Abstract:

Domain:

History:

OpenAI Gym:

Environment:

Hypothesis:

Jaiden (2):

Alex (2):

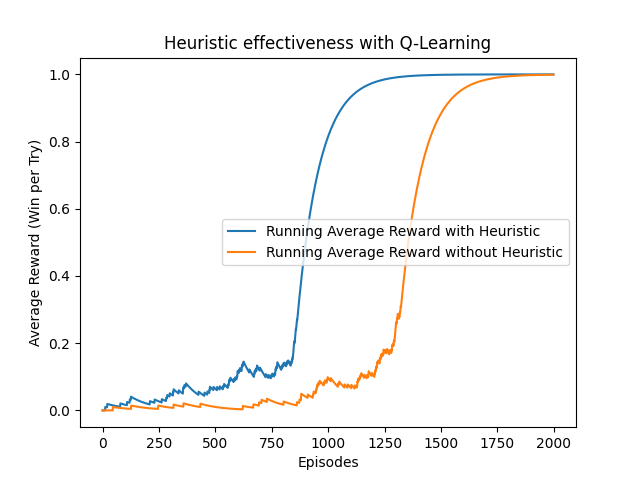
3. Adding heuristics to a Q-Learning algorithm will improve the average reward per episode.

4. Finding the optimal values of gamma, lambda, epsilon, and alpha will improve the average reward per episode.

Experiments:

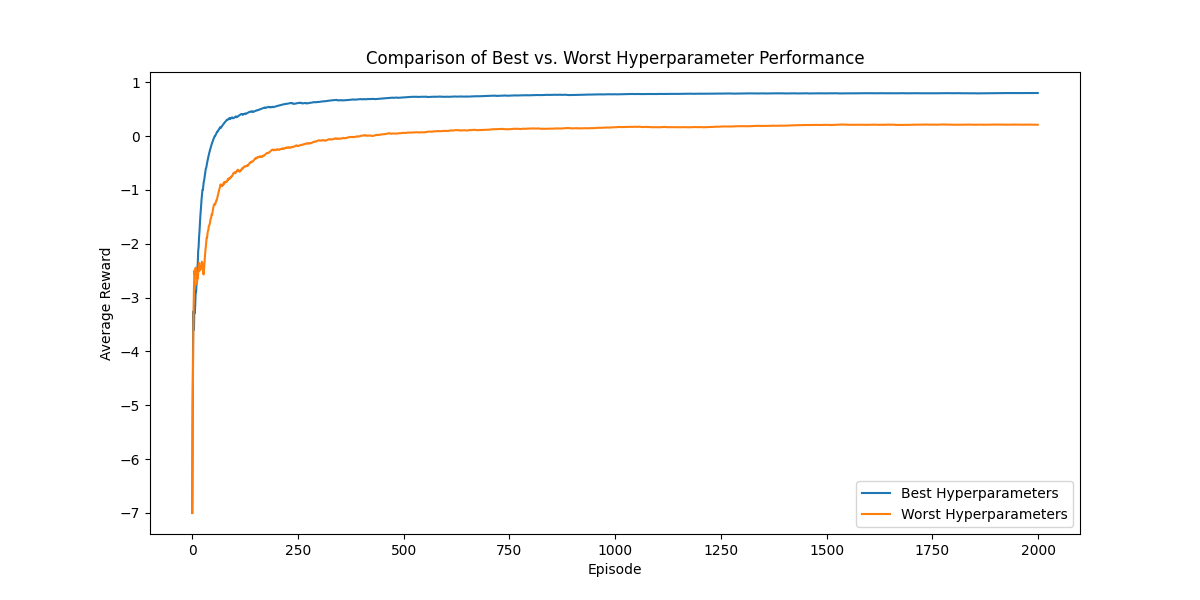
Jaiden (2):

Alex (2):



*Figure X*. The average reward per episode with the use of Q-Learning. The results show the performance due to heuristics and without the use of heuristics.

