CS885 Assignment 1

Angelo Rajendram

September 2022

1 Part 1 Write Up

Increasing partial policy evaluation iterations results in fewer policy improvement steps for modified policy iteration. This occurs because more policy evaluation iterations provide a better estimate of the current policy's value function, on which the policy improvement step relies upon.

It is worth noting, however, that beyond 5 policy evaluation iterations there is no further decrease in the final number of policy improvement steps in modified policy iteration. This is because after the value function for a given policy has converged sufficiently, there is no further advantage achieved by obtaining a more accurate value estimate.

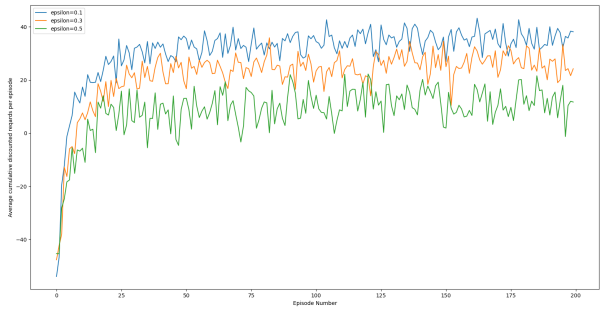
Ideally, partial evaluation iterations should be close to the minimum required to minimize the number of policy improvement steps. Modified policy iteration exhibits computational complexity in-between that of Value iteration (low complexity) and Policy iteration (high complexity) per step, while also exhibiting a convergence rate in-between them as well (Policy iteration with fast convergence and Value iteration with slow convergence). Modified policy iteration represents a compromise between the benefits and disadvantages of the other two methods.

In practice, and as demonstrated by this exercise, modified policy iteration results in a similar or slightly greater number of policy improvement steps as policy iteration, while avoiding the high computational costs of perfect policy evaluation.

- Value Iteration:
  - Each Iteration: $O(|S|^2|A|)$
  - Many Iterations: Linear Convergence
- Policy Iteration:
  - Each Iteration: $O(|S|^3 + |S|^2|A|)$
  - Few Iterations: Linear-quadratic convergence
- Modified Policy Iteration:
  - Each Iteration: $O(k|S|^2 + |S|^2|A|)$
  - Few Iterations: Linear-quadratic convergence

2 Part 2 Write Up

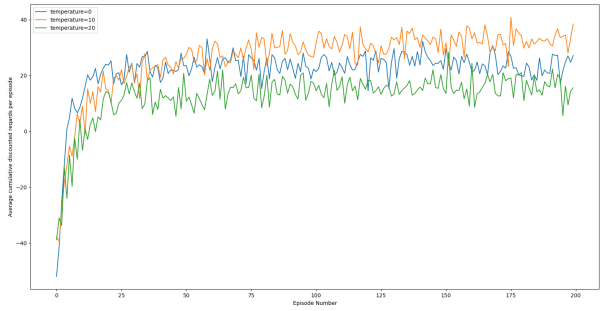
2.1 ϵ-greedy Exploration



For ϵ-greedy exploration the highest average cumulative rewards are achieved with an ϵ of 0.1. This corresponds to an exploration probability of 10%. The choices the agent makes upon learning improved policies will deviate from those learned policies more frequently during an episode as the exploration probability increases. This reduces the cumulative reward achieved relative to that which could have been achieved by following the learned policy strictly.

Nevertheless, eliminating the exploration probability altogether will result in an agent that avoids states it hasn't explored if that state's current value estimate happens to be lower than that of already explored states. This would result in learning a sub-optimal policy.

2.2 Boltzmann Exploration



Similar to ϵ-greedy exploration, the agent displays a greater tendency to explore as the temperature increases. As the temperature parameter approaches positive infinity, the agent starts to select actions with equal probability. On the other hand, as the temperature parameter approaches zero the agent will select the greedy action based on the Q-value estimates. For temperatures between zero and infinity, the agent selects actions randomly but its selection probability is weighted according to the Q-value estimates. The higher the temperature, the greater the stochasticity.

As seen in the plot, the agent using a Boltzmann temperature of 10 tends to outperform the agent with a temperature of 0. This is because a temperature of 0 corresponds to an agent that only exploits and does not explore, resulting in a sub-optimal policy as described earlier. On the other hand, the agent with a temperature of 20 is too exploratory and under-performs relative to both of the other cases.

