TEMPORAL DIFFERENCE LEARNING

Scott O'Hara
Metrowest Developers Machine Learning Group
6/24/2020

REFERENCES

Sample-based Learning Methods ***(M. White and A. White), University of Alberta, Alberta Machine Intelligence Institute, Coursera.***

- https://www.coursera.org/learn/sample-based-learning-methods/

Reinforcement learning: An Introduction R. S. Sutton and A. G. Barto, Second edition. Cambridge, Massachusetts: The MIT Press, 2018.

CS188 - Introduction to Artificial Intelligence ***course at University of California, Berkeley:***

- http://ai.berkeley.edu/home.html
- http://gamescrafters.berkeley.edu/~cs188/sp20/

WHAT IS TEMPORAL DIFFERENCE (TD) LEARNING?

- TD-Learning is a kind of *prediction learning* that takes advantage of the temporal structure of learning to predict. (Sutton video).

- In prediction learning:
  - you make a prediction about what will happen next.
  - you wait to see what happens.
  - You learn by comparing what happens to what you predicted.

Credit: Sample-based Learning Methods *(M. White and A. White), University of Alberta, Alberta Machine Intelligence Institute, Coursera.*

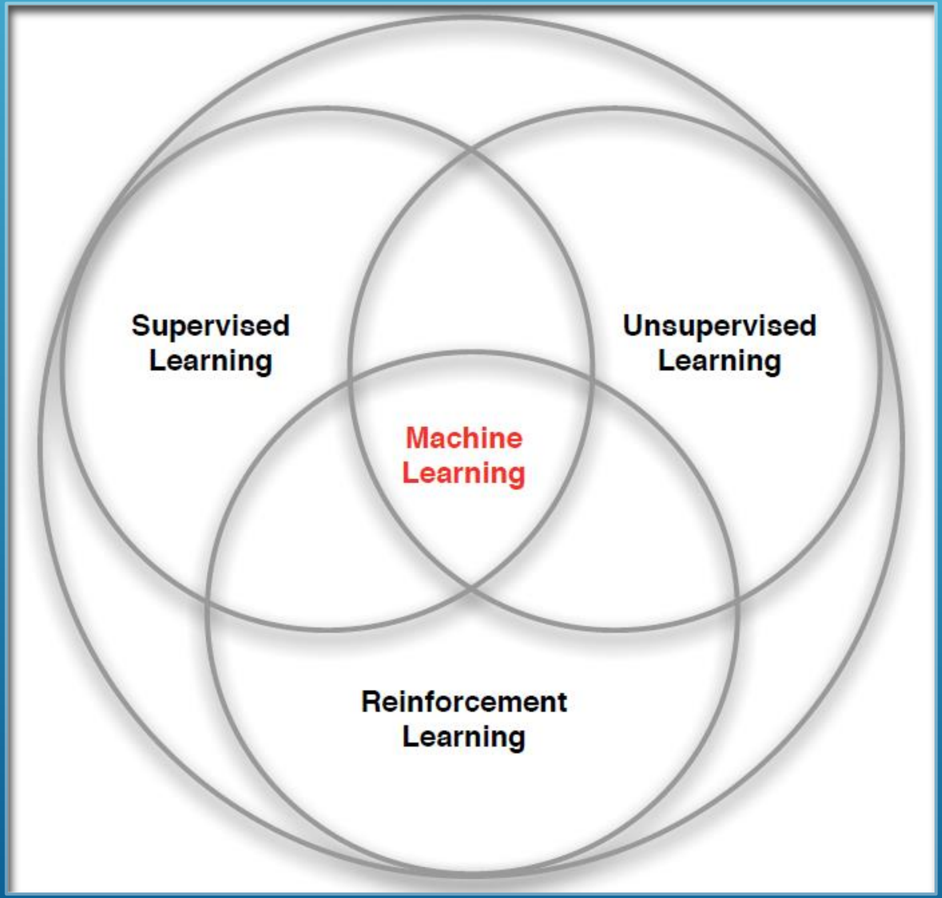
WHAT IS TEMPORAL DIFFERENCE (TD) LEARNING?

- TD-Learning is one of the most fundamental ideas in reinforcement learning.
- From *Reinforcement Learning: An Introduction*: "If one had to identify one idea as central and novel to reinforce learning, it would be temporal difference learning." (page 119, Chapter 6.)

Credit: Sample-based Learning Methods *(M. White and A. White), University of Alberta, Alberta Machine Intelligence Institute, Coursera.*

WHAT IS REINFORCEMENT LEARNING?

Reinforcement learning is a kind **of *unsupervised supervised learning,*** so it combines aspects of the other two forms of machine learning.



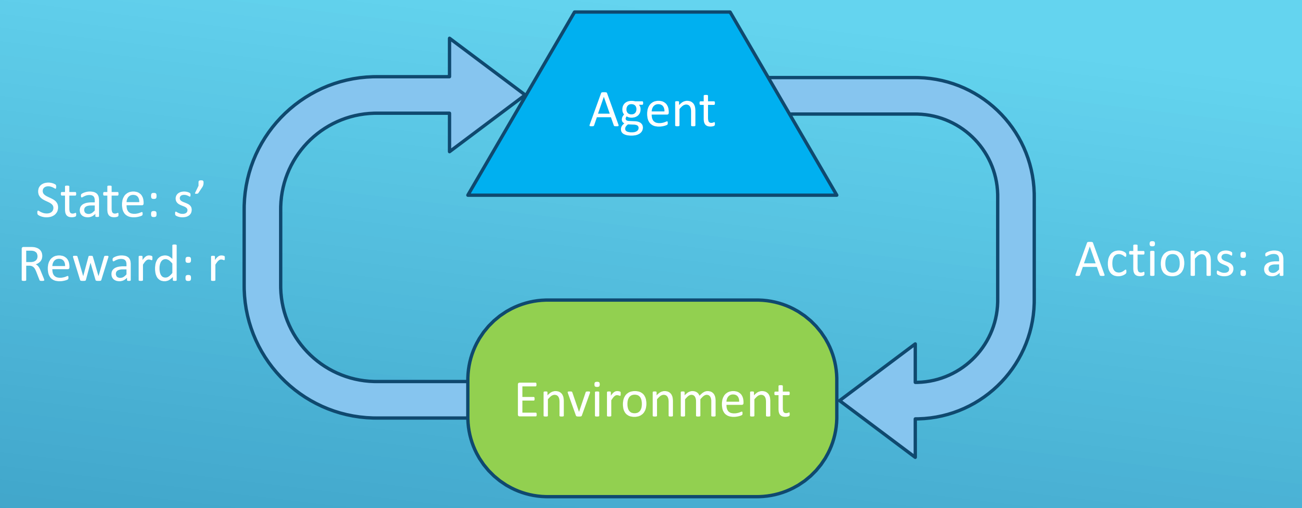
Supervised Learning – Learn a function from labeled data that maps input attributes to an output.

Unsupervised Learning – Find classes, patterns or generalizations in unlabeled data.

Reinforcement Learning –An agent learns to maximize rewards while acting in an uncertain environment.

Credit: image borrowed from lecture slides David Silver, DeepMind, "Introduction to Reinforcement Learning

THE REINFORCEMENT LEARNING PROBLEM



- Agent must learn to act to maximize expected rewards.
- Agent knows the current state s, takes an action ***a***, receives a reward ***r*** and observes the next state ***s'***.

$$S_0, A_0, R_0, S_1, A_1, R_2, S_2, A_2, R_2, \ldots, S_n, A_n, R_n, S_T$$

- Agent has **no access** to the reward model ***r(s,a,s')*** or the transition model ***T(s,a,s')***.

