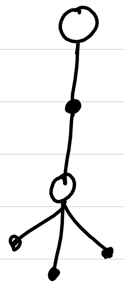
SARSA

TD(λ), 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



state s

Action a → Reward

state s'

Actions a' → Expected rewards (Q)

① choose action a w.r.t. policy from Q (ε-greedy)

② Take a, observe where agent lands s'

③ Chose a' (from s') w.r.t. policy from Q (ε-greedy)

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

γ is assigned to states

TD(λ), 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

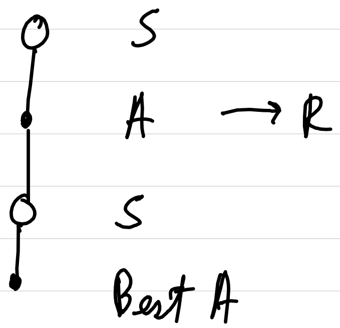
Q - Learning

① choose action a w.r.t policy from Q (ε-greedy)

② Take a, observe where agent lands s'

③ Chose a' (from s') w.r.t policy from Q
Only greedy

$$Q(s,a) \leftarrow Q(s,a) + \alpha\left[R + \gamma \max_{a} Q(s',a') - Q(s,a)\right]$$



Because QL 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.

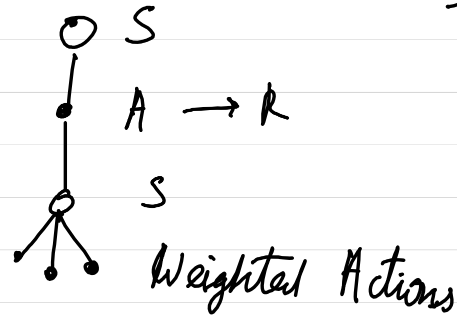
Expected SARSA

① choose action a w r t policy from Q (ε-greedy)

② Take a, observe where agent lands s'

③ Take weighted avg of all a' (from s') w r t policy from π

$$Q(s, a) \leftarrow Q(s, a) + \alpha \Big[R + \gamma \sum \pi(a \mid s_{t+1}) \times Q(s_{t+1}, a) - Q(s_t, A_t) \Big]$$



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. Chose a' (from s') wrt policy from Q_i (ε-greedy)

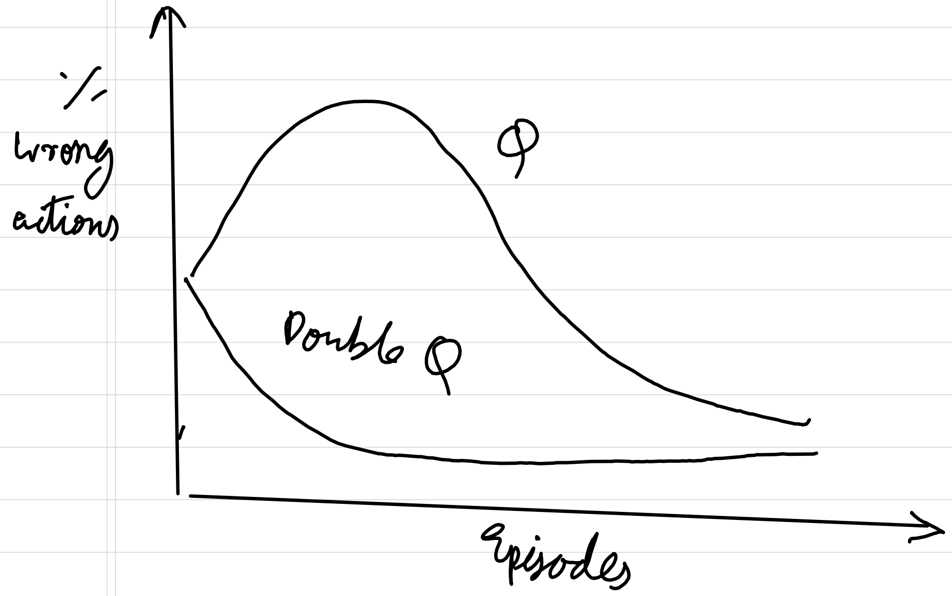
0.5 Prob →

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

0.5 Prob →

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

Q vs Double Q



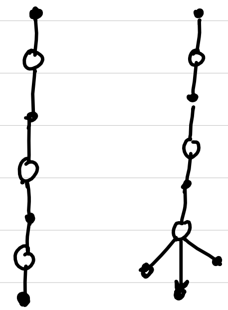
Problem of Model free RL
(MC, TDL, Q, SARSA)

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

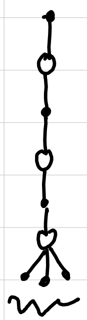


n- SARSA n- Expected SARSA

Just like TD but start & end with actions

We are storing the last n states, actions and rewards

Requires a lot of Bookkeeping



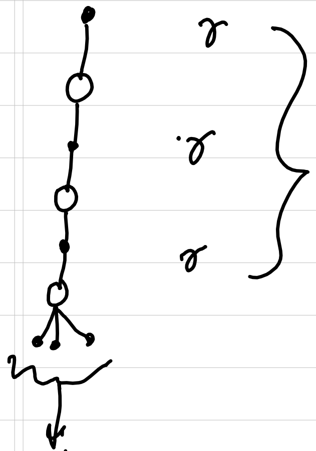
Max

n- Q learning

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

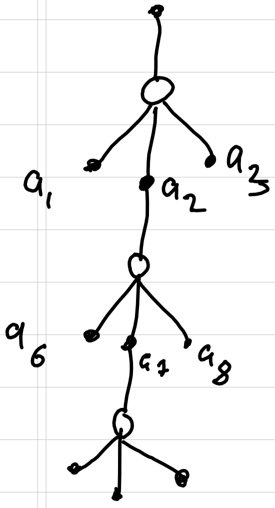
n - step Backup tree

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



estimates at Bottom

Now we will consider the actions not taken as well



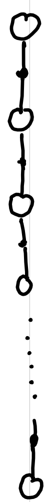
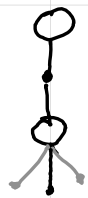
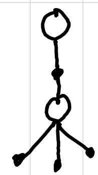
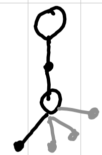
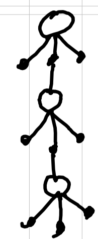
$$r_1 + \sum_{(a_1, a_3 ..)} \pi(a|s) \times Q(s,a)$$

$$\pi(a_2|s) \Big(r_2 \times \sum_{(a_6, a_8 ..)} \pi(a|s) \times Q(s,a)$$

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

MC	TD	SARSA	Expected SARSA	Q Learning	Tree Backup
					
		random	π weighted sum	Best	