REINFORCEMENT LEARNING

FUNDAMENTALS + APPLICATIONS

WEEK 5

TEMPORAL DIFFERENCE LEARNING

Chapter 6

stimulus, s, called $V(s)$ (for value). Also assume that these predictions determine the animal's conditioned response to whichever stimulus is observed. Then upon observing stimulus s_k (e.g., the bell on trial k) and receiving a reward on that trial, r_k, the prediction error is

$$\delta_k = r_k - V_k(s_k) \tag{15.1}$$

As we will see below, this prediction error (with further refinements) appears to be carried by dopaminergic neurons (Houk *et al.*, 1995; Montague et al., 1996; Schultz et al., 1997).

The animal then updates the prediction in the direction of the prediction error, so as to reduce it. Thus, the predicted value on the next trial, $k+1$, of the stimulus s_k is:

$$V_{k+1}(s_k) = V_k(s_k) + \alpha \cdot \delta_k \tag{15.2}$$

Most methods we have covered so far

are closely related.

They only differ in how they define

reality.

Look, and see.

THE UNDERLYING FORM

Recall:

Incremental Mean

New Mean = Old Mean + Step Size • [New Data - Old Mean]

Gradient Descent

$$\text{New Parameter} = \text{Old Parameter} - \text{Step Size} \cdot \frac{\partial(\text{Reality} - \text{Expectation})}{\partial \text{Old Parameter}}$$

THE UNDERLYING FORM

New Estimate = *Old Estimate* + *Step Size* • [*Target* - *Old Estimate*]

$$\textit{New Expectation} = \textit{Expectation} + \textit{Step Size} \cdot \underbrace{[\ \textit{Reality} - \textit{Expectation}\ ]}_{\textbf{Prediction Error}}$$

K-ARMED BANDITS

New Expectation = *Expectation* + *Step Size* • [*Reality* - *Expectation*]

$$q(s) = q(s) + \alpha[R - q(s)]$$

MONTE CARLO METHODS

New Expectation = *Expectation* + *Step Size* • [*Reality* - *Expectation*]

$$V(s) = V(s) + \alpha[G - V(s)]$$

$$Q(s,a) = Q(s,a) + \alpha[G - Q(s,a)]$$

$$V_{n+1} = V_n + \frac{W_n}{C_n}[G_n - V_n]$$

TD-LEARNING METHODS

New Expectation = *Expectation* + *Step Size* • [*Reality* - *Expectation*]

TD(0)	$V(s) = V(s) + \alpha[R + \gamma V(s') - V(s)]$
SARSA	$Q(s,a) = Q(s,a) + \alpha[R + \gamma Q(s',a') - Q(s,a)]$
Q-Learning	$Q(s,a) = Q(s,a) + \alpha[R + \gamma \max Q(s',a) - Q(s,a)]$
Expected SARSA	$Q(s,a) = Q(s,a) + \alpha[R + \gamma \sum \pi(a \mid s') Q(s',a) - Q(s,a)]$

TD(0)

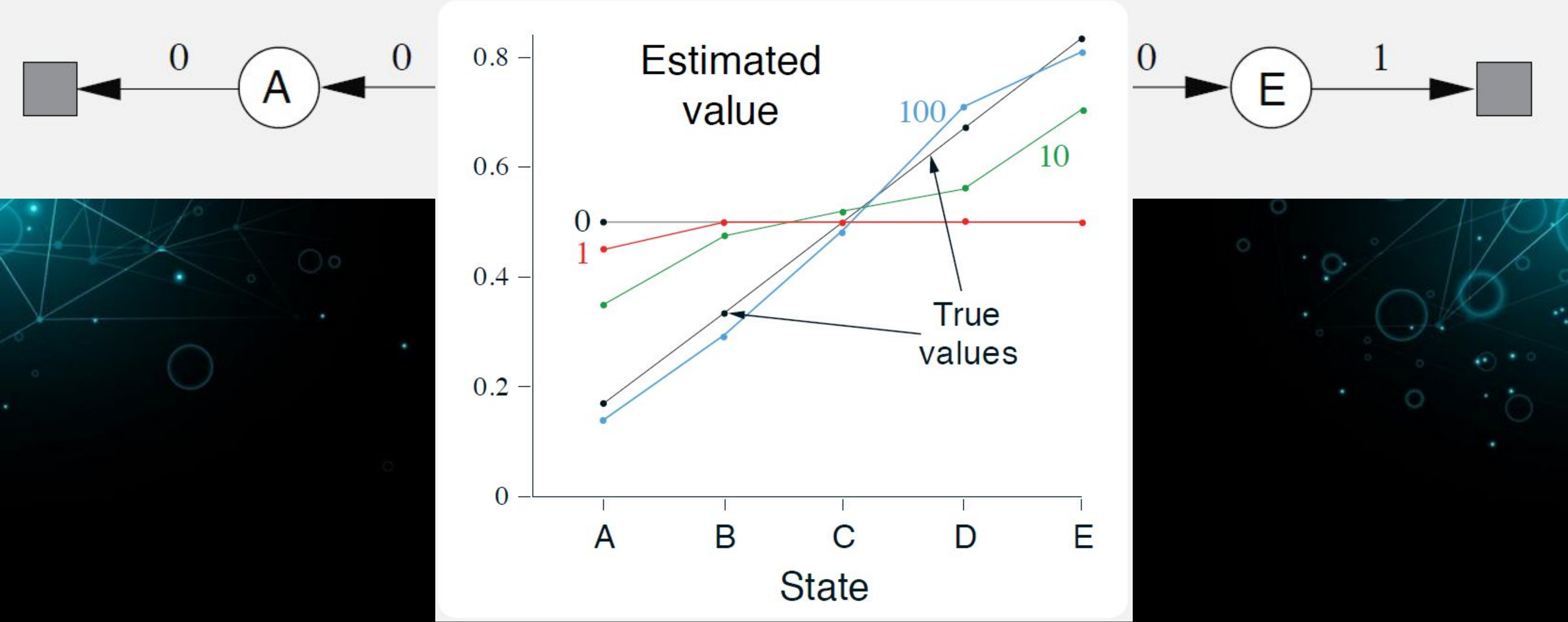
On-policy prediction, but can be off-policy too. Moves the values of states toward the next states visited.

$$V(s) = V(s) + \alpha[\underbrace{R + \gamma V(s') - V(s)}]$$

"TD Error"
(prediction error)

$$\delta_t = R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$$

Exercise 6.3 From the results shown in the left graph of the random walk example it appears that the first episode results in a change in only $V(\mathsf{A})$. What does this tell you about what happened on the first episode? Why was only the estimate for this one state changed? By exactly how much was it changed? □



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 $Q(\mathit{terminal}, \cdot) = 0$

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

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'
    $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma \max_a Q(S', a) - Q(S, A)\big]$
    $S \leftarrow S'$
  until S is terminal

just chose.

Expected SARSA **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'
    $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma$ ∑π(a|s')Q(s',a) - Q(s,a) $\big]$
    $S \leftarrow S'$
  until S is terminal

of next actions available.

TD-LEARNING METHODS

New Expectation = *Expectation* + *Step Size* • [*Reality* - *Expectation*]

TD(0)	The current state-value estimate vs. the immediate reward + the discounted state-value of all successor states.
SARSA *(on-policy)*	The current action-value estimate vs. the immediate reward + the discounted action-value of actions available from all successor states.
Q-Learning *(off-policy)*	The current action-value estimate vs. the immediate reward + the discounted max action-value available from all successor states.
Expected SARSA	The current action-value estimate vs. the immediate reward + the discounted policy-averaged action-values available from all successor states.

TD-LEARNING METHODS

New Expectation = *Expectation* + *Step Size* • [*Reality* - *Expectation*]

TD(0)	"I update my state-values toward the reward I just got plus the state that I ended up in afterward, one step at a time."
SARSA *(on-policy)*	"I update my action-values toward the reward I just got plus the state-action combo I used right after, one step at a time."
Q-Learning *(off-policy)*	"I update my action-values toward the reward I just got plus the best action available from the next state..
Expected SARSA	"I update my actions using the average of the actions my available from the next state, weighted by how likely I am to choose them."

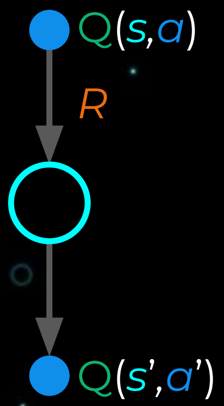
TD-LEARNING METHODS

TD(0)

$\alpha[R + \gamma V(s') - V(s)]$

Update toward next state in chain.

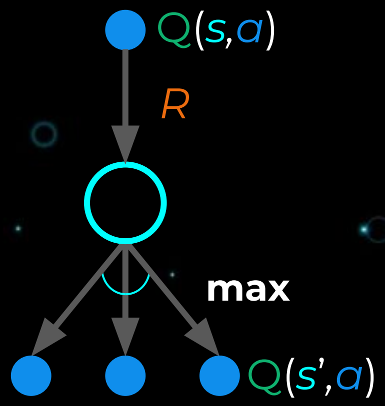
SARSA



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

Update toward next action chosen.

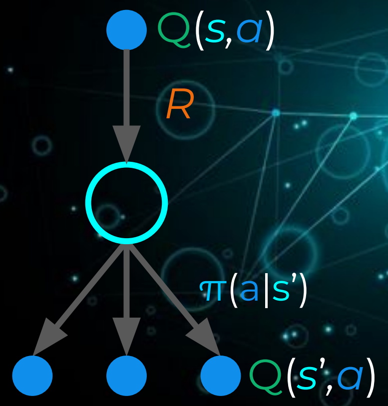
Q-Learning



$\alpha[R + \gamma \mathbf{max} Q(s',a) - Q(s,a)]$

Update toward max of next actions experienced by behavioural policy.

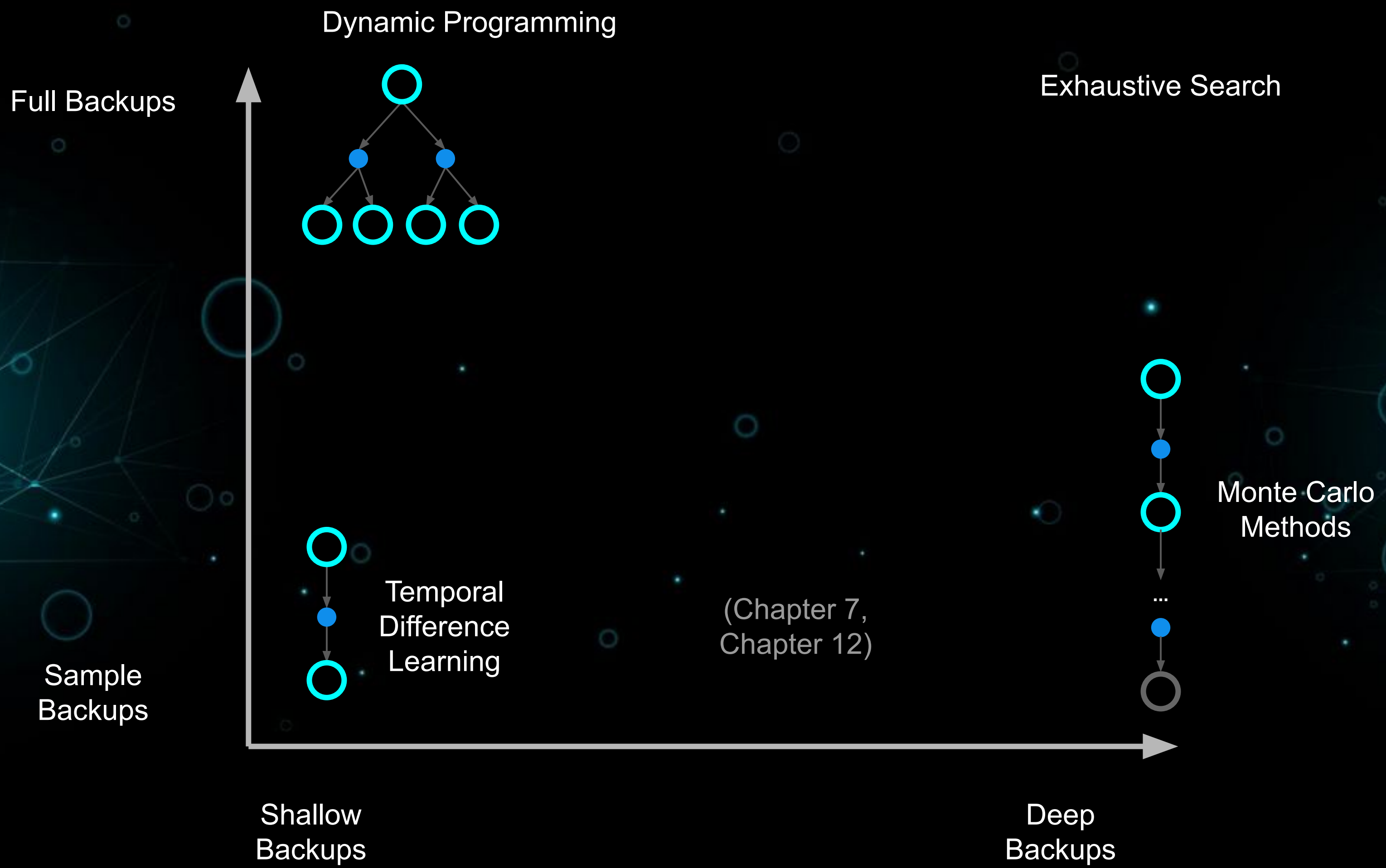
Expected SARSA



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

Update toward target-policy-weighted average of next actions experienced by behavioural policy.

	Monte Carlo	T-D Learning	Dynamic Programming
Backup Method	One chain (episode) at a time.	One link (transition) of one chain at a time	All links in all possible chains, all at once.
Bootstrapping	No	Yes	Yes
Sampling	Yes	Yes	No
Compute Efficiency	Far more efficient than DP, but can be inefficient due to randomness.	Usually more efficient than MC	LOL NO
Initialization	Not sensitive to initialization	More sensitive to initialization due to reliance on $V(S_{t+1})$	Doesn't matter.
Convergence	Very desirable convergence properties (law of large numbers)	Good convergence but not always for function approximation.	Yes
Return	Uses actual returns.	Uses estimated returns.	Estimated returns using actual system dynamics.
Bias	Unbiased	Some bias in the "guess" of $V(S_{t+1})$	Unbiased
Variance	Higher variance (full trajectory set of random events per update)	Lower variance (one set of random events per update)	N/A
Markov Property	Does not exploit, better in non-Markov environs.	Does exploit, better in Markov environments	Does exploit



Adapted from Silver (2015)