

# Reinforcement Learning Formula Sheet

Eddie Guo

## Multi-Armed Bandit Problem

Expected reward of action  $a$ :  $q_*(a) \equiv \mathbb{E}[R_t \mid A_t = a]$

Estimate of  $q_*(a)$  at time  $t$ :  $Q_t(a) \equiv \frac{\sum_{i=1}^{t-1} R_i \cdot \mathbb{1}_{A_i=a}}{\sum_{i=1}^{t-1} \mathbb{1}_{A_i=a}}$

Optimization:  $Q_{n+1} = Q_n + \frac{1}{n}(R_n - Q_n)$

$\lim_{t \rightarrow \infty} Q_t(a) = q_*(a)$  by LLN

Greedy action selection:  $A_t = \underset{a}{\operatorname{argmax}} Q_t(a)$

$\epsilon$ -greedy selection: greedy most of time but selects random action w/ small probability  $\epsilon$

Nonstationary problems: constant step-size parameter

$Q_{n+1} \equiv Q_n + \alpha(R_n - Q_n), \quad \alpha \in [0, 1)$

$Q_{n+1} = (1 - \alpha)Q_1 + \sum_{i=1}^n \alpha(1 - \alpha)^{n-i} R_i$

Notice exponentially decaying past rewards.

### 1: A simple bandit algorithm

Initialize, for  $a = 1$  to  $k$ :

$Q(a) \leftarrow 0$

$N(a) \leftarrow 0$

Loop:

$A \leftarrow \begin{cases} \underset{a}{\operatorname{argmax}} Q(a), & \text{with probability } 1 - \epsilon \\ \text{random action,} & \text{with probability } \epsilon \end{cases}$

$R \leftarrow \text{bandit}(A)$

$N(A) \leftarrow N(A) + 1$

$Q(A) \leftarrow Q(A) + \frac{1}{N(A)}[R - Q(A)]$

## Upper Confidence Bound (UCB) Action Selection

“Optimism in the face of uncertainty”

Same as greedy except initialize  $Q_t(a)$  to a high value, select value that optimizes an action  $A_t$ , and updates the upper bound to  $Q_t(a)$ .

$A_t \equiv \underset{a}{\operatorname{argmax}} \left[ Q_t(a) + c \sqrt{\frac{\ln t}{N_t(a)}} \right]$

## Finite Markov Decision Processes

State:  $S_t \in \mathcal{S}$ , Action:  $A_t \in \mathcal{A}(s)$ , Reward:  $R_{t+1} \in \mathcal{R} \subset \mathbb{R}$

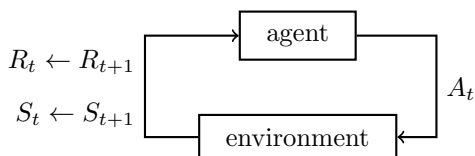
Transition dynamics fn (joint PMF):

Joint prob of next state  $s'$  and reward  $r$  given state  $s$  and action  $a$ .

$p(s', r \mid s, a) \equiv \Pr\{S_t = s', R_t = r \mid S_{t-1} = s, A_{t-1} = a\}$

$p: \mathcal{S} \times \mathcal{R} \times \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$

$\sum_{s' \in \mathcal{S}} \sum_{r \in \mathcal{R}} p(s', r \mid s, a) = 1, \quad \forall s \in \mathcal{S}, \forall a \in \mathcal{A}(s)$



## State-Transition Probabilities (Alternative Forms)

$p(s', r \mid s, a) \equiv \Pr\{S_t = s', R_t = r \mid S_{t-1} = s, A_{t-1} = a\}$

$p(s' \mid s, a) = \Pr\{S_t = s' \mid S_{t-1} = s, A_{t-1} = a\} = \sum_{r \in \mathcal{R}} p(s', r \mid s, a)$

$r(s, a) \equiv \mathbb{E}[R_t \mid S_{t-1} = s, A_{t-1} = a] = \sum_{r \in \mathcal{R}} r \sum_{s' \in \mathcal{S}} p(s', r \mid s, a)$

$r(s, a, s') \equiv \mathbb{E}[R_t \mid S_{t-1} = s, A_{t-1} = a, S_t = s'] = \sum_{r \in \mathcal{R}} r \frac{p(s', r \mid s, a)}{p(s' \mid s, a)}$

Markov property: future states of Markov process depend only on present state and not on past events.