The third hypothesis states that integrating strategic knowledge or heuristics into Q-learning can enhance its efficiency. Heuristics, such as prioritizing specific actions or altering rewards to favor desirable outcomes, can streamline the learning process. To implement this theory, we found that the agent in most cases will never go up or left. We also found that an elf should never go to its previous state or try to go beyond the boundaries of the map. For example, if it is in the far-right column, it should never go right. Therefore, to use these findings we penalized the agents' action choices based on its current state by adding a negative reward for the action choice (-10) for moving up, left, to the previous state, or beyond the boundaries. The agent is then biased to not pursue an already low rewarding action due to the way our agent chooses the next action. Overall, the results (see figure 1) prove that this helps the agent learn faster since it does not waste its effort trying unfruitful actions, and thus increasing the average reward per episode.

*Figure X.* The average reward per episode with the use of SARSA(λ). The results show effectiveness of choosing optimal values of alpha, lambda, epsilon, and gamma.

The experiment conducts a hyperparameter optimization search across alpha (learning rate), lambda (eligibility trace decay rate), epsilon (exploration rate in epsilon-greedy policy), and gamma (discount factor for future rewards) for SARSA(λ). The objective is to determine the optimal combination of these parameters that maximizes the agent's performance. To do this we created a predefined list of different values to test and created a quadruple nested loop that will work through each possible paring of values. Throughout execution the program will keep track of the greatest and worst performer by measuring the total average reward after 2000 episodes. With our predefined list we found the best values (α = 0.1, λ = 0.6, ε = 0.05, γ = 0.9) and worst values (α = 0.1, λ = 0.8, ε = 0.2, γ = 0.95). The results (see figure X) prove that these parameters are worth optimizing. For this run, the average total reward for the best parameters was *0.8265* as compared to the worst parameters sitting at a *0.157*. Theoretically speaking, this is the difference of the agent finding the goal most of the time versus the agent almost never finding it which in turn proves the hypothesis correct.

Comparisions:

Jaiden (1):

Alex (2):

