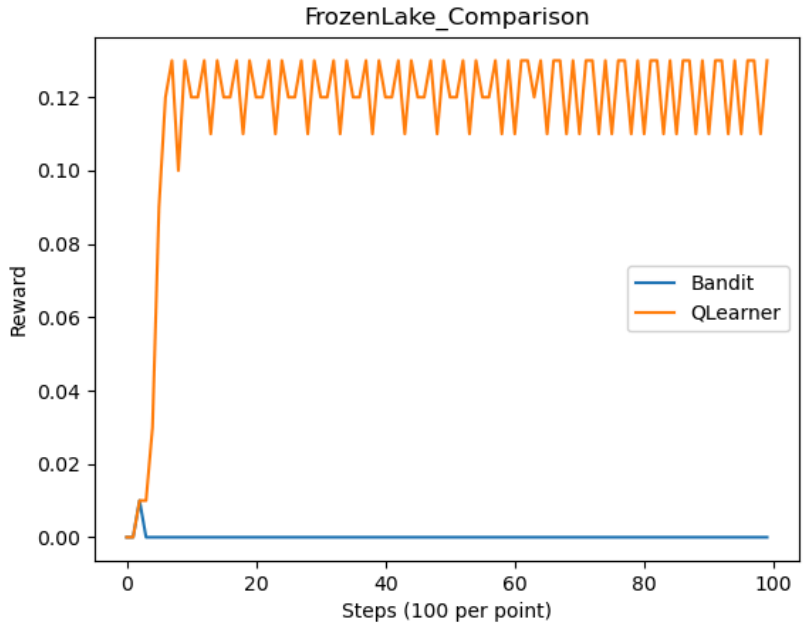
Team: Group 4

Q2. Bandits vs. Q-Learning (2.0 points):

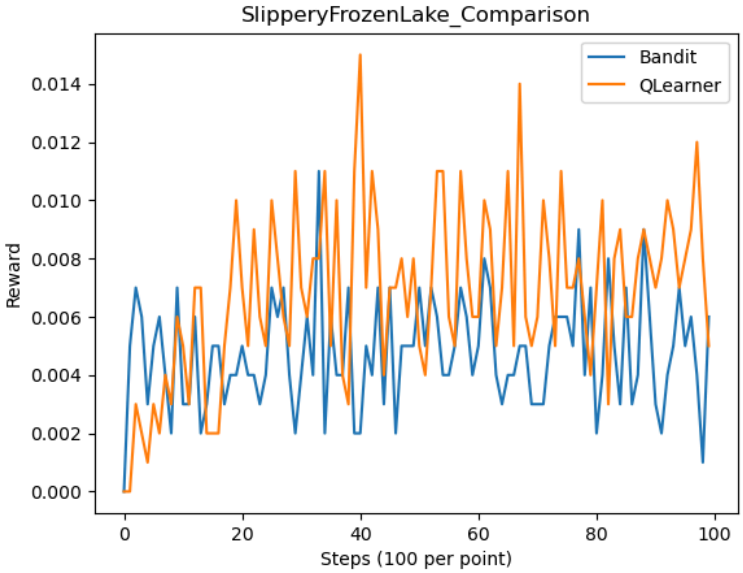
a)

I. Plot 1**:** This graph compares the performance of two reinforcement learning algorithms - Multi-Armed Bandit and Q-Learning - on the FrozenLake environment.

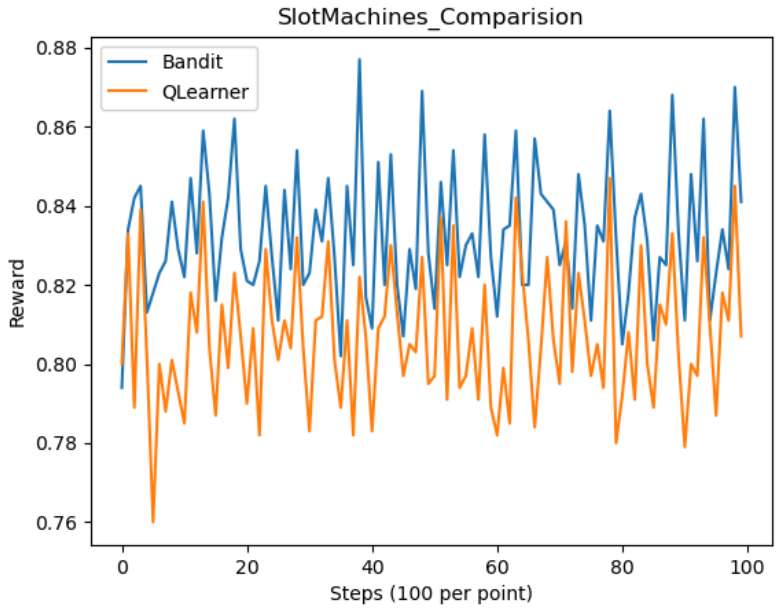
Q-Learning Significantly outperforms Multi-Armed Bandit in the FrozenLake environment by quickly learning and maintaining a higher reward level (around 0.12) while bandit method fails to learn effectively, staying near zero reward throughout training.