MARKOV DECISION PROCESSES

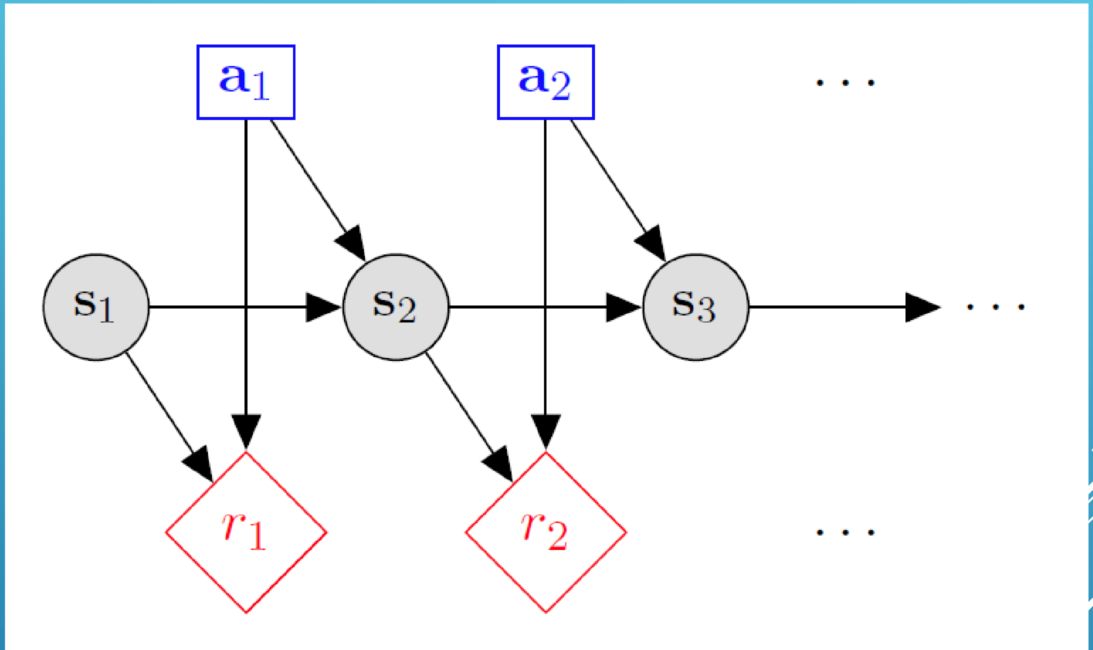
- **States:** $s_1, \ldots, s_n$
- **Actions:** $a_1, \ldots, a_m$
- **Reward model:**

$$R(s, a, s') \in R$$

- **Transition model:**

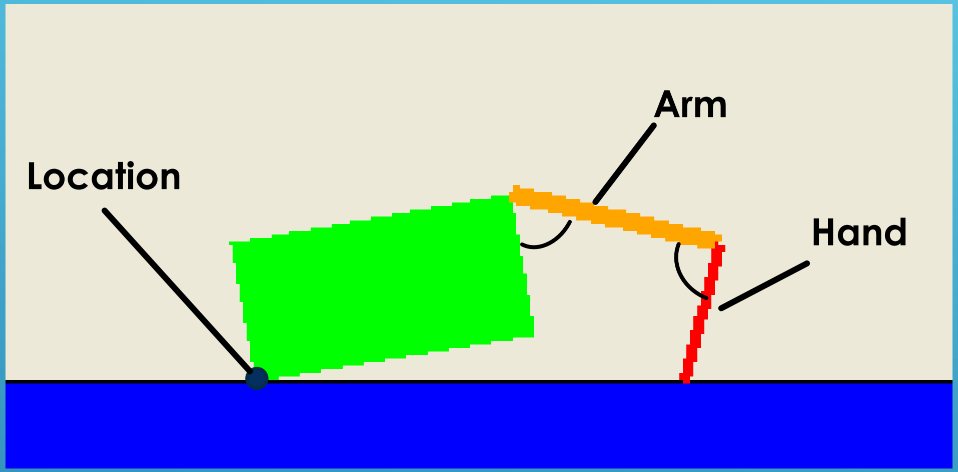
$$T(s, a, s') = P(s' | s, a)$$

- **Discount factor:** $\gamma \in [0, 1]$



<https://en.wikipedia.org/w/index.php?title=Markov_decision_process&oldid=855934986> [accessed 11 September 2018]

