CMPSCI 687 Homework 4

Due November 25, 2018, 11:55pm Eastern Time

Instructions: This homework assignment consists of a written portion and a programming portion. Collaboration is not allowed on any part of this assignment. Submissions must be typed (hand written and scanned submissions will not be accepted). You must use LaTeX. The assignment should be submitted on Moodle as a .zip (.gz, .tar.gz, etc.) file containing your answers in a .pdf file and a folder with your source code. Include with your source code instructions for how to run your code. You may not use any reinforcement learning or machine learning specific libraries in your code (you may use libraries like C++ Eigen and numpy though). If you are unsure whether you can use a library, ask on Piazza. If you submit by October 14, you will not lose any credit. The automated system will not accept assignments after 11:55pm on October 25.

- Solution 1:
- E-greedy Sarsa GridWorld Hyperparameters
- $lr = 0.01$ $epsilon = 0.1$

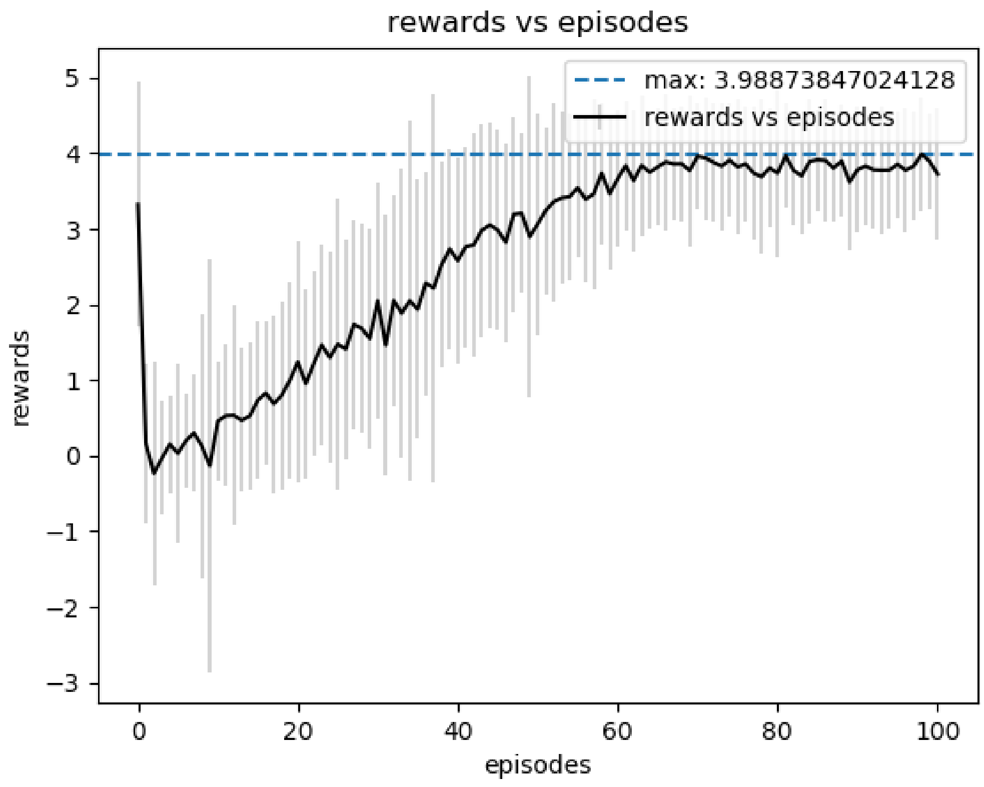


Figure 1: E-greedy Sarsa GridWorld

- E-greedy QLearning GridWorld Hyperparameters
- $lr = 0.000316$ $epsilon = 0.05, 0.01$

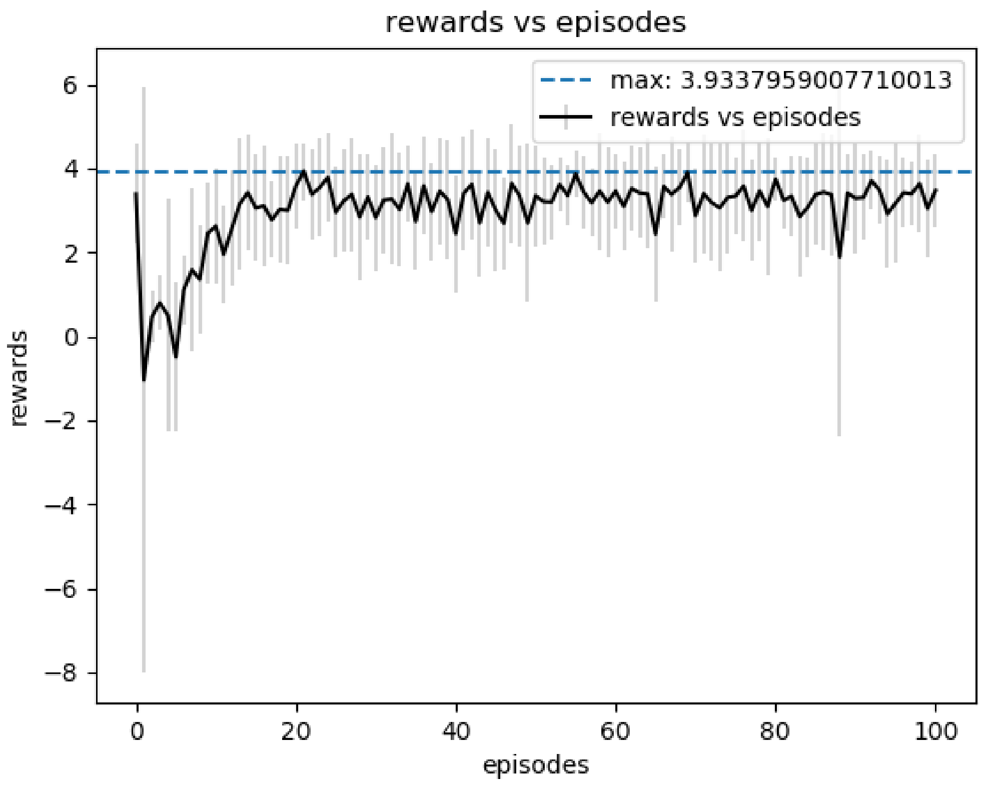


Figure 2: E-greedy QLearning GridWorld

- E-greedy Sarsa CartPole Hyperparameters
- $lr = 0.001$ $order = 5$ $epsilon = 0.1$

