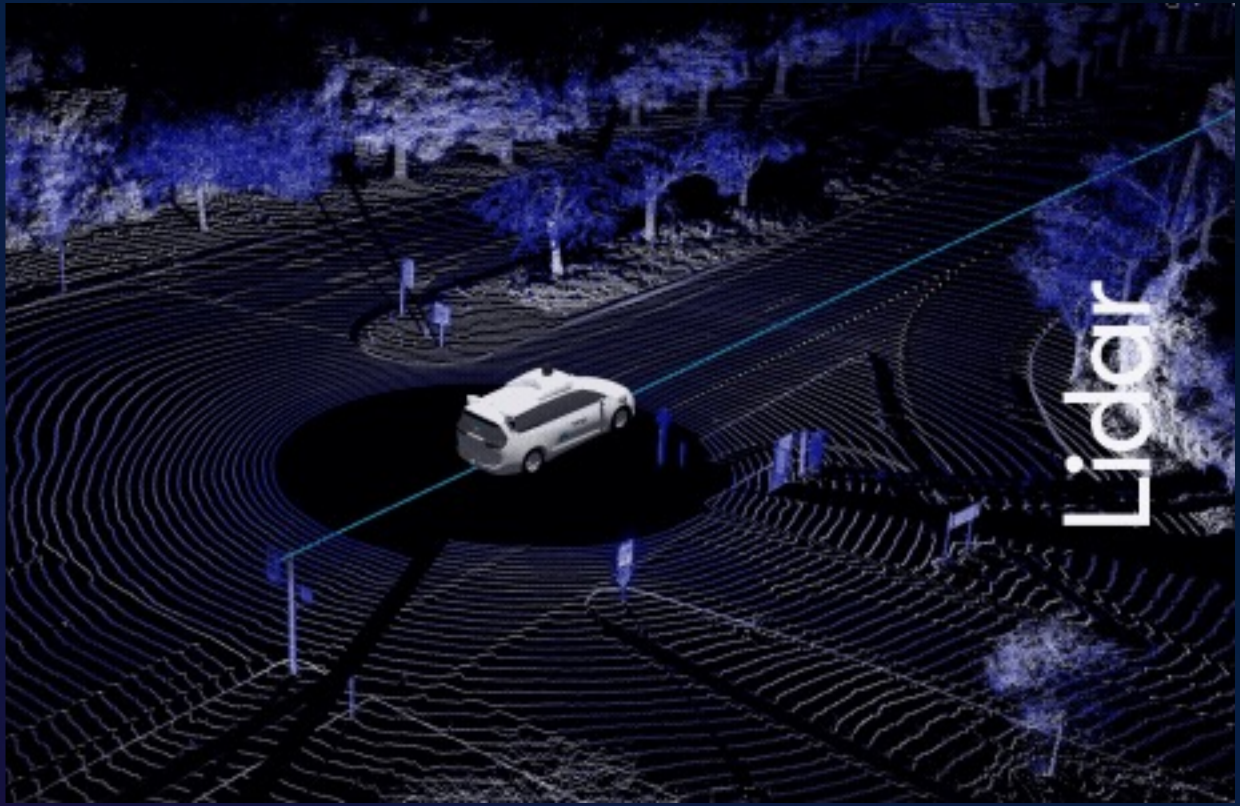
RECAP

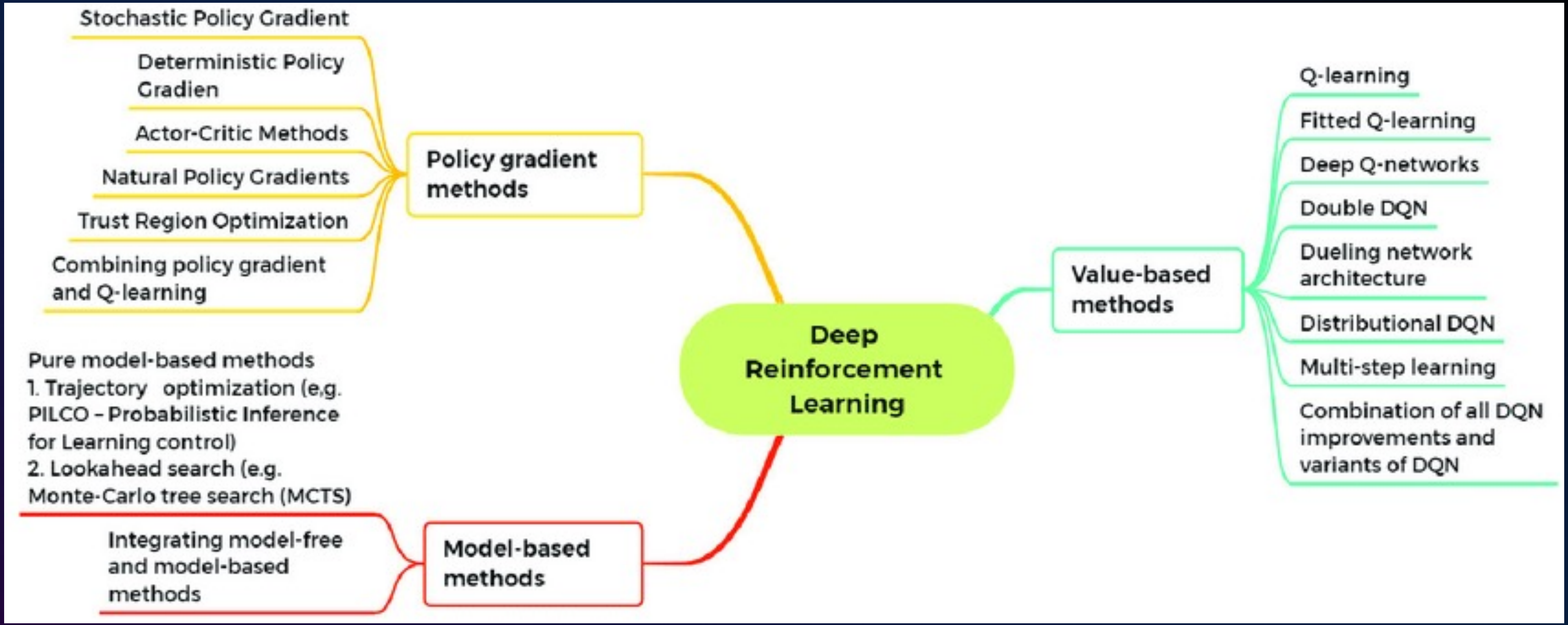
Value based learning algorithms

Irshad Chohan

Principal Solutions Architect
AWS India


Lidar

RL – Learning algorithms



RL algorithms

Value based learning : Find optimal Q(s,a) / Q(s)

Policy based learning: Find optimal π(s)

Bellman equation:

$$V^{\pi}(s) = \sum_{a} \pi(s,a) \sum_{s'} P^{a}_{ss'} \left[r^{a}_{ss'} + \gamma V^{\pi}(s') \right] = \sum_{a} \pi(s,a) Q^{\pi}(s,a)$$

Value-Based Reinforcement Learning

- finding the optimal value function allows deriving the optimal policy.
- Temporal-difference learning is commonly used to update value estimates.
- Algorithms like Q-learning and SARSA are value-based approaches.

So in summary, policy-based methods directly optimize the policy while value-based techniques aim to find the optimal value function, which in turn provides the best policy. Both can achieve the end goal of maximizing rewards over time.