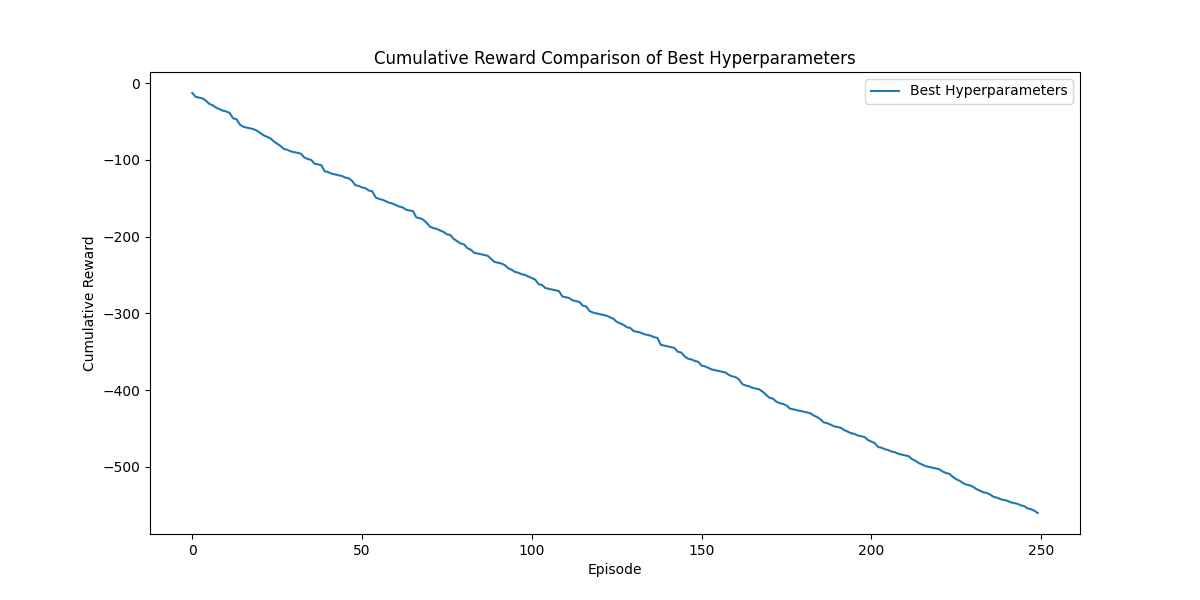
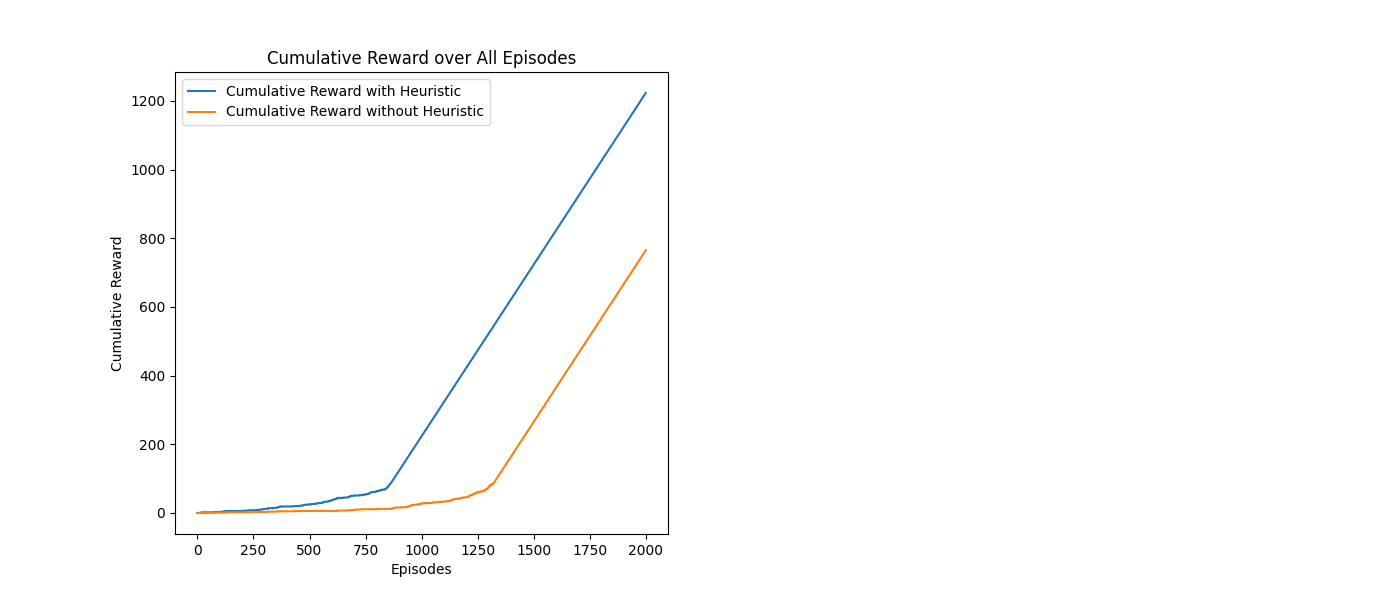
Comparing my enhanced temporal difference Q-learning algorithm with OpenAI's standard version [1] reveals a significant performance advantage in my approach, thanks to the integration of heuristics. My algorithm using heuristics boasts a *1200* cumulative reward while gymnasiums implementation shows *1100* at the same episode while using the same map setup. These heuristics boost efficiency and decision-making, allowing my algorithm to slightly outperform OpenAI's when heuristics are applied. Without heuristics, OpenAI’s performance outweighs my own, with my non-heuristic approach having a cumulative reward of about *790*.

My second experiment's algorithm underperforms compared to online benchmarks [2]. Though functional, it is slower, pointing to potential inefficiencies in exploration or updates. At episode 250 my cumulative reward is about *–600* whereas the other is at about *–180*.

This performance gap suggests my implementation might lack optimization in hyperparameter selection or algorithm structure but should be taken with a grain of salt considering that our implementations may reward and punish our agents with different severities. The main takeaway to compare would be the learning curve, which for our Q-Learning approach looks like it performs quite well and our SARSA(λ) performs about average.