Agent-envir interactions: episode  $\rightarrow$  terminal state  $\rightarrow$  reset

Goal of agent: maximize expected return,  $G_t$

Episodic tasks:  $G_t \equiv R_{t+1} + R_{t+2} + \dots + R_T$

## Continuing Tasks (no terminal state)

$G_t \equiv R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots + \gamma^{k-1} R_{t+k} + \dots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$

$G_t = R_{t+1} + \gamma G_{t+1}, \quad \sum_{k=0}^{\infty} \gamma^k = \frac{1}{1 - \gamma}, \quad \gamma \in [0, 1) \text{ is discount rate}$

$G_t \equiv \sum_{k=t+1}^T \gamma^{k-t-1} R_k \quad T = \infty \text{ or } \gamma = 1 \text{ (but not both)}$

Notice that future rewards are discounted more.

$\gamma = 0$ : agent only cares about immediate reward (greedy).

$\gamma \rightarrow 1$ : future rewards contribute more.

## Policies

Law of total expectation:  $\mathbb{E}[X] = \mathbb{E}[\mathbb{E}[X \mid Y]]$

Partition formula:  $\mathbb{E}[X] = \sum_i \mathbb{E}[X \mid A_i] P(A_i)$

Policy: mapping from states to probs of selecting each possible action.

$\pi(a \mid s) = p(a \mid s) = \Pr\{A_t = a \mid S_t = s\}$

Expectation of  $R_{t+1}$  in terms of  $\pi$  and  $p$ :

$\mathbb{E}[R_{t+1} \mid S_t = s] = \sum_a \pi(a \mid S_t) \sum_{s', r} p(s', r \mid s, a) r$

## Value Functions

Value fns give expected return  $G_t$  when starting in state  $s$  and following policy  $\pi$  thereafter.

State-value fn:  $v_\pi(s) \equiv \mathbb{E}_\pi[G_t \mid S_t = s] \quad G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$

Value of terminal state is always 0.

Action-value fn:  $q_\pi(s, a) \equiv \mathbb{E}_\pi[G_t \mid S_t = s, A_t = a]$

$v_\pi$  in terms of  $q_\pi$  and  $\pi$ :  $v_\pi(s) = \sum_a \pi(a \mid S_t) q_\pi(s, a)$

$q_\pi$  in terms of  $v_\pi$  and  $p$ :  $q_\pi(s, a) = \sum_{r, s'} p(s', r \mid s, a) [r + \gamma v_\pi(s')]$

## Bellman Equations

$$v_{\pi}(s) = \sum_a \pi(a | s) \sum_{s', r} p(s', r | s, a) [r + \gamma v_{\pi}(s')]$$

$$q_{\pi}(s, a) = \sum_{s', r} p(s', r | s, a) \left[ r + \gamma \sum_{a'} \pi(a' | s') q(s', a') \right]$$

Optimal value fns:  $\pi_1 \geq \pi_2 \iff v_{\pi_1}(s) \geq v_{\pi_2}(s), \forall s \in \mathcal{S}$

$$v_*(s) = \max_{\pi} v_{\pi}(s) = \max_a \sum_{s', r} p(s', r | s, a) [r + \gamma v_*(s')], \quad \forall s \in \mathcal{S}$$

$$q_*(s, a) = \max_{\pi} q_{\pi}(s, a) = \sum_{s', r} p(s', r | s, a) [r + \gamma \max_{a'} q_*(s', a')]$$

$$\forall s \in \mathcal{S}, \forall a \in \mathcal{A}$$

## Policy Evaluation

$$\pi_* = \operatorname{argmax}_a \sum_{s', r} p(s', r | s, a) [r + \gamma v_*(s')]$$

### 2: Iterative Policy Evaluation

Input  $\pi$ , the policy to be evaluated

$$\vec{V} \leftarrow \vec{0}, \vec{V}' \leftarrow \vec{0}$$

loop:

$$\Delta \leftarrow 0$$

loop for each  $s \in \mathcal{S}$ :

$$V'(s) \leftarrow \sum_a \pi(a | s) \sum_{s', r} p(s', r | s, a) [r + \gamma V(s')]$$

$$\Delta \leftarrow \max(\Delta, |V'(s) - V(s)|)$$

$$V \leftarrow V'$$

until  $\Delta < \theta$  (small positive number)

return  $V \approx v_{\pi}$

Policy improvement thm:  $q_{\pi}(s, \pi'(s)) \geq v_{\pi}(s), \quad \forall s \in \mathcal{S}$

$$\pi'(s) \equiv \operatorname{argmax}_a q_{\pi}(s, a) = \operatorname{argmax}_a \sum_{s', r} p(s', r | s, a) [r + \gamma v_{\pi}(s')]$$

### 3: Policy Iteration

1. Initialization

$V(s) \in \mathbb{R}$  and  $\pi(s) \in \mathcal{A}(s)$  arbitrarily  $\forall s \in \mathcal{S}$