Value functions

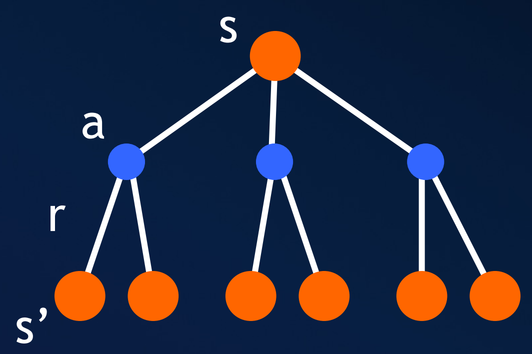
state value function: $V^{\pi}(s)$

expected return when starting in *s* and following π

state-action value function: $Q^{\pi}(s,a)$

expected return when starting in *s*, performing *a*, and following π

Q(s,a) can be derived from V(s) using the Bellman equation for Q-values



useful for finding the optimal policy

can estimate from experience

pick the best action using $Q^{\pi}(s,a)$

Bellman equation $V^{\pi}(s) = \sum_a \pi(s,a) \sum_{s'} P^a_{ss'} \left[r^a_{ss'} + \gamma V^{\pi}(s') \right] = \sum_a \pi(s,a) Q^{\pi}(s,a)$

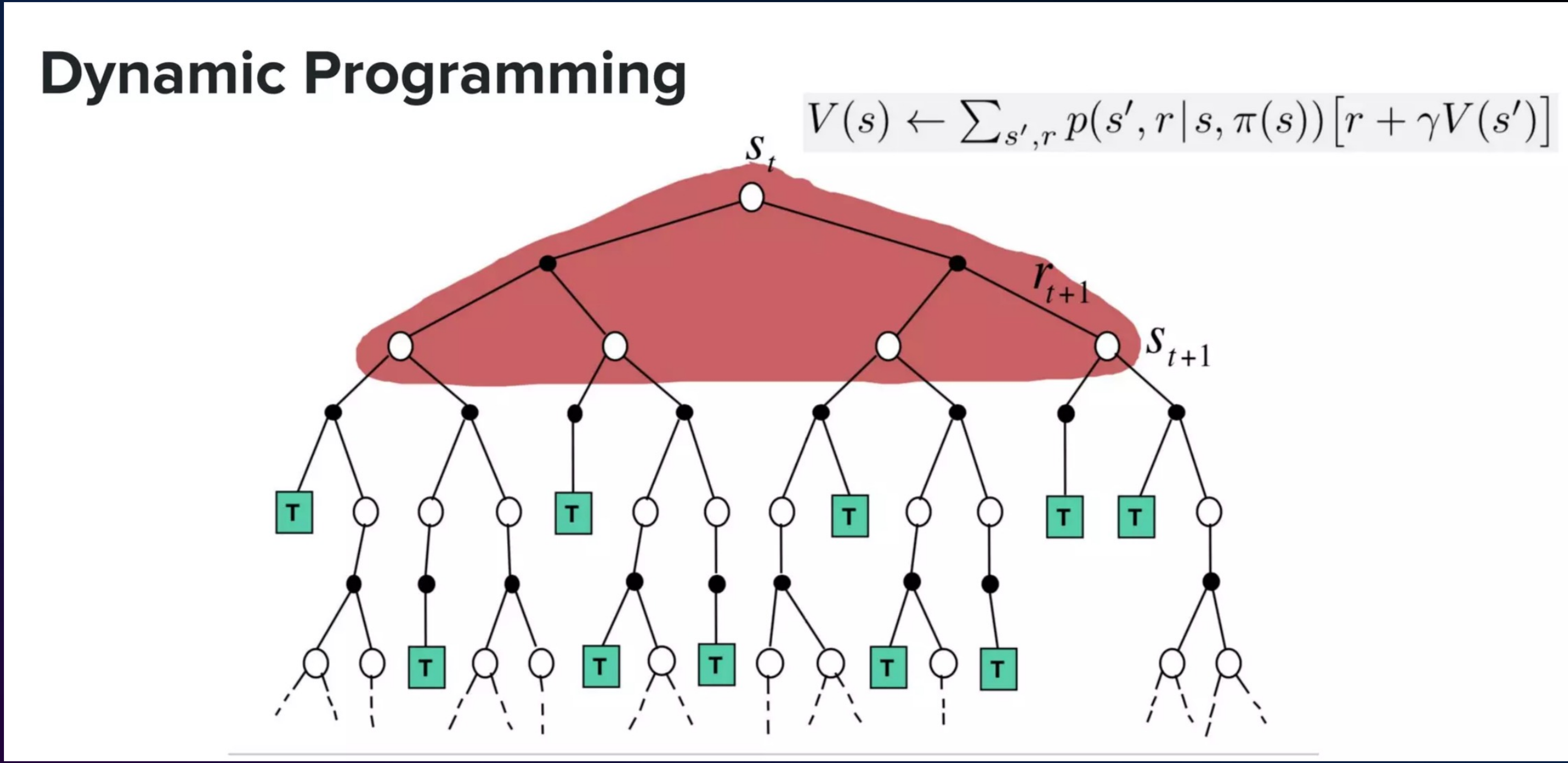
Key algorithms that use both policy and value functions