[1] "FrozenLake\_v0 Tutorial - Gymnasium Documentation," Farama Foundation. [Online]. Available: https://gymnasium.farama.org/tutorials/training\_agents/FrozenLake\_tuto/. [Accessed: dd-Month-YYYY].

[2] An effective asynchronous framework for small scale reinforcement learning problems - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/Performances-of-different-algorithms-in-frozen-lake-problem\_fig9\_333706618 [accessed 24 Mar, 2024]



Novelty:

Alex:

The novelty of my implementation lies in the expansion of the traditional 4x4 grid to a 9x9 grid for the Q-learning environment, significantly impacting the state space and algorithm performance. In mathematical terms, a 4x4 grid offers 16 possible states, whereas a 9x9 grid expands this to 81 possible states. This exponential increase in state space elevates the complexity of finding an optimal path, as the algorithm must navigate a much larger environment, thereby affecting both the exploration required and the computational load. Moreover, the expanded grid tests the scalability and adaptability of Q-learning algorithms to more complex scenarios, offering insights into performance under heightened complexity.

Literature:

Jaiden (2):

Alex (3):

**D. S. Nair and P. Supriya, "Comparison of Temporal Difference Learning Algorithm and Dijkstra's Algorithm for Robotic Path Planning," 2018 Second International Conference on Intelligent Computing and Control Systems (ICICCS), Madurai, India, 2018, pp. 1619-1624, doi: 10.1109/ICCONS.2018.8663020. keywords: {Path planning;Navigation;Collision avoidance;Robot sensing systems;Reinforcement learning;Machine learning algorithms;Modified Temporal Difference Learning;Dijkstra's algorithm;Path planning;Obstacle avoidance},**

Robotic navigation, particularly path planning amidst static obstacles, stands as a critical challenge in the realm of automation and robotics. This work introduces a novel approach named Modified Temporal Difference Learning for path planning and obstacle avoidance, specifically designed for environments with static obstacles. Utilizing MATLAB, the algorithm is executed within a 4x4 grid setup, accompanied by a GUI that facilitates user input regarding obstacles' number, positions, and types.

**M. Thill, S. Bagheri, P. Koch and W. Konen, "Temporal difference learning with eligibility traces for the game connect four," 2014 IEEE Conference on Computational Intelligence and Games, Dortmund, Germany, 2014, pp. 1-8, doi: 10.1109/CIG.2014.6932870. keywords: {Coherence},**

In the realm of machine learning, particularly in the context of board games, systems have significantly evolved through self-play and temporal difference (TD) learning, as showcased by TD-Gammon and Lucas' Othello agent. A notable advancement in this field is the introduction of eligibility traces to enhance the learning process of board games like Connect Four, a game of moderate complexity. This study delves into the incorporation of eligibility traces—a method previously unexplored in large-scale systems, involving millions of training games and a substantial initial feature set.

**D. Xu, Y. Fang, Z. Zhang and Y. Meng, "Path Planning Method Combining Depth Learning and Sarsa Algorithm," 2017 10th International Symposium on Computational Intelligence and Design (ISCID), Hangzhou, China, 2017, pp. 77-82, doi: 10.1109/ISCID.2017.145. keywords: {Noise reduction;Feature extraction;Neurons;Algorithm design and analysis;Path planning;Data mining;Stacking;Sarsa(?) algorithm;depth learning;Stacking Denoising AutoEncoders},**

This paper addresses the limitations of the traditional Sarsa(λ) algorithm in path planning, such as its slow environmental learning and neglect of valuable information. It introduces a novel approach by integrating Stacked Denoising AutoEncoders (SDAE) to enhance environmental feature extraction while mitigating noise impact. The method employs SOM neural networks for position mapping, leveraging position data to determine the R value for Sarsa(λ) updates, thereby facilitating more accurate path planning.

Conclusion:

Future Work:

Contributions:

References: