HARVARD
School of Engineering
and Applied Sciences

Reinforcement Learning

Temporal Difference Methods

Pablo Ruiz Ruiz

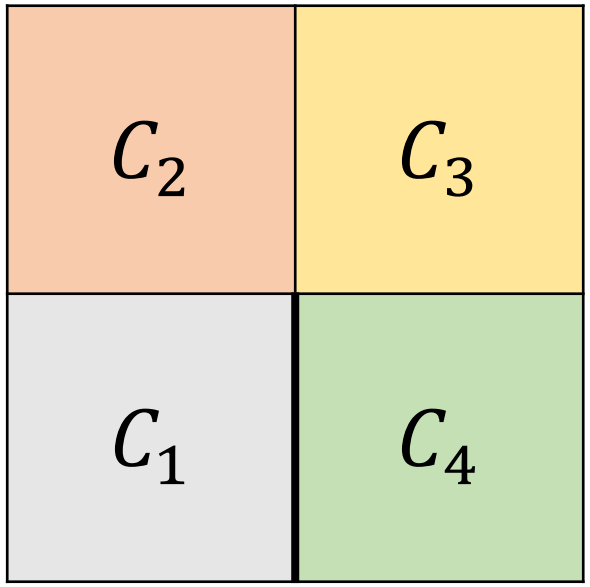
Deep Learning Intern Researcher @ Harvard University

Index

- SARSA
- SARSAMAX – Q-Learning
- Expected SARSA

Environment Example

States



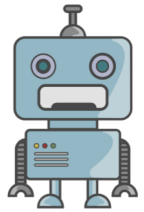
Actions

- a_1
- a_2
- a_3
- a_4

Rewards

- $+10$ for reaching C_4
- -1 otherwise

We call the states as C_i because in the formulas they use s_i to represent the state at each time step.
Thus, **there are 4 possible states** in this environment