Plot 2: In the Slippery FrozenLake environment, both Q-Learning and Multi-Armed Bandit perform poorly with high variability and similar low rewards (around 0.002-0.014), indicating that the added stochasticity makes the environment significantly more challenging for both algorithms.



Plot 3: In the SlotMachines environment, the Multi-Armed Bandit consistently outperforms Q-Learning with higher average rewards (around 0.84 vs 0.80), which is expected since slot machines are a stateless problem better suited for bandit algorithms.



b) Based on the plots, Q-Learning performs significantly better than MultiArmedBandit in the FrozenLake (non-slippery) environment.

This happens because:

1. Q-Learning is designed to handle sequential decision-making problems where actions affect future states. In FrozenLake, you need to plan multiple steps ahead to reach the goal - moving from the start to the goal requires making a sequence of correct moves through a grid.
2. Q-Learning's key advantage here is that it learns state-action values (Q-values) that account for both immediate rewards and potential future rewards through its temporal difference learning mechanism. The gamma (discount factor) parameter allows it to consider future consequences of actions, which is crucial in FrozenLake where reaching the goal requires several correct sequential moves.
3. In contrast, the MultiArmedBandit treats each action as independent and doesn't consider state transitions or future rewards. It can't understand that taking a particular action in state A might lead to a better state B, which then leads to the goal. This makes it ineffective at solving sequential decision problems like FrozenLake.

This explains why Q-Learning achieves and maintains a higher reward level (around 0.12) while the MultiArmedBandit's performance stays near zero - it's simply not capable of learning the sequential nature of the task.

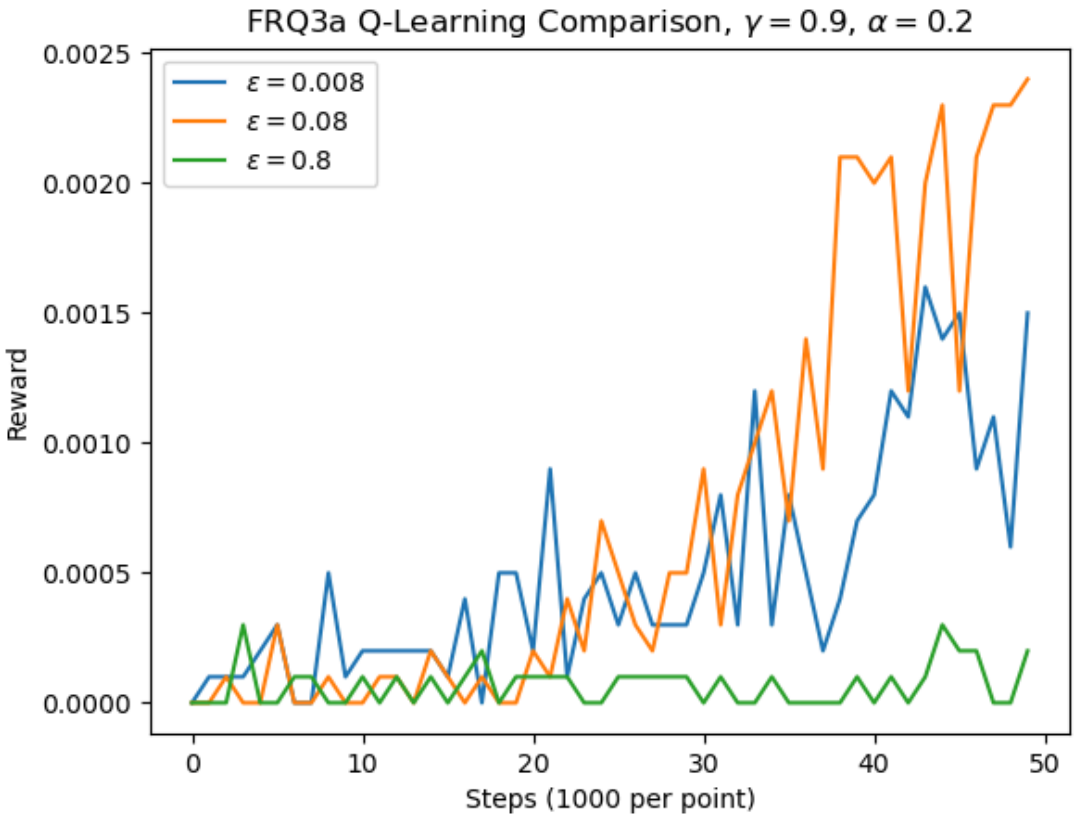
c) No, changing MultiArmedBandit's hyperparameters cannot match Q-Learning's performance in FrozenLake because MultiArmedBandit fundamentally cannot learn sequential state-action relationships - it treats each action independently, while FrozenLake requires understanding how current actions affect future states to reach the goal.

