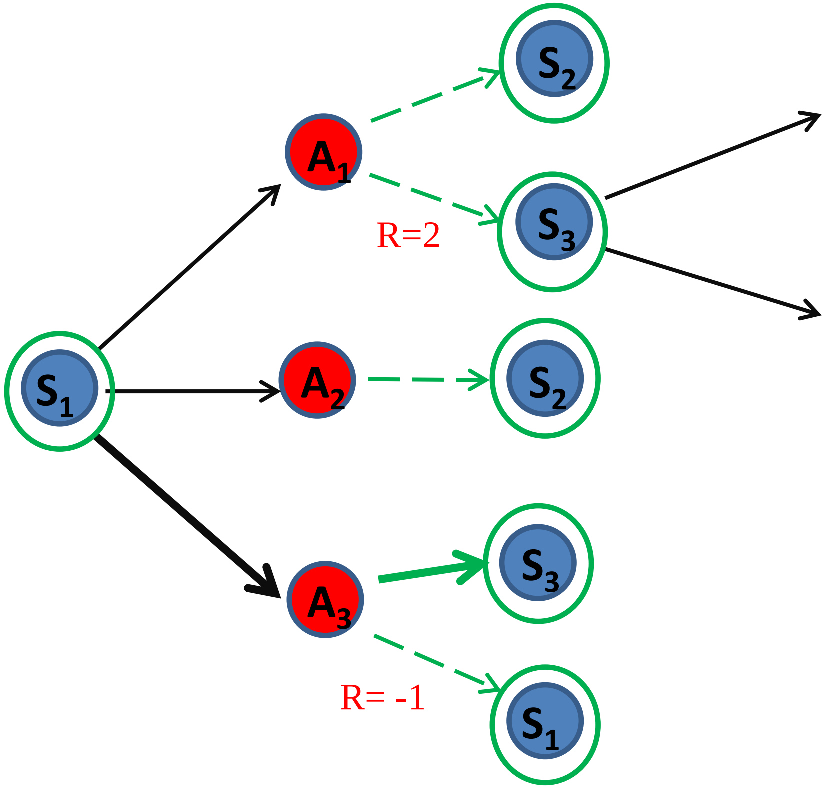
Model-Free Methods

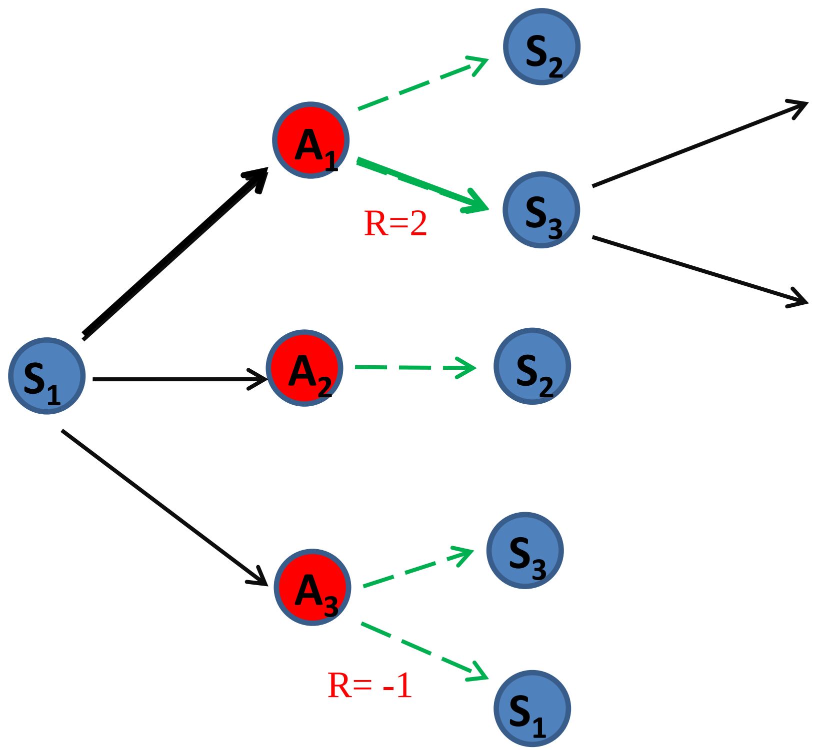
Model-Free Methods

Model-based: use all branches



In model-based we update $V_{\pi}(S)$ using all the possible S'
In model-free we take a step, and update based on this sample

Model-based: use all branches

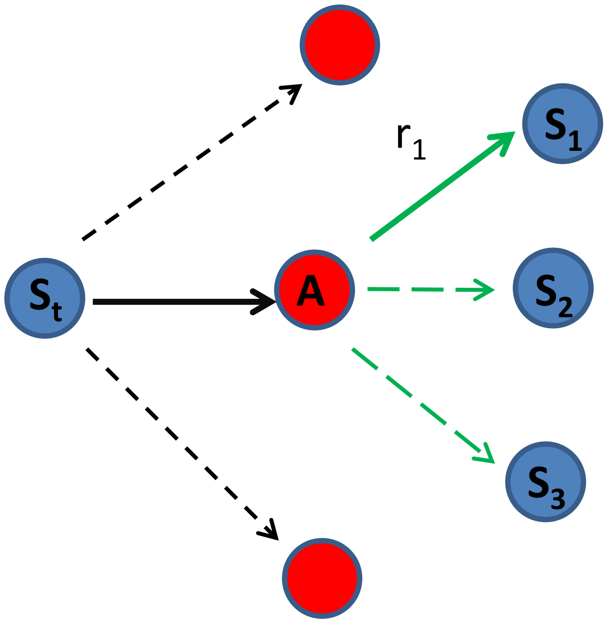


In model-free we take a step, and update based on this sample

$$\langle V \rangle \leftarrow \langle V \rangle + \alpha\,(V - \langle V \rangle)$$

$$V(S_1) \leftarrow V(S_1) + \alpha\,[r + \gamma\, V(S_3) - V(S_1)]$$

On-line: take an action A, ending at S_1



$$V(s_t) \leftarrow V(s_t) + \alpha\Big[r_{t+1} + \gamma V(s_{t+1}) - V(s_t)\Big].$$

$\langle V\rangle \leftarrow \langle V\rangle + \alpha\,(V - \langle V\rangle)$

TD Prediction Algorithm

Terminology: Prediction -- computing V_π (S) for a given π

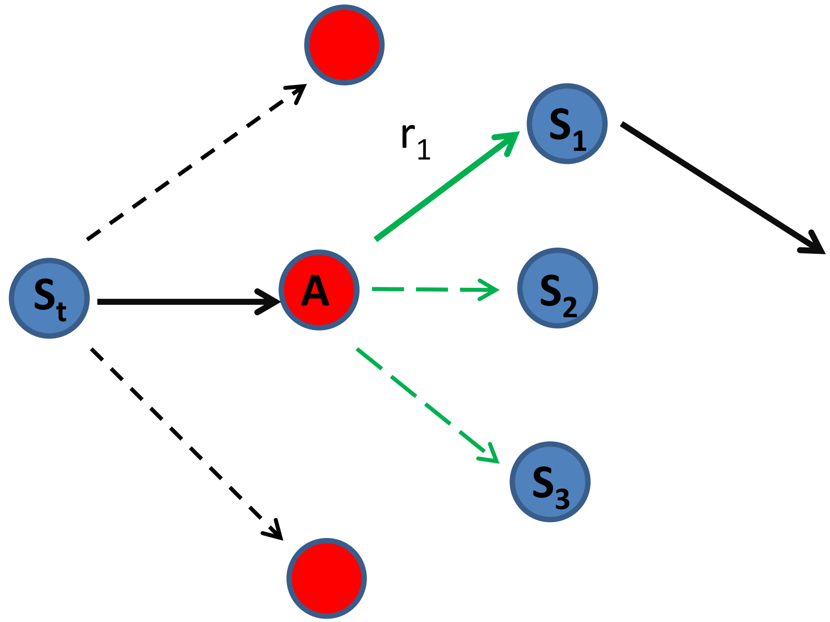
Initialize $V(s)$ arbitrarily, π to the policy to be evaluated
Repeat (for each episode):
    Initialize s
    Repeat (for each step of episode):
        $a \leftarrow$ action given by π for s
        Take action a; observe reward, r, and next state, s'
        $V(s) \leftarrow V(s) + \alpha\big[r + \gamma V(s') - V(s)\big]$
        $s \leftarrow s'$
    until s is terminal

Prediction error: $[r + \gamma V(S') - V(S)]$

Expected : V(S), observed: $r + \gamma V(S')$

Learning a Policy: Exploration problem:

take an action A, ending at S_1



$$V(s_t) \leftarrow V(s_t) + \alpha\Big[r_{t+1} + \gamma V(s_{t+1}) - V(s_t)\Big].$$

Update S_t then update S_1

May never explore the alternative actions to A

From Value to Action

- Based on V(S), action can be selected
- ‘Greedy’ selection is not good enough
  (Select action A with current max expected future reward)
- Need for ‘exploration’
- For example: ‘ε-greedy’
- Max return with p = 1-ε, and with p=ε one of the other actions
- Can be a more complex decision
- Done here in episodes

TD Policy Learning

Initialize $V(s)$ arbitrarily, π to the policy to be evaluated
Repeat (for each episode):
  Initialize s
  Repeat (for each step of episode):
    $a \leftarrow$ action given by ε-greedy
    Take action a; observe reward, r, and next state, s'
    $V(s) \leftarrow V(s) + \alpha\big[r + \gamma V(s') - V(s)\big]$
    $s \leftarrow s'$
  until s is terminal

ε-greedy performs exploration
Can be more complex, e.g. changing ε with time or with conditions

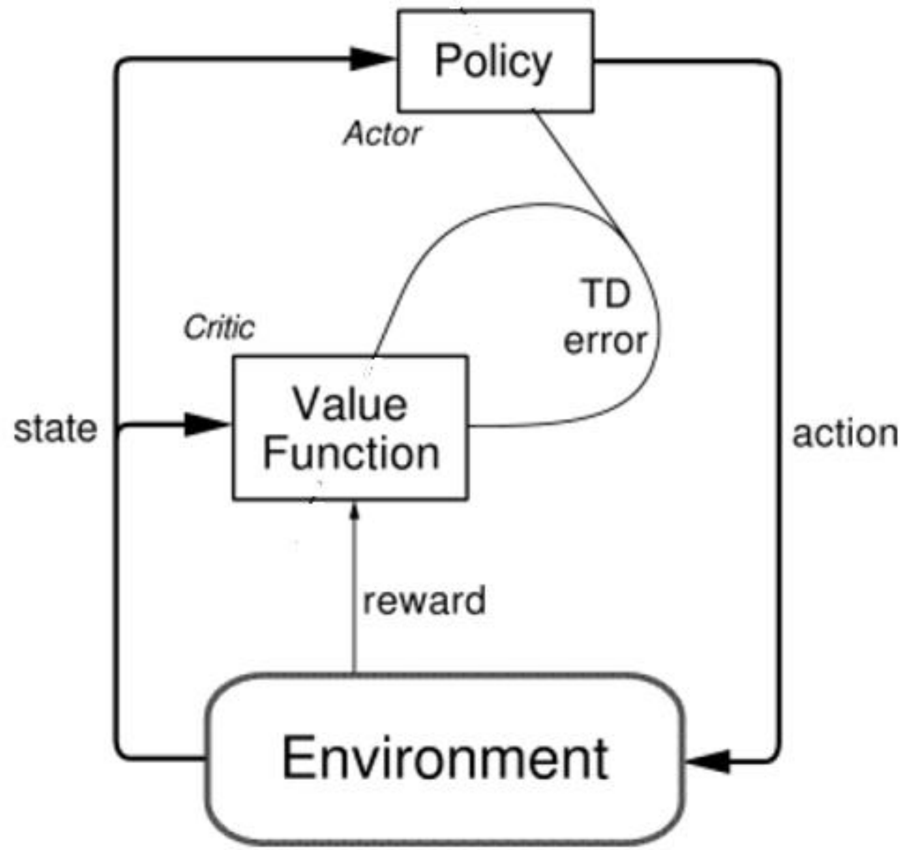
TD 'Actor-Critic'