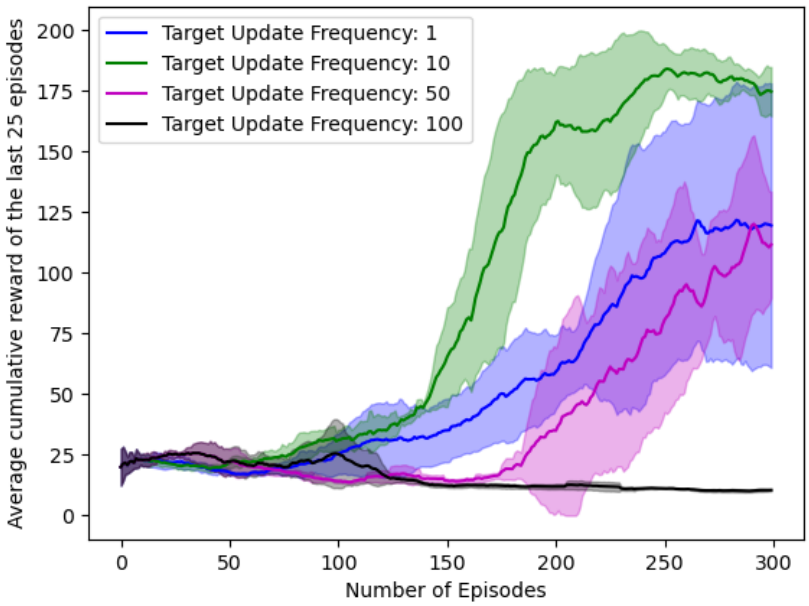
2.3 Exploration and Exploitation

The final Q-values for each state and the final policy are different for the various choices of exploration/exploitation parameters (ϵ or temperature) since the stochasticity of the agent's choice of action interacts with the stochastic transition model. This results in differences in Q-value for each state under each of the various exploration/exploitation parameters.

Ideally, the exploration/exploitation parameters should be chosen such that it permits the agent to explore infrequently visited state-action pairs while allowing the agent to reap the benefits from the learned policy as its estimate of the Q-values improve. Q-learning converges to optimal Q-values if every state is visited infinitely often (due to exploration) and if the action selection becomes greedy as time approaches infinity.

3 Part 3 Write Up

3.1 Target Network Update Frequency



The target network update frequency parameter that maximizes average cumulative rewards is 10, followed by 1, 50, and 100, in order of decreasing performance. Updating the target network in deep Q-learning has parallels with Value iteration.

In value iteration, the transition dynamics are explicitly defined, however, in deep Q-learning we do not have an explicit transition model though we can approximate it by collecting samples. Every target network update corresponds to a step in value iteration albeit with an estimate based on collected samples. The Q network is updated via back-propagation and the gradients from the target network should not affect the updates.

Updating the target network infrequently (keeping all else constant) is akin to making value iteration steps with a frozen/infrequently updated next state value estimate, and so the next state value estimate on which the update is based remains poor even after several episodes/steps.
On the other hand, if the target network is updated too frequently then this would correspond to making value estimates with few samples of the transition model, also resulting in poor estimates.

Value Iteration:
Repeat for all s:

$$\overbrace{\underbrace{V(s)}_{\text{update}}}^{\substack{\text{Value estimate}\\ \text{at current state}}} \leftarrow \overbrace{\max_a R(s) + \gamma \sum_{s'} Pr(s'|s,a)\, \underbrace{\bar{V}(s')}_{\text{target}}}^{\text{Estimate using value estimate at next state}}$$

