Reinforcement Learning Cheat Sheet

Prof. Kuan-Ting Lai

2020/6/19

Symbols & Terminologies

1. The Problem

S_t	state at time t
A_t	action at time t
R_t	reward at time t
γ	discount rate (where $0 \leq \gamma \leq 1$)
G_t	discounted return at time t ($\sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$)
$\mathcal{S}$	set of all nonterminal states
$\mathcal{S}^+$	set of all states (including terminal states)
$\mathcal{A}$	set of all actions
$\mathcal{A}(s)$	set of all actions available in state s
$\mathcal{R}$	set of all rewards
$p(s', r \vert s, a)$	probability of next state s' and reward r, given current state s and current action a ($\mathbb{P}(S_{t+1} = s', R_{t+1} = r \vert S_t = s, A_t = a)$)

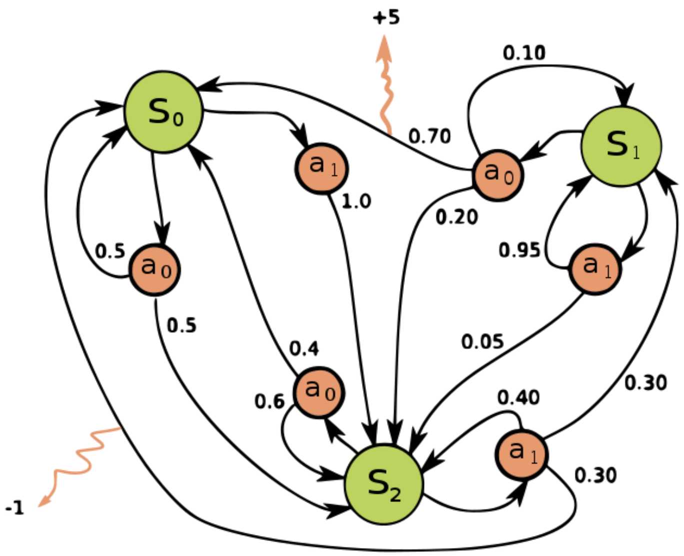
2. The Solution

π	policy *if deterministic*: $\pi(s) \in \mathcal{A}(s)$ for all $s \in \mathcal{S}$ *if stochastic*: $\pi(a \vert s) = \mathbb{P}(A_t = a \vert S_t = s)$ for all $s \in \mathcal{S}$ and $a \in \mathcal{A}(s)$
v_π	state-value function for policy π ($v_\pi(s) \doteq \mathbb{E}[G_t \vert S_t = s]$ for all $s \in \mathcal{S}$)
q_π	action-value function for policy π ($q_\pi(s, a) \doteq \mathbb{E}[G_t \vert S_t = s, A_t = a]$ for all $s \in \mathcal{S}$ and $a \in \mathcal{A}(s)$)
v_*	optimal state-value function ($v_*(s) \doteq \max_\pi v_\pi(s)$ for all $s \in \mathcal{S}$)
q_*	optimal action-value function ($q_*(s, a) \doteq \max_\pi q_\pi(s, a)$ for all $s \in \mathcal{S}$ and $a \in \mathcal{A}(s)$)

Bellman Equations

$$v_\pi(s) = \sum_{a\in\mathcal{A}(s)} \pi(a|s) \sum_{s'\in\mathcal{S}, r\in\mathcal{R}} p(s', r|s, a)(r + \gamma v_\pi(s'))$$

$$q_\pi(s, a) = \sum_{s'\in\mathcal{S}, r\in\mathcal{R}} p(s', r|s, a)(r + \gamma \sum_{a'\in\mathcal{A}(s')} \pi(a'|s')q_\pi(s', a'))$$



Bellman Optimality Equations

$$v_*(s) = \max_{a \in \mathcal{A}(s)} \sum_{s' \in \mathcal{S}, r \in \mathcal{R}} p(s', r | s, a)(r + \gamma v_*(s'))$$

$$q_*(s, a) = \sum_{s' \in \mathcal{S}, r \in \mathcal{R}} p(s', r | s, a)(r + \gamma \max_{a' \in \mathcal{A}(s')} q_*(s', a'))$$