Terminology: Prediction is the same as policy evaluation. Computing V_π (S)

Initialize $V(s)$ arbitrarily, π to the policy to be evaluated
Repeat (for each episode):
  Initialize s
  Repeat (for each step of episode):
    $a \leftarrow$ action given by 'actor'
    Take action a; observe reward, r, and next state, s'
    $V(s) \leftarrow V(s) + \alpha\big[r + \gamma V(s') - V(s)\big]$
    $s \leftarrow s'$
  until s is terminal

Motivated by brain modeling

‘Actor-critic’ scheme -- standard drawing

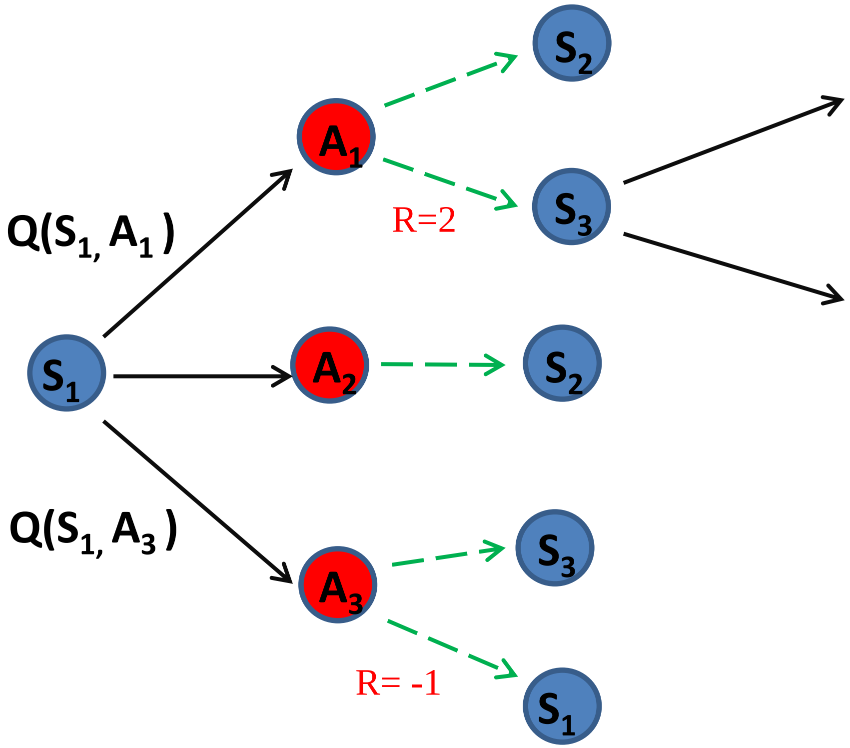


Motivated by brain modeling
(E.g. Ventral striatum is the critic, dorsal striatum is the actor)

Q-learning

- The main algorithm used for model-free RL

Q-values (state-action)



Q_π (S,a) is the expected return starting from S, taking the action a, and thereafter following policy π

Q-value (state-action)