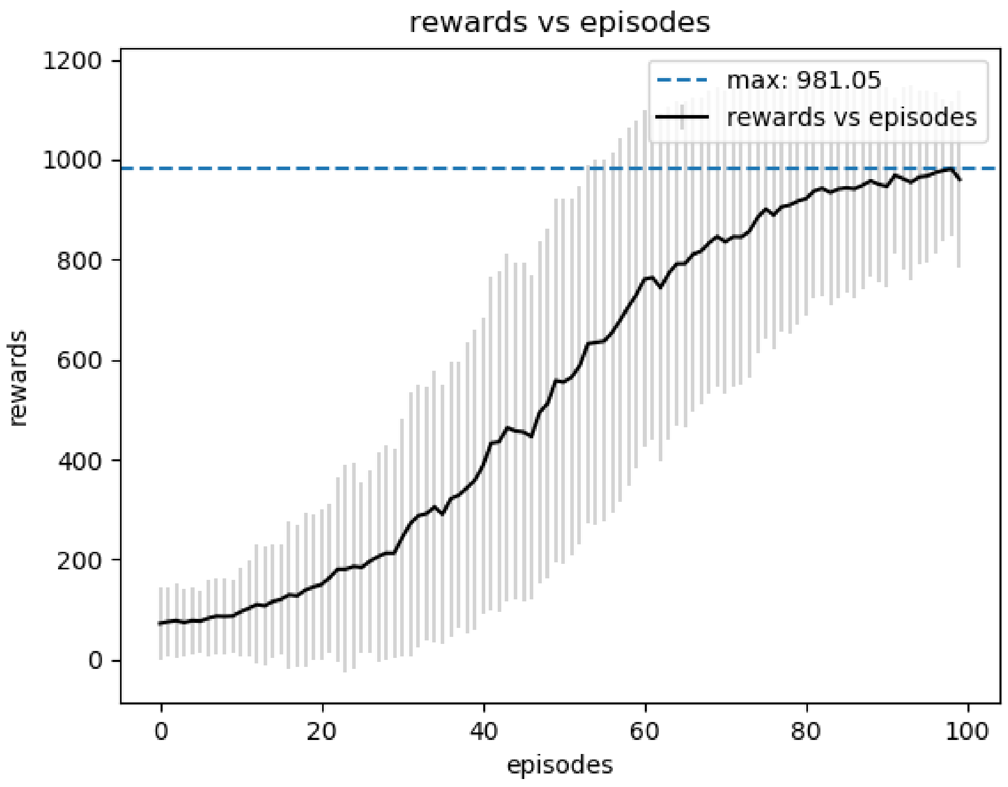


Figure 3: E-greedy Sarsa CartPole

- E-greedy QLearning Cartpole Hyperparameters
- $lr = 0.001$ $order = 5$ $epsilon = 0.05$

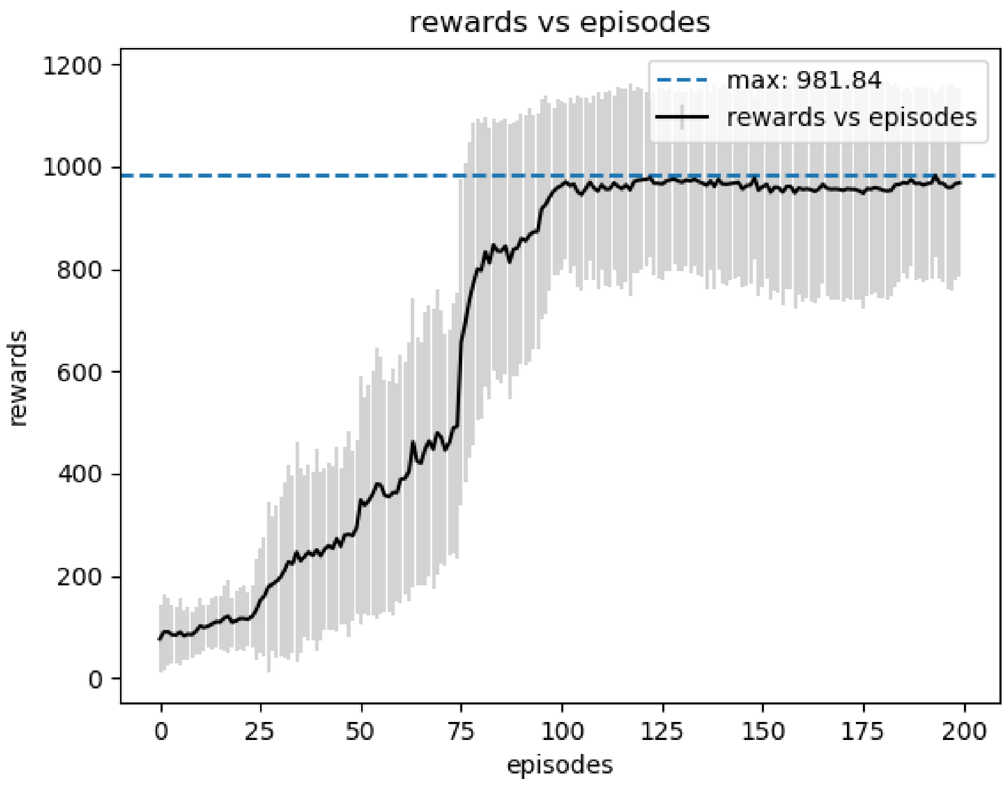


Figure 4: E-greedy QLearning Cartpole

- Observations:
  Q Learning in the case of gridworld required smaller learning rate to converge as compared to sarsa for gridworld.
- Solution 2:
- E-greedy Sarsa Polynomial Cartpole Hyperparameters
- $lr = 0.01$ $order = 8$ $epsilon = 1.0$ decaying

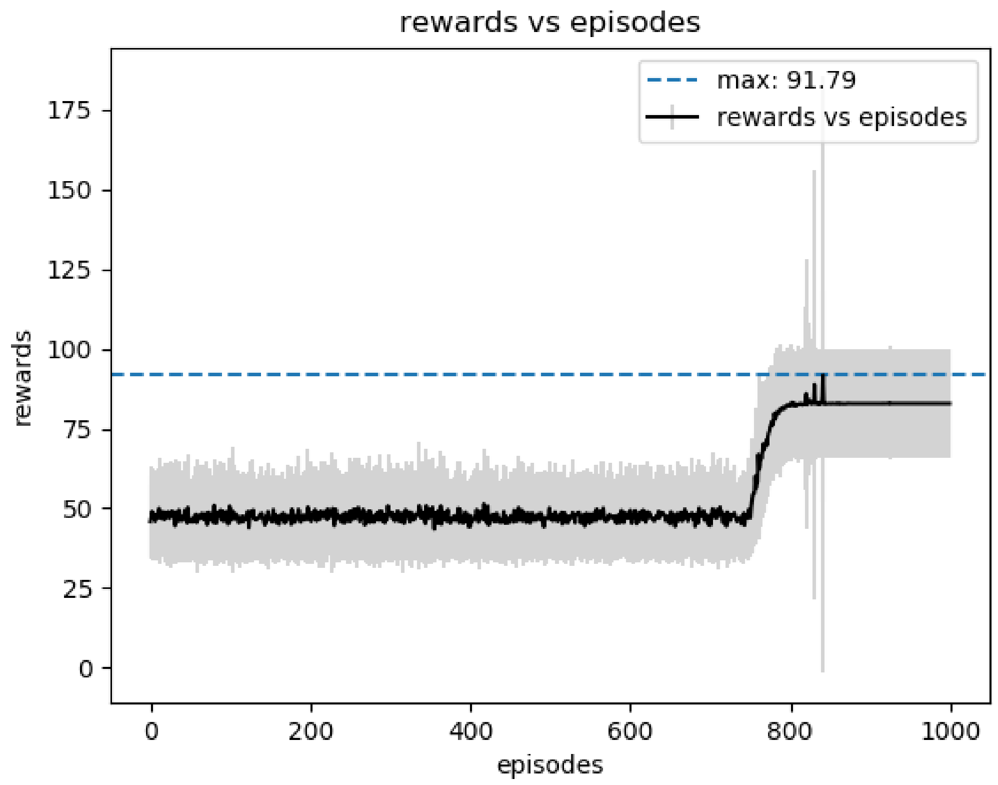


Figure 5: E-greedy Sarsa Polynomial Cartpole

- E-greedy QLearning Polynomial Cartpole Hyperparameters
- $lr = 0.01$ $order = 8, 9$ $epsilon = 1.0$ decaying

