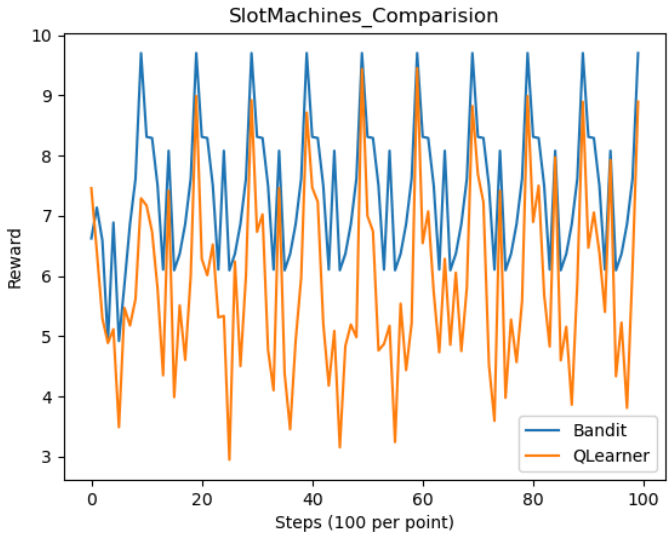
Nicole Birova, Jack Hanke, Dan Plotkin, Vrishani Shah, Hanna Zelis
MSAI 349 – 01
David Demeter
9 December 2024

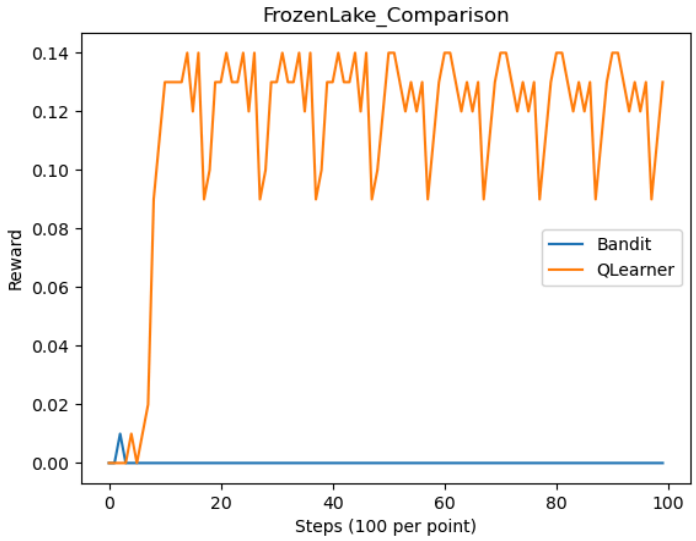
Free Responses

1. **Code in zip file**

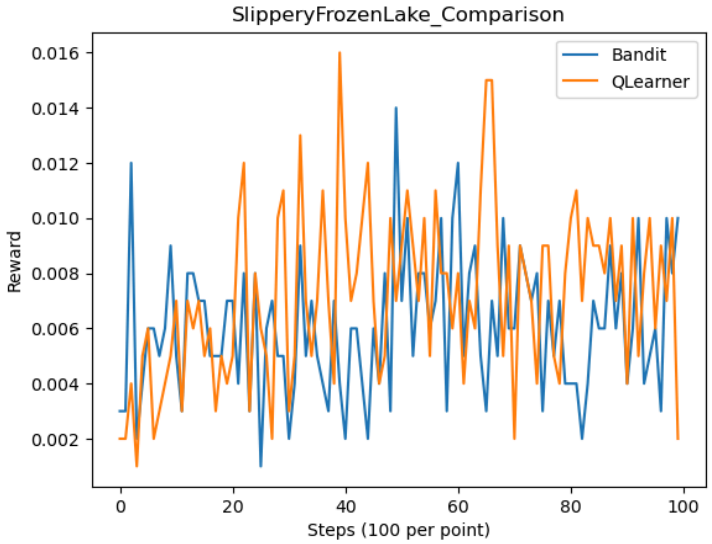
2. **Bandits vs. Q-Learning**
   a. Run python -m q2.py; it will create three plots: 2a_SlotMachines_Comparison.png, 2a_FrozenLake_Comparison.png, and 2a_SlipperyFrozenLake_Comparison.png. Please read about the FrozenLake environment (https://gymnasium.farama.org/environments/toy_text/frozen_lake/) . Each plot will show a comparison of your MultiArmedBandit and QLearning models on the named environment (e.g., SlotMachines). Include those plots here. For each plot, provide a one-sentence description of the most notable trend. Pay attention to the scale on the y-axis.