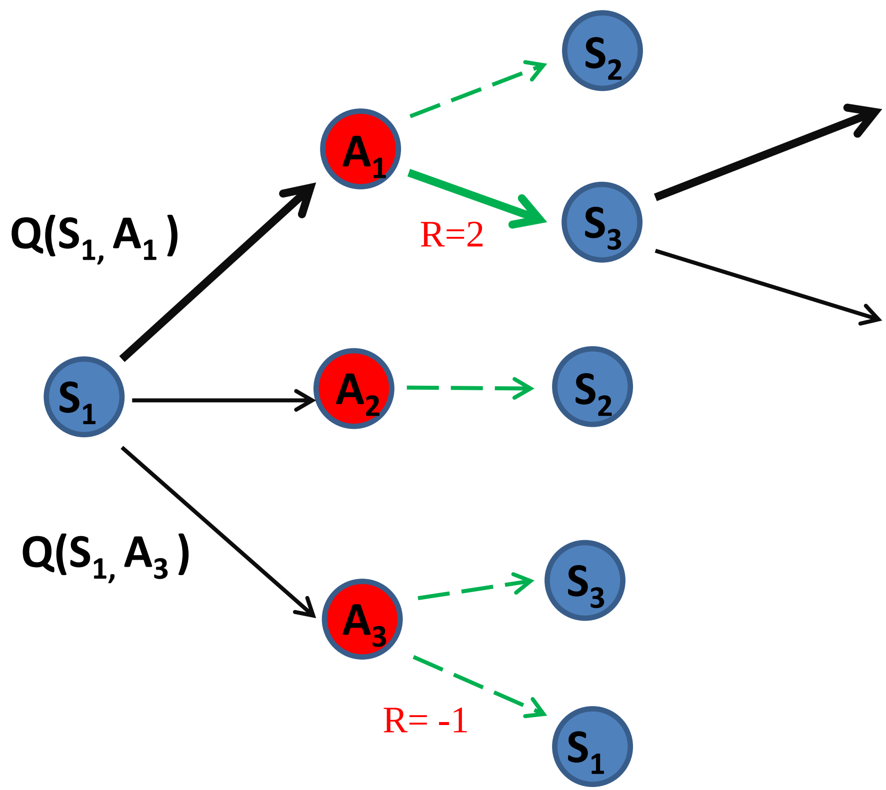
- The same update is done on Q-values rather than on V

- Used in most practical algorithms and some brain models

- Q_π (S,a) is the expected return starting from S, taking the action a, and thereafter following policy π:

$$Q^\pi(s,a) = E_\pi\{R_t | s_t = s, a_t = a\} = E_\pi\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \,\middle|\, s_t = s, a_t = a\right\}.$$

Q-values (state-action)



$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \Big[r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t) \Big].$$

SARSA

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \Big[r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t) \Big].$$

It is called SARSA because it uses s(t) a(t) r(t+1) s(t+1) a(t+1)
A step like this uses the current π, so that each S has its a = π(S)

SARSA RL Algorithm

Initialize $Q(s,a)$ arbitrarily
Repeat (for each episode):
    Initialize s
    Choose a from s using policy derived from Q (e.g., ε-greedy)
    Repeat (for each step of episode):
        Take action a, observe r, s'
        Choose a' from s' using policy derived from Q (e.g., ε-greedy)
        $Q(s,a) \leftarrow Q(s,a) + \alpha\big[r + \gamma Q(s',a') - Q(s,a)\big]$
        $s \leftarrow s'$; $a \leftarrow a'$;
    until s is terminal

Epsilon greedy: with probability epsilon do not select the greedy action, but with equal probability among all actions

On Convergence

- Using episodes:
- Some of the states are '*terminals*'
- When the computation reaches a terminal s, it stops.
- Re-starts at a new state s according to some probability
- At the starting state, each action has a non-zero probability (exploration)
- As the number of episodes goes to infinity, Q(S,A) will converge to Q^*(S,A).