Reinforcement Learning Assignment 2

Yifan Bai(260562421), Bhavya Patwa(260964036)

March 2, 2020

1 Question 1, Track A

1.1 Part i

Our goal is to estimate state function v_π via updating an estimate of it, V_π using the following rule:

$$V_\pi(S_t) += \alpha(Target - V_\pi(S_t))$$

For MC, return G_t is used as a target, and it is a discounted sum of rewards from state S_t, until the end of episode. Recall also that the definition of state function:

$$v_\pi(S_t) = E[G_t]$$

For TD, instead of waiting till the end of episode, the Target is sum of immediate reward and estimate of future rewards, i.e. Target $= R_{t+1} + \gamma V_\pi(S_{t+1})$. We use an estimated value to update the same kind of estimated value.

For reference, bias of an estimator $\hat{\theta}$ is defined as $E[\hat{\theta}] - \theta$. For us, it is then $E[Target] - v_\pi(S_t)$. Recall also that value function is $E[Target] = E[G_t] = v_\pi(S_t)$. This corresponds to MC means that MC has no bias. However, TD has bias, since $V_\pi(S_{t+1}$ is only an estimate of value function but not the true one, thus bias will not be zero. And it is expected that this bias will be huge at the beginning of training process, where V_π is far away from v_π. As training goes on, the estimates tend to improve and converge, thus reducing bias.

Variance measures 'noisiness' of an estimator. For MC, since G_t is sum of ALL rewards till the end of episode, it would be affected by all actions taken from state S_t. For instance, if there are two trajectories after taking an action and the two returns are very different, individual returns G_t would be far from the true value function $v_\pi(S_t)$. This means high variance. It is analogous to supervised regression in the sense that if we try to follow all 'points', we would then have to make many turns to connect all the data points. The end result is an overfitted, high variance function.

In contrast, TD has low variance. Target $R_{t+1} + \gamma V_\pi(S_{t+1})$ only depends on immediate reward R_{t+1}. Contrary to MC target of discounted sum, there are way less variables, thus less variance.

In terms of scalability, MC has some challenges. For instance, when estimating action values, many state-action pair may not be visited. For deterministic policy, in following π we only see one of actions at each state instead of averages. This means MC estimates of other actions will not improve with experience.

TD on the other hand is slightly better since we update estimations based on existing estimations and current state, the amount of computations would be less and the vulnerability to greater 'noise' is lower.

1.2 Part ii

Using Hoeffding's Inequality, one can prove that MC's error bound. The reason for it to be independent of its state space is that since it updates at each end of episode, we may either 1) Not need to go to the end of state space 2) We are only updating the exact rewards. The number of sample sizes thus dictate the error bounds, i.e. end of sampling determined by the need.

Since it is independent of the size of the state-space, we could tweak other two parameters which are constants of values in [0, 1]. This way, we could have a certain degree of accuracy with not too many samples. This reduces sampling complexity and efforts. The disadvantage is that since it is proportional to inverse squared value of sample size, to reduce the error to zero would be very costly and it would even be costly when we want really small errors.

1.3 Part iii

As described above, TD estimates the target by sum of immediate rewards and use it with current state to together estimate future rewards. We can see it as learning a partial or an implicit image of the part of the world it has been exploring, and at the same time, update the new estimates 'on the fly', improving its future estimations (bias reduction). We do not need to wait till the end of an episode to get an update. This is fundamentally different to MC, as we need to explore all states and only update our 'understanding' (reward) at the end of each episode.

Take Page 127 of the textook, Example 6. Batch MC always find estimates minimizing MSE on training set, but TD always find one that might not minimize MSE but exactly correct for maximum likelihood (ML) model of Markov Process. ML of a parameter gives the value with greatest probability of generating data. Here, the ML is the model of Markov Process formed based on observed episodes. This is analogous to learning a partial/implicit image of known world. We could then estimate the perfectly correct value function given the model is exactly correct (i.e. same 'world').

