An Analysis of Q Learning Methods for Reinforcement Learning Using OpenAI Gym

John Alex Keszler, Nicholas Pham, Jackson Wagner
CS182 Fall 2018

Abstract

In this analysis we explored a number of q learning algorithms applied to the OpenAI cartpole problem, including approximate q learning with hill climbing and simulated annealing on linear approximation weights, and epsilon-greedy tabular q learning over a discretized state space. The exploration of approximation algorithms for the cart-inverted pendulum problem demonstrated the success of these algorithms in solving a simple physical system. The exploration of tabular q learning over a discretized state space revealed the necessity of an epsilon greedy approach and yielded an optimal state space size of 18 for the cartpole system. The Deep Q Network algorithm was then applied to the lunar lander problem and the cartpole problem. Here we observed convergence in both cases. For the lunar lander problem, the optimal discount factor (gamma) was found to be .99, and the optimal interpolation parameter of the DQN q table (tau) to be .01. For the cartpole problem, convergence took greater than 1000 episodes, significantly more than the aforementioned simpler algorithms on the same problem.

Introduction

Q learning is a reinforcement learning technique used to learn effective policies that inform artificial agents how to act under various circumstances. Q learning is powerful, because it does not require a model of the environment to generate good action recommendations. The most basic forms of q learning are tabular q learning and approximate q learning. Whereas tabular q learning maintains a table of q values associated with (state, action) pairs, approximate q learning maintains a series of weights associated with features that are updated upon experiencing rewards.

While these methods are highly effective for simple problems with small state spaces, when the size of the state space becomes large policy convergence is not guaranteed. This led to the development of deep q learning. In this methodology, instead of storing a q value for each (state, action) pair, a neural network is used to approximate the q values associated with actions given a state. Coefficients are used to approximate the function relating inputs to outputs, and are learned through iterative adjustments along gradients that promise less error.

In this study, we compare the ability of these various veins of q learning to solve problems of various levels of difficulty. We also introduce unique modifications to standard algorithms as a means of improving the performance of our agents.

Methods

In this analysis, we utilized the OpenAI cart-pole and lunar lander interfaces to develop and test a variety of reinforcement learning algorithms. We began by implementing tabular q learning with a discretized state space and epsilon-greedy updates, and approximate q learning using local search techniques to evaluate weights, including a modified hill climbing algorithm and simulated annealing in the cartpole environment. We then attempted to apply similar algorithms to the OpenAI lunar lander system, at which point we realized the limitations of these simple algorithms. In order to create an effective agent in this new environment, we employed a deep q learning method. To evaluate the performance of our various agents, we swept hyperparameter values, including learning rate, decay rate, and filter size, and compared results across cuts. To the right are images taken from the lunar lander (top) and cart-pole (bottom) environments.