Deriving the Bellman Equations

$$v_\pi(s) = \sum_{a \in \mathcal{A}(s)} \pi(a|s) q_\pi(s,a)$$

$$v_*(s) = \max_{a \in \mathcal{A}(s)} q_*(s,a)$$

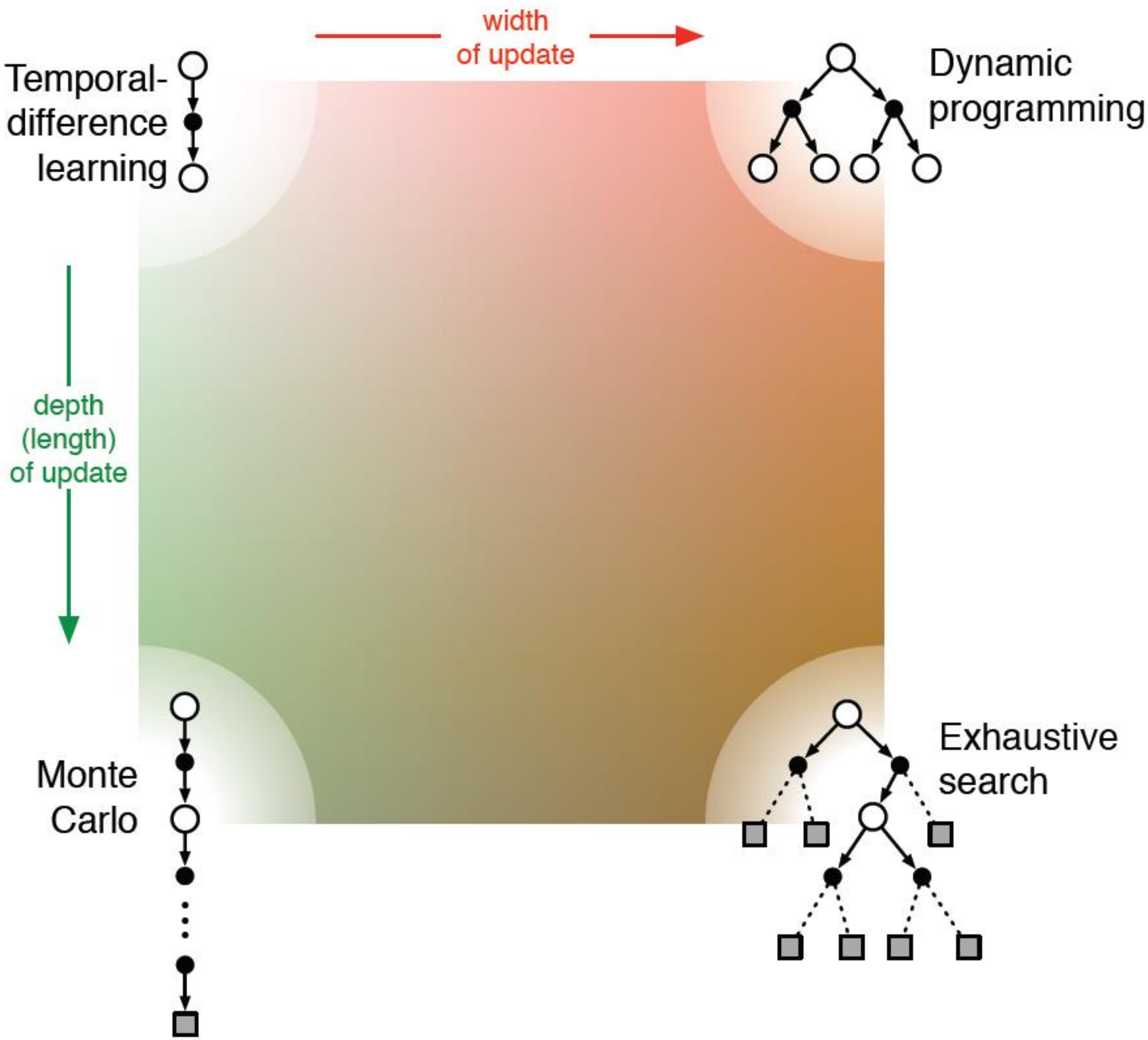
$$q_\pi(s,a) = \sum_{s' \in \mathcal{S}, r \in \mathcal{R}} p(s', r|s,a)(r + \gamma v_\pi(s'))$$

$$q_*(s,a) = \sum_{s' \in \mathcal{S}, r \in \mathcal{R}} p(s', r|s,a)(r + \gamma v_*(s'))$$

