Sherwyn Braganza

South Dakota School of Mines and Technology

Advanced Topics in AI, Fall 2022

CSC 549 - M01

Exam 2

85%

1. Consider the following pseudocode for a faulty SARSA algorithm. Find all the mistakes in the algorithm. Explain why they are mistakes and correct them.

The image below shows the faults I have identified in the SARSA Algorithm.

procedure SARSA(number of episodes $N \in \mathbb{N}$
discount factor $\lambda \in (0,1]$
learning rate $\alpha_n = \frac{1}{\log(n+1)}$) 1
  Initialize matrices $Q(s,a)$ and $n(s,a)$ to $0, \forall s,a$
  for episode $k \in 1,2,3,\ldots,n$ **do**
    $t \leftarrow 1$
    Initialize s_1
    Choose a_1 from a uniform distribution over the actions 2
    while Episode k is not finished **do**
      Take action a_t: observe reward r_t and next state s_{t+1}
      Choose a_{t+1} from s_{t+1} using μ_t: an ε-greedy policy with respect to Q
      if The current state is terminal **then** ▷ *Compute target value*

$$y_t = 0$$

Terminal reward should be r.

      else

$$y_t = r_t + \max_a Q(s_{t+1}, a)$$

This is off-policy, but Sarsa should be on-policy

      end if
      $n(s_t,a_t) \leftarrow n(s_t,a_t) + 1$
      Update Q function:

$$Q(s_{t+1}, a_{t+1}) \leftarrow Q(s_t,a_t) - \alpha_{n(s_t,a_t)}\left(y_t - Q(s_t,a_t)\right)$$

3

      $t \leftarrow t+1$
    end while
  end for
end procedure

(i) The learning rate ($\alpha_n = \dfrac{1}{log(n+1)}$) would be greater than 1 if the base of the log term is not **2.**

(ii) Choosing a_1 from a uniform distribution over the actions would result in an off -policy action which is not what we want. A more appropriate choice would be to **choose a_1 from s_1 using the policy μ_t: an ϵ-greedy policy w.r.t. Q.**

(iii) This algorithm contains all the characteristics of a SARSA (λ) algorithm; so I feel it is safe to assume that this algorithm was designed to replicate a SARSA (λ) algorithm. What is missing here is the backtracking update (**Eligibility Trace Update**). The corrections for (3) are shown below:

For all $s \in S, a \in A(s)$:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \ y_t \ n(s,a)$$

$$n(s,a) \leftarrow \lambda n(s,a)$$

* here we assumed that $\lambda = 1$

2. **Your friend found a variant of SARSA which is defined through a sequence of policies π_t (where t >= 1), and consist of just changing (in the previous algorithm after corrections) the way the target is computed. The target becomes:**

$$y_t = r_t + \lambda \sum_a \pi_t(a \mid s_{t+1})\ Q(S_{t+1}, a)$$

Where $\pi_t(a \mid s)$ is the probability that a is selected in state s under policy π_t.

a) **What sequence of policies (π_t) should you choose so that the corresponding variant of SARSA is on-policy? The variant is called Expected SARSA.**

A. Expected SARSA would still be on-policy if the chosen action (a) by π_t resulted in the largest value of $\pi_t(a \mid s)\ Q(S_{t+1}, a)$ when compared to other actions in the state. Or in other words, a chosen should reflect the dominant $\pi_t(a \mid s)\ Q(S_{t+1}, a)$ in the term $\sum_a \pi_t(a \mid s_{t+1})\ Q(S_{t+1}, a)$. This would only happen if the sequence of policies followed are determined by an ϵ**-greedy policy w.r.t. Q where ϵ = 1.**

b) **Consider an off policy variant of SARSA corresponding to a stationary policy $\pi = \pi_t \forall t$. Under this algorithm, do the Q values converge? If so, what are the limiting Q values? Justify your answer.**

A. In the trivial case, if the stationary policy being followed doesn't reach the terminal state, the Q values will not converge as states which are looped over would have a Q value of $-\infty$.

The Q values would converge and would be dictated by the Q value of the terminal state. The Q values for each state would converge to the sum of the Q values of the next state plus the reward for taking the action that leads to that next state. For example, the Q value of the state just before the Terminal State in the policy would have a limiting Q value of the Terminal State (0) plus the the reward for taking the action that led to the Terminal State (-1) which equals to -1.

My thought process behind this was based of value iteration from Chapter 4. When the episodes reach infinity, the update ($\alpha(y_t - Q(s_t, a_t))$) should tend to 0. And from observing the grid world problem, the stable value of each state in the policy was somewhat of the form:

$$Q(s_t, a_t) = Q(s_{t+1}, a_t) + r_t$$

And since the terminal states Value function is fixed at 0, it dictates the values of all preceding states by this equation.