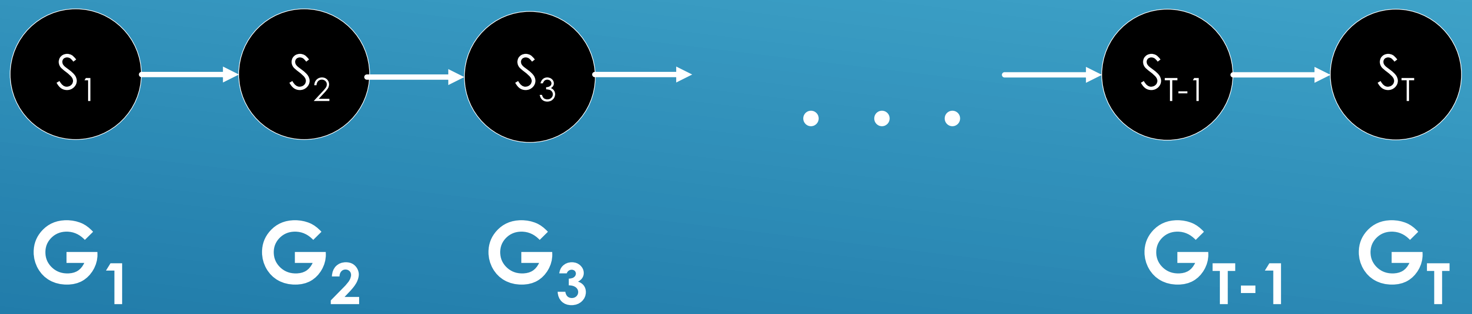
APPLICATION: CRAWLER ROBOT



- **States:** <Location, Arm angle, Hand angle>
- **Actions:** increase Arm angle, decrease Arm angle,
  increase Hand angle, decrease Hand angle.
- **Reward model:** +1 if robot moves right, -1 if robot moves left.
- **Transition model:** model of box movement caused by arm movements.

The Discounted Return from An Episode

$$G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots$$



QUANTITIES TO OPTIMIZE

- The value (utility) of a **state s:**

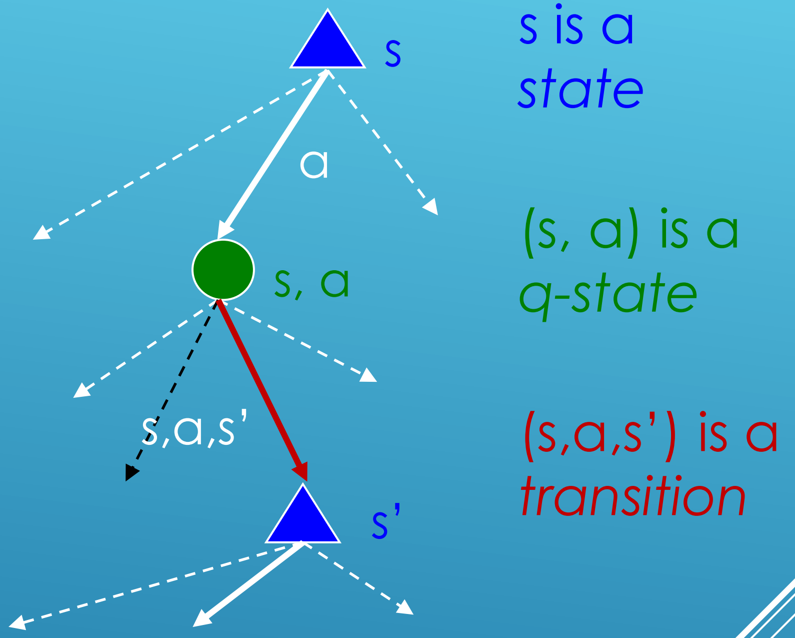
  V(s) = expected utility starting in s and acting optimally thereafter.

- The value (utility) of a **q-state (s,a)**:

  Q(s,a) = expected utility when taking action a from state s and acting optimally thereafter.

- The policy π:

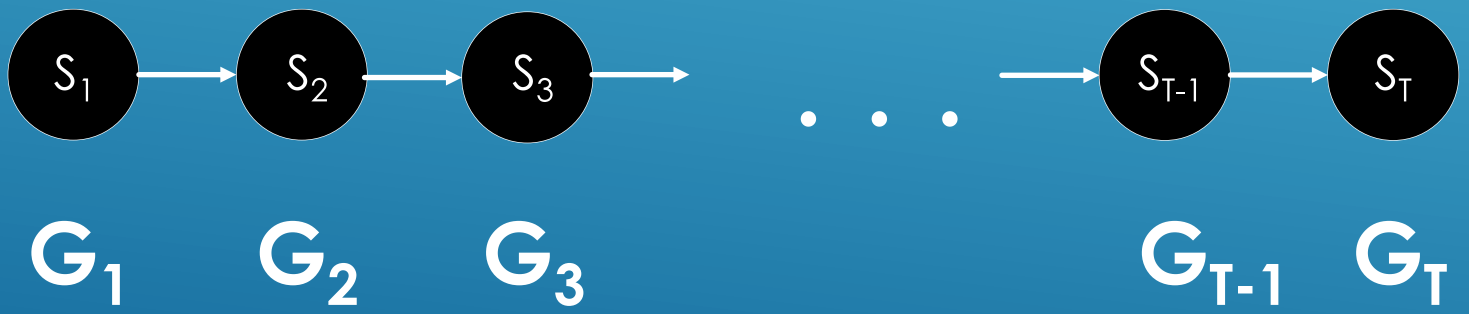
  π(a | s) = probability of action a from state s