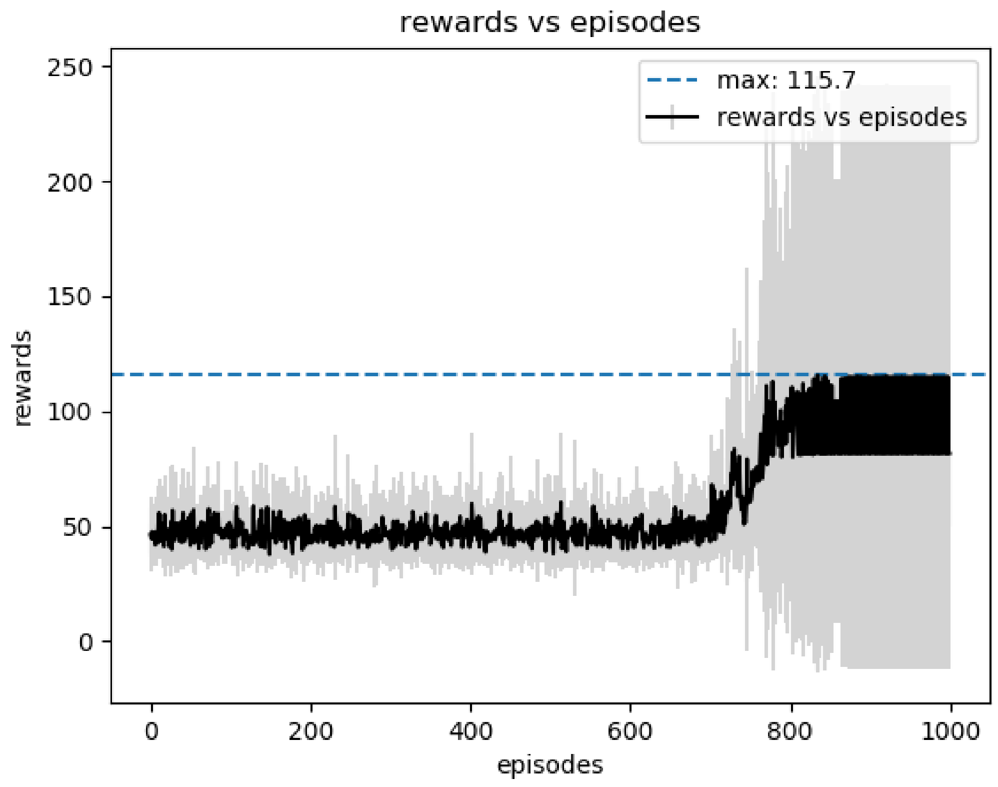


Figure 6: E-greedy QLearning Polynomial

- E-greedy Sarsa Tile CartPole Hyperparameters
- $lr = 0.03$ $tilegrid = 10X10X10X10$ $tiles = 4096$ $Tilings = 32$ $epsilon = 0.2$

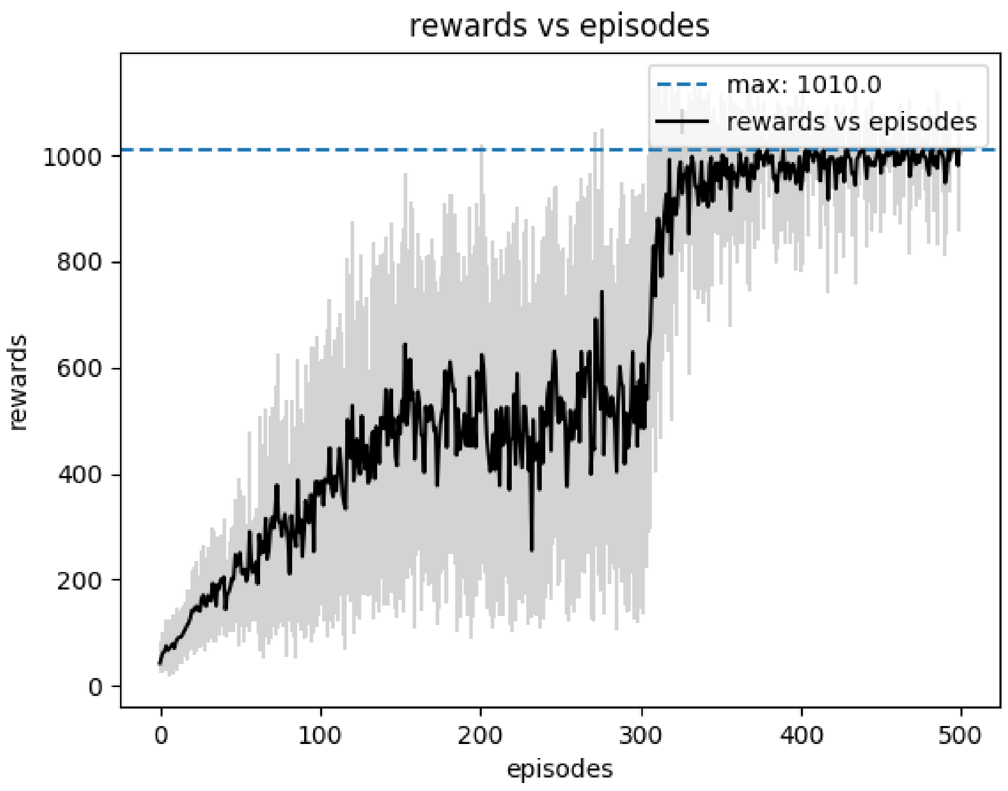


Figure 7: E-greedy Sarsa Tile CartPole

- E-greedy QLearning Tile Cartpole Hyperparameters
- $lr = 0.02$ $tilegrid = 10X10X10X10$ $tiles = 4096$ $Tilings = 32$ $epsilon = 0.1$