2 Question 2, Track A

2.1 Part i

Algorithm 1: SARSA algorithm with function approximation that uses Boltzman exploration

```
Input: a differentiable action-value function parameterization $\hat{q} : S \times A \times \mathbb{R}^d \to \mathbb{R}$;
Algorithm parameters: step size $\alpha > 0$, small $\epsilon > 0$;
Initialize value-function weights $\mathbf{w} \in \mathbb{R}^d$ arbitrarily (e.g., $\mathbf{w} = \mathbf{0}$);
Loop for each episode;
    S, A ← initial state and action of episode (eg. Boltzmann $P_A(t+1) = \frac{e^{\hat{q}(S,a,\mathbf{w}_t)}/\tau}{\sum_{a_i} e^{\hat{q}(S,a_i,\mathbf{w}_t)}/\tau}$);
    Loop for each episode;
        Take action A, observe R, S';
        If S' is terminal:;
            $\mathbf{w} \leftarrow \mathbf{w} + \alpha[R - \hat{q}(S, A, \mathbf{w})]$;
            Go to next episode;
        Choose A' as a function of $\hat{q}(S', ., \mathbf{w})$ (eg. Boltzmann $P_{A'}(t+1) = \frac{e^{\hat{q}(S',a,\mathbf{w}_t)}/\tau}{\sum_{a_i} e^{\hat{q}(S',a_i,\mathbf{w}_t)}/\tau}$);
        $\mathbf{w} \leftarrow \mathbf{w} + \alpha[R + \gamma\hat{q}(S', A', \mathbf{w}) - \hat{q}(S, A, \mathbf{w})]\nabla\hat{q}(S, A, \mathbf{w})$;
        $S \leftarrow S'$;
        $A \leftarrow A'$;
```

2.2 Part ii

- SARSA with ϵ-greedy policy may fail to converge for an MDP because the probability of visiting a given state can change discontinuously when the Q function fluctuates slightly.
- For a two path MDP where one trajectory has higher reward and other has lower, it will lead to an infinite loop where each path appears better than the other and thus the cycle of learning Q values oscillate forever.
- By using Boltzman exploration, it no longer chooses an action greedily or due to randomness introduced by the ϵ factor, which are responsible for not keeping the aggregate Q stable.
- Boltzman exploration can help smoothen the Q values and assigns probability to each state-action pair which would be used for choosing the next action.
- Thus, the softmax property ensures that the aggregate Q value always converges and doesn't oscillate like in the ϵ-greedy case.

2.3 Part iii

- The SARSA and Q-Learning are both TD Control algorithms which uses State-Action Pairs inorder to update the policy which we call Q. Q-Learning directly learns the optimal policy, while SARSA learns a near-optimal policy whilst exploring greedily.
- SARSA is on-policy, as it updates Q after each action while Q-Learning uses maximum Q return over all possible actions before updating.
- Q-Learning update step: $Q(S, A) \leftarrow Q(S, A) + \alpha[R + \gamma max_a Q(S', a) - Q(S, A)]$
  Expected SARSA update rule: $Q(S, A) \leftarrow Q(S, A) + \alpha[R + \gamma\Sigma_a \pi(a|S')Q(S', a) - Q(S, A)]$

- In Expected SARSA, we take into account how likely each action is under the current policy. Given the next state S', the Expected SARSA algorithm moves deterministically in the same direction as the Q-Learning moves greedily.
- Instead of taking the expected value over the next state-action pairs, we take the maximum value, which makes Q-Learning a special case of Expected SARSA.

3 Statement of Contribution

Both team members contributed equally to written part of this assignment, from brainstorming ideas, formulating solutions, to reviewing solutions and typing out the document. We hereby state that all the work presented in this report is that of the authors. The links to Google Colab notebook for programming questions are:

3.1 Question 1

Continuous Random Walk (Bhavya)
`https://colab.research.google.com/drive/1gTMDA-xn8CsZUmiyIrNBpJYCIbxnDGuV`

3.2 Question 2

Cartpole (Yifan)
Q-learning
`https://colab.research.google.com/drive/1T17Y267DB4bQP3cWesqPWiXYYaUbi-Sa`
Sarsa
`https://colab.research.google.com/drive/10xDdPaS5hS_s1LxL8yMyME-14_k9-1x8#scrollTo=uCUPei6ditZH`
Expected-Sarsa
`https://colab.research.google.com/drive/1dRbApM006FAOrjkeUNGHxWzBx6aWXRIQ`