Rl ASSIGNMENT-3

Contributions-

Anya- SE22UCSE033
Section 5-1,3

Lakshmi-SE22UCSE149
Section 5- 2
Section 6

Stuti Garg-SE22UCSE263
Section 4-1,2

Section 4

SECTION 4 Date: / / /

Question 1) MDP Formulation

a) States (S)

$S = (q_1, q_2, g, T)$

where;

q_1 → Queue length of road 1 (East-west)

q_2 → Queue length of road 2 (North-South)

g → which road is currently green (road 1 or road 2)

T → Time taken since last switch

constraints;

$q_1, q_2 \in \{0, 1, 2, \ldots 17\}$ (∵ Given less than 20 per road)

(queue length will not exceed 1810 since, max queue length in 1800 s will be 1800 and 10 s more for cars to depart (decay from green to red).

$g \in \{1, 2\}$

$T \in \{0, 10\}$

Date: / / /

a) State Space (S)

$S = \{(q_1, q_2, g, T) \mid 0 \le q_1, q_2 \le 17,\ g \in \{1,2\},\ T \in \{0,10\}\}$

Max state space = $18 \times 18 \times 2 \times 11$

$= 7128$

b) Action (a)

$a \in \begin{cases} 0 & ; \text{keep it green currently (No switch)} \\ 1 & ; \text{switch to the other road after 10s have passed. } (\because T \le 10) \end{cases}$

Action space (A)

$A = \{0, 1\}$ (only 2 actions possible).

c) Reward (R)

$R(S, a) = -(q_1 + q_2)$

(negative reward because shorter queues are better. ∴

Date: / / /

(d) State Transitions

Arrival Transitions

Road 1 queue update

$$q_1 = \begin{cases} \min(q_1 + 1, 17) & \text{wp } 0.28 \text{ (car arrives)} \\ q_1 & \text{wp } 0.72 \text{ (no arrival)} \end{cases}$$

[+1 since car added → max queue length]

Road 2 queue update

$$q_2 = \begin{cases} \min(q_2 + 1, 17) & \text{wp } 0.4 \text{ (car arrives)} \\ q_2 & \text{wp } 0.6 \text{ (no arrival)} \end{cases}$$

[+1 since car added to queue → max length]

Departure Transitions

If road is green

Let K = road 1 or 2 (applicable for both)

$$q_K = \begin{cases} q_K - 1 & \text{wp } 0.9 \text{ (car departs)} \\ q_K & \text{wp } 0.1 \text{ (no departure)} \end{cases}$$

Constraint: $q_1, q_2 > 0$

Date: / / /

If road is red;

~~let $S_E \leq 10$~~

$$q_E = \begin{cases} q_E - 1 & \text{if } S_t \leq 10 \text{ up } 0.9\left(1 - \frac{S^2}{100}\right) \text{ decay formula} \\ & \text{constraint } q_E > 0 \\ q_E & \text{otherwise (if } S > 10\text{) no departure.} \end{cases}$$

Time update

$$S = \begin{cases} 0 & \text{if } a_t = 1 \text{ \& } S_t \geq 10 \text{ (switched)} \\ S+1 & \text{otherwise no switch} \end{cases}$$

Green road update

$$g_{t+1} = \begin{cases} 0 & \text{if } g_t = 1 \text{ \& } a_t = 1 \text{ \& } S \geq 10 \\ 1 & \text{if } g_t = 0 \text{ \& } a_t = 1 \text{ \& } S \geq 10 \\ g_{t+1} & \text{otherwise} \end{cases}$$

1. Pseudocode for the Variant of SARSA

Input:

- Environment with state space S and action space A
- Transition probabilities P(x' | x, a)
- Reward function r(x, a)
- Learning rate α
- Discount factor β
- Exploration factor ε
- Number of episodes N

Initialize:

- Value function V(x) = 0 for all $x \in S$
- Policy $\pi(x)$ is derived from V(x)

For each episode:

- Initialize state x from the environment
- Choose action a using ε-greedy policy derived from V(x)

While the episode is not done:

- Take action a, observe reward r(x, a), and next state x'
- Compute the Q-value for the current state-action pair:

  $Q(x, a) = r(x, a) + \beta * V(x')$
- Update the value function using the TD update rule:

  $V(x) = V(x) + \alpha * (r(x, a) + \beta * V(x') - V(x))$
- Choose the next action a' using ε-greedy policy derived from V(x')
- Set x = x', a = a'

Return:

- Value function V(x) for each state x

- Optimal policy $\pi(x)$ derived from $V(x)$

2. Advantage of the Variant of SARSA over Value/Policy Iteration

Value/Policy Iteration:

In value/policy iteration, we are given the transition and reward probabilities, and the process involves iteratively updating the value function and policy to converge to the optimal policy.

Value iteration directly computes the optimal value function, and policy iteration computes an optimal policy based on the value function.

Advantage of the Variant of SARSA:

The main advantage of the variant of SARSA over value/policy iteration is its online learning ability.

Value/Policy iteration requires a complete model of the environment (i.e., the transition probabilities and reward probabilities) to compute the optimal value function and policy.

SARSA, on the other hand, can be used in environments where the transition and reward probabilities are not fully known or are difficult to compute. It can learn the optimal policy through interactions with the environment without needing a complete model.

SARSA learns incrementally by interacting with the environment, so it can start making improvements to the policy even without knowing the full transition model.

3. Advantage of the Variant of SARSA over the Original SARSA

Original SARSA:

In original SARSA, you update the Q-values directly based on the observed rewards and the current Q-values of the next state-action pair. This means that you are estimating and updating the Q-function for every state-action pair.

Advantage of the Variant of SARSA:

The primary advantage of the variant of SARSA is that you are estimating fewer parameters. Instead of updating a Q-value for each state-action pair, you are only updating the value function $V(x)V(x)V(x)$ for each state. This reduces the number of parameters you need to track and update, which can be advantageous in high-dimensional state spaces.

In original SARSA, for each state xxx, you maintain a Q-value for each possible action aaa, which results in more parameters to estimate and update.

In the value-based SARSA variant, you only need to estimate and update a single value for each state V(x)V(x)V(x), which is computationally less expensive, especially in large environments with a large action space.