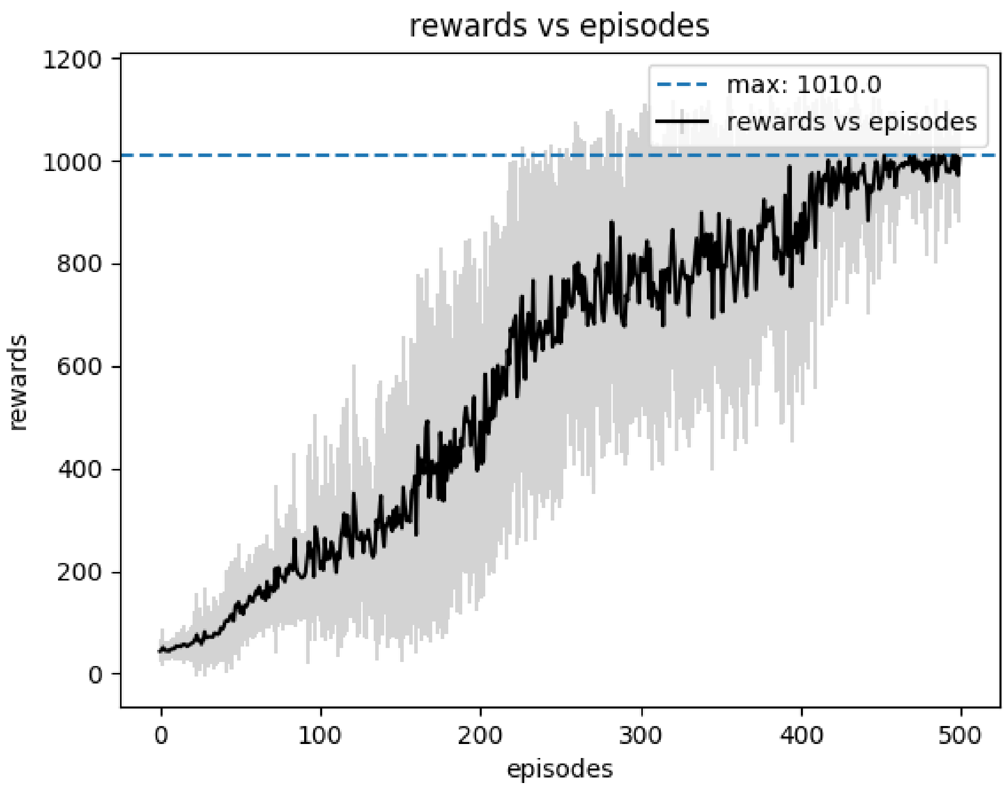


Figure 8: E-greedy QLearning Tile Cartpole

- Solution 3:
- Comparison between CE. Sarsa and Q-Learning

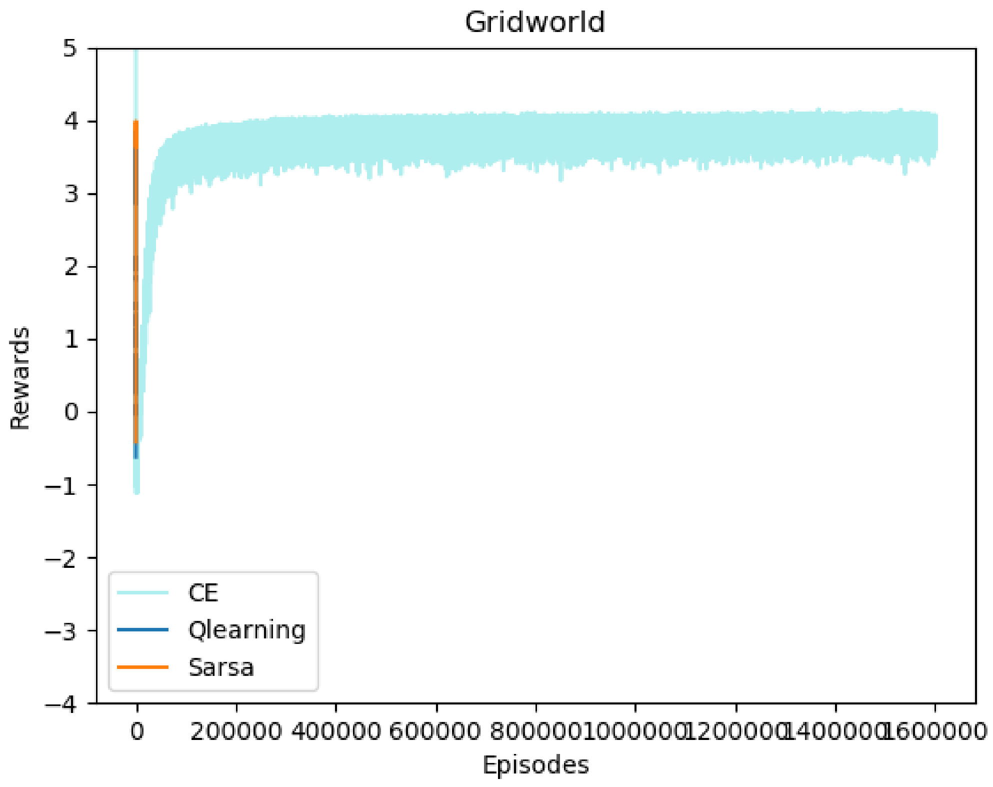


Figure 9: Gridworld CE, Sarsa and Qlearning