Deriving the Bellman Equations

$$
\begin{aligned}
q_\pi(s,a) &\doteq \mathbb{E}_\pi[G_t | S_t = s, A_t = a] \quad (1)\\
&= \sum_{s' \in \mathcal{S}, r \in \mathcal{R}} \mathbb{P}(S_{t+1} = s', R_{t+1} = r | S_t = s, A_t = a)\mathbb{E}_\pi[G_t | S_t = s, A_t = a, S_{t+1} = s', R_{t+1} = r] \quad (2)\\
&= \sum_{s' \in \mathcal{S}, r \in \mathcal{R}} p(s', r | s, a)\mathbb{E}_\pi[G_t | S_t = s, A_t = a, S_{t+1} = s', R_{t+1} = r] \quad (3)\\
&= \sum_{s' \in \mathcal{S}, r \in \mathcal{R}} p(s', r | s, a)\mathbb{E}_\pi[G_t | S_{t+1} = s', R_{t+1} = r] \quad (4)\\
&= \sum_{s' \in \mathcal{S}, r \in \mathcal{R}} p(s', r | s, a)\mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1} | S_{t+1} = s', R_{t+1} = r] \quad (5)\\
&= \sum_{s' \in \mathcal{S}, r \in \mathcal{R}} p(s', r | s, a)(r + \gamma \mathbb{E}_\pi[G_{t+1} | S_{t+1} = s']) \quad (6)\\
&= \sum_{s' \in \mathcal{S}, r \in \mathcal{R}} p(s', r | s, a)(r + \gamma v_\pi(s')) \quad (7)
\end{aligned}
$$

- (1) by definition ($q_\pi(s,a) \doteq \mathbb{E}_\pi[G_t | S_t = s, A_t = a]$)
- (2) Law of Total Expectation
- (3) by definition ($p(s', r | s, a) \doteq \mathbb{P}(S_{t+1} = s', R_{t+1} = r | S_t = s, A_t = a)$)
- (4) $\mathbb{E}_\pi[G_t | S_t = s, A_t = a, S_{t+1} = s', R_{t+1} = r] = \mathbb{E}_\pi[G_t | S_{t+1} = s', R_{t+1} = r]$
- (5) $G_t = R_{t+1} + \gamma G_{t+1}$
- (6) Linearity of Expectation
- (7) $v_\pi(s') = \mathbb{E}_\pi[G_{t+1} | S_{t+1} = s']$



width
of update
Temporal-
difference
learning
Dynamic
programming
depth
(length)
of update
Monte
Carlo
Exhaustive
search

Dynamic Programming

Algorithm 1: Policy Evaluation

Input: MDP, policy π, small positive number θ
Output: $V \approx v_\pi$
Initialize V arbitrarily (e.g., $V(s) = 0$ for all $s \in \mathcal{S}^+$)
repeat
    $\Delta \leftarrow 0$
    for $s \in \mathcal{S}$ **do**
        $v \leftarrow V(s)$
        $V(s) \leftarrow \sum_{a \in \mathcal{A}(s)} \pi(a|s) \sum_{s' \in \mathcal{S}, r \in \mathcal{R}} p(s', r|s, a)(r + \gamma V(s'))$
        $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
    end
until $\Delta < \theta$;
return V

Algorithm 2: Estimation of Action Values

Input: MDP, state-value function V
Output: action-value function Q
for $s \in \mathcal{S}$ **do**
    for $a \in \mathcal{A}(s)$ **do**
        $Q(s, a) \leftarrow \sum_{s' \in \mathcal{S}, r \in \mathcal{R}} p(s', r|s, a)(r + \gamma V(s'))$
    end
end
return Q

Dynamic Programming

Algorithm 3: Policy Improvement

Input: MDP, value function V
Output: policy π'
for $s \in \mathcal{S}$ **do**
  for $a \in \mathcal{A}(s)$ **do**
    $Q(s,a) \leftarrow \sum_{s' \in \mathcal{S}, r \in \mathcal{R}} p(s', r|s,a)(r + \gamma V(s'))$
  end
  $\pi'(s) \leftarrow \arg\max_{a \in \mathcal{A}(s)} Q(s,a)$
end
return π'