- **Q-Learning:** Finds optimal policy while estimating value of state-action pairs

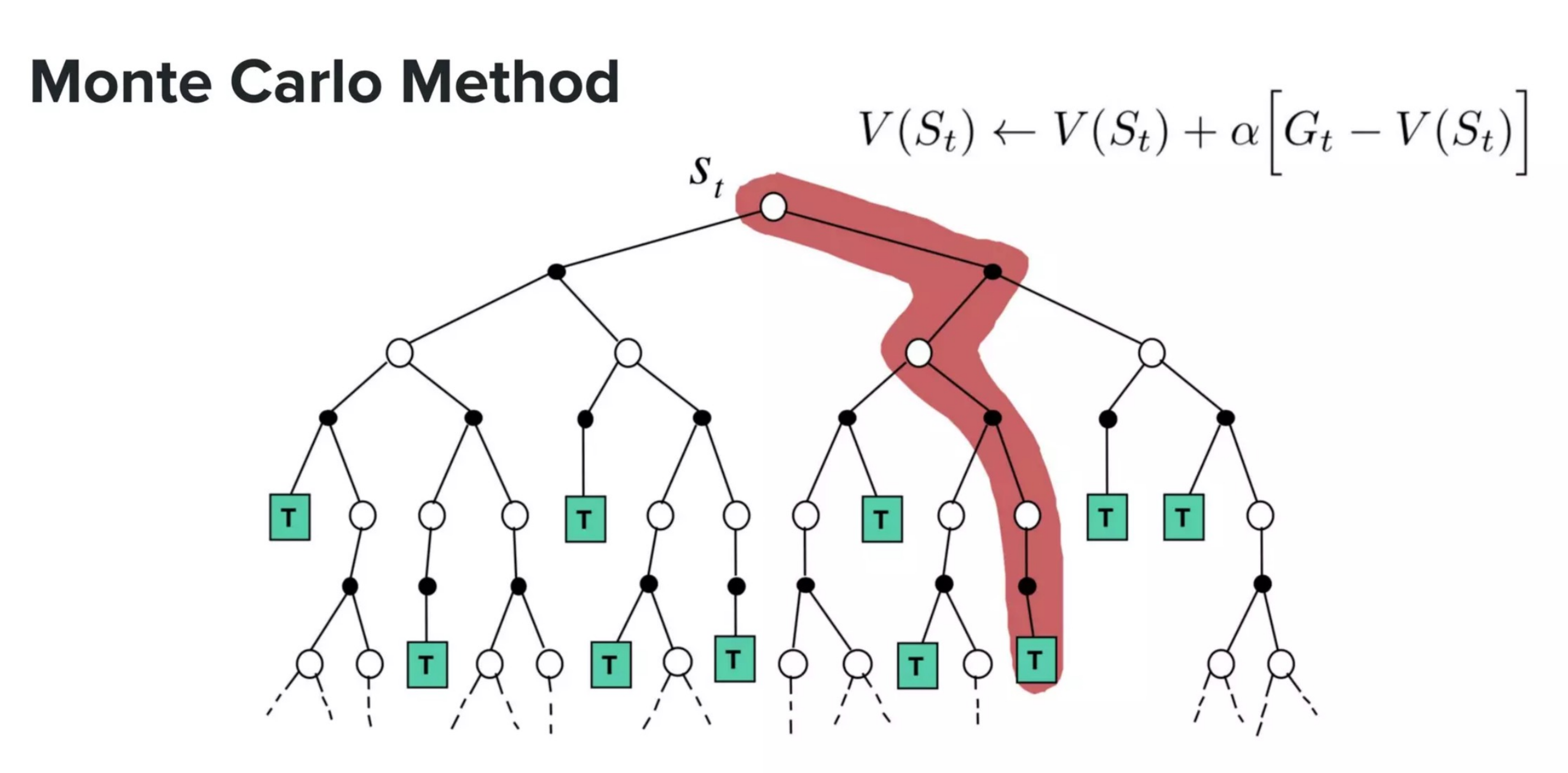
- **SARSA:** On-policy method that learns state-action values to determine policy

- **Actor-Critic:** Has separate policy network (Actor) and value network (Critic)

Dynamic Programming



Monte Carlo Method



DP & MC

Dynamic Programming

- Update per step, using bootstrapping
- Need model
- Computation cost

Monte Carlo method

- Update per episode
- Model-free
- Hard to be applied to continuous task

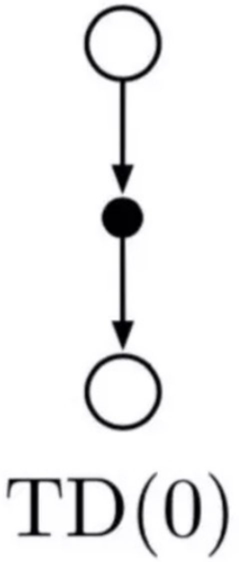
Let's combine their advantages!!!

Temporal-difference learning

Different from MC method, each sample of TD learning is just **a few steps**, not the whole trajectory. TD learning bases its update in part on an existing estimate, so it's also a ***bootstrapping*** method.

TD method is an **policy evaluation method** (without control), which is used to predict the value of fixed policy.

$$V(S_t) \leftarrow V(S_t) + \alpha\Big[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)\Big]$$



backup diagram of TD(0)

Temporal-difference learning

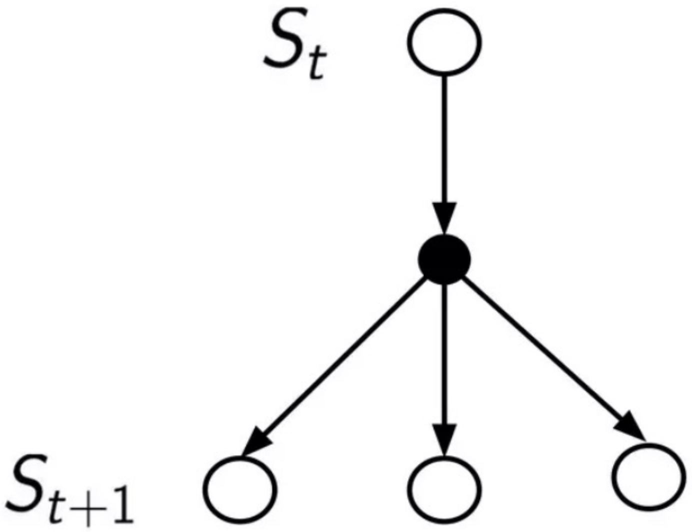
In model-free RL, we use samples to estimate the expectation of future total rewards.

sample 1: $R(S_t, A_t, S_{t+1}) + \gamma V_k(S_{t+1})$

sample 2: $R(S_t, A_t, S_{t+1}) + \gamma V_k(S_{t+1})$

...

sample n: $R(S_t, A_t, S_{t+1}) + \gamma V_k(S_{t+1})$



Temporal-difference learning

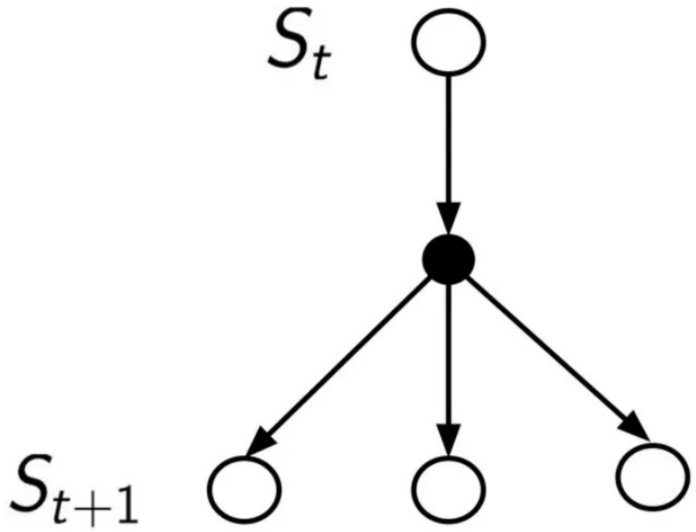
In model-free RL, we use samples to estimate the expectation of future total rewards.

sample 1: $R(S_t, A_t, S_{t+1}) + \gamma V_k(S_{t+1})$

sample 2: $R(S_t, A_t, S_{t+1}) + \gamma V_k(S_{t+1})$

...

sample n: $R(S_t, A_t, S_{t+1}) + \gamma V_k(S_{t+1})$



Temporal-difference learning

- Model-free
- Online learning (fully incremental method)
  - Can be applied to continuous task
- Better convergence time
  - Faster than Monte Carlo method

Temporal-difference learning

Different from MC method, each sample of TD learning is just **few steps.** Not the whole trajectory.

TD learning bases it's update in part on existing estimate, so it's also a bootstrapping method.

TD method is an **policy evaluation method** (without control), which is used to predict the value of fixed policy.

We **don't rewind time** to get sample after sample from s_t (unlike Monte Carlo), we use **weightage format**

$$V(s) = (1 - \alpha)V(s) + (\alpha)sample$$

which is equal to

$$V(s) = V(s) + \alpha(sample - V(s))$$

The α can be a kind of learning rate.

Temporal-difference with Control

SARSA

- Inspired by policy iteration, an on-policy TD control
- Replace value function by action-value function
- Behaviour policy is same as Target policy

$$Q_{\boxed{\pi}}(s,a) \leftarrow \sum_{s'} P(s'|s,\boxed{\pi(s)})\left[R(s,\boxed{\pi(s)},s') + \gamma Q_{\boxed{\pi}}(s',\boxed{\pi(s')})\right]$$

- In model-free method, we don’t know the transition probability. Hence we use experience sample.
- Here that is, SARSA (s,a,r,s’,a’)
- Sarsa update

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

Sarsa

Sarsa (on-policy TD control) for estimating $Q \approx q_*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(\textit{terminal}, \cdot) = 0$

Loop for each episode:
  Initialize S
  Choose A from S using policy derived from Q (e.g., ε-greedy)
  Loop 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

On-policy!