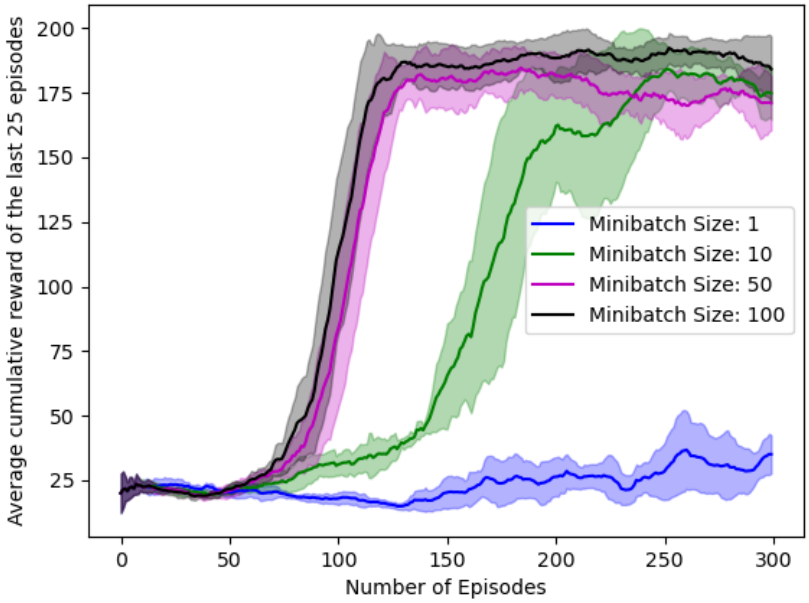
$$\bar{V} \leftarrow V$$

Deep Q-Learning:
Repeat for each (s, a, r, s') in mini-batch:

$$\boldsymbol{w} \leftarrow \boldsymbol{w} - \alpha_t [\, \overbrace{\underbrace{Q_{\boldsymbol{w}}(s,a)}_{\text{update}}}^{\substack{\text{Q-value}\\ \text{estimate at}\\ \text{current state}}} - \overbrace{(r + \gamma \max_{a'} \underbrace{Q_{\bar{\boldsymbol{w}}}(s',a'))}_{\text{target}}}^{\substack{\text{Q-value estimate using}\\ \text{value estimate at next state}}}] \frac{\partial Q_{\boldsymbol{w}}(s,a)}{\partial \boldsymbol{w}}$$

$$\bar{\boldsymbol{w}} \leftarrow \boldsymbol{w}$$

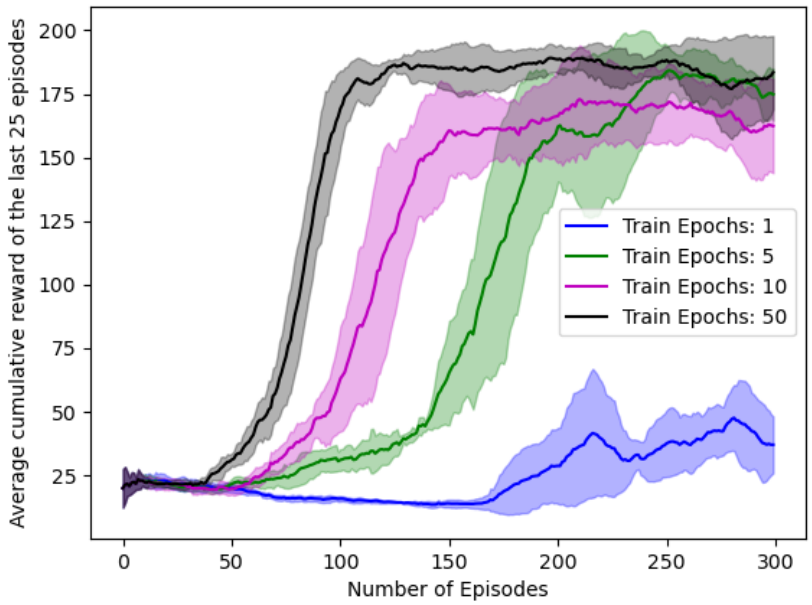
3.2 Mini-Batch Size



The mini-batch size is the number of samples processed before the model is updated. The average cumulative reward achieved and rate of convergence appears to improve as the mini-batch size increases. The larger the mini-batch size the more samples available to the agent to estimate the reward and transition dynamics of the system, thereby improving its estimates.

Exact gradient descent is when the neural network is able to take a step toward the optimal point in the loss landscape by taking all the data into consideration. As the size of the mini-batch approaches the full set of (s, a, r, s') (an infinite set in continuous time scenarios) this becomes exact gradient descent in the limit.

3.3 Replay Buffer Training Epochs



The replay buffer training epochs are the number of training passes taken through the data accumulated in the replay buffer. As the number of training epochs increases so does the average cumulative reward achieved by the agent. An optimal number of training epochs should be chosen to mitigate over-fitting to the data.

For the scenarios with more epochs, there appears to be a slight degradation in performance in later episodes. This may be due to over-fitting the data set. More training epochs will allow the agent to take more gradient steps towards improved performance, however, this performance is measured with regards to the training data set and may not reflect actual performance in more general situations (due to the over-fitting issue).

Similar to mini-batch size, the number of training epochs affects the gradient descent steps. Whereas the mini-batch size affects the confidence with which a step can be taken in the direction of improved performance, the number of epochs affects the number of steps overall taken towards this direction.