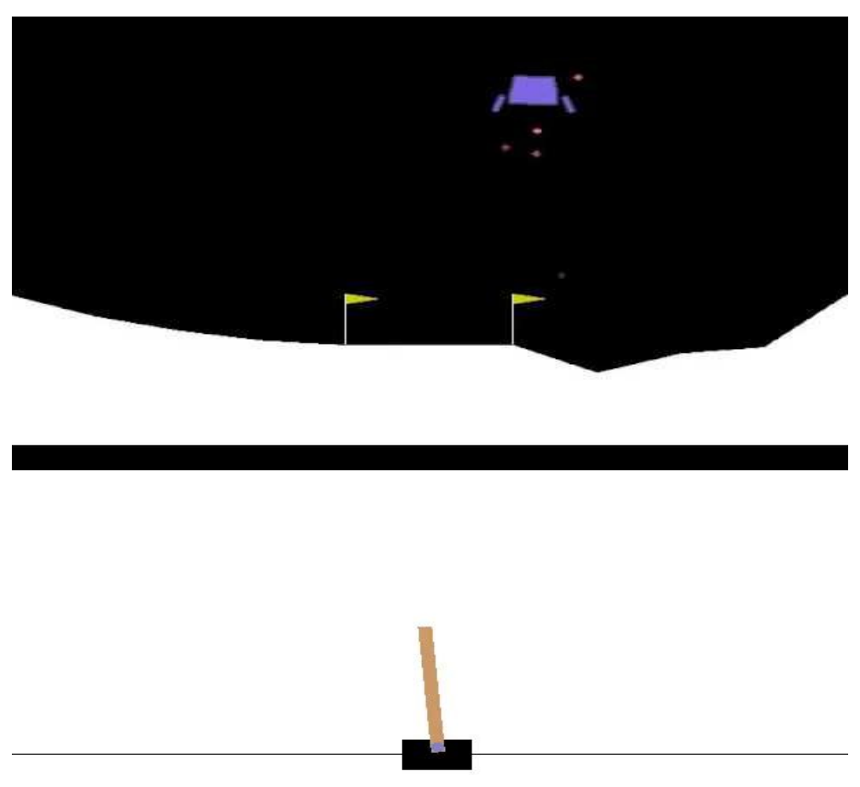
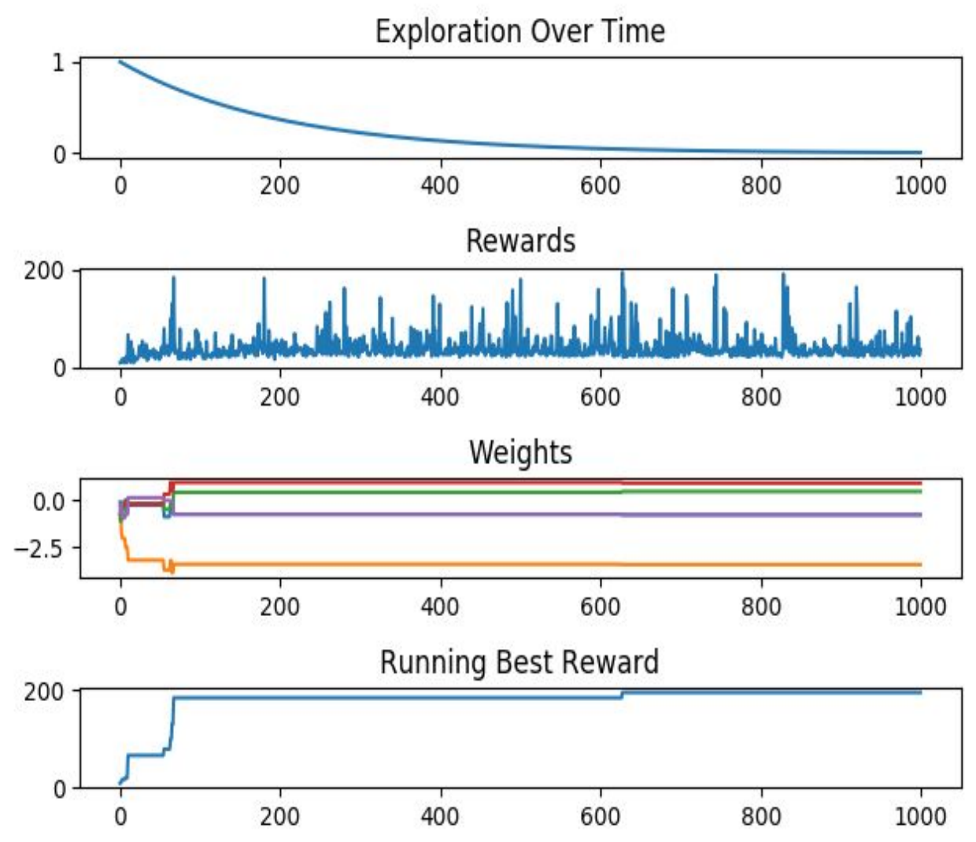


