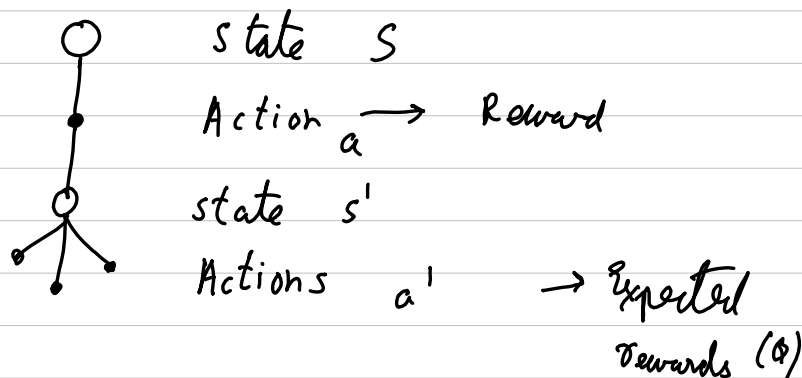


SARSA

TD, we considered only the goodness of next state.

But in SARSA & Q learning, we consider the goodness of actions.

V in TD gets replaced by Q



- ① choose action a wrt policy from Q (ϵ -greedy)
- ② Take a , observe where agent lands s'
- ③ chose a' (from s') wrt policy from Q (ϵ -greedy)

$$Q(s, a) \leftarrow Q(s, a) + \alpha [R + \gamma Q(s', a') - Q(s, a)]$$

γ is assigned to states

TD, state & values are considered
Actions taken are not considered
Which state to go is decided by which state is best.
How good states are

γ is assigned to actions

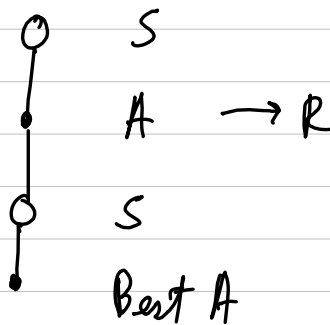
In SARSA, state, Actions & values are considered
Actions are considered

Which state to go is not decided,
which action to take is decided
How good actions are

Q-Learning

- ① choose action a w.r.t policy from Q (ϵ -greedy)
- ② Take a , observe where agent lands s'
- ③ choose a' (from s') w.r.t policy from Q
only greedy

$$Q(s, a) \leftarrow Q(s, a) + \alpha [R + \gamma \max_a Q(s', a) - Q(s, a)]$$



Because Q_L uses Best A for the second action, it can be referred as off policy

Q_L favours exploitation in updates useful in deterministic environments but may lead to suboptimal policies in stochastic environments

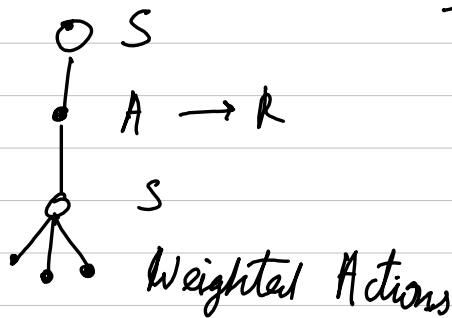
Q_L can converge faster than SARSA & expected SARSA in deterministic environments

But for stochastic environments, it can lead to oscillations, and updates chase fluctuating high reward estimates.

Expected SARSA

- ① choose action a wrt policy from Q (ϵ -greedy)
- ② Take a , observe where agent lands s'
- ③ Take a' (from s') wrt policy from π
weighted
avg of all

$$Q(s, a) \leftarrow Q(s, a) + \alpha \left[R + \gamma \sum_{a'} \pi(a' | s_{t+1}) \times Q(s_{t+1}, a') - Q(s_t, A_t) \right]$$



We weigh the expected rewards by the probability of choosing the reward

Expected SARSA is more balanced in exploration
Exploitation

Middle between Q_L & SARSA

Double Q Learning

Use 2 Q functions Q_1, Q_2

- ① choose action a wrt policy from $Q_1 + Q_2$ (ϵ -greedy)
- ② Take a , observe where agent lands s'
- ③ choose a' (from s') wrt policy from Q_i (ϵ -greedy)