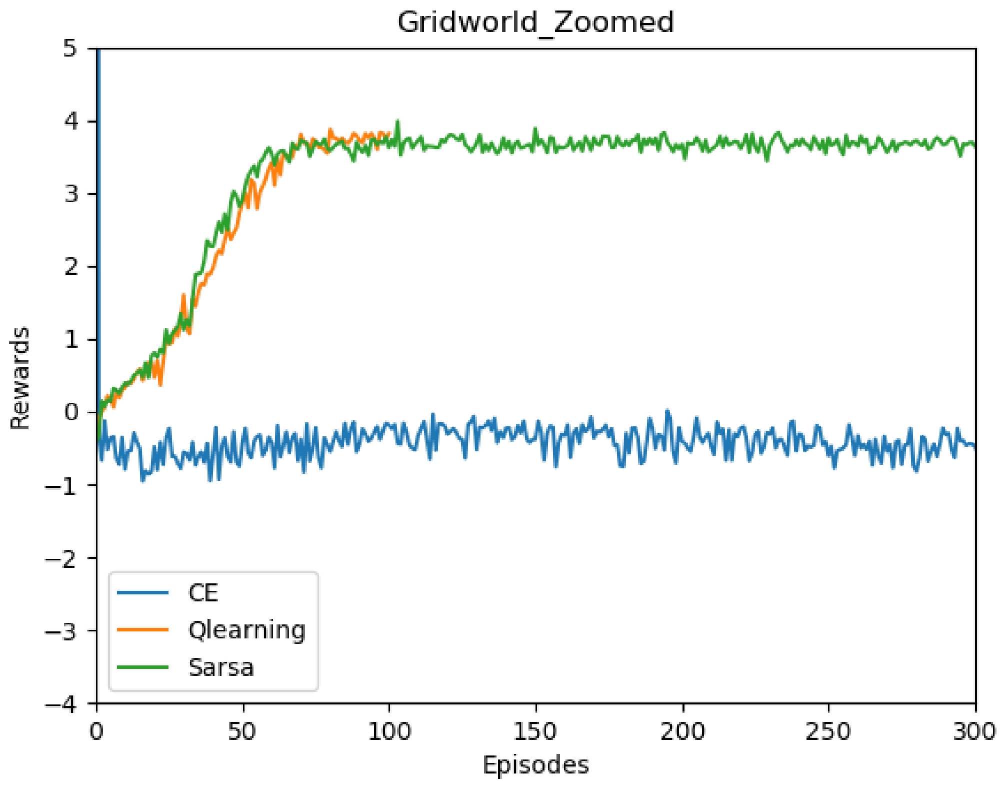


Figure 10: Gridworld CE, Sarsa and Qlearning Zoomed

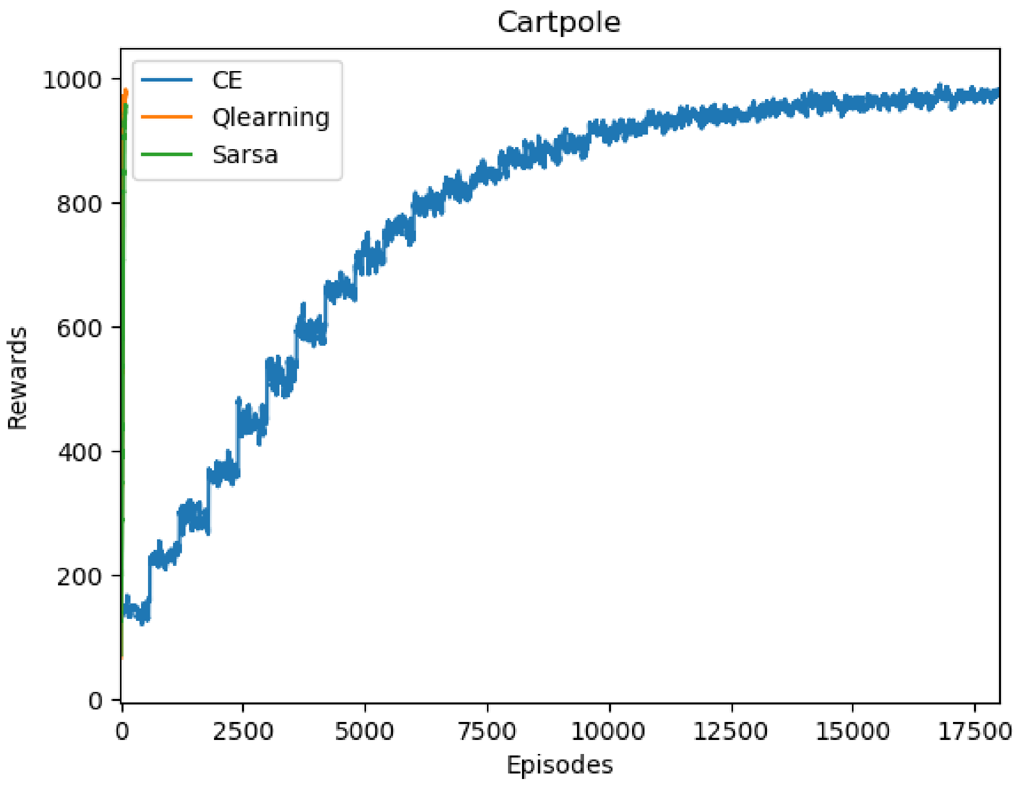
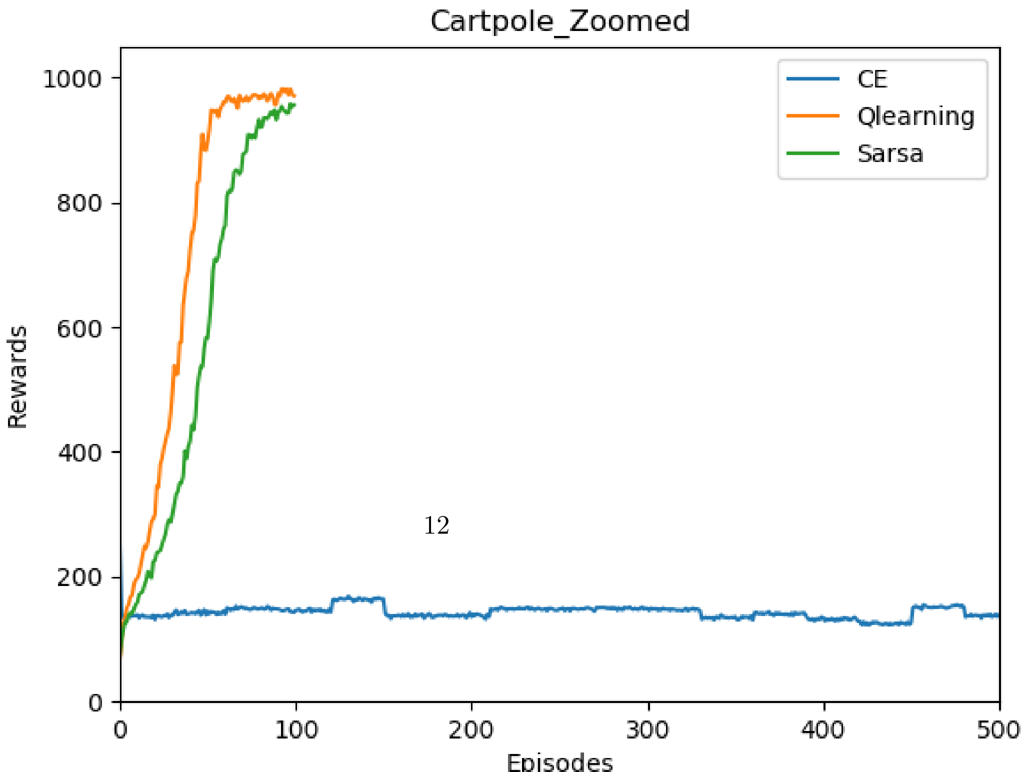
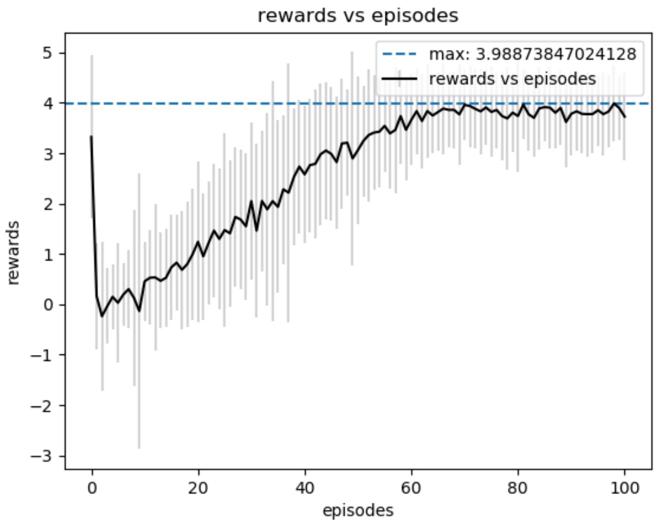