2. Policy Evaluation

Loop:

$$\Delta \leftarrow 0$$

Loop for each  $s \in \mathcal{S}$ :

$$v \leftarrow V(s)$$

$$V(s) \leftarrow \sum_{s', r} p(s', r | s, \pi(s)) [r + \gamma V(s')]$$

$$\Delta \leftarrow \max(\Delta, |v - V(s)|)$$

3. Policy Improvement

*policy-stable*  $\leftarrow$  true

For each  $s \in \mathcal{S}$ :

$$\text{old-action} \leftarrow \pi(s)$$

$$\pi(s) \leftarrow \operatorname{argmax}_a \sum_{s', r} p(s', r | s, a) [r + \gamma V(s')]$$

If *old-action*  $\neq \pi(s)$ , then *policy-stable*  $\leftarrow$  false

If *policy-stable*, then stop and return  $V \approx v_*$  and  $\pi \approx \pi_*$ ; else go to 2.

## Monte Carlo Methods

- Reqs only sample sequences of states, actions, rewards from interactions w/ envir. Works in RL by averaging sample returns.
- MC only for episodic tasks b/c only upon completion of episode are value estimates and policies changed.

### 4: First-visit MC prediction for estimating $V \approx v_{\pi}$

Input: policy  $\pi$  to be evaluated

Initialize:

$$V(s) \in \mathbb{R}, \text{ arbitrarily } \forall s \in \mathcal{S}$$

$$\text{Returns}(s) \leftarrow \text{empty list } \forall s \in \mathcal{S}$$

Loop (for each episode):

Generate episode following  $\pi$

$$G \leftarrow 0$$

Loop for each step of episode,  $t = T - 1, T - 2, \dots, 0$ :

$$G \leftarrow \gamma G + R_{t+1}$$

Unless  $S_t$  appears in  $S_0, S_1, \dots, S_{t-1}$ :

Append  $G$  to  $\text{Returns}(S_t)$

$$V(S_t) \leftarrow \text{average}(\text{Returns}(S_t))$$

### MC Estimation of Action Values

$$\pi(s) \equiv \operatorname{argmax}_a q(s, a), \quad q_{\pi_k}(s, \pi_{k+1}(s)) \geq q_{\pi_k}(s, \pi_k(s)) \geq v_{\pi_k}(s)$$

### 5: First-visit MC prediction for estimating $V \approx v_{\pi}$

Input: policy  $\pi$  to be evaluated

Initialize:

$$\pi(s) \in \mathcal{A}(s), \text{ arbitrarily } \forall s \in \mathcal{S}$$

$$Q(s, a) \in \mathbb{R}, \text{ arbitrarily } \forall s \in \mathcal{S}, \forall a \in \mathcal{A}$$

$$\text{Returns}(s) \leftarrow \text{empty list } \forall s \in \mathcal{S}, \forall a \in \mathcal{A}$$

Loop (for each episode):

Choose  $S_0 \in \mathcal{S}, A_0 \in \mathcal{A}(S_0)$  randomly st all pairs have probabilities greater than 0

Generate episode from  $S_0, A_0$  following  $\pi$

$$G \leftarrow 0$$

Loop for each step of episode,  $t = T - 1, T - 2, \dots, 0$ :

$$G \leftarrow \gamma G + R_{t+1}$$

Unless  $S_t$  appears in  $S_0, A_0, \dots, S_{t-1}, A_{t-1}$ :

Append  $G$  to  $\text{Returns}(S_t)$

$$Q(S_t, A_t) \leftarrow \text{average}(\text{Returns}(S_t, A_t))$$

$$\pi(S_t) \leftarrow \operatorname{argmax}_a Q(S_t, a)$$

Note the last three lines can be made more efficient:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{1}{n} (G - Q(S_t, A_t))$$

$$\pi(S_t) \leftarrow \operatorname{argmax}_a Q(S_t, a)$$

### MC Control w/o Exploring Starts

On-policy: tries to evaluate or improve policy used to make decisions.

Off-policy: same as on-policy but policy is different from that used to generate data: target policy + behaviour policy.

$\epsilon$ -soft policy: all nongreedy actions given minimal probability of selection  $\epsilon/|\mathcal{A}(s)|$  whereas greedy action given probability  $1 - \epsilon + \epsilon/|\mathcal{A}(s)|$ .

## 6: On-policy first-visit MC control (for $\epsilon$ -soft policies, $\pi \approx \pi_*$ )

Algorithm parameter: small  $\epsilon > 0$

Initialize:

$\pi \leftarrow$  arbitrary  $\epsilon$