Computing the State-Value Function **V** from An Episode

$$v_\pi(s) \doteq \mathbb{E}_\pi[G_t \mid S_t = s]$$

$$V(S_t) \leftarrow V(S_t) + \alpha[G_t - V(S_t)]$$



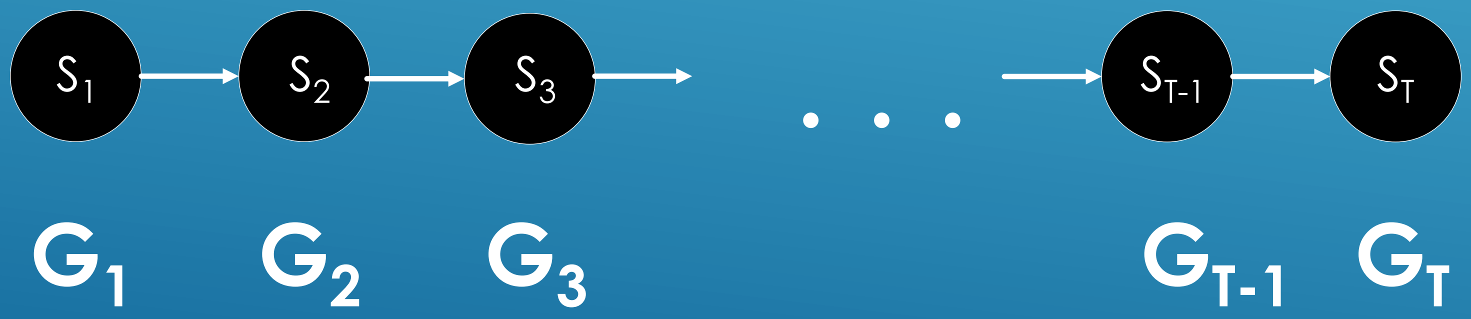
Credit: Sample-based Learning Methods *(M. White and A. White), University of Alberta, Alberta Machine Intelligence Institute, Coursera.*

Computing the State-Value Function **V** from An Episode

Problem: We don't know $\mathbf{G_t}$ until the end of the episode.

$$v_\pi(s) \doteq \mathbb{E}_\pi[G_t \mid S_t = s]$$

$$V(S_t) \leftarrow V(S_t) + \alpha[G_t - V(S_t)]$$



Credit: Sample-based Learning Methods *(M. White and A. White), University of Alberta, Alberta Machine Intelligence Institute, Coursera.*

Expressing the State-Value Function Recursively

$$G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots$$

$$= R_{t+1} + \gamma G_{t+1}$$

$$v_\pi(s) \doteq \mathbb{E}_\pi[G_t \mid S_t = s]$$

$$= \mathbb{E}_\pi[R_{t+1} + \gamma \boxed{G_{t+1}} \mid S_t = s]$$

$$= R_{t+1} + \gamma v_\pi(S_{t+1})$$

$$G_t \approx R_{t+1} + \gamma V(S_{t+1})$$

Credit: Sample-based Learning Methods *(M. White and A. White), University of Alberta, Alberta Machine Intelligence Institute, Coursera.*

The Temporal Difference Error (TD-Error δ_t)

We can think of the value of the next state $V(S_{t+1})$ as a stand-in for the return until the end of the episode.

$$G_t \approx R_{t+1} + \gamma V(S_{t+1})$$

$$V(S_t) \leftarrow V(S_t) + \alpha\left[G_t - V(S_t)\right]$$

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

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

Credit: Sample-based Learning Methods *(M. White and A. White), University of Alberta, Alberta Machine Intelligence Institute, Coursera.*