The SlotMachines plot shows periodic behavior in both agents, with the Bandit agent consistently outperforming the QLearner.



For the FrozenLake plot, the Bandit fails to achieve any consistent reward, while the QLearner Agent learns to achieve reward quickly, and then converges to a periodic pattern.



For the SlipperyFrozenLake plot, both agents obtain some reward in a non-periodic fashion, with the QLearner outperforming the Bandit in general.

b. In which of the above plots does Q-Learning appear to receive higher rewards on average than MultiArmedBandit? Provide an explanation for why that happens, based on your understanding of Q-Learning.

The QLearner achieves higher average reward than the Bandit in both the FrozenLake and SlipperyFrozenLake environments. As a QLearner tracks the value of a specific state, and the Lake environments have states of varying true value, the QLearner can make state-based decisions while the Bandit cannot. In the Lake environment, the Bandit stumbles into the reward once, in which it then learns that right is the most valuable action. It proceeds to move right a large majority of the time, never receiving +1 reward again. In the SlipperlyLake environment, the Bandit, though not making state-based decisions, can randomly "slip" into occasionally reaching the goal, leading to more reward collected over time.

c. Following b.: in the environment(s) where MultiArmedBandit was the worse model, is there any way you could change your choice of hyperparameters so that MultiArmedBandit would perform as well as Q-Learning? Why or why not?

Though increasing the value of epsilon would increase the Bandit's ability to randomly wander into the +1 reward, it is unlikely that this random wandering outperforms Q-Learning.

d. In which of the above plots does MultiArmedBandit appear to receive higher rewards on average than Q-Learning? Provide an explanation for why that happens, based on your understanding of MultiArmedBandit.

The Bandit receives higher reward in the SlotMachines environment. This is because SlotMachines is an environment where the true reward does not change over time. The MultiArmedBandit assumes stationary true rewards, which allows for a more informed estimate of the true reward. Q-Learning, on the other hand, is designed to handle non-stationary reward environments. This means that the QLearner in the stationary environment has a biased estimate of a slot machines true value based on the slot machines recent performance. This bias in estimate leads to the Bandit outperforming the QLearner.

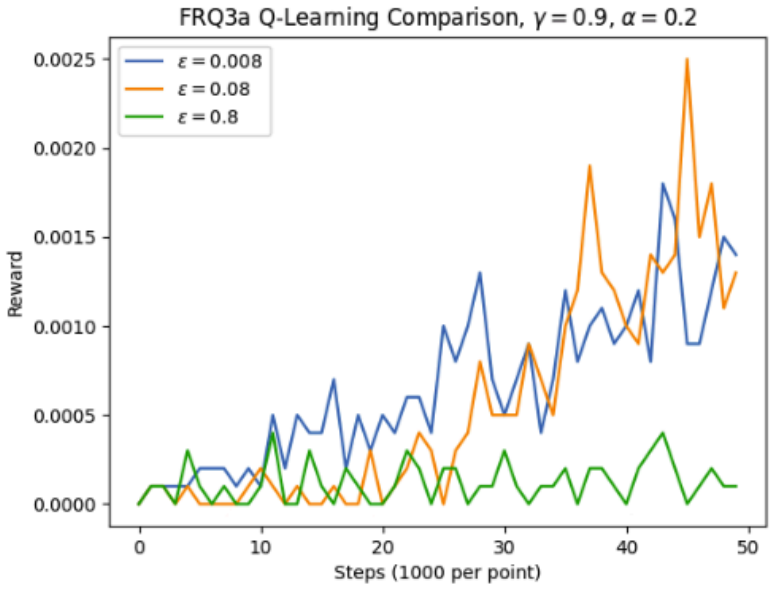
e. Following d.: in the environment(s) where Q-Learning was the worse model, is there any way you could change your choice of hyperparameters so that QLearning would perform as well as MultiArmedBandit? Why or why not?

Decreasing our value of alpha corresponds with taking more of the recent history into account when calculating our current valuation of a specific action. This would correspond to a less biased estimate of the true return of a specific slot machine, leading to performance that is similar to a Bandit, but would also likely fall a little short of Bandit performance.