d) MultiArmedBandit performs better than Q-Learning in the SlotMachines environment because this is a stateless problem that exactly matches the bandit setting - each "arm" (slot machine) has a fixed reward distribution, and the task is simply to identify and stick with the best option. MultiArmedBandit is specifically designed for this scenario where you only need to learn which action gives the highest immediate reward without worrying about future states or sequential decision-making, making it more efficient than Q-Learning's more complex state-action value learning approach.

e) Yes, setting gamma=0 in Q-Learning would make it ignore future rewards and act like a bandit algorithm, though this essentially strips away Q-Learning's defining features and makes it unnecessarily complex for a simple slot machine task.

Q3.

Plot:



a) This code is comparing Q-Learning performance with different epsilon values (0.008, 0.08, and 0.8) on an 8x8 Slippery FrozenLake environment. For each epsilon value, it runs 10 trials of training for 50,000 steps, keeps track of rewards across 50 bins, and then plots the average performance of each epsilon setting to visualize how different exploration rates affect learning. The code uses fixed values for gamma (0.9) and alpha (0.2) while varying only epsilon to understand the impact of exploration vs exploitation.

b) Based on the plot, epsilon = 0.08 (orange line) appears to be the best value as it achieves the highest and most consistently improving rewards over time. This makes sense because:

1. ε = 0.008 (blue) explores too little, making it slow to find good policies
2. ε = 0.8 (green) explores too much, preventing it from exploiting good policies
3. ε = 0.08 provides a good balance - enough exploration to discover good paths but also enough exploitation to capitalize on what's learned

The middle epsilon value works best because the Slippery FrozenLake environment requires both exploration to find successful paths and exploitation to reinforce them, especially given its stochastic nature.

c) Here's a one-sentence explanation for each epsilon value:
   i) ε = 0.008: With more timesteps, it would eventually achieve better performance but very slowly, as its low exploration rate means it rarely discovers new potentially better paths.
   ii) ε = 0.08: The medium epsilon would likely maintain its superior performance and continue improving steadily as it has the right balance of exploration vs exploitation even over longer periods.
   iii) ε = 0.8: Even with more timesteps, performance would remain poor as the high exploration rate means it spends too much time trying random actions rather than exploiting learned good paths.

d) Testing with too many epsilon values (30 or 300) risks overfitting to the specific environment, as we might find an epsilon that works exceptionally well for the training domain but performs poorly when the agent is deployed to a new domain with different characteristics - this is essentially hyperparameter overfitting, similar to overfitting model parameters in supervised learning.

Q4: Tic-Tac-Toe (2.0 points): Suppose we want to train a Reinforcement Learning agent to play the game of Tic-Tac-Toe (see https://en.wikipedia.org/wiki/Tic-tac-toe), and need to construct an environment with states and actions. Assume our agent will simply choose actions based on the current state of the game, rather than trying to guess what the opponent will do next.

a. What should be the states and actions within the Tic-Tac-Toe Reinforcement Learning environment? Don't try to list them all, just describe how the rules of the game define what states and actions are possible. How does the current state of the game affect the actions you can take?

Answer: Each state is represented by a 3x3 grid depicting a possible game configuration. Each of the nine cells in the grid can either contain an X, an O, or remain empty. This results in a total of [obj] theoretical configurations. However, due to the alternating turns of the players during the game, not all of these configurations are achievable. An action consists of the agent placing their symbol (X or O) in a selected cell within the grid. The available actions at any moment depend on the game's current state.

b. Design a reward function for teaching a Reinforcement Learning agent to play optimally in the Tic-Tac-Toe environment. Your reward function should specify a reward value for each of the 3 possible ways that a game can end (win, loss, or draw) as well as a single reward value for actions that do not result in the end of the game (e.g., your starting move). Explain your choices.

Answer: The reward function is defined as follows:

+1: Reward for a win, incentivizing the agent to take actions that lead to victory, as the goal is to maximize rewards.
-1: Penalty for a loss, discouraging the agent from making moves that result in defeat.
0: Neutral reward for a draw, neither encouraging nor discouraging this outcome.
-0.1: Penalty for non-terminal moves, motivating the agent to win as quickly as possible and avoid unnecessary actions, as each move not leading directly to a win incurs a -0.1 penalty

c. Suppose you were playing a more complicated game with a larger board, and you want the agent to learn to win as fast as possible. How might you change your reward function to encourage speed?