Agent

Environment Example

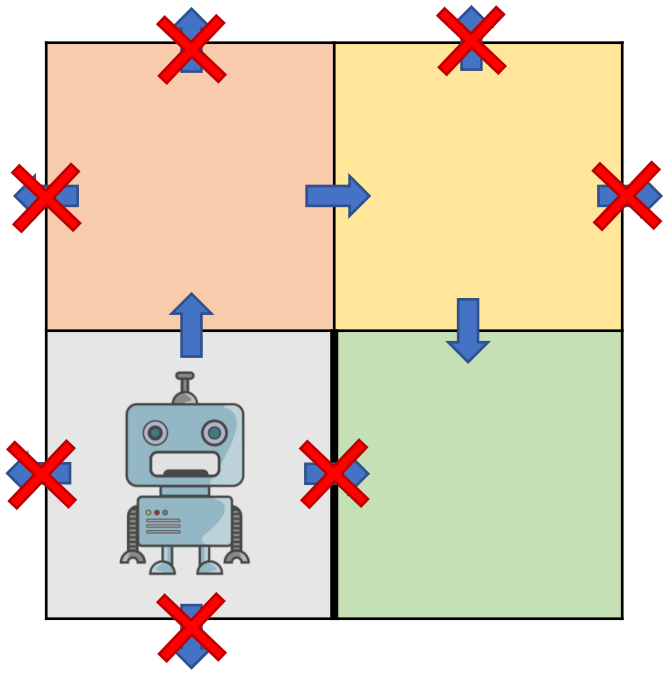


Table of values and Counter Table

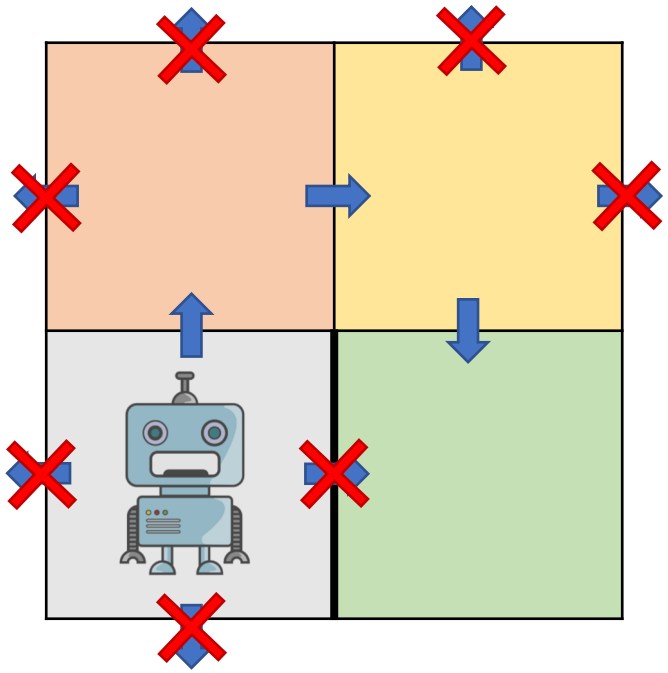
	a_1	a_2	a_3	a_4
c_1				
c_2				
c_3				

X represents that if the agent takes that action, it will remain in the same cell in the next state

The Table of values is trying to answer the natural question: *which action is the best one at each state?*

Counter Table is keeping track of how many time each state has been visited

Environment Example



Optimal policy π^*: c_1 ↑ c_2 → c_3 ↓ c_4

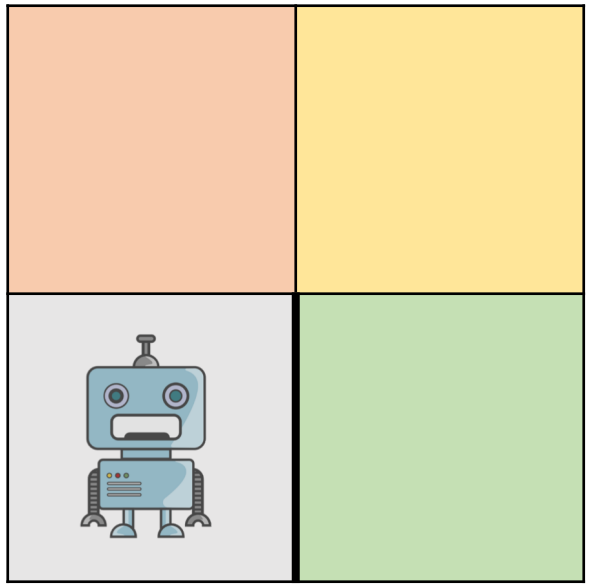
The agent has to find this policy by **interacting** with the environment

Temporal Difference Methods

Do NOT need to run the entire episode to update the Q-Table

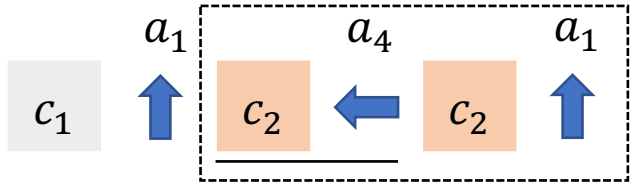
Instead, we **update every time step**

SARSA – SARSA(0)