Algorithm 4: Policy Iteration

Input: MDP, small positive number θ
Output: policy $\pi \approx \pi_*$
Initialize π arbitrarily (e.g., $\pi(a|s) = \frac{1}{|\mathcal{A}(s)|}$ for all $s \in \mathcal{S}$ and $a \in \mathcal{A}(s)$)
policy-stable $\leftarrow$ *false*
repeat
  $V \leftarrow$ **Policy_Evaluation**(MDP, π, θ)
  $\pi' \leftarrow$ **Policy_Improvement**(MDP, V)
  if $\pi = \pi'$ **then**
    policy-stable $\leftarrow$ *true*
  end
  $\pi \leftarrow \pi'$
until *policy-stable* $=$ *true*;
return π

Algorithm 5: Truncated Policy Evaluation

Input: MDP, policy π, value function V, positive integer *max_iterations*
Output: $V \approx v_\pi$ (if *max_iterations* is large enough)
counter $\leftarrow 0$
while *counter* $<$ *max_iterations* **do**
  for $s \in \mathcal{S}$ **do**
    $V(s) \leftarrow \sum_{a \in \mathcal{A}(s)} \pi(a|s) \sum_{s' \in \mathcal{S}, r \in \mathcal{R}} p(s', r|s,a)(r + \gamma V(s'))$
  end
  counter $\leftarrow$ *counter* $+ 1$
end
return V

Dynamic Programming

Algorithm 6: Truncated Policy Iteration

Input: MDP, positive integer *max_iterations*, small positive number θ
Output: policy $\pi \approx \pi_*$
Initialize V arbitrarily (e.g., $V(s) = 0$ for all $s \in \mathcal{S}^+$)
Initialize π arbitrarily (e.g., $\pi(a|s) = \frac{1}{|\mathcal{A}(s)|}$ for all $s \in \mathcal{S}$ and $a \in \mathcal{A}(s)$)
repeat
  $\pi \leftarrow$ **Policy_Improvement**(MDP, V)
  $V_{old} \leftarrow V$
  $V \leftarrow$ **Truncated_Policy_Evaluation**(MDP, π, V, *max_iterations*)
until $\max_{s \in \mathcal{S}} |V(s) - V_{old}(s)| < \theta$;
return π

Algorithm 7: Value Iteration

Input: MDP, small positive number θ
Output: policy $\pi \approx \pi_*$
Initialize V arbitrarily (e.g., $V(s) = 0$ for all $s \in \mathcal{S}^+$)
repeat
  $\Delta \leftarrow 0$
  for $s \in \mathcal{S}$ **do**
    $v \leftarrow V(s)$
    $V(s) \leftarrow \max_{a \in \mathcal{A}(s)} \sum_{s' \in \mathcal{S}, r \in \mathcal{R}} p(s', r|s, a)(r + \gamma V(s'))$
    $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
  end
until $\Delta < \theta$;
$\pi \leftarrow$ **Policy_Improvement**(MDP, V)
return π

Monte Carlo Methods

Algorithm 8: First-Visit MC Prediction (*for state values*)

Input: policy π, positive integer *num_episodes*
Output: value function V ($\approx v_\pi$ if *num_episodes* is large enough)
Initialize $N(s) = 0$ for all $s \in \mathcal{S}$
Initialize $returns\_sum(s) = 0$ for all $s \in \mathcal{S}$
for $i \leftarrow 1$ *to num_episodes* **do**
    Generate an episode $S_0, A_0, R_1, \ldots, S_T$ using π
    for $t \leftarrow 0$ *to* $T - 1$ **do**
        if S_t *is a first visit (with return* G_t*)* **then**
            $N(S_t) \leftarrow N(S_t) + 1$
            $returns\_sum(S_t) \leftarrow returns\_sum(S_t) + G_t$
    end
end
$V(s) \leftarrow returns\_sum(s)/N(s)$ for all $s \in \mathcal{S}$
return V

Algorithm 9: First-Visit MC Prediction (*for action values*)

