CSC 549 Advanced Topics in Artificial Intelligence
Exam 2

90%

Justin Lewis

November 30, 2022

1. (40 pts) Consider the faulty SARSA algorithm:

-10

```
1  procedure SARSA( number of episodes N ∈ ℕ
2                  discount factor λ ∈ (0, 1]
3                  learning rate α_n = 1/log(n+1) )
4       Initialize matrices Q(s,a) and n(s,a) to 0, ∀s, a
5       for episode k ∈ 1, 2, 3, ..., n do
6           t ← 1
7           Initialize s_1
8           Choose a_1 from a uniform distribution over the actions
9           while Episode k is not finished do
10              Take action a_t: observe reward r_t and next state s_{t+1}
11              Choose a_{t+1} from s_{t+1} using μ_t: an ϵ−greedy policy with respect to Q
12              if The current state is terminal then
13                  y_t = 0
14              else
15                  y_t = r_t + max_a Q(s_{t+1}, a)
16              end if
17              n(s_t, a_t) ← n(s_t, a_t) + 1
18              Update Q function:
19                  Q(s_{t+1}, a_{t+1}) ← Q(s_t, a_t) − α_{n(s_t,a_t)}(y_t − Q(s_t, a_t))
20              t ← t + 1
21          end while
22      end for
23  end procedure
```

I would note that λ, labeled as the discount factor, is typically for the trace decay rate. Also regarding λ is that it isn't actually used in the algorithm.

It's unclear based off of how the do-while loop is written, but it seems to me that the if "If the current state is terminal..." part is not useful since you set $t = t + 1$, which makes state reference the terminal state, so when you loop back to the the top, it should exit the loop since the episode is now finished.

The most significant error with this is the action-value function. There's three things wrong with it, the action-value function is supposed to be using a_{t+1} instead of of a (making the current a Q style update). It also isn't using the discount factor (λ in this case).

This is fixed by changing line 15 to $y_t = r_t + \lambda Q_{(s_{t+1}, a_{t+1})}$.

The third thing wrong with the value function is that it has a subtract instead of an add on line 19 (right before the α).

Besides that, there's just a couple other small things I found that are a little iffy. On line 8 when you choose a_1, it is choosing a random action from the action space. This isn't inherently incorrect depending on how your actions space is defined. However, for some problems it may be that that actions you can take are dependent on the current state. For example in some grid world problems, when you're against an edge it might be defined that you cannot go into the edge (though in our class we've just defined it as saying you

can for a punishment, but that's besides the point). This can be fixed by saying to choose from a uiform distributions over the actions given s_1, often notated in the book as $A(s)$.

Another mistake is that it is updating Q of the next step instead of the current step.

There's an interesting thing to note where the step size is based on how many times you've visited a state action pair. I dind't encounter any issues with this, but thinking about it it would seem like it removes punishment for repeating the same state actions pairs, which may cause it to learn slower since it isn't quite as punished for taking longer.

I'm not sure whether to rewrite the algorithm with the corrections made. I don't think I would need to since the important part is the explanations anyway and the re-write would basically be the textbook SARSA algorithm, except written in the strange notation that implies all past state and actions are just saved in memory. But here it is anyway. Note that any future references to the algorithm will use line numbers from the PREVIOUSLY written algorithm, not the following.

1 **procedure** correctedSARSA(number of episodes $N \in \mathbb{N}$
2 discount factor $\gamma \in (0, 1]$
3 learning rate $\alpha_n = \frac{1}{\log(n+1)}$) May not converge. Use 1/n
4 **Initialize** matrices Q(s,a) and n(s,a) to 0, $\forall s, a$
5 **for** episode $k \in 1, 2, 3, \ldots, n$ **do**
6 $t \leftarrow 1$
7 **Initialize** s_1
8 Choose a_1 from a uniform distribution of possible actions gi uniform distribution?
9 **while** Episode k is not finished **do**
10 Take action a_t: observe reward r_t and next state s_{t+1}
11 Choose a_{t+1} from s_{t+1} using μ_t: an ϵ–greedy policy with respect to Q
12
13 $y_t = r_t + \gamma Q(s_{t+1}, a_{t+1})$
14
15 $n(s_t, a_t) \leftarrow n(s_t, a_t) + 1$
16 Update Q function:
17 $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha_{n(s_t,a_t)}(y_t - Q(s_t, a_t))$
18 $t \leftarrow t + 1$ decay?
19 **end while**
20 **end for**
21 **end procedure**

2. Consider Expected SARSA, where line 15 would look like $y_t = r_t + \lambda \sum_a \pi_t(a|s_{t+1})Q(S_{t+1}, a)$.

   (a) What sequence of policies should you choose so that the corresponding variant of SARSA is on-policy?

   'To make it on-policy, you need the behavior and target policy to be identical'. Or, in my own words, as long as when it is computing the target value it uses the action that it actually takes (which is chosen on line 8) then it will be on-policy. However, to answer in more detail as I was discussing with peers (Colton) this brought up at what point does it become on policy when the computation is made. I initially thought it would also need to be deterministic to be completely on-policy. This would make it essentially no different from standard SARSA, as only the action that would make it into the summation is the same as was chosen on the aforementioned line 8. Colton argued that it doesn't matter if it's deterministic or not. I'm not sure what the answer is, I double checked the textbook and couldn't find anything that gave a conclusive answer, just more information that seemed to support my point of view where it would need to be deterministic. I am, however, thinking that it can be stochastic just because if I assume the statement that the behavior (where the a_{t+1} is chosen from) and the target policy need to be identical, and I consider that on-policy agents "commit to always exploring", then I see no reason for it to need to be deterministic.