Figure 1 -- screenshots of lunar lander (top) and cartpole (bottom) interfaces

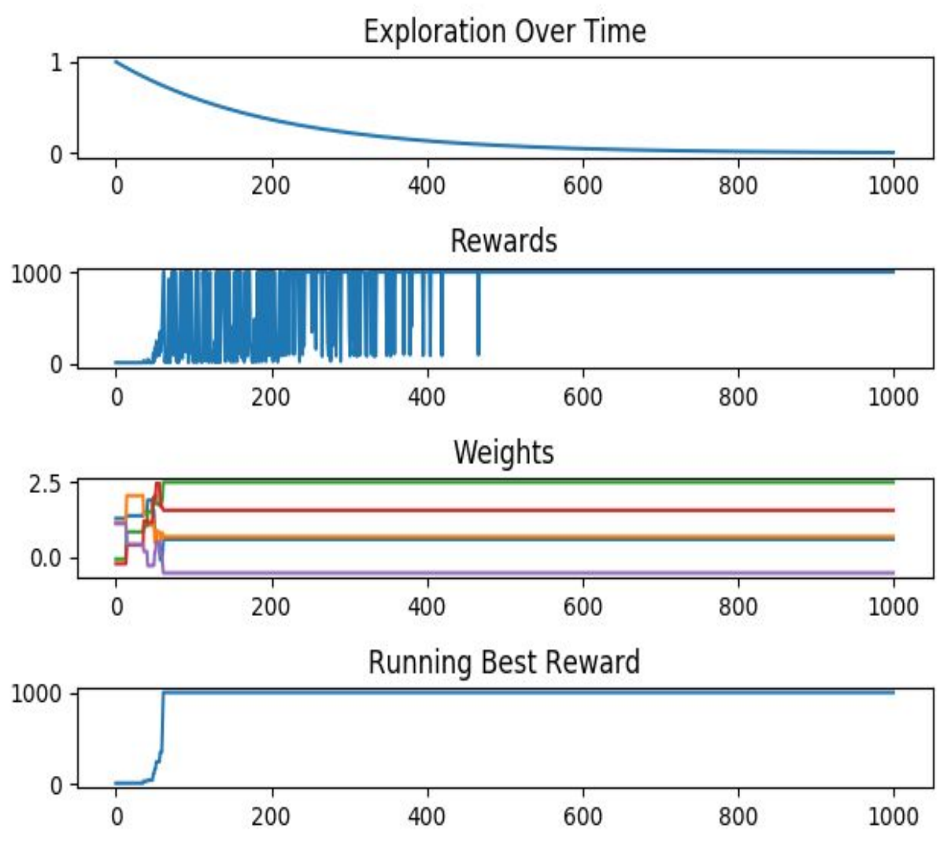
Experiments and Data

Approximate learning on Cartpole

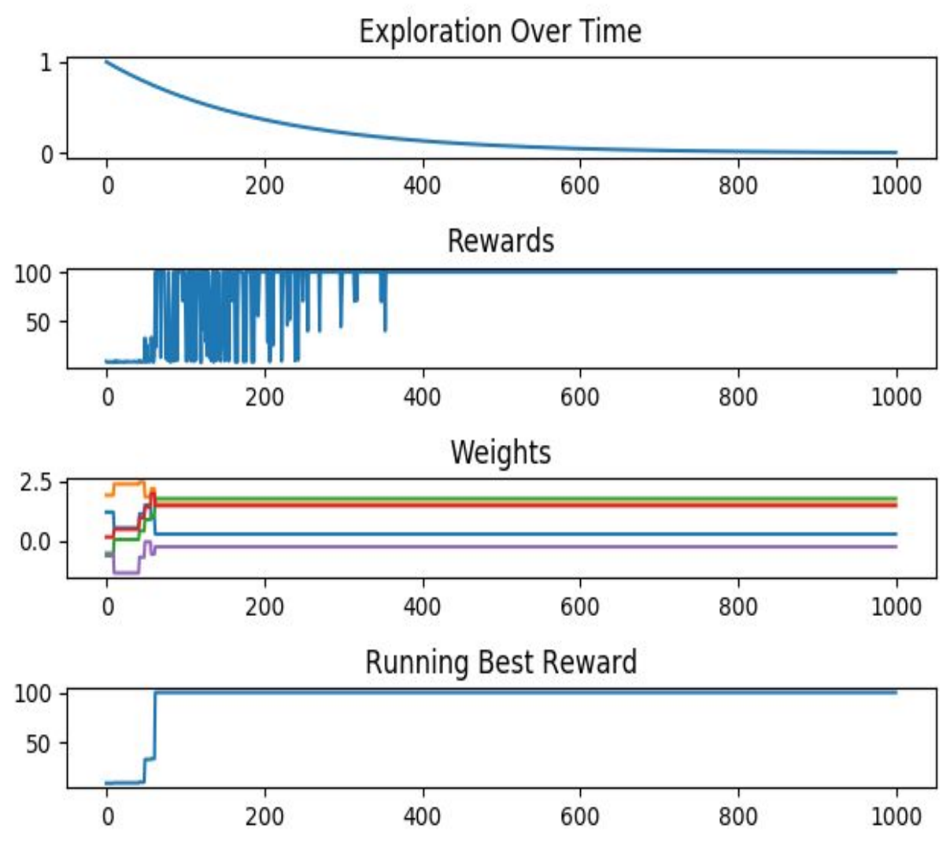
Each of these experiments, the model was trained for 1000 epochs, before being tested on the final configuration for 100 trails. The goal was to average a score of 195.0 or more on this 100 trial test..