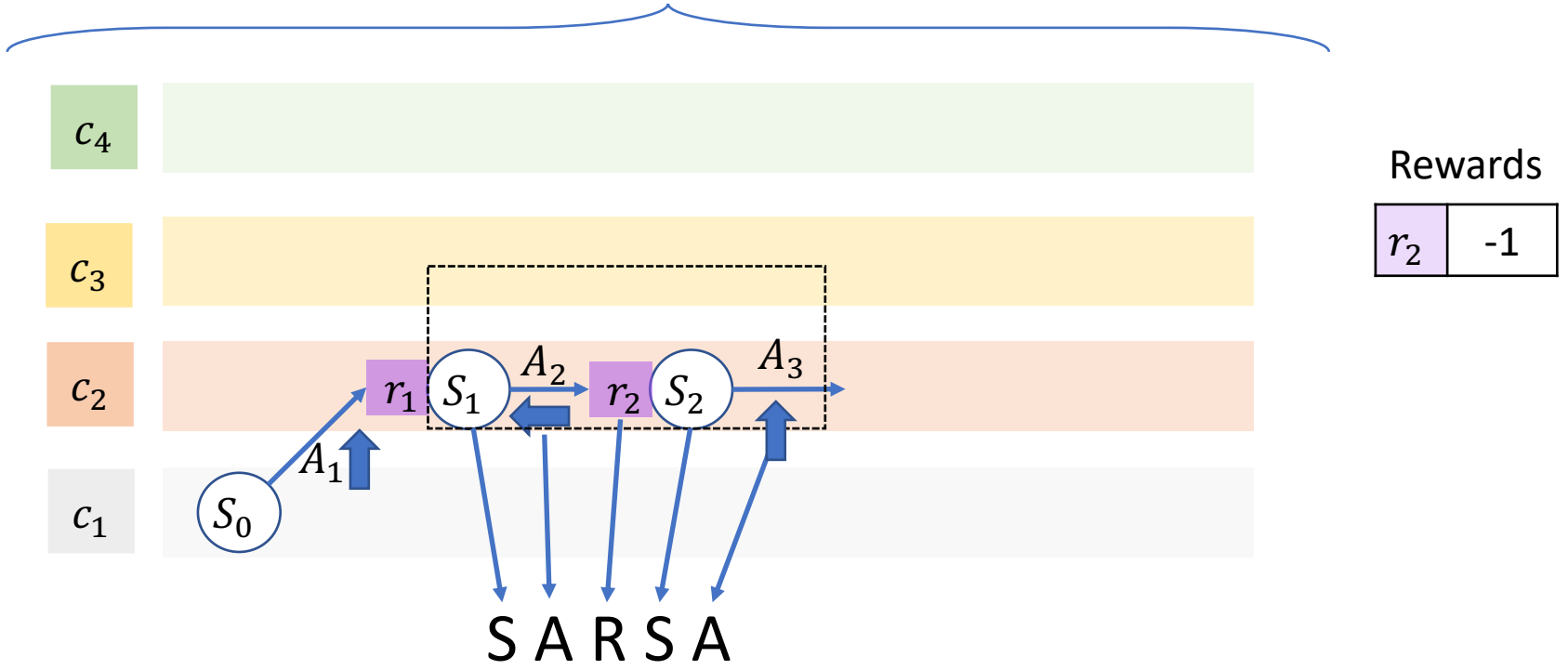


Episode 1:

The Q-Table is initialized with all values set to 0



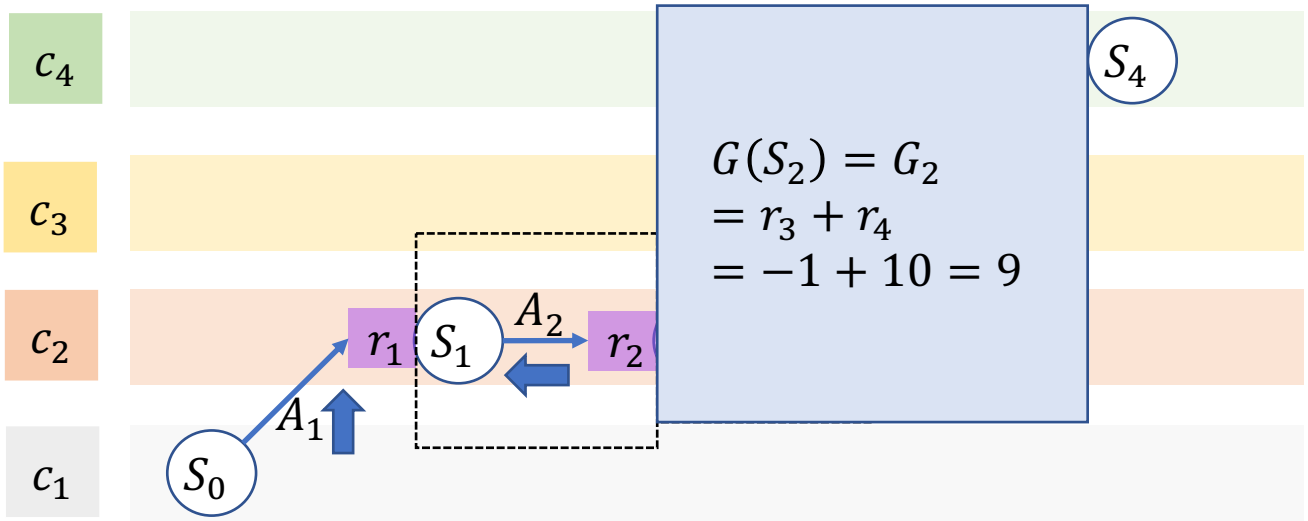
Not entire episode, just 1 time step ahead