   Another debate is whether on-off-policy terminology even applies to Expected SARSA. On-policy can be thought of as whether an agent learning based off of what it is doing (loose description). To get a better understanding of what I mean by that is to look at the difference between a standard SARSA algorithm and a Q-learning algorithm. The primary difference between the two is that Q learning will update the state-action values based on which would be optimal while SAR'S'A' updates the state-action

value based off of which action has already been selected. They both turn into on-policy if the policy is greedy (as opposed to ϵ-greedy, in which only SARSA is on-policy between the two). So that has a parallelism to the above argument, where if the target and behavioral policy is the same, then it's the same as SARSA. Meaning the target policy would only give a 1 for the action that was already selected. This gets weird when stochasticity is involved, but it's a weird question.

(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.

Recall that a policy is just the probability of taking an action given conditions ($\pi(a|s)$). So that may be helpful to keep in mind while considering this.

The notation here is a little peculiar, it seems like it should be saying $\pi_t = \pi \forall t$. But I digress. The only answer that can be confidently given is that it depends on the environment. I found a source (item 4 below) that discusses how in a stationary policy an optimal policy may exist, though not much detail is gone into. From an extensive discussion with Jonathan regarding stationary policies, we came to the conclusion that this question is significantly more vague than we initially thought.

Some information that an answer would depend on is whether we split the policy into behavioral and target policy, where it's the behavioral policy that is stationary and you hope that the target policy (which would be greedy with respect to the state-action values) converges to the optimal policy (only conceptually, this can pretty much be disregarded).

Although upon writing that out I'm not sure that really matters, because what that is basically stating is "if you follow a stationary behavioral policy, do the relative values converge" (i.e. if you consider GPI where the values may continue changing indefinitely, you stop the iteration when your policy does not change between iterations, because the values preserved the relationship).

Anyway, one of the most significant conclusions to draw from this is the stationary policy's ability to explore the state space (or rather, state-action space). But even at this, it somewhat depends on your perspective, because if the policy keeps the agent from exploration, then yes the Q-values still will converge to the same 'relative values', but many of the values may be left unexplored, or 0. Versus if the state-action space is thoroughly explored, then the same would up, the values should be able to converge. Possibly not though. This is a very dependent question. Also a stationary policy is very bizarre.

What a stationary policy would entail it seems the agent isn't really learning much. Sure it's filling out Q values, but it's not changing it's behavior or getting any better. Since at this point the only value to gain from doing this is to have the policy capable of exploring the entirety of the state-action space thoroughly enough to converge the relative values. I definitely don't think it's likely for the agent to find the optimal relative values though it is possible. In any case this kind of entails one of two situations, right? Either the agent is following a stationary policy that is already the optimal policy, in which case why bother using reinforcement learning because you were able to hard-code essentially the ideal solution. Alternatively, you could get an agent that is basically moving around completely randomly and not learning but still noting down what happens. I'm not certain whether that would have any value, because again I'm not sure if it would converge to the optimal policy. I know it definitely could, just not sure if it would.

If you consider a grid world problem, then theoretically an agent that always picks from the four directions completely randomly would have something close to an optimal policy (by which I mean if you made a policy that was greedy with respect to the found values, it would be optimal) since theoretically over time the tiles closer to the goal would have lower values and those further away would have much much higher values (since it wanders aimlessly for a very long time before eventually finding it's way to the goal). That said, there's also the possibility that the agent could just start close to the goal, wander away, take ages wracking up punishment, then go back to the goal. This would mess things up! And at this point you could just do generalized policy iteration anyway (although I suppose that eliminates the stationary policy constraint, but that's besides the point).

Anyway, there's also policies you could choose for grid world that would be close to ideal without starting as the optimal policy, such as if your goal is bottom right and start is top left, then have your stationary policy be to have a 50-50 chance of going down or right any turn. This should be very close to optimal! It should converge as well!

This is such a bizarre question, but hopefully I demonstrated I put in a significant amount of time into trying to figure something out. Sadly the best I could come up with is that it depends on the environment, but it really seems quite silly. Everything that's been taught in the class and those David

Silver lectures and I believe the textbook as well though obviously I haven't been able to read all of it really has the agent capable of actually learning with some amount of exploration (i.e. most agents have a balance of exploitation versus exploration whereas this algorithm seems to involve an agent that has exclusively exploration (possibly... Depends on the policy).

Resources used while trying to figure these out:

- https://ai.stackexchange.com/a/10907, which discusses Expected SARSA vs SARSA
- https://www.geeksforgeeks.org/expected-sarsa-in-reinforcement-learning/, which discusses Expected SARSA vs SARSA vs Q-learning
- Reinforcement Learning, an Introduction by Sutton and Barto, especially section 6.6
- https://ai.stackexchange.com/a/15429, which discusses stationary and non stationary policies
- https://www.quora.com/What-is-the-meaning-of-a-stationary-Policy-in-the-context-of-reinforcement-learning, I looked at this as well but I didn't think it had any value. I'm linking it in case there was something mentioned that subconsciously made its way into my answer.