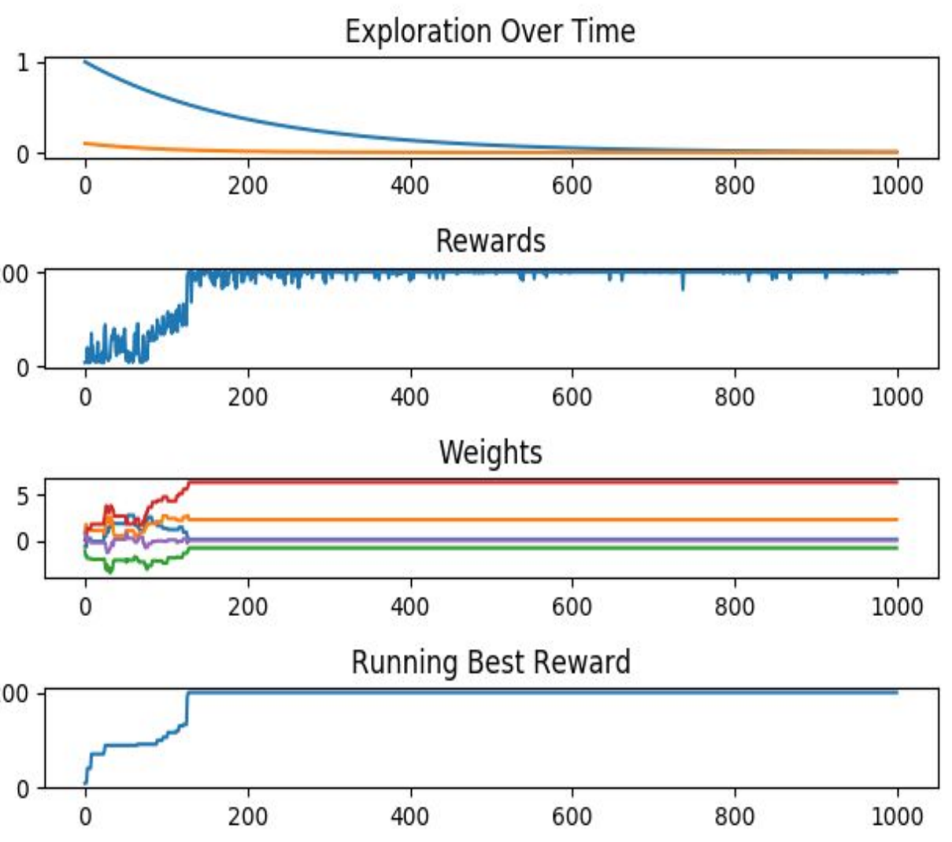
Hill Climbing on Linear Approximation Weights. Neighbors are generated by adding a random vector scaled to an exploration rate, decreasing by 0.5% per epoch. Note the spikes preventing the average from improving. Converged on suboptimal policy after 650 epochs. Average test score 42.9 over 100 trials.



Hill Climbing with increased maximum episode length. Despite the spikes, with a higher ceiling the controller eventually learns to continually balance the pendulum. Converged on suitable policy after 100 epochs.
Average test score 200.0 over 100 trails.