Updating from a Prediction

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

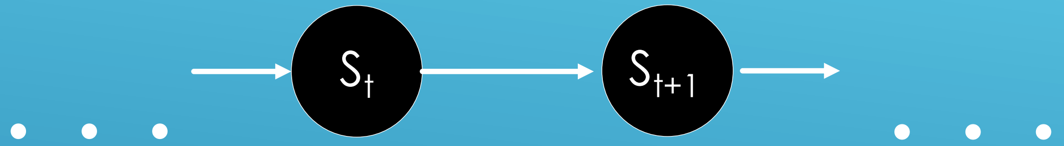
The TD-target

The TD-prediction

Credit: Sample-based Learning Methods *(M. White and A. White), University of Alberta, Alberta Machine Intelligence Institute, Coursera.*

$$S_0, A_0, R_1, S_1, A_1, R_2, S_2, \ldots, S_t, A_t, R_{t+1}, S_{t+1}$$

Updating from a Prediction



$\mathbf{V}(S_T)$ $\quad\quad$ $\mathbf{V}(S_T) \sim \mathbf{R}_{T+1} + \gamma \mathbf{V}(S_{T+1})$

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

Credit: Sample-based Learning Methods *(M. White and A. White), University of Alberta, Alberta Machine Intelligence Institute, Coursera.*

Temporal Difference Learning

Tabular TD(0) for estimating v_π

Input: the policy π to be evaluated
Algorithm parameter: step size $\alpha \in (0, 1]$
Initialize $V(s)$, for all $s \in \mathcal{S}^+$, arbitrarily except that $V(terminal) = 0$

Loop for each episode:
    Initialize S
    Loop for each step of episode:
        $A \leftarrow$ action given by π for S
        Take action A, observe R, S'
        $V(S) \leftarrow V(S) + \alpha\big[R + \gamma V(S') - V(S)\big]$
        $S \leftarrow S'$
    until S is terminal

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 $Q(terminal, \cdot) = 0$

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)
        $Q(S, A) \leftarrow Q(S, A) + \alpha\left[R + \gamma Q(S', A') - Q(S, A)\right]$
        $S \leftarrow S'$; $A \leftarrow A'$;
    until S is terminal

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

Expected SARSA

Same as Q-Learning, but substitute expected state-action value for the max state-action value.

$$
\begin{aligned}
Q(S_t, A_t) &\leftarrow Q(S_t, A_t) + \alpha\Big[R_{t+1} + \gamma \mathbb{E}_\pi[Q(S_{t+1}, A_{t+1}) \mid S_{t+1}] - Q(S_t, A_t)\Big] \\
&\leftarrow Q(S_t, A_t) + \alpha\Big[R_{t+1} + \gamma \sum \pi(a|S_{t+1}) Q(S_{t+1}, a) - Q(S_t, A_t)\Big]
\end{aligned}
$$

EPISODIC SARSA WITH FUNCTION APPROXIMATION

Episodic Semi-gradient Sarsa for Estimating $\hat{q} \approx q_*$

Input: a differentiable action-value function parameterization $\hat{q} : \mathcal{S} \times \mathcal{A} \times \mathbb{R}^d \to \mathbb{R}$
Algorithm parameters: step size $\alpha > 0$, small $\varepsilon > 0$
Initialize value-function weights $\mathbf{w} \in \mathbb{R}^d$ arbitrarily (e.g., $\mathbf{w} = \mathbf{0}$)

Loop for each episode:
  $S, A \leftarrow$ initial state and action of episode (e.g., ε-greedy)
  Loop for each step of episode:
    Take action A, observe R, S'
    If S' is terminal:
      $\mathbf{w} \leftarrow \mathbf{w} + \alpha\big[R - \hat{q}(S, A, \mathbf{w})\big]\nabla\hat{q}(S, A, \mathbf{w})$
      Go to next episode
    Choose A' as a function of $\hat{q}(S', \cdot, \mathbf{w})$ (e.g., ε-greedy)
    $\mathbf{w} \leftarrow \mathbf{w} + \alpha\big[R + \gamma\hat{q}(S', A', \mathbf{w}) - \hat{q}(S, A, \mathbf{w})\big]\nabla\hat{q}(S, A, \mathbf{w})$
    $S \leftarrow S'$
    $A \leftarrow A'$

Credit: *Reinforcement Learning: an Introduction, R. Sutton and A. Barto, 2018.*