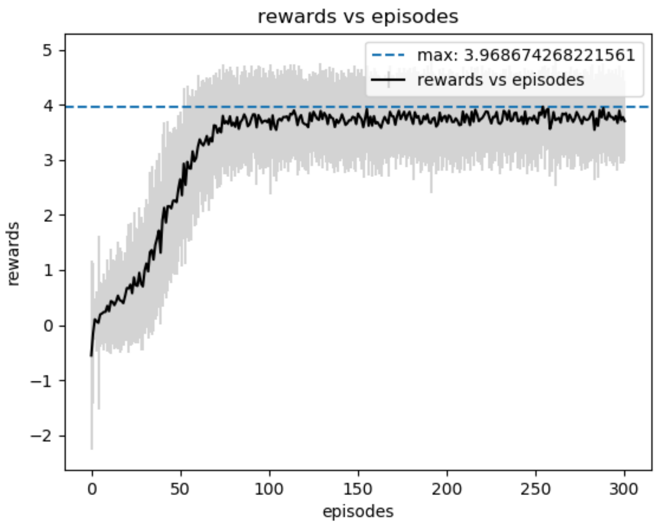
Figure 11: Cartpole CE, Sarsa and Qlearning



- Solution 4:
- Softmax Sarsa GridWorld Hyperparameters
- $lr = 5$ $sigma = 0.1$
- $lr = 10$ $epsilon = 0.0316$



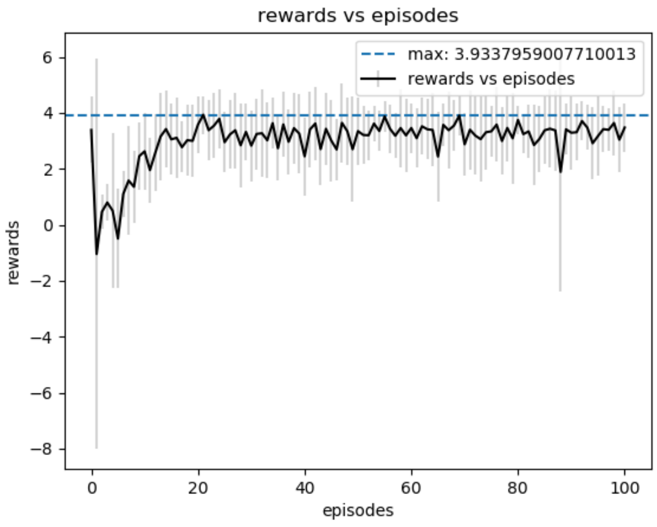
(a) E-greedy Sarsa Gridworld



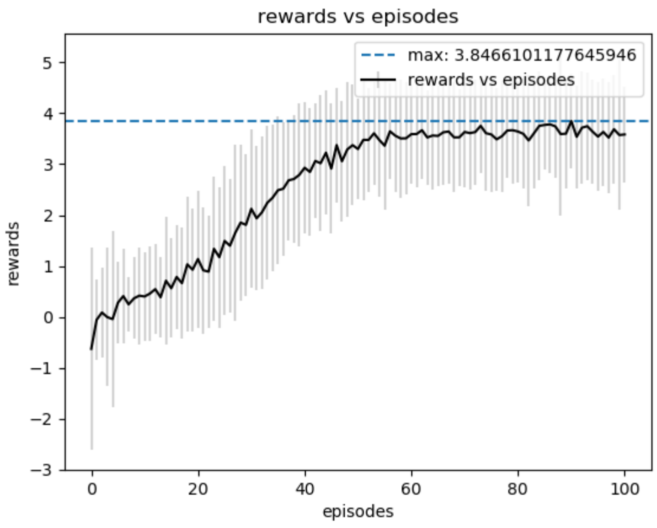
(b) Softmax Sarsa GridWorld

Figure 13: Softmax Sarsa GridWorld vs E-greedy Sarsa Gridworld

- Softmax QLearning GridWorld Hyperparameters
- $lr = 10, 15$ $sigma = 0.1$



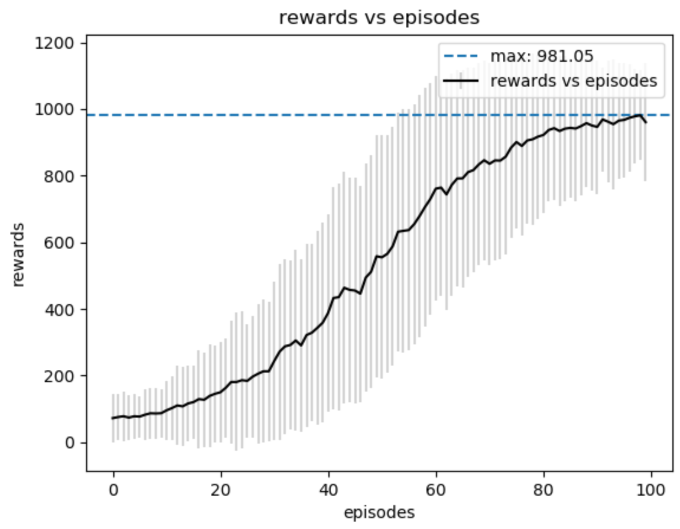
(a) E-greedy QLearning Gridworld



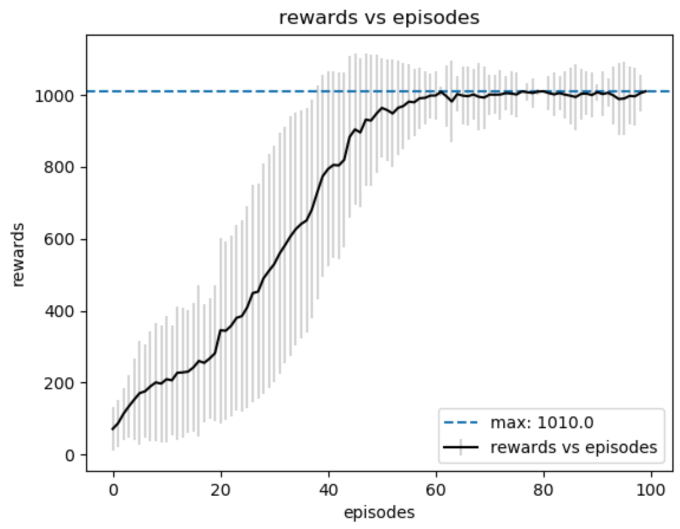
(b) Softmax QLearning GridWorld

Figure 14: Softmax QLearning GridWorld vs E-greedy QLearning Gridworld

- Softmax Sarsa CartPole Hyperparameters
- $lr = 0.001$ $order = 5$ $sigma = 0.3$