Input: policy π, positive integer *num_episodes*
Output: value function Q ($\approx q_\pi$ if *num_episodes* is large enough)
Initialize $N(s, a) = 0$ for all $s \in \mathcal{S}, a \in \mathcal{A}(s)$
Initialize $returns\_sum(s, a) = 0$ for all $s \in \mathcal{S}, a \in \mathcal{A}(s)$
for $i \leftarrow 1$ *to num_episodes* **do**
    Generate an episode $S_0, A_0, R_1, \ldots, S_T$ using π
    for $t \leftarrow 0$ *to* $T - 1$ **do**
        if (S_t, A_t) *is a first visit (with return* G_t*)* **then**
            $N(S_t, A_t) \leftarrow N(S_t, A_t) + 1$
            $returns\_sum(S_t, A_t) \leftarrow returns\_sum(S_t, A_t) + G_t$
    end
end
$Q(s, a) \leftarrow returns\_sum(s, a)/N(s, a)$ for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$
return Q

Monte Carlo Methods

Algorithm 10: First-Visit GLIE MC Control

Input: positive integer *num_episodes*, GLIE $\{\epsilon_i\}$
Output: policy π ($\approx \pi_*$ if *num_episodes* is large enough)

```
Initialize Q(s, a) = 0 for all s ∈ S and a ∈ A(s)
Initialize N(s, a) = 0 for all s ∈ S, a ∈ A(s)
for i ← 1 to num_episodes do
    ε ← ε_i
    π ← ε-greedy(Q)
    Generate an episode S_0, A_0, R_1, ..., S_T using π
    for t ← 0 to T − 1 do
        if (S_t, A_t) is a first visit (with return G_t) then
            N(S_t, A_t) ← N(S_t, A_t) + 1
            Q(S_t, A_t) ← Q(S_t, A_t) + (1 / N(S_t, A_t)) (G_t − Q(S_t, A_t))
    end
end
return π
```

Algorithm 11: First-Visit Constant-α (GLIE) MC Control

Input: positive integer *num_episodes*, small positive fraction α, GLIE $\{\epsilon_i\}$
Output: policy π ($\approx \pi_*$ if *num_episodes* is large enough)

```
Initialize Q arbitrarily (e.g., Q(s, a) = 0 for all s ∈ S and a ∈ A(s))
for i ← 1 to num_episodes do
    ε ← ε_i
    π ← ε-greedy(Q)
    Generate an episode S_0, A_0, R_1, ..., S_T using π
    for t ← 0 to T − 1 do
        if (S_t, A_t) is a first visit (with return G_t) then
            Q(S_t, A_t) ← Q(S_t, A_t) + α(G_t − Q(S_t, A_t))
    end
end
return π
```

Temporal-Difference Methods

Algorithm 12: TD(0)

Input: policy π, positive integer *num_episodes*
Output: value function V ($\approx v_\pi$ if *num_episodes* is large enough)
Initialize V arbitrarily (e.g., $V(s) = 0$ for all $s \in \mathcal{S}^+$)
for $i \leftarrow 1$ *to num_episodes* **do**
    Observe S_0
    $t \leftarrow 0$
    repeat
        Choose action A_t using policy π
        Take action A_t and observe R_{t+1}, S_{t+1}
        $V(S_t) \leftarrow V(S_t) + \alpha(R_{t+1} + \gamma V(S_{t+1}) - V(S_t))$
        $t \leftarrow t + 1$
    until S_t *is terminal*;
end
return V

Algorithm 13: Sarsa

Input: policy π, positive integer *num_episodes*, small positive fraction α, GLIE $\{\epsilon_i\}$
Output: value function Q ($\approx q_\pi$ if *num_episodes* is large enough)
Initialize Q arbitrarily (e.g., $Q(s, a) = 0$ for all $s \in \mathcal{S}$ and $a \in \mathcal{A}(s)$, and $Q(\textit{terminal-state}, \cdot) = 0$)
for $i \leftarrow 1$ *to num_episodes* **do**
    $\epsilon \leftarrow \epsilon_i$
    Observe S_0
    Choose action A_0 using policy derived from Q (e.g., ϵ-greedy)
    $t \leftarrow 0$
    repeat
        Take action A_t and observe R_{t+1}, S_{t+1}
        Choose action A_{t+1} using policy derived from Q (e.g., ϵ-greedy)
        $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t))$
        $t \leftarrow t + 1$
    until S_t *is terminal*;
end
return Q