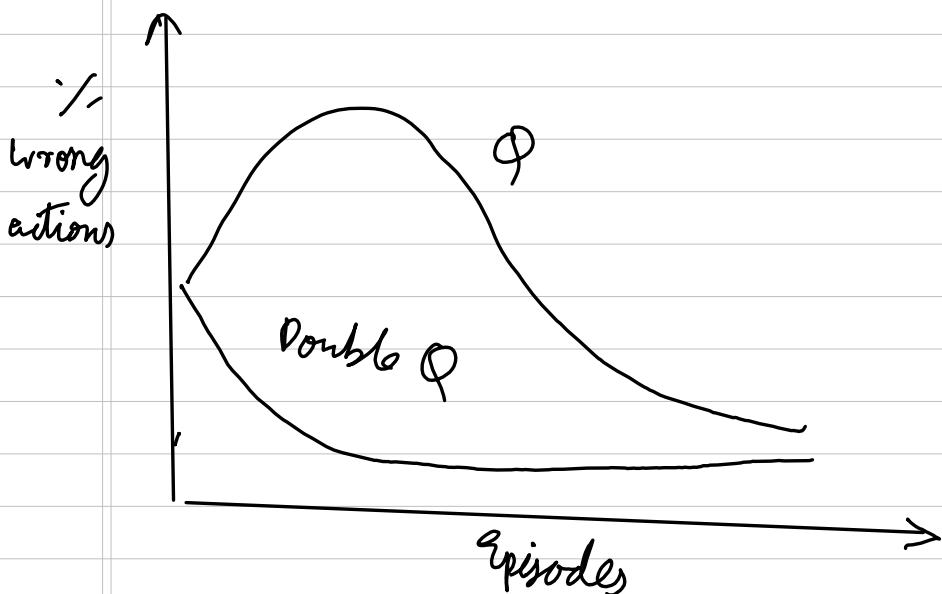
0.5 Prob \rightarrow

$$Q_1(s, a) \leftarrow Q_1(s, a) + \alpha [R + \gamma \underbrace{Q_1(s', a')}_{\text{chose max from } Q_2} - Q_1(s, a)]$$

0.5 Prob \rightarrow

$$Q_2(s, a) \leftarrow Q_2(s, a) + \alpha [R + \gamma \underbrace{Q_2(s', a')}_{\text{chose max from } Q_1} - Q_2(s, a)]$$

Q vs Double Q



Problem of Model free RL (MC, TD, Q, SARSA)

- 1) Rewards may be sparse
- 2) Q table needs matrix of $|A| \times |S|$ large table
- 3) If a state is never visited then Q estimate is absent

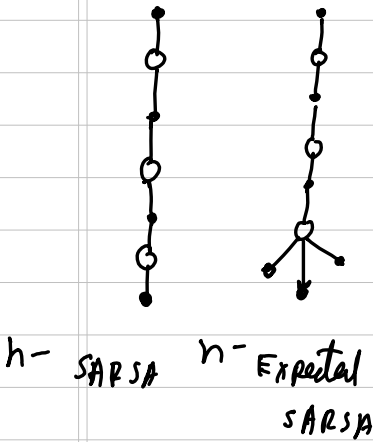
In order to overcome this problem of lookup tables, function approximation RL must be used.

