CSC 449 Advanced Topics in Artificial Intelligence

Deep Reinforcement Learning
Exam 2
Fall, 2022

Name: Colton Snyder ID#: 7533299 Score: 95

Your solutions to these problems should be uploaded to D2L as a single pdf file by the deadline. You may turn in the solution up to two days late, with a penalty of 10% per day, and you should only upload one version of your solutions.
This exam is individual and open book. You may consult any reference work. If you make specific use of a reference outside those on the course web page in solving a problem, include a citation to that reference.
You may discuss the course material in general with other students, but you must work on the solutions to the problems on your own.
It is difficult to write questions in which every possibility is taken into account. As a result, there may sometimes be "trick" answers that are simple and avoid addressing the intended problem. Such trick answers will not receive credit. As an example, suppose we said, use the chain rule to compute $\frac{\partial z}{\partial x}$ with $z = \frac{7}{y}$ and $y = x^2$. A trick answer would be to say that the partial deriviative is not well defined because y might equal 0. A correct answer might note this, but would then give the correct partial derivative when $y \neq 0$.

1. (40 pts) Consider the following pseudo-code for a faulty SARSA algorithm:

procedure SARSA(number of episodes $N \in \mathbb{N}$
discount factor $\lambda \in (0,1]$
learning rate $\alpha_n = \frac{1}{\log(n+1)}$)

may not converge, but 1/n would

Initialize matrices $Q(s,a)$ and $n(s,a)$ to $0, \forall s,a$
for episode $k \in 1,2,3,\ldots,\cancel{n}$ N **do**
$t \leftarrow 1$
Initialize s_1
~~Choose a_1 from a uniform distribution over the actions~~

Choose a_1 from a s_1 using a policy derived from Q (e.g. ε-Greedy). Choosing a random starting action is for the Monte Carlo Exploring Starts algorithm

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*

s_{t+1} not current state, if the state that we reach on the next timestep is terminal then the target is 0. Also, if the current state were terminal then we would have already ended the episode.

$$y_t = 0$$

else

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

This is the target for a Q-Learning algorithm. For SARSA, this should be
$y_t = r_t + \lambda Q(s_{t+1}, a_{t+1})$

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

This needs to be a plus not a minus, so that we move towards the target not away from it.

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

$Q(s_t, a_t)$, updating value of current state-action pair not future state-action

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

Find all of the mistakes in the algorithm. Explain why they are mistakes, and correct them.

2. (60 pts) Your friend found a variant of SARSA which is defined through a sequence of policies π_t (where $t \geq 1$), and consists of just changing (in the previous algorithm **after corrections**) the way the target is computed. The target becomes

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

where $\pi_t(a|s)$ 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? This variant is called Expected SARSA.

To be on-policy each policy π_t used to determine the target value should be the same as the μ_t policy used for deciding behavior at that time step, and these policies should all be based on the values of Q.

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.

Under this algorithm with a stationary policy used at each time step to determine the target value, then the Q values will converge to q_π, the true state-action values for the stationary policy, π. This is because at each time step, the target calculated is $v_\pi(s_{t+1})$, the value of the state s_{t+1} under the policy π, so the Q update step moves the value of $Q(s_t, a_t)$ towards $v_\pi(s_{t+1})$, or the value of the state that you land in after taking action a_t from state s_t.

ok