Temporal-Difference Methods

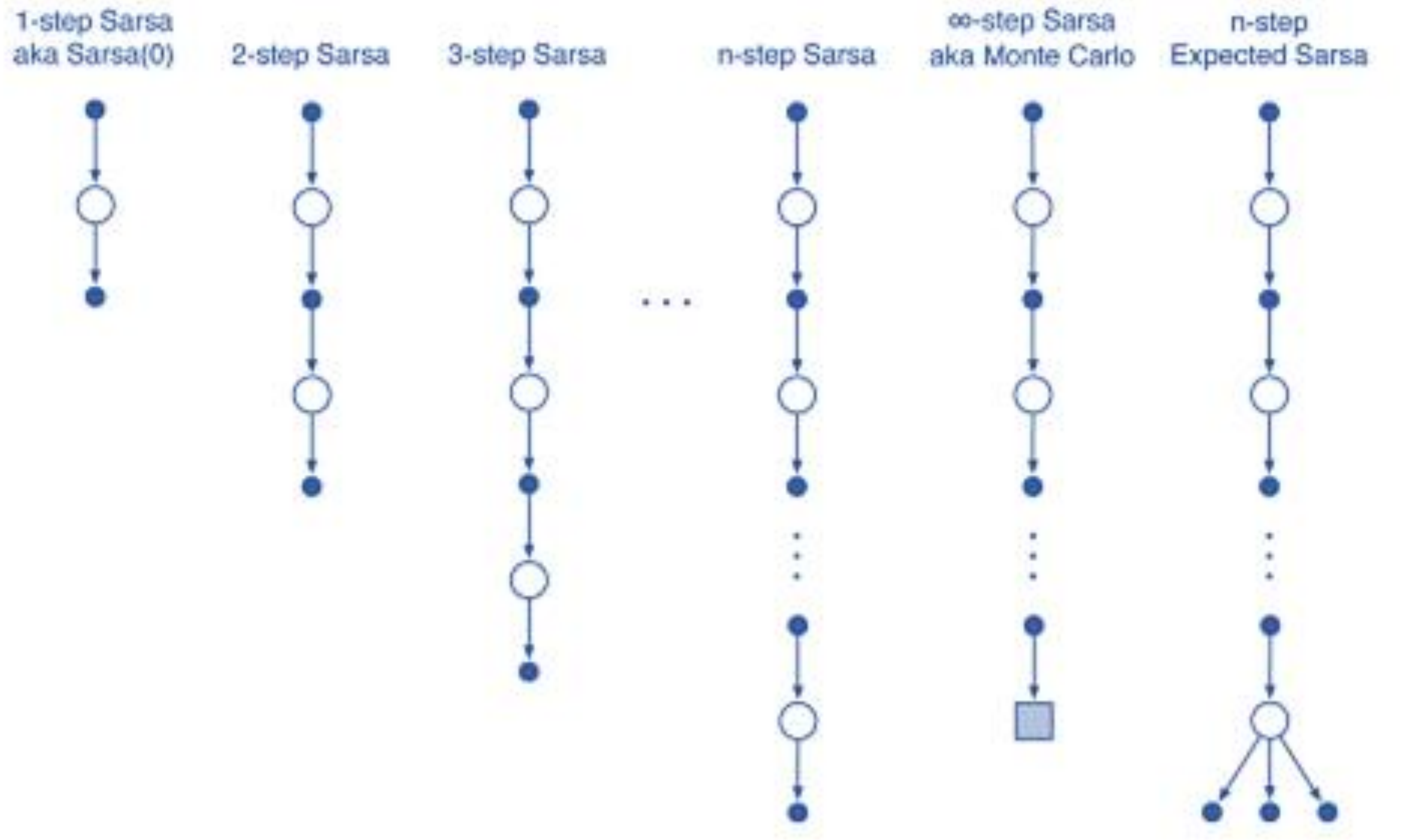
Algorithm 14: Sarsamax (Q-Learning)

Input: policy π, positive integer *num_episodes*, small positive fraction α, GL
Output: value function Q ($\approx q_\pi$ if *num_episodes* is large enough)
Initialize Q arbitrarily (e.g., $Q(s,a) = 0$ for all $s \in \mathcal{S}$ and $a \in \mathcal{A}(s)$, and $Q(ter$
for $i \leftarrow 1$ *to num_episodes* **do**
  $\epsilon \leftarrow \epsilon_i$
  Observe S_0
  $t \leftarrow 0$
  repeat
    Choose action A_t using policy derived from Q (e.g., ϵ-greedy)
    Take action A_t and observe R_{t+1}, S_{t+1}
    $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t))$
    $t \leftarrow t + 1$
  until S_t *is terminal*;
end
return Q