		Lookup Table	function Approximator
Policy			Policy Gradient Actor Critic
Control	State Action Value	MC Control SARSA SARSA backward Q learning	Deep Q networks
Prediction	state Value	MC Prediction TDL TD (λ) backward	

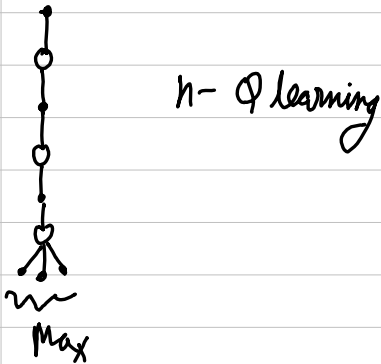
n step SARSA

Just like TD, but
start & end with actions

We are storing the last
n states, actions and
rewards



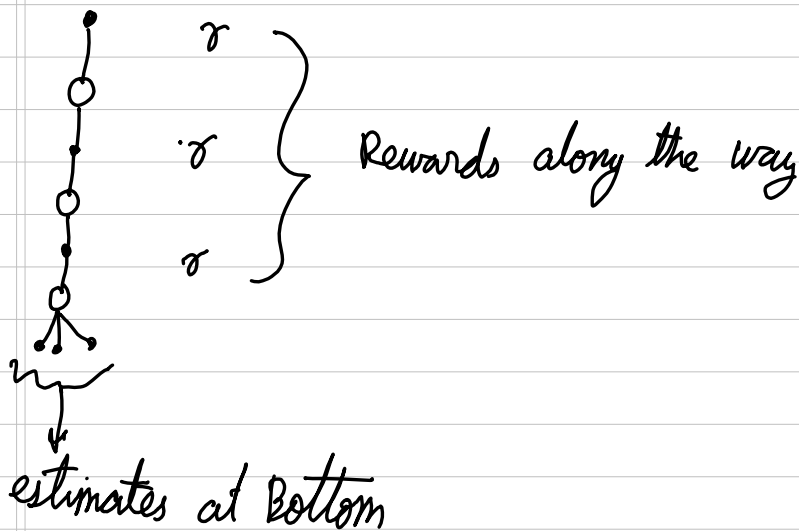
Requires a lot of Bookkeeping



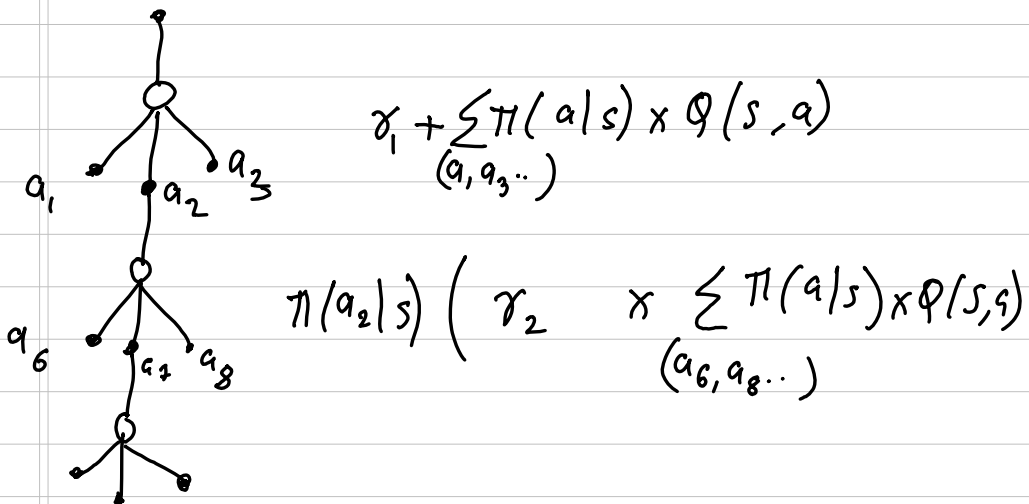
$$G_t = \underbrace{r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \dots + \gamma^n Q(s_{t+n}, a_{t+n})}_{\downarrow \text{Real}}$$

n - step Backup tree

so far the target for the update node was combining the rewards along the way and estimates at bottom



Now we will consider the actions not taken as well




each first level action has weight $\pi(a|s_{t+1})$

Action taken does not contribute

Probability of action taken contributes at next level rewards

A hand-drawn diagram of a molecule. It consists of two circles, one on the left and one on the right, connected by a horizontal line. In the center of this line is a solid black dot. The circles are drawn with a single continuous line, and the connecting line is also a single continuous line. The entire diagram is drawn in black ink on a white background.

Random



π weights

Sum