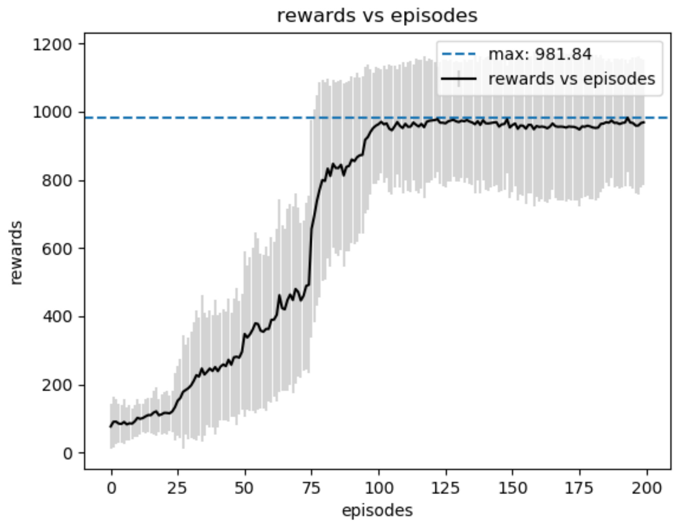
(a) E-greedy Sarsa Cartpole



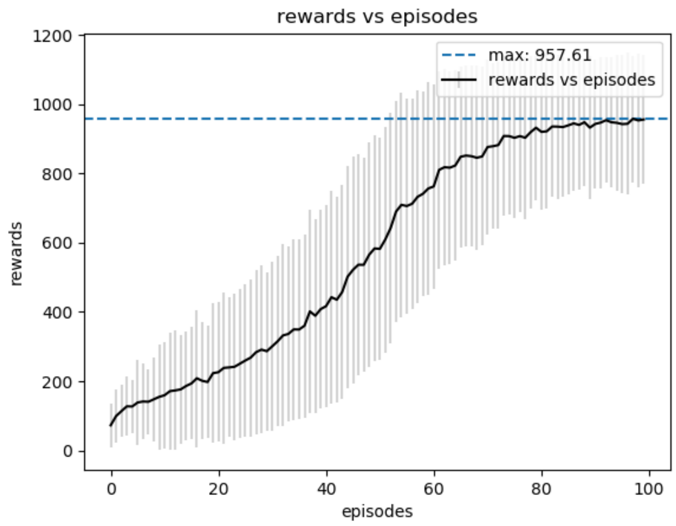
(b) Softmax Sarsa Cartpole

Figure 15: Softmax Sarsa Cartpole vs E-greedy Sarsa Cartpole

- Softmax QLearning Cartpole Hyperparameters
- $lr = 5$ $order = 0.001$ $sigma = 0.3$



(a) E-greedy QLearning Cartpole



(b) Softmax QLearning Cartpole

Figure 16: Softmax QLearning Cartpole vs E-greedy QLearning Cartpole

- Solution 5:
- Softmax Sarsa Mountain Car Hyperparameters
- $lr = 0.001778$ $order = 5$ $epsilon = 0.0001$

- $lr = 0.00316$ $order = 3$ $epsilon = 0.0001$
- $lr = 0.001778$ $order = 2$ $epsilon = 0.0007$

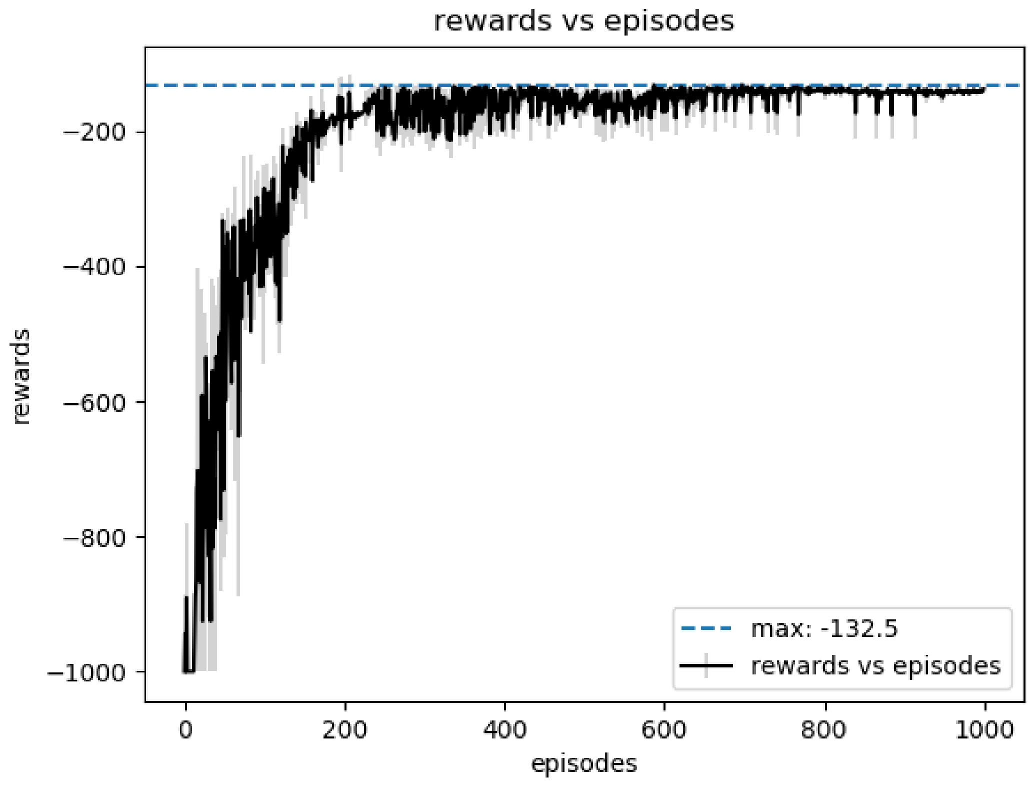


Figure 17: Softmax Sarsa Mountain Car

- Softmax QLearning Mountain Car Hyperparameters
- $lr = 0.00316$ $order = 2$ $epsilon = 0.006$
- $lr = 0.00316$ $order = 2$ $epsilon = 0.0007$