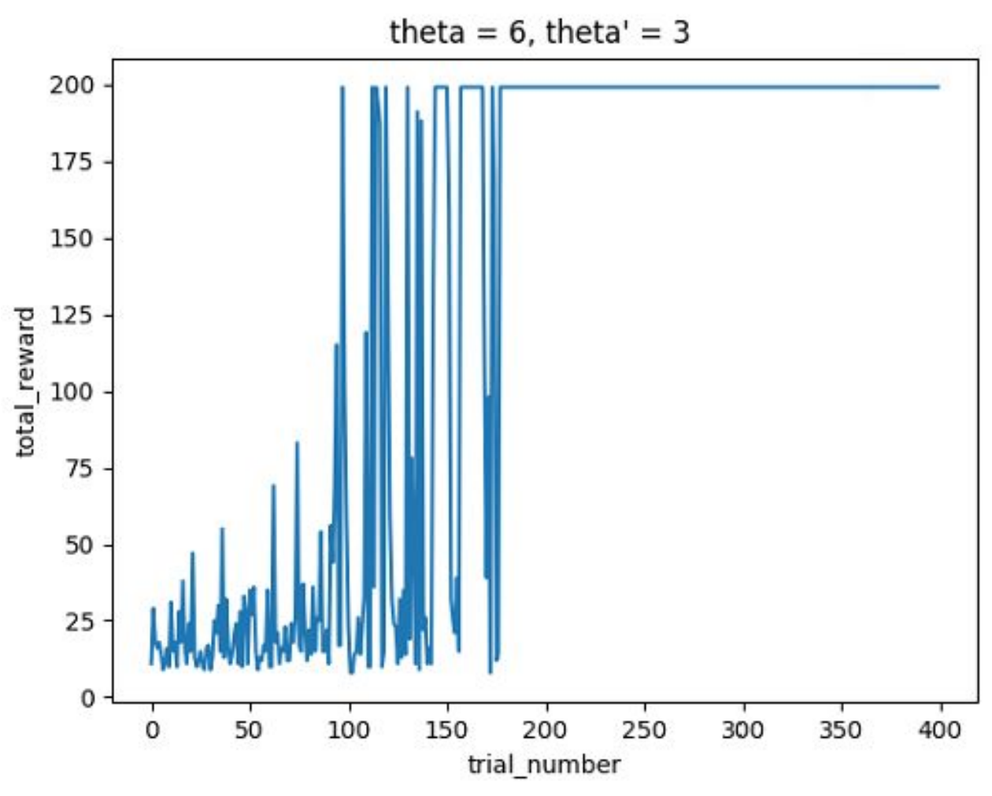


Hill Climbing with averaging filter (over 3 samples). Interestingly, even with the maximum episode length set to 100 during the learning phase (reducing computation time), the controller can result in successful learned weights which can balance the pendulum for longer than the trial period. Converged on suitable policy after 400 epochs with half-length trials. Average test score 200.0 over 100 trials.



Simulated Annealing on Linear Approximation Weights. Much faster convergence on the first try. Converged on acceptable policy in 150 epochs. Average test score 199.5 over 100 trials.

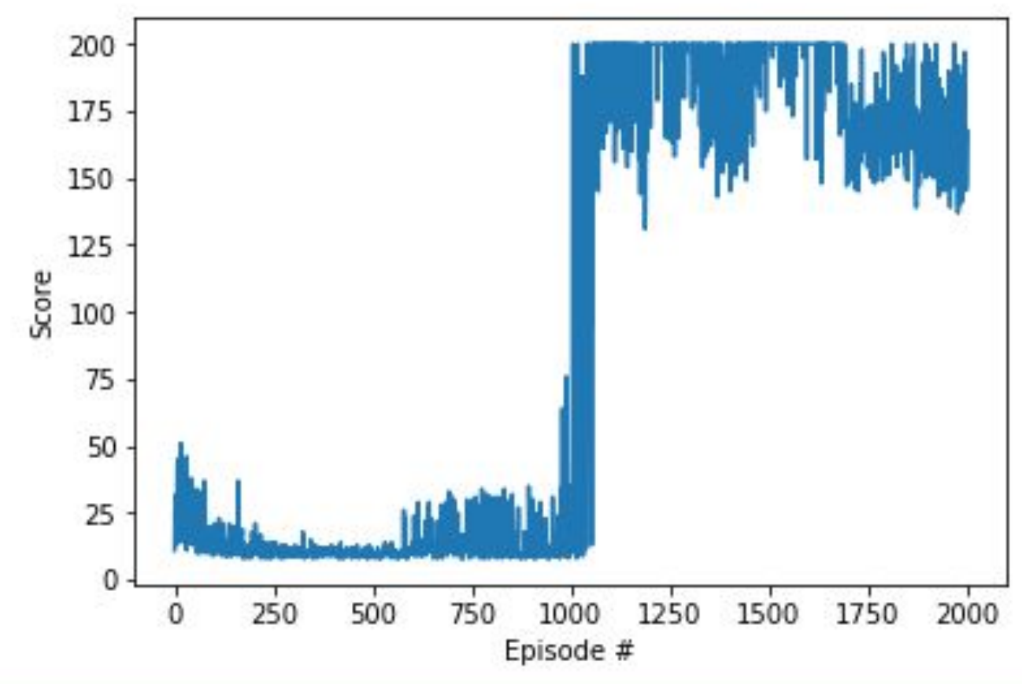
Epsilon greedy tabular learning on Cartpole with discretized state space



With the theta and theta' features broken up into 6 and 3 buckets respectively, we see consistent policy convergence to a successful policy after less than 200 iterations.