3. **Exploration vs. Exploitation**
    a. Look at the code in q3.py and run python -m q3.py and include the plot it creates (free_response/3a_g0.9_a0.2.png) as your answer to this part. In your own words, describe what this code is doing.

The code in *q3.py* implements a Q-learning agent on the *FrozenLake-v1* environment to analyze the impact of different values of epsilon (ε) (0.008, 0.08, and 0.8) (line 13) on learning performance. The environment, an 8x8 slippery grid (line 11), shows a reinforcement learning task where the agent must balance exploration and exploitation to maximize rewards. Exploration is trying new actions to potentially discover better rewards while exploitation is choosing actions that are currently believed to yield the best rewards. The agent is trained over 50,000 steps (line 16) for each epsilon value, with hyperparameters such as the discount factor (γ = 0.9) (line 17) and a learning rate (α = 0.2) (line 18) influencing how the agent updates its action-value estimates.

For each epsilon, the code runs 10 trials (line 15) to ensure results are statistically robust, storing the overall rewards across training episodes. The agent uses the *QLearing* class (line 30), where the epsilon-greedy strategy determines whether it selects a random action (exploration) or the current best action (exploitation). The results are averaged across trials (line 39) and plotted to compare the performance of the different ε values. This code shows a clear visualization of how varying the ε affects the agent's ability to balance exploration and exploitation, via the graph made, shown below, and saved as *3a_g0.9_a0.2.png* in the *free_response* file.



    b. Using the above plot, describe what you notice. What seems to be the ``best'' value of epsilon? What explains this result?

The plot shows that the model's performance varies significantly across the three ε values. An ε value of 0.08 results in the best overall performance, with the highest and most consistent rewards by the end of training. This happens because it finds a balance between exploration and exploitation, allowing the model to discover better strategies while effectively utilizing its knowledge. However, the ε value of 0.008 performs worse as it prioritizes exploitation too early, limiting its ability to seek optimal strategies. Finally, an ε value of 0.8 performs poorly because the high amount of exploration prevents the model from focusing on high-reward actions, resulting in slower performance growth and lower overall rewards.

c. The above plot trains agents for 50,000 timesteps each. Suppose we instead trained them for 500,000 or 5,000,000 timesteps. How would you expect the trends to change or remain the same for each of the three values of epsilon? Give a one-sentence explanation for each value.

If the models were trained for longer (500,000 or 5,000,000 timesteps), the trends for each epsilon value would be as follows:

- ε = 0.008: This model would likely continue to perform relatively well because it heavily exploits its learned strategies. However, its lack of initial exploration might mean it never finds the globally optimal strategy, restricting its potential performance.
- ε = 0.08: The performance of this model would likely remain the best among the three, as it balances exploration and exploitation even with extended training, allowing it to refine its strategy further.
- ε = 0.8: This model would improve over time as it continues exploring, but its performance still would be behind the others due to the high amount of randomness in action selection.

d. When people use reinforcement learning in practice, it can be difficult to choose epsilon and other hyperparameters. Instead of trying three options like we did above, suppose we tried 30 or 300 different choices. What might be the danger of choosing epsilon this way if we wanted to use our agent in a new domain?

Testing 30 or 300 different ε values might lead to overfitting the choice of the epsilon to the specific training setup. In a new environment, the model's performance could decline because the hyperparameter tuning was too specific to the original environment. This overfitting risks finding an epsilon that works well only under certain conditions, limiting the flexibility of the model's behavior. Also, the computational cost of testing a large range of values could become too much, especially for more intricate environments or longer training periods. A better approach might involve flexible techniques, such as gradually reducing epsilon over time, to handle a wider variety of situations.

4. **Tic-Tac-Toe**
    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?

The states would be the position of the board, and an action would be placing either an ‘X’ or an ‘O’ in a specific unoccupied square. At some specific state, what actions are legal is dependent on what states are unoccupied.

    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.

Let a win be a reward of +1, a loss -1, and draw 0. Also let non-terminal actions have a reward of 0. We choose this to signal to the algorithm that winning is the intended goal, losing is to be avoided, and a draw is not significant. All other moves have no inherent value, and so are also assigned 0 reward.

    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?

To prefer a fast win, a small negative reward should be received for every non-terminal action, say -1/1000. This encourages the agent to not only seek out a source of positive reward, but to do so quickly to minimize the small accrued losses on the path to this source.