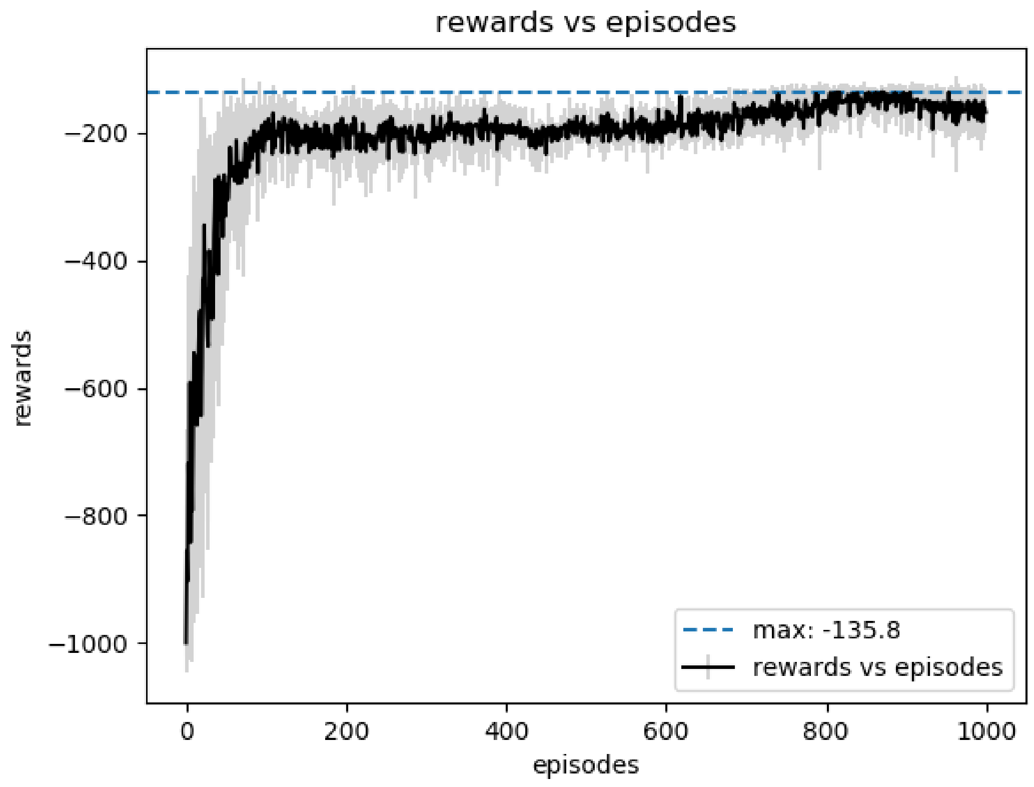


Figure 18: Softmax QLearning Mountain Car

Observations for Problem 2: It was very difficult to get convergence for polynomial, radial basis as compared to finding hyperparameters for fourier basis.In the case of polynomial and radial, I tried setting exploration parameter high and gradually decaying the exploration parameter or/and learning rate,I also tried tuning the learning rate and order over a large range of values. (10^{-1} to 10^{-10} for order and 0.1 to 0.000001 for learning rate). Doing so, resulted in a reward of around 100 for polynomial and around 300 for radial. I feel the reason why this could have happened is that optimal policy in cartpole in fairly simple and so ideally, should be the value function. The value function in this case should be representable using a linear basis which indicates that the weights vector for polynomial may need to be sparse. It could be possible that since $phi(s)$ has mostly non zeros values due to the properties of polynomial basis, it could never make the weight vector completely sparse and hence it could never converge to optimal policy. Whereas in the case of fourier, exploring orders in the range 1 to 5 and learning rate in the range 0.1 to 0.0001 gave decent results very quickly. I also tried using Neural network with replay buffer, 2 hidden layers with different combinations of no of nodes and decaying epsilon. Although the rewards would increase over time and sometimes would reach 1010, the returns observed had a lot of variance in the returns. Also the rewards would drop immediately to 29-30 after reaching some high value which makes it seem like the network immediately forgets what it learnt due to some kind of over-fitting and instead learns a policy which performs worse than random policy. Convergence for non linear function approximator is not guaranteed under any conditions so it is expected that the Neural network will be more difficult to tune than a linear function approximator. I was able to get good convergence with linear function approximator via tile coding as it seems it can better represent the value function of cartpole as compared to polynomial. I used 10X10X10X10 tiling and number of tilings is 32.

- The reason why tile coding worked is because tile coding is similar to using different tabular function approximators with different weight vectors for each tabular function approximator. Since convergence is guaranteed in tabular function approximator under standard assumptions, tile coding being similar should be able to achieve convergence too.

- In Q Learning and Sarsa we are parametrising the value function instead of policy. Q Learning and Sarsa use TD for update wherein convergence to v_{Π} is guaranteed only under standard assumptions like decaying step size and also there should exist a weight vector w such that $v_w = v^{\pi}$. So if Q Learning and Sarsa has to result in value function of an optimal policy (v^{π^*}) then, there should exist a weight vector w^* such that $v_{w^*} = v^{\pi^*}$. It seems like linear function approximator using Tile coding and fourier basis could best represent nearly optimal value function for cartpole among other basis.

  Second, since they are (stochastic) gradient algorithms, policy gradient algorithms tend to have convergence guarantees when value-function based methods do not (e.g., using non-linear policy parameterizations is not a problem for policy gradient methods). Furthermore, whereas value-based methods approximately optimize an objective (minimizing some notion of expected Bellman error), this objective is merely a surrogate for the primary objective: maximizing expected return. Policy gradient methods take a more direct approach and directly maximize the expected return.

- **Observations for problem 3:** In cross entropy, policy is parametrised and learnt directly whereas in Qlearning and Sarsa policy, value function is parameterised which approximately optimizes an objective which minimizes Bellman error and indirectly maximizes expected returns hence improving policy. Hence it is quite likely that cross entropy can perform better than Qlearning and Sarsa. In cases where policy is known and is fairly simple as compared to value function in terms of modelling, cross entropy can be used.

- Cross entropy cannot be affected by any kind of bias whereas target in QLearning and Sarsa could be biased which can affect convergence.

- In cases where rewards are stochastic and sparse and received only at the end of the episode, policy/value function can oscillate between extremities due to variance whereas CE is highly robust as it uses mean rewards over several episodes for its update.

- Policy for MDPs with continuous action spaces can be modeled in CE using any regression which would represent a policy and spit out next action. But this is computationally expensive and almost impossible in Q-Learning as stated by Baird and Klopf.

- Lastly, the reason why CE is not so popular as compared to Qlearning and sarsa is because in QLearning/Sarsa, value function is updated in every time step which results in faster convergence and a lot less episodes than CE which requires some hundreds of episodes to be run before an update.

- Observations from experiments: In the case of gridworld, Q Learning with fourier basis and softmax action selection gave slightly lower returns (3.86) as compared to CE (3.95) whereas Sarsa with fourier basis and softmax action selection gave results similar to CE. Likewise, Q learning and Sarsa resulted in slightly better mean returns than CE . It was more difficult to tune CE since it would take many more episodes for it to converge if we increase exploration which made it difficult to try out wide range of hyperparameters.

  Observations for problem 4: Softmax results in much higher returns in cartpole and almost similar (maybe slightly better) returns in gridworld over 100 episodes.

- Explanation: E-greedy exploration uses $0 \leq \epsilon \leq 1$ as parameter of exploration to decide which action to perform using $Qt(s_t, a)$. The agent chooses the action with the highest Q-value in the current state with probability 1ϵ and a random action otherwise. A larger value for ϵ means more exploration actions are selected by the agent. Randomness is necessary for an agent navigating through a stochastic environment to learn the optimal policy.

- In e-greedy exploration, the exploration action is selected uniform randomly from the set of possible actions. Therefore, it is as likely to choose the worst appearing action as it is to choose the second-best appearing action if an exploration action is selected and hence may delay convergence. That is why Boltzmann or softmax exploration uses the Boltzmann distribution function to assign a probability $\pi(s_t, a)$ to the actions in order to create a graded function of estimated value.

- $\pi(s_t, a) = e^{Q_t(s_t,a)/T} / \sum_{i=1}^{\infty} e^{Q_t(s_t,a_i)/T}$ When temperature parameter T = 0 the agent does not explore at all, and when T the agent selects random actions. Using softmax exploration with intermediate values for T, the agent still most likely selects the best action, but other actions are ranked instead of randomly chosen.

- Also, e-greedy can lead to choosing non optimal actions which may have highest Q value in the initial phase of learning with a much higher probability than that assigned to a non optimal action with the highest Q value by Softmax . Hence Softmax will always give better results than e-greedy.