Temporal-difference with Control

Q-learning

- Inspired by value iteration

$$Q(s,a) \leftarrow \sum_{s'} P(s'|s,a)\left[R(s,a,s') + \gamma V^*(s')\right]$$

- Q-learning update

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

- Offline policy TD control

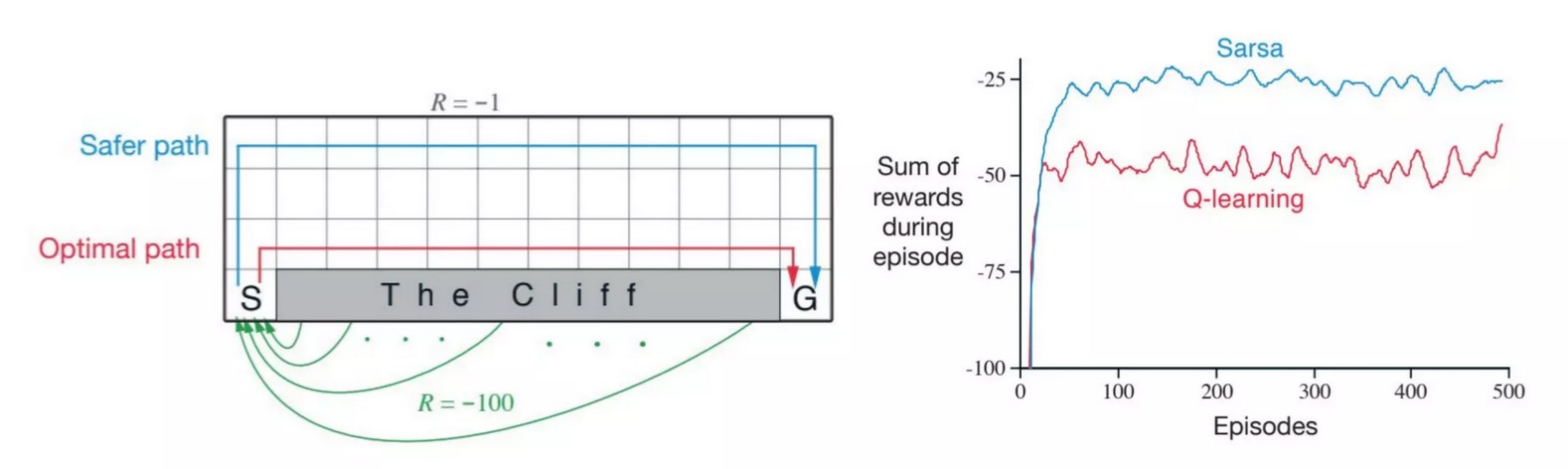
Q-learning

Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(terminal, \cdot) = 0$

Loop for each episode:
  Initialize S
  Loop for each step of episode:
    Choose A from S using policy derived from Q (e.g., ε-greedy)
    Take action A, observe R, S'
    $\boxed{Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma \max_a Q(S', a) - Q(S, A)\big]}$
    $S \leftarrow S'$
  until S is terminal

SARSA vs Q-Learning



TD Demo

- Present a simple grid world example to illustrate both algorithms
- Demonstrate how Q-Learning and SARSA perform in this environment
- Discuss the differences in learned policies between the two algorithms
- Show visualizations of value functions and policies

TD Demo

Hands-on Exercise

- Provide a simple environment (e.g., a small grid world or CartPole)
- Guide through implementing either Q-Learning or SARSA
- Encourage experimentation with different hyperparameters
- Discuss how to evaluate and visualize the results

Quiz time::

Between Q-Learning and SARSA which one is faster to converge?

a) Q-Learning

b) SARSA

Between Q-Learning and SARSA which one is faster to converge?

a) Q-Learning

b) SARSA

How does TD learning differ from Monte Carlo methods?

A. TD learning updates estimates based on current states, while Monte Carlo uses complete episodes

B. TD learning does not use rewards

C. Monte Carlo methods update estimates every step

D. TD learning requires complete knowledge of the environment

What is the difference between Value based and Policy based algorithms?

A. Value-based estimates value functions; Policy-based optimizes policies.

B. Value-based updates policies; Policy-based updates value functions.

C. Value-based uses exploration; Policy-based uses exploitation.

D. Value-based handles discrete states; Policy-based handles continuous states.

Thank you!

For Tomorrow!

What is RLHF? Where in the world it is getting used?

What is Actor-critic method?

What RL algorithm is being used by AlphaGo?