REINFORCEMENT LEARNING

1. On-Off-Policy

Q-learning is an off-policy. Notice when you want to take action a_t at time t+1, you chooise action based on $argmax\{Q\}$. This action whatever it may be, is not still realized. When you to the next time step t+1, you use the ϵ greedy approach to choose A_{t+1}. This time you take this action actually. So during updating you use action a based on $argmax\{Q\}$, but during interaction with the simulated environment you choose your action based on ϵ greedy approach applied to $Q(S_{t+1}, A_{t+1})$. This discripancy between action chosen during update for time $t+1$ and during trajectory/simulation generation is called off-policy. SARSA does not exhibit this issue and is an on-line policy. Both methods attempt to find the optimal Q^* and policy.

2. 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
$p(s', r|s, a)$
probability of next state s' and reward r, given current state s and current action a ($\mathbb{P}(S_{t+1} = s', R_{t+1} = r|S_t = s, A_t = a)$)

3. The Solution

π policy
if deterministic: $\pi(s) \in \mathcal{A}(s)$ for all $s \in \mathcal{S}$
if stochastic: $\pi(a|s) = \mathbb{P}(A_t = a|S_t = s)$ for all $s \in \mathcal{S}$ and $a \in \mathcal{A}(s)$

v_π	state-value function for policy π ($v_\pi(s) \doteq \mathbb{E}[G_t \mid S_t = s]$ for all $s \in \mathcal{S}$)
q_π	action-value function for policy π ($q_\pi(s, a) \doteq \mathbb{E}[G_t \mid S_t = s, A_t = a]$ 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 $a \in \mathcal{A}(s)$)

4. Bellman Equations

4.1. Bellman Expectation Equations.

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

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

4.2. Bellman Optimality Equations.

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

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

4.3. Useful Formulas for Deriving the Bellman Equations.

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

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

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

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

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

The reasoning for the above is as follows:

- (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) $v_\pi(s') = \mathbb{E}_\pi[G_{t+1}|S_{t+1} = s']$

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

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 $a \in \mathcal{A}(s)$)
$policy\text{-}stable \leftarrow false$
repeat
    $V \leftarrow$ **Policy_Evaluation**$(\text{MDP}, \pi, \theta)$
    $\pi' \leftarrow$ **Policy_Improvement**(MDP, V)
    if $\pi = \pi'$ **then**
        $policy\text{-}stable \leftarrow true$
    end
    $\pi \leftarrow \pi'$
until $policy\text{-}stable = true$;
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

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**$(\text{MDP}, \pi, V, max\_iterations)$
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

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

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 $s \in \mathcal{S}$ and $a \in \mathcal{A}(s)$
Initialize $N(s, a) = 0$ for all $s \in \mathcal{S}, a \in \mathcal{A}(s)$
for $i \leftarrow 1$ ***to*** *num_episodes* **do**
    $\epsilon \leftarrow \epsilon_i$
    $\pi \leftarrow \epsilon\text{-greedy}(Q)$
    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**
            $N(S_t, A_t) \leftarrow N(S_t, A_t) + 1$
            $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{1}{N(S_t, A_t)}(G_t - Q(S_t, A_t))$
    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 $s \in \mathcal{S}$ and $a \in \mathcal{A}(s)$)
for $i \leftarrow 1$ ***to*** *num_episodes* **do**
    $\epsilon \leftarrow \epsilon_i$
    $\pi \leftarrow \epsilon\text{-greedy}(Q)$
    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**
            $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(G_t - Q(S_t, A_t))$
    end
end
return π

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

Algorithm 14: Sarsamax (Q-Learning)

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}
    $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t))$
    $t \leftarrow t + 1$
  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