Algorithm 15: Expected Sarsa

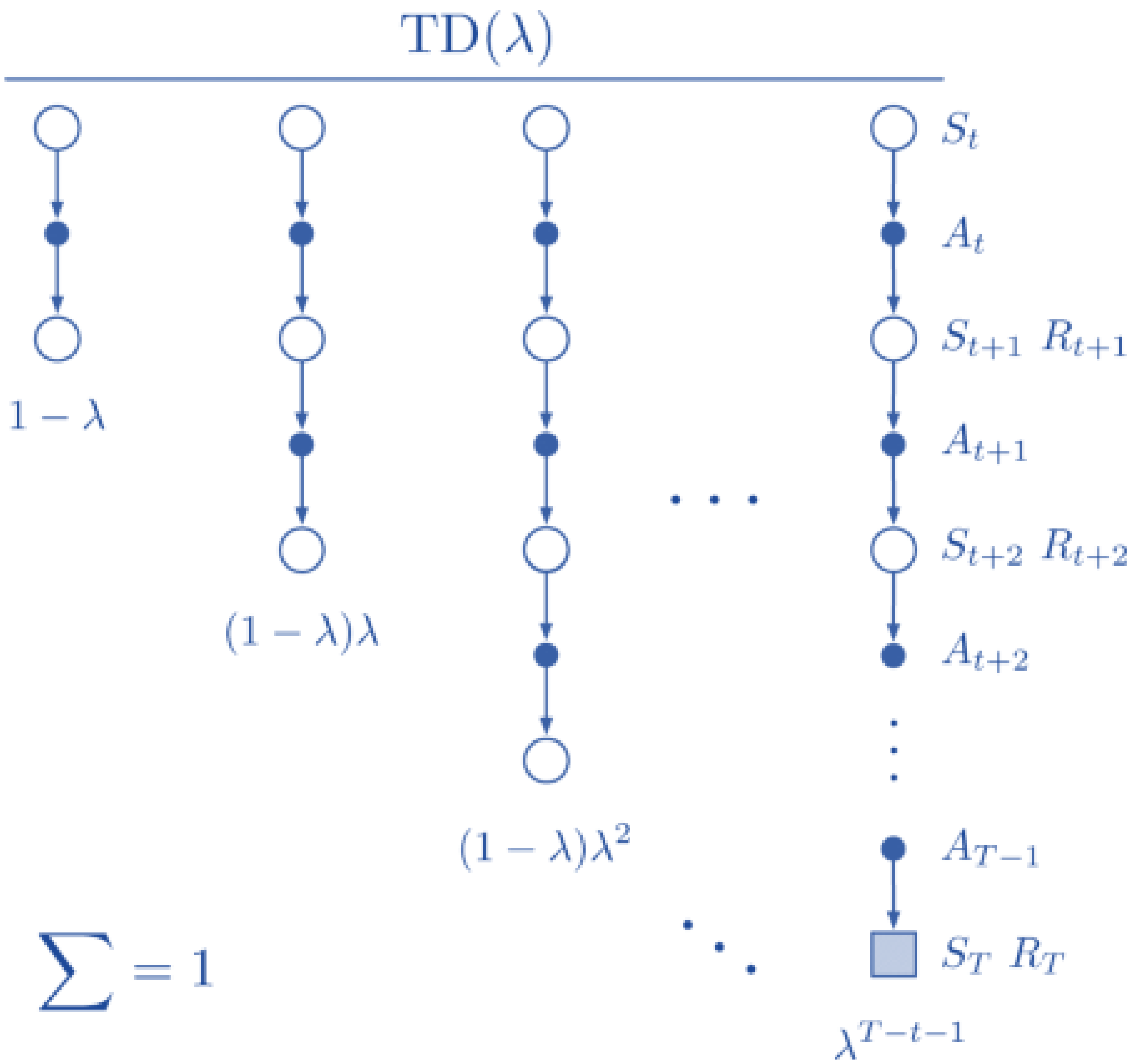
Input: policy π, positive integer *num_episodes*, small positive fraction α, GLIE $\{\epsilon_i\}$
Output: value function Q ($\approx q_\pi$ if *num_episodes* is large enough)
Initialize Q arbitrarily (e.g., $Q(s,a) = 0$ for all $s \in \mathcal{S}$ and $a \in \mathcal{A}(s)$, and $Q(\textit{terminal-state}, \cdot) = 0$)
for $i \leftarrow 1$ *to num_episodes* **do**
  $\epsilon \leftarrow \epsilon_i$
  Observe S_0
  $t \leftarrow 0$
  repeat
    Choose action A_t using policy derived from Q (e.g., ϵ-greedy)
    Take action A_t and observe R_{t+1}, S_{t+1}
    $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma \sum_a \pi(a|S_{t+1}) Q(S_{t+1}, a) - Q(S_t, A_t))$
    $t \leftarrow t + 1$
  until S_t *is terminal*;
end
return Q