5. **Fair ML in the Real World**
    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.

In *"Gender Shades: Intersectional Accuracy Disparities in Commercial Gender Classification,"* Buolamwini and Gebru define Positive Predictive Value (PPV) as the proportion of predicted positives that are true positives and False Positive Rate (FPR) as the proportion of actual negatives incorrectly classified as positives. Complementary metrics like Negative Predictive Value (NPV) and False Negative Rate (FNR) show the proportion of predicted negatives that are true negatives and the rate at which actual positives are misclassified as negatives, respectively. If applying for a loan, one would prefer a system with a high FPR rather than a high FNR. A high FPR would indicate that some ineligible applicants are approved, increasing chances of being approved even if misclassified, while a high FNR risks unfairly denying eligible applicants. A high FPR errs on the side of inclusion, which is preferable for an applicant seeking access to financial resources.

    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.

When a machine learning classifier determines loan approval, a system with a high Positive Predictive Value (PPV) is preferable to one with a high Negative Predictive Value (NPV). A high PPV ensures that when the system predicts approval, it is highly likely to be correct, meaning approved applicants are genuinely qualified. This enhances the fairness and reliability of the approval process, minimizing errors that could lead to unwarranted approvals or rejections. In contrast, a high NPV ensures accuracy in denying loans to ineligible applicants, which primarily benefits lenders but does not directly improve outcomes for qualified applicants seeking approval. Therefore, a system with a high PPV better supports fairness and access for eligible individuals while maintaining trust in the approval process.

    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?

Buolamwini and Gebru emphasize the importance of accountability and transparency in machine learning systems to mitigate bias and ensure fairness. They recommend practices such as auditing algorithms for performance disparities across demographic groups, using disaggregated evaluation metrics (e.g., PPV and FPR) to measure fairness at the intersection of race and gender,

and providing detailed documentation about datasets, including demographic composition and preprocessing steps. They also advocate for public accountability mechanisms, such as requiring companies to disclose the results of fairness audits and ensuring that systems are designed to minimize harm to marginalized groups.

These recommendations are directly tied to metrics like PPV and FPR because these metrics can reveal fairness issues in how systems treat different groups. For example, disparities in PPV between demographic groups indicate that the system is less accurate in identifying positive outcomes for certain groups, undermining fairness and trust. Similarly, high FPR for specific groups may show that the system disproportionately misclassifies individuals from those groups as belonging to a positive category, leading to unfair outcomes. By focusing on these metrics in an intersectional and transparent manner, Buolamwini and Gebru aim to highlight and address systemic biases in ML systems.

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

The analysis conducted by Buolamwini and Gebru is intersectional because it examines performance disparities across multiple overlapping demographic categories, specifically focusing on both race and gender. Rather than evaluating accuracy metrics for these characteristics in isolation, the authors assess how they interact, identifying compounded disparities for individuals at the intersection of multiple marginalized identities (e.g., Black women). This intersectional approach acknowledges that individuals can face unique forms of bias not captured when demographic attributes are considered separately.

The analysis shows that commercial gender classification systems perform worse for darker-skinned individuals and that the error rates are highest for darker-skinned women. For example, the systems demonstrate significantly higher misclassification rates for this group compared to lighter-skinned men, who experience the lowest error rates. These findings highlight how biases in training data and algorithm design disproportionately affect certain groups, emphasizing the need for fairness audits that consider intersectional demographics to ensure equitable outcomes.

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?

In Section 4.7, when the authors state that their "findings ... do not appear to be confounded by the quality of sensor readings," they mean that the observed disparities in classification performance are not caused by external factors such as poor image quality, lighting conditions, or other technical issues related to the sensors used to capture the data. In this context,

"confounded" refers to a situation where an observed effect (e.g., disparities in accuracy across demographic groups) could be falsely attributed to the machine learning model itself when it is actually caused by unrelated factors like the quality of sensor data.

It is crucial to establish that their findings are not confounded because the paper's thesis is centered on bias in the design and training of machine learning algorithms, not on hardware or data capture limitations. By ruling out sensor quality as a source of error, the authors strengthen their argument that the disparities in classification performance are due to systemic issues such as biased training data or algorithmic design flaws. This distinction is essential to ensure the validity of their claims about the need for more equitable and transparent machine learning systems.