Rewards

r_2	-1

This was the Monte Carlo approach



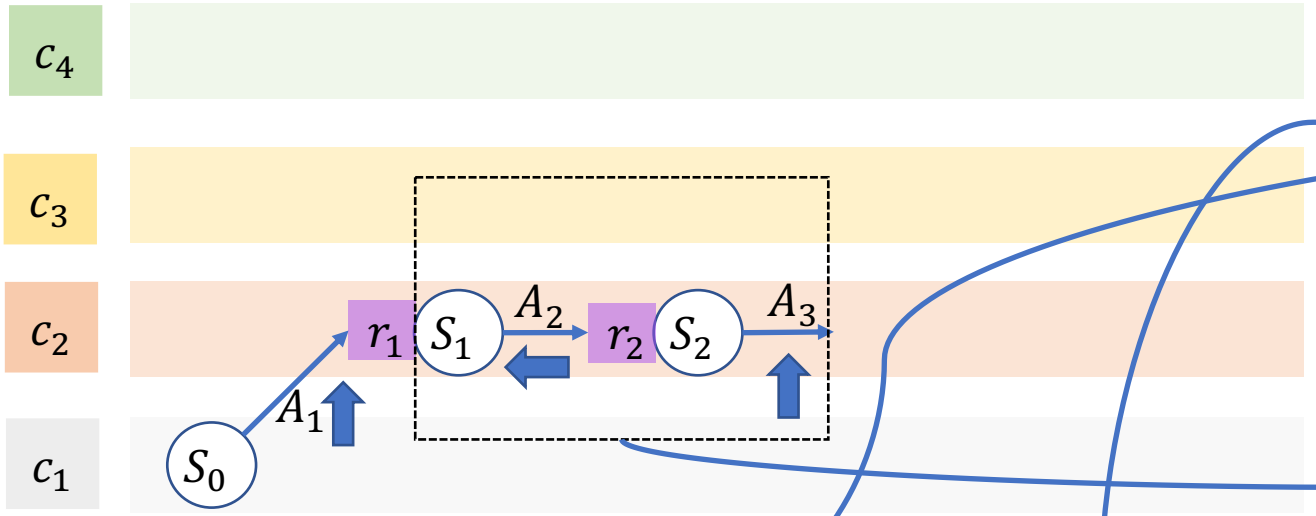
General Update Rule: $Q(s_t, a_{t+1}) = r_{t+1} + G_{t+1}$

But we haven't finished our episode:

We do NOT know the value of G

SARSA(0)

How do we update the table?



Q-Table

	a_1 ↑	a_2 ↓	a_3 ←	a_4 →
c_1	+7	+6	+5	+6
c_2	+8	+7	~~+8~~ +7.84	+9
c_3	+10	+7	+8	+8

Update Rule: $Q(c_2, a_3) = Q(c_2, a_3) + \alpha\big(r_2 + \gamma Q(c_2, a_4) - Q(c_2, a_3)\big) = 8 + \frac{1}{5}(0 + 0.9 * 8 - 8) = 7.84$

General Update Rule: $Q(S_t, A_t) = Q(S_t, A_t) + \alpha\big(R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)\big) =$

$Current + step_{size} * (observed\ rewards + disctounted\ Q_{next} - current) =$

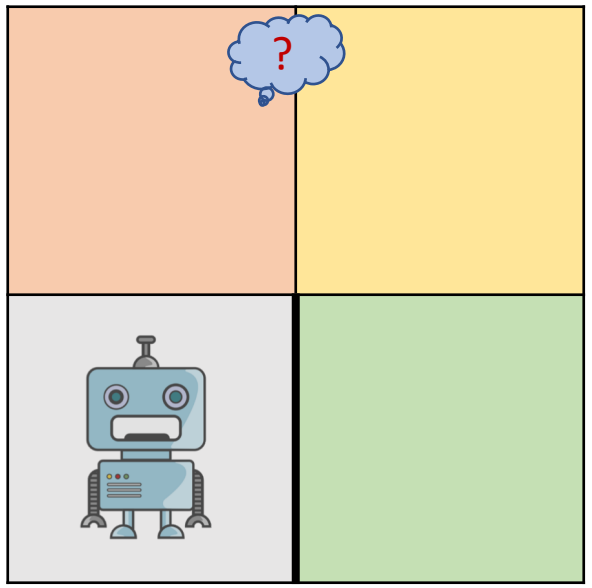
The update in Sarsa is done after each transition from a non-terminal state

Algorithm 13: Sarsa

Input: policy π, positive integer *num_episodes*, small positive fraction α, GLIE $\{\epsilon_i\}$
Output: value function Q ($\approx q_\pi$ if *num_episodes* is large enough)
Initialize Q arbitrarily (e.g., $Q(s,a) = 0$ for all $s \in \mathcal{S}$ and $a \in \mathcal{A}(s)$, and $Q(\textit{terminal-state}, \cdot) = 0$)
for $i \leftarrow 1$ ***to*** *num_episodes* **do**
  $\epsilon \leftarrow \epsilon_i$
  Observe S_0
  Choose action A_0 using policy derived from Q (e.g., ϵ-greedy)
  $t \leftarrow 0$
  repeat
    Take action A_t and observe R_{t+1}, S_{t+1}
    Choose action A_{t+1} using policy derived from Q (e.g., ϵ-greedy)
    $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t))$
    $t \leftarrow t + 1$
  until S_t *is terminal*;
end
return Q

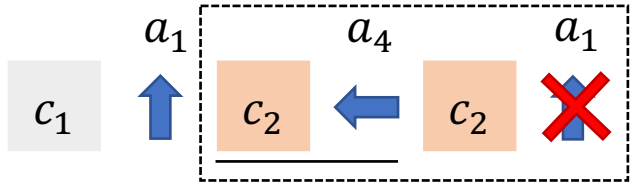
SARSAMAX – Q-Learning

How do we update the table?



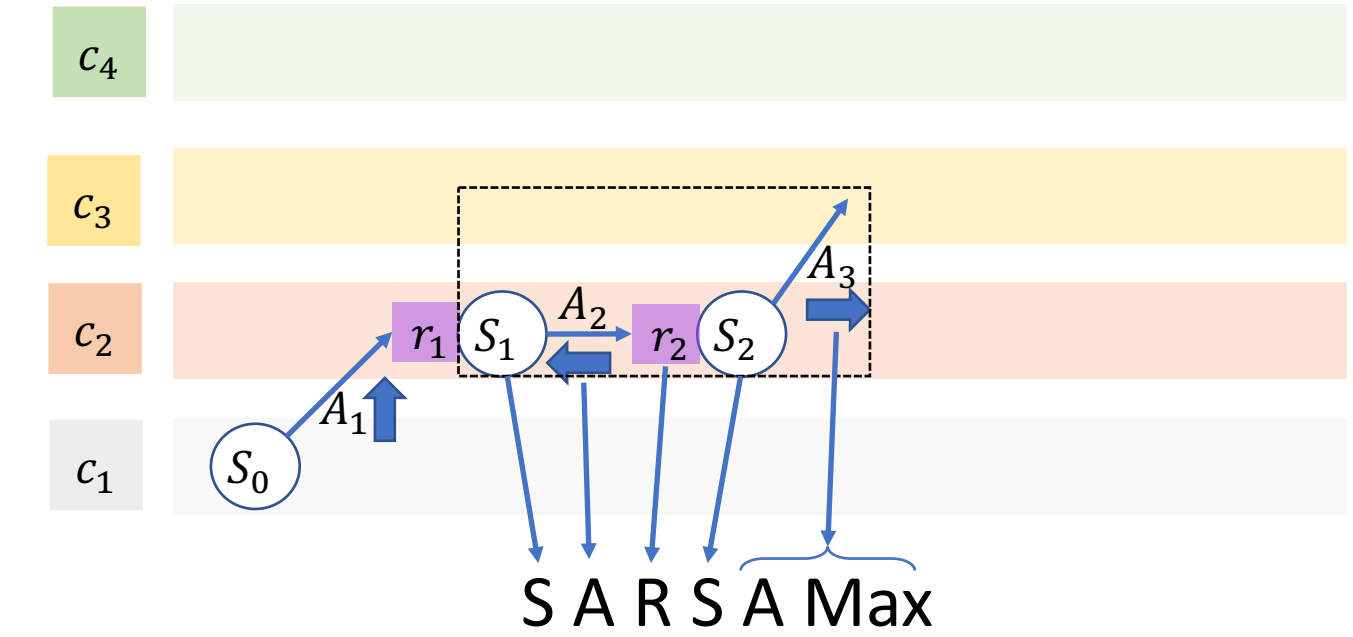
Episode 1:

The Q-Table is initialized with all values set to 0



Update the policy before choosing next action

By choosing the action with the highest Q-value

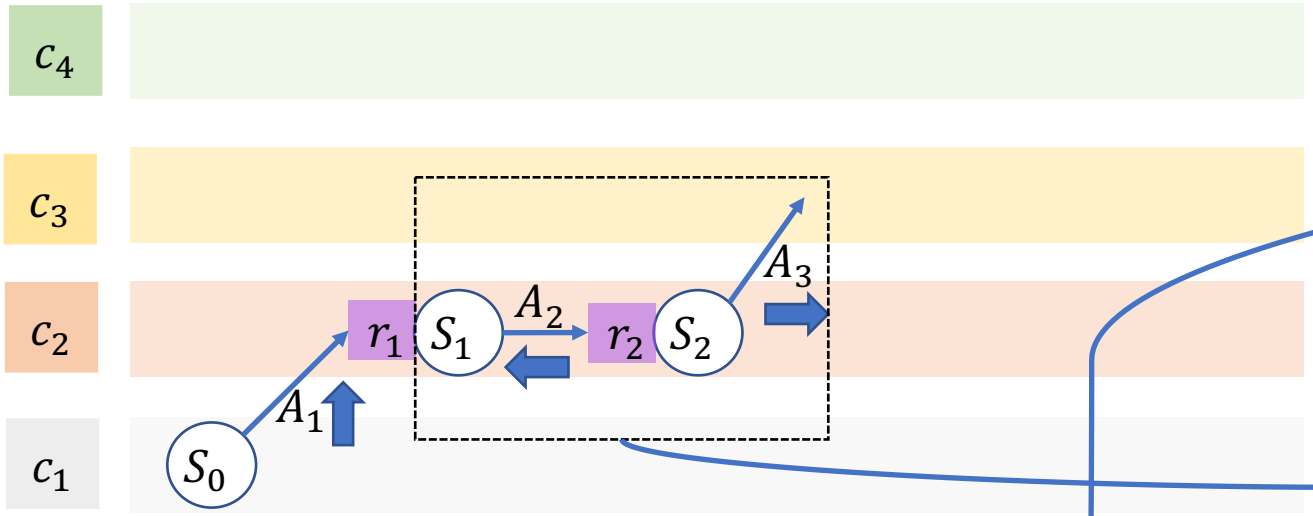


Rewards

r_2	-1

SARSAMAX – Q-Learning

How do we update the table?



Q-Table

	a_1 ↑	a_2 ↓	a_3 ←	a_4 →
c_1	+7	+6	+5	+6
c_2	+8	+7	~~+8~~ +8.02	+9
c_3	+10	+7	+8	+8

Update Rule: $Q(c_2, a_3) = Q(c_2, a_3) + \alpha\big(r_2 + \gamma Q(c_2, a_4) - Q(c_2, a_3)\big) = 8 + \frac{1}{5}(8 + 0.9 * 9 - 8) = 8.02$

General Update Rule: $Q(S_t, A_t) = Q(S_t, A_t) + \alpha\left(R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)\right) =$

$$Current + step_{size} * (observed\ rewards + disctounted\ Q_{of\ action\ that\ maximizes\ Q} - current) =$$

The update in Sarsa pushes the values closer to evaluating whatever the ε-greedy policy is currently following

ON-POLICY

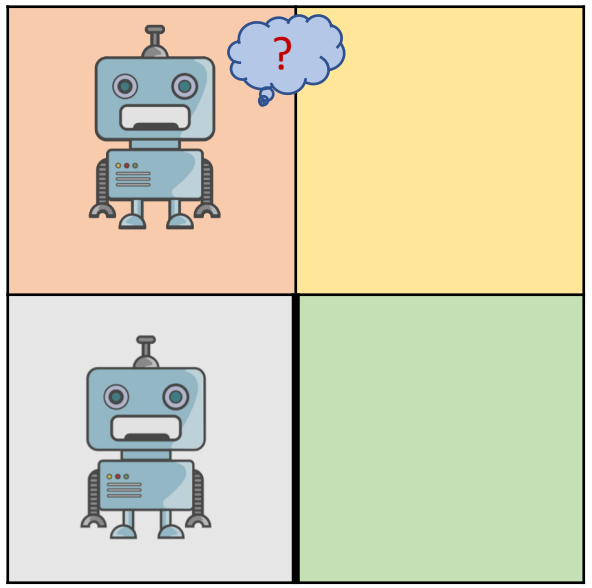
The update in Sarsamax directly attempts to approximate the optimal value function at every time step

OFF-POLICY

Algorithm 14: Sarsamax (Q-Learning)

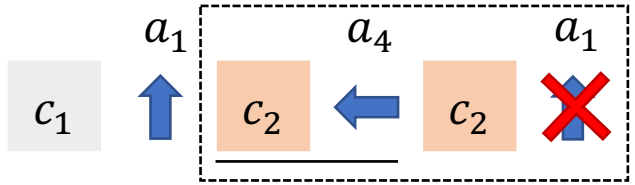
Input: policy π, positive integer *num_episodes*, small positive fraction α, GLIE $\{\epsilon_i\}$
Output: value function Q ($\approx q_\pi$ if *num_episodes* is large enough)
Initialize Q arbitrarily (e.g., $Q(s,a) = 0$ for all $s \in \mathcal{S}$ and $a \in \mathcal{A}(s)$, and $Q(\textit{terminal-state}, \cdot) = 0$)
for $i \leftarrow 1$ ***to*** *num_episodes* **do**
  $\epsilon \leftarrow \epsilon_i$
  Observe S_0
  $t \leftarrow 0$
  repeat
    Choose action A_t using policy derived from Q (e.g., ϵ-greedy)
    Take action A_t and observe R_{t+1}, S_{t+1}
    $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t))$
    $t \leftarrow t + 1$
  until S_t *is terminal*;
end
return Q

Expected SARSA



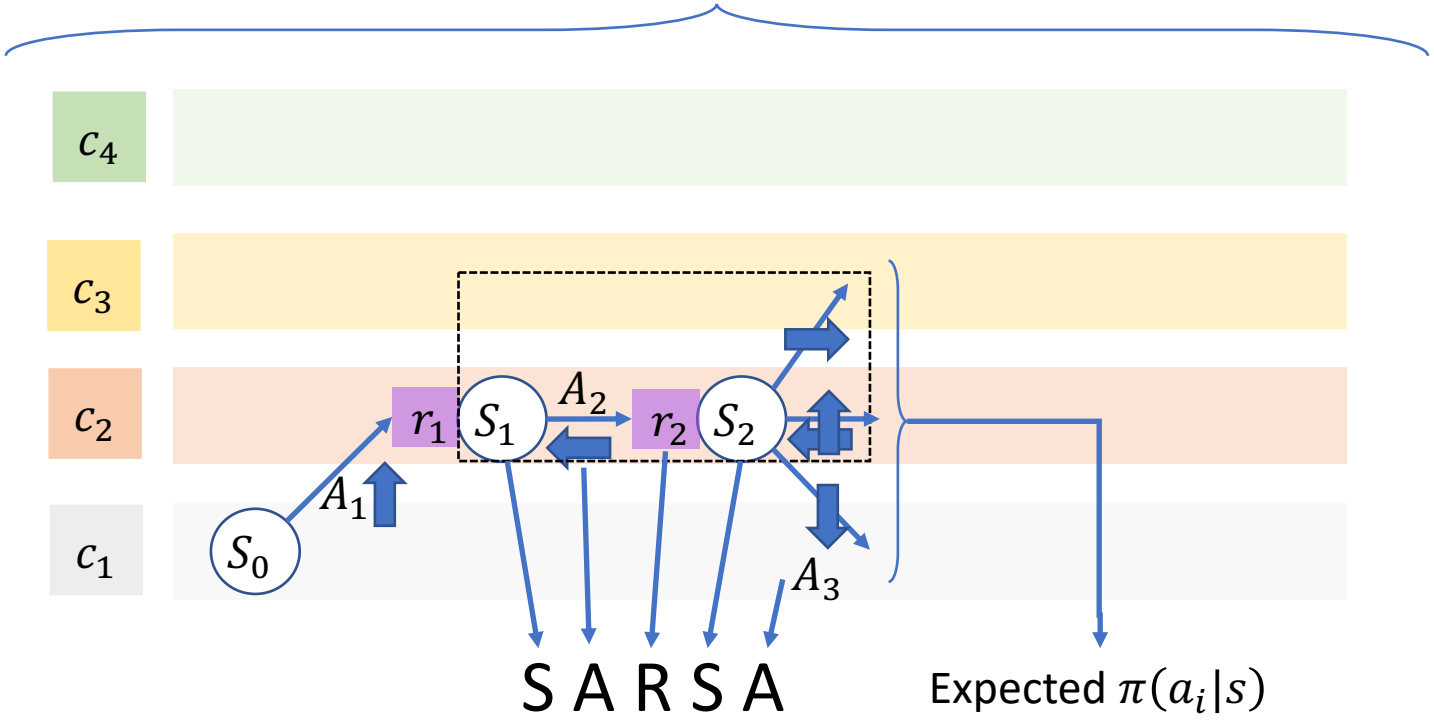
Episode 1:

The Q-Table is initialized with all values set to 0



Also update the policy before choosing next action

Takes into account all actions and its probabilities

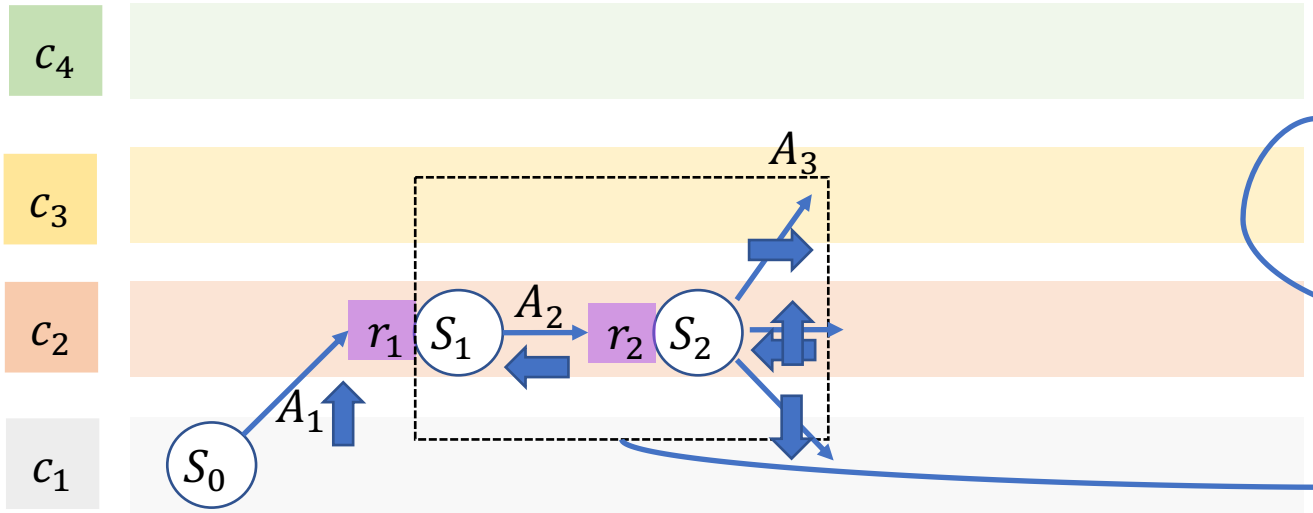


Rewards

r_2	-1

Expected SARSA

How do we update the table?



Q-Table

	a_1 ↑	a_2 ↓	a_3 ←	a_4 →
c_1	+7	+6	+5	+6
c_2	+8	+7	+8 +8.02	+9
c_3	+10	+7	+8	+8

Update Rule: $Q(c_2, a_3) = Q(c_2, a_3) + \alpha\big(r_2 + \gamma\{\pi(a_1|s)Q(c_3, a_1) + \pi(a_2|s)Q(c_3, a_2) + \pi(a_3|s)Q(c_3, a_3) + \pi(a_4|s)Q(c_3, a_4)\} - Q(c_2, a_3)\big) =$

General Update Rule: $Q(S_t, A_t) = Q(S_t, A_t) + \alpha(R_{t+1} - \gamma \sum_a \pi(a|s)\, Q(S_{t+1}, a) - Q(S_t, A_t)) =$

$Current + step_{size} * (observed\ rewards + disctounted\ weighted\ average\ of\ Q_{next} - current) =$