N-step SARSA


1-step Sarsa
aka Sarsa(0)
2-step Sarsa
3-step Sarsa
n-step Sarsa
∞-step Sarsa
aka Monte Carlo
n-step
Expected Sarsa

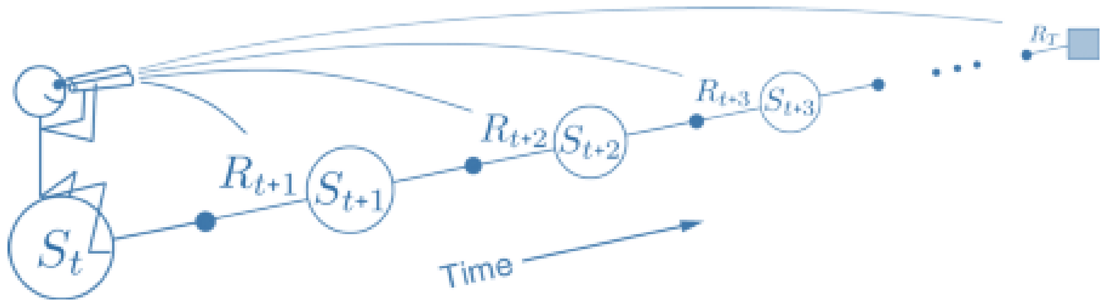
TD(λ)

$$G_t^\lambda \leftarrow (1-\lambda)\sum_{n=1}^{\infty}\lambda^{n-1}G_{t:t+n}$$

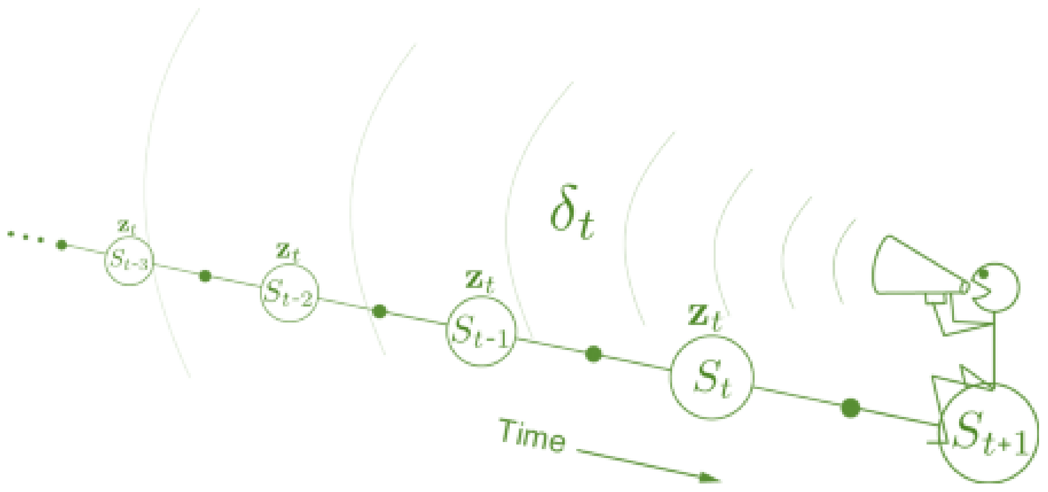


Forward View vs. Backward View

- N-step TD (and DP) are based on forward view



- TD(λ) is oriented backward in time



Reference

1. Udacity, https://github.com/udacity/deep-reinforcement-learning/blob/master/cheatsheet/cheatsheet.pdf