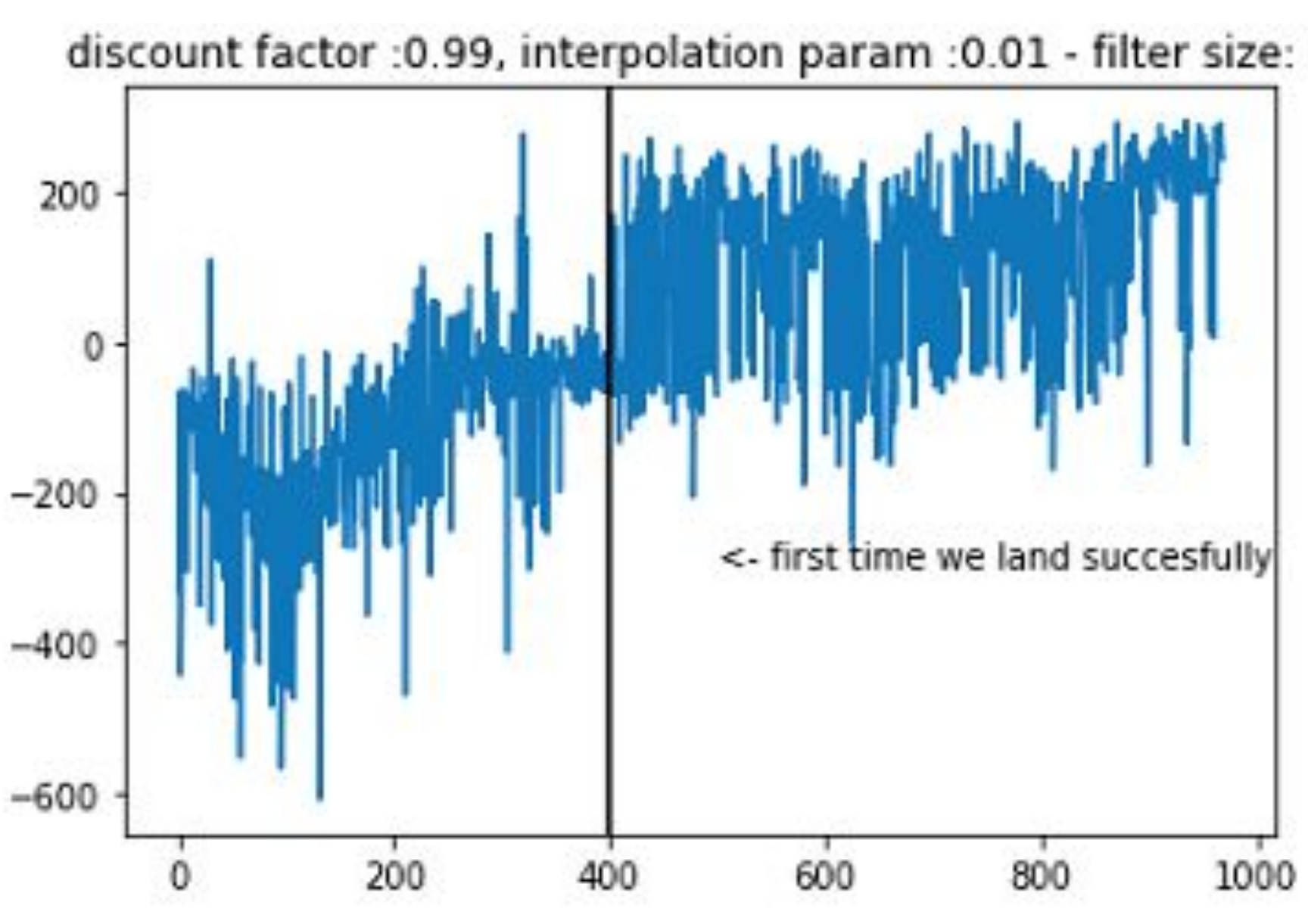
Deep Q Network on Cartpole

```
accuracy:
training   (min:    8.000, max:  200.000, cur:  146.000)
Episode 2000    Average Score: 165.48
Episode 2000    Average Score: 165.48
```

