Exam 2: Advanced AI

Caleb Ehrisman

11/30/2022

100

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 $\alpha_n = \frac{1}{\log(n+1)}$)
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
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*

$$y_t = 0$$

else

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

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

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

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

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

Initially it can be seen that this faulty SARSA algorithm was intended to be the SARSA(λ) implementation due to the $n(s,a)$ matrix (eligibility trace) and the visit update of $n(s_t,a_t)$ += 1. However, there are multiple mistakes that need to be fixed. It is assumed that γ is set to 1; so, in all instances where γ would be typically be included are not written.

(1) The learning rate a_n is undefined at 0 as the $\log(t+1)$ when n is 0. Also the learning rate is greater than 1 until $n = 9$, so it is not ideal as it will cause issues with convergence and take longer to find the optimal solution. To fix this and still use the Log term in the denominator of the learning rate decay would be to change the constant within the computation. So under the assumption that this is using Log_{10} it can be changed to $\alpha_n = \frac{1}{log(t+10)}$ or to make it work with all log bases: $\alpha_n = \frac{1}{log_{base}(t+base)}$

The n variable also needs to be changed to t to match the time step variable in the given algorithm and avoid ambiguity with n.

(2) Since the learning rate was updated to not be undefined at 0, it is now possible to have time step t start at 0 instead of 1. This also resolves the issue of having a_0 and s_0 never being used. With the assumption that initial values are typically using the notation of $value_0$.

(3) a_0 needs to be chosen from starting state by using a greedy method not an uniform distribution to fall in line with SARSA(λ).

(4) The computation for y_t is incorrect. The correct computation would be

$$y_t = r_t + Q(s_{t+1}, a_{t+1})$$

The reason it is wrong is the current y_t equation is the same method used for Q-learning and it results in an off-policy learning style since it will choose actions based on the max reward of available actions. Where SARSA learns the Q values by taking actions given by the policy and updates the policy by interacting with the environment that has a chance to take a non-optimal action.

(5) The update section of the pseudo-code is missing several components needed to properly do the updates in the SARSAλ algorithm and is the most problematic section. Firstly, the alpha value notation is not correct. It should be α_t to match the notation given in the parameter section. The $Q(s_{t+1}, a_{t+1}$ update is lacking the eligibility trace $(n(s_t, a_t))$ being applied to y_t. The second term should be added to the Q value not subtracted. The n matrix is not also not being updated so it would not be decreasing based on the discount factor as it is in the SARSA(λ) algorithm. The last thing missing is that during the update step is that ALL states and actions in the state space and action space are also updated. Not just the current and next state/action pair. With all these components in the update being corrected or added it results in the following:

for each $s \in$ **S**, $a \in$ **A(s) do**
$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha_t * n(s_t, a_t)[y_t - Q(s, a)])$
$n(s_t, a_t) \leftarrow \lambda * n(s_t, a_t)$
end for

(6) For this I am not entirely sure if it is a mistake but it is not exactly clear. "while Episode k is not finished" does not have a reference statement of when k is finished.

The wording should be changed to "while Episode $k(t)$ is not terminal" to be explicitly clear that when the state in the current step within the episode is in a terminal state then to exit the while loop.

(7) The policy notation μ should be changed to π to fit with standard notation as we have had in the book.

(8) The initialization of the eligibility trace $n(s,a)$ to $0, \forall s, a$ needs to be done for each episode. So it should be moved to inside the for episode loop.

Corrected SARSA(λ) Algorithm

---

procedure SARSA(number of episodes $N \in \mathbb{N}$
discounting factor $\lambda \in (0,1]$
learning rate $\alpha_t = \frac{1}{log(t+10)}$)
initialize matrices $Q(s,a)$ $0, \forall s, a$
for episode $k \in$ 1,2,3,...,n **do**
$t \leftarrow 0$
Initialize s_0 and $n(s,a)$ to $0, \forall s, a$
Choose a_0 from s_0 using policy derived from Q (Greedy)
while Episode $k(t)$ is not terminal **do**
Take action a_t; observe reward r_t and next state s_{t+1}
Choose a_{t+1} from s_{t+1} using $\boldsymbol{\pi_t}$: an ϵ-greedy policy with respect to Q
if Current state is terminal **then**
$y_t = 0$
else
$y_t = r_t + Q(s_{t+1}, a_{t+1})$
end if
$n(s_t, a_t) = n(s_t, a_t) + 1$

for each $s \in \mathbf{S}$, $a \in \mathbf{A(s)}$ **do** ▷ Update Q_values and n_values
$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha_t * n(s_t, a_t)[y_t - Q(s_t, a_t)]$
$n(s_t, a_t) \leftarrow \lambda * n(s_t, a_t)$
end for
$t \leftarrow t + 1$
end while
end for
end procedure

---

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

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

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

Expected SARSA(λ) Algorithm from Corrected

procedure SARSA(number of episodes $N \in \mathbb{N}$
discounting factor $\lambda \in (0, 1]$
learning rate $\alpha_t = \frac{1}{log(t+10)}$)
initialize matrices $Q(s, a)$
for episode $k \in$ 1,2,3,...,n **do**
$t \leftarrow 0$
Initialize s_0 and $n(s, a)$ to $0, \forall s, a$
Choose a_0 from s_0 using policy derived from Q (Greedy)
while Episode $k(t)$ is not terminal **do**
Take action a_t; observe reward r_t and next state s_{t+1}
Choose a_{t+1} from s_{t+1} using $\boldsymbol{\pi_t}$: an ϵ-greedy policy with respect to Q
if Current state is terminal **then**
$y_t = 0$
else
$y_t = r_t + \lambda \sum \pi_t(a|s_{t+1}) Q(s_{t+1}, a)$
end if
$n(s_t, a_t) = n(s_t, a_t) + 1$

for each $s \in$ **S**, $a \in$ **A(s) do** ▷ Update Q_values and n_values
$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha_t * n(s_t, a_t)[y_t - Q(s_t, a_t)])$
$n(s_t, a_t) \leftarrow \lambda * n(s_t, a_t)$
end for
$s_t \leftarrow s_{t+1}, a_t \leftarrow a_{t+1}$
$t \leftarrow t + 1$
end while
end for
end procedure

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.

Expected SARSA can be both off-policy and on-policy, so how actions are decided upon is important. To be on-policy the chosen action (a) from π_t has to result in best

value from $\pi_t(a|s)Q(S_{t+1},a)$ with regard to the other actions in that state. This shows that the action, a is the most dominant action in the term $\lambda\sum\pi_t(a|s_{t+1})Q(s_{t+1},a)$. The action decision should be based on a greedy policy that is derived from Q that will always follow the best reward.

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 stationary policy is just a policy that does not change in respect to time. So a ϵ-greedy policy without a decaying ϵ would be considered stationary, since if the same conditions are met it would always chose the same action. Regardless, if the target policy is following a stationary greedy method then the algorithm will converge to an optimal solution.

An expected SARSA algorithm could have two different ϵ-greedy policies that use different values and this would create an off-policy method. Or the target policy could be a deterministic greedy method.

As far for convergence, there always exists an optimal stationary, deterministic policy which is inherently greedy so if the target policy is acting greedy with a behavioral doing exploration then within a infinite number of steps it will converge onto the optimal policy. Temporal Difference evaluation and then updating the Q_values will be the steps taken to do so.

The limiting Q_value would be the terminal state. This is due to all other values being dependent on the state distance from the terminal state. The limitation on state values would be the sum of the all rewards from all actions needed to reach the terminal state while following the optimal policy.