Answer: For a more complex game with a larger board, the reward function can be adjusted as follows:
Reward Function:

- {1+(1/Number of Moves)}: Reward for a win, where the additional (1/Number of moves) encourages the agent to win in fewer moves, as fewer moves yield a higher reward.
- -1: Penalty for a loss, discouraging actions that lead to defeat.
- 0: Neutral reward for a draw, neither promoting nor discouraging the outcome.
- -0.1: Penalty for non-terminal moves, pushing the agent to act efficiently and avoid unnecessary moves.

This setup emphasizes the importance of speed, with a higher reward for faster wins and a penalty for moves that do not contribute to ending the game efficiently

Q5. Fair ML in the Real World (2.0 points): Read Joy Buolamwini and Timnit Gebru, 2018. "Gender shades: Intersectional accuracy disparities in commercial gender classification." Conference on fairness, accountability and transparency, then use it to help answer the following questions (see https://proceedings.mlr.press/v81/buolamwini18a/buolamwini18a.pdf).

a. Buolamwini and Gebru use PPV and FPR as metrics to measure fairness. Find the definition of these in the paper, then look up the corresponding definition for NPV and FNR (these appear in the slides). Assuming you were applying for a loan and you know a ML classifier is deciding whether to grant it to you: would you rather have that decision made by a system with a high FPR or a high FNR? Why? Provide a detailed justification.

Answer:

Positive Predictive Value (PPV): The percentage of predicted positive outcomes that are accurate.
False Positive Rate (FPR): The fraction of actual negative instances that are mistakenly classified as positive.
Negative Predictive Value (NPV): The percentage of predicted negative outcomes that are correct.
False Negative Rate (FNR): The fraction of actual positive instances that are incorrectly classified as negative.

b. Assuming you were applying for a loan and you know a ML classifier is deciding whether to grant it to you: would you rather have that decision made by a system with a high PPV or a high NPV? Why? Provide a detailed justification.

Answer:The ML model operates with two categories: "Grant Approved" and "Grant Rejected." I would prefer a classifier with high PPV because it indicates a strong likelihood that candidates classified as "Grant Approved" truly deserve approval. In other words, the model is highly accurate in identifying qualified candidates for approval.On the other hand, a high NPV suggests that the model is effective at correctly rejecting undeserving candidates as "Grant Rejected." However, in a scenario where the primary goal of an agency is to approve loans, ensuring high NPV is less critical.

c. What recommendations do Buolamwini and Gebru make regarding accountability and transparency of ML systems? How does this relate to specific metrics such as PPV or FPR?

Answer: Buolamwini and Gebru suggest the following recommendations:

1. Promote transparency in model training and evaluation by publicly documenting the demographic and phenotypic makeup of the datasets used.
2. Present performance metrics broken down by demographic and phenotypic subgroups (e.g., intersections of gender and skin tone).
3. Address dataset imbalances by utilizing balanced samples to ensure all subgroups are adequately represented in the model's learning process.
4. Provide users the flexibility to adjust thresholds for metrics like True Positive Rate (TPR) and False Positive Rate (FPR), enabling tailored decision-making based on specific trade-offs.

Relation to Metrics like PPV and FPR:

Positive Predictive Value (PPV): A high PPV reflects the reliability of positive predictions. Analyzing PPV across subgroups highlights any biases in how accurately predictions are made for different demographics.

False Positive Rate (FPR): A high FPR indicates a greater frequency of incorrect positive classifications. Disaggregating FPR by subgroup uncovers disparities in error rates across different demographic groups.

d. What is intersectional about the analysis conducted by the authors? What does that analysis show?

Answer: The authors analyzed combined gender and skin tone subgroups, such as darker-skinned females, darker-skinned males, lighter-skinned females, and lighter-skinned males, referring to this approach as intersectional analysis. Their findings revealed that darker-skinned females faced the poorest model performance, while lighter-skinned males had the highest accuracy. This highlights the compounded impact of biases related to both gender and skin tone in the dataset.

e. In Section 4.7, the authors say that their "findings ... do not appear to be confounded by the quality of sensor readings." What do they mean by "confounded" in this context? Why is it important to the thesis of this paper to check whether their findings are confounded?

Answer: In this thesis, stating that the findings are not confounded by sensor quality implies that the results are unaffected by external factors such as image or sensor quality. Verifying whether the findings are influenced by such factors is crucial to ensure that the observed results accurately reflect biases in the algorithms rather than being a consequence of poor data quality.