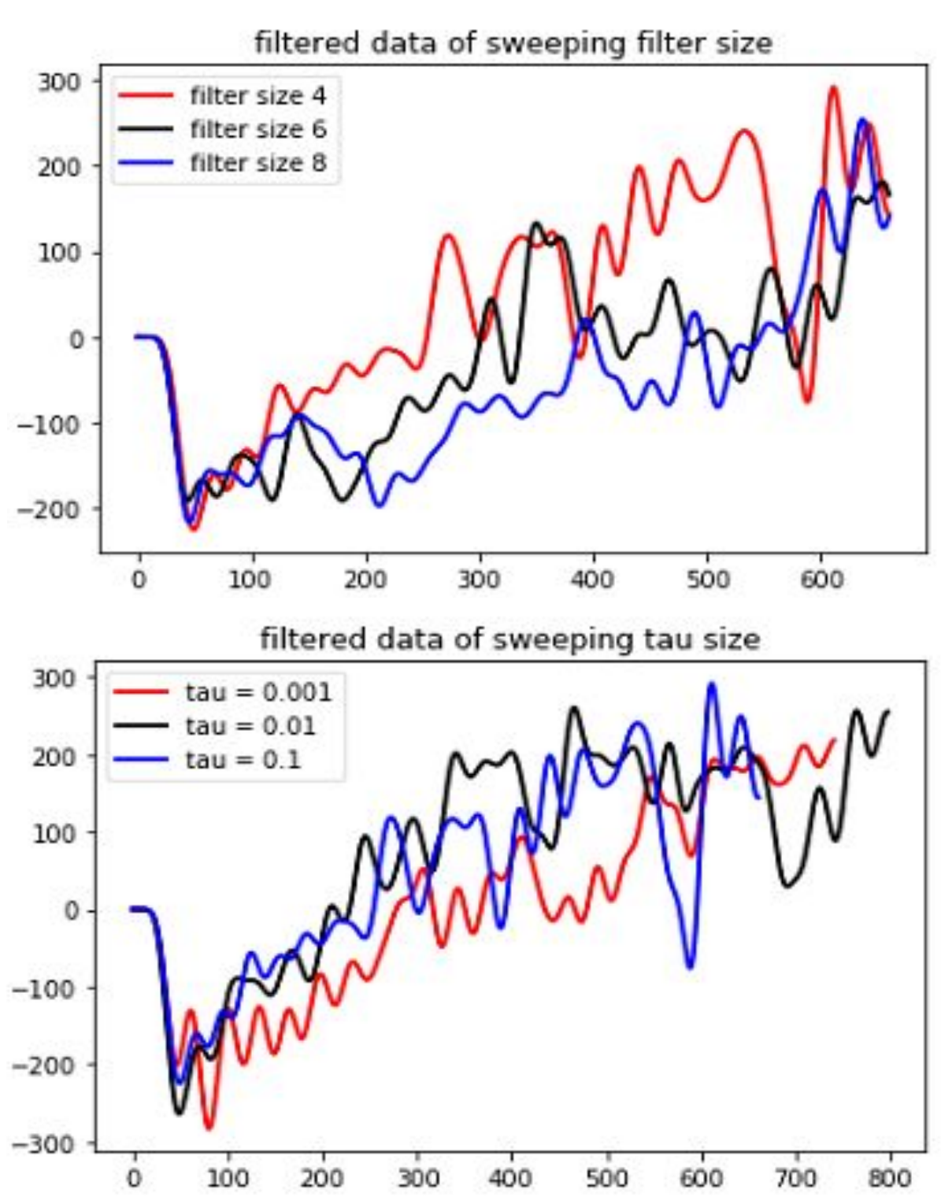


The DQN took over 1000 iterations to converge on the cartpole problem.

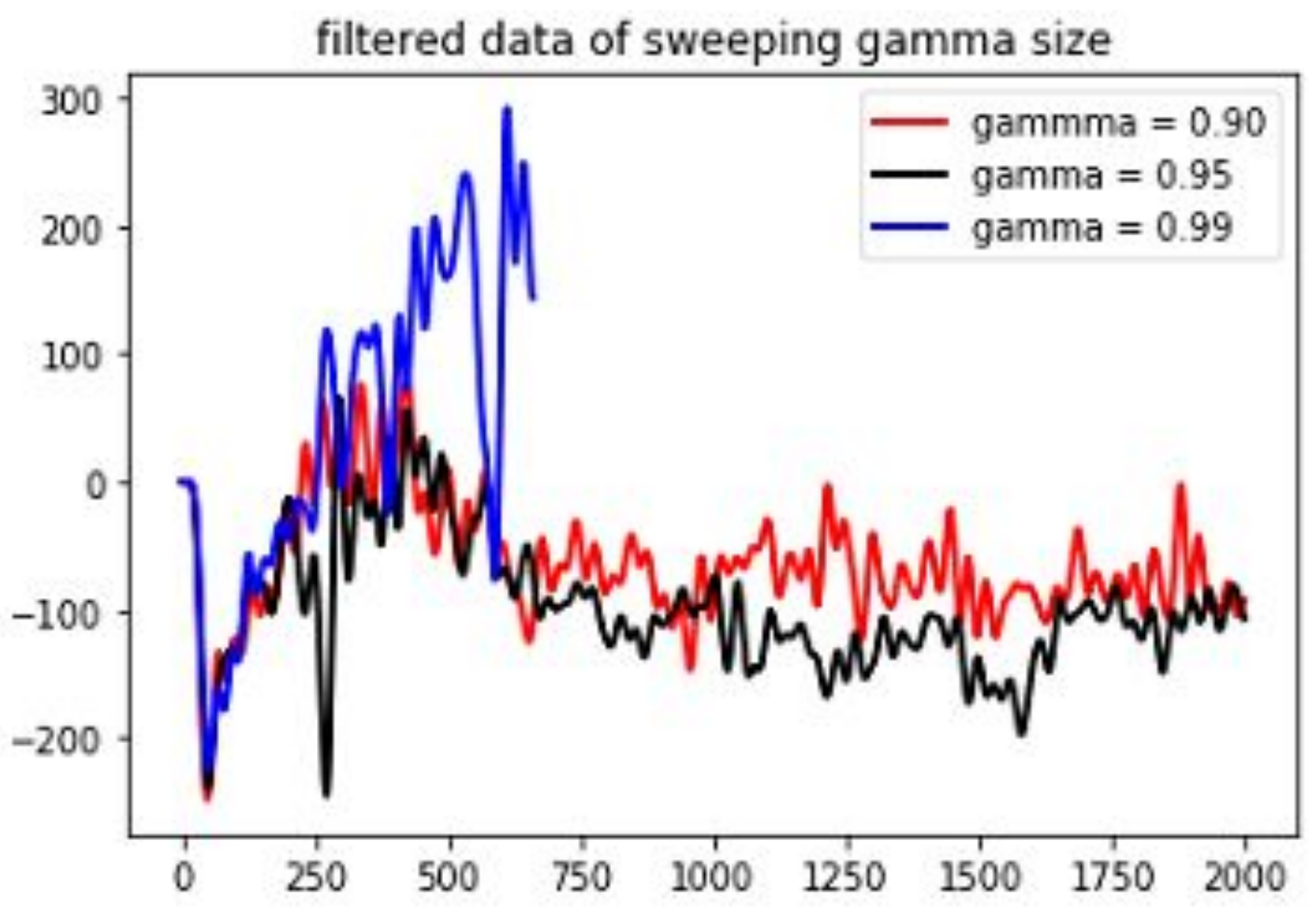
Deep Q Network on Lunar Lander



Here we see a convergence of policy for the lunar lander after 500 iterations with a discount factor of 0.99 and an interpolation parameter of 0.01.



Here we see an analysis of filter size in our DQN. The filter size represents the number of samples we require to update the weights in our neural network. We observe that filter size has a negligible effect on convergence timing across the range explored above.



In this plot we see that a hyperparameter sweep suggests the discount factor to have the greatest effect on policy convergence. Our experiment suggests that gamma > 0.97 is required for convergence, and that gamma closer to 0.99 is optimal. This is consistent with our understanding of bellman updates, because as gamma → 1, we place greater weight on expected future outcomes and thus minimize the effect of local minima.

Future Work

As we continue our research, we plan to conduct further experiments with the DQN framework. Specifically, we plan to modify the architecture of the Neural Network responsible for creating vectors of likely q values associated with states. In doing so, we will expand our analysis to include a deeper emphasis on forms of underlying ML.

Conclusions

Based on our exploration of these learning algorithms of varying complexity, we have seen that though simple algorithms have success with problems of smaller state spaces, even moving to a moderately complex problem can result in issues, necessitating the use of more complex techniques. As the input state vector for the cart-pole inverted pendulum problem has just four elements with magnitudes proportional to the *error* in the system, a linear approximation model using just these state vectors can be used, and learned via local search algorithms to great effect. Performance can be improved via tweaks such as increasing maximum episode lengths to avoid hitting a ceilings and FIR filtering to reduce the effects of spikes. Similarly, tabular q learning with a discretized state space can be implemented to great success when defining only 18 states.

However, when these methods were applied to the lunar lander problem with an eight element state vector, issues with convergence lead us to more complex methods, such as the DQN algorithm, to extract relevant features for the learning process. However, when DQN was applied to the cartpole, convergence occurred, but only after many iterations. This is because the neural network used is quite large, consisting of a first layer of 256 and a second layer of 16 fully connected, resulting in the exploration of a number of local minima before convergence. This progression demonstrates the need to match solution complexity to problem complexity. However, most practical problems are more complex than these and can involve noisy measurements that necessitate advanced processing.

References

[1] Sutton, R. & Barto, A. *Reinforcement Learning: An Introduction*(MIT Press, 1998)
[2] Taitler, Ayal, and Nahum Shimkin. "Learning Control for Air Hockey Striking Using Deep Reinforcement Learning." *2017 International Conference on Control, Artificial Intelligence, Robotics & Optimization (ICCAIRO)*, 2017, doi:10.1109/iccairo.2017.14.
[3] Mnih, Volodymyr, et al. "Human-Level Control through Deep Reinforcement Learning." *Nature*, vol. 518, no. 7540, 2015, pp. 529–533., doi:10.1038/nature14236.
[4] RUSSELL, STUART NORVIG PETER. *ARTIFICIAL INTELLIGENCE: a Modern Approach*. PEARSON, 2018.
[5] Simonini, Thomas. "An Introduction to Deep Q-Learning: Let's Play Doom." *Medium*, 2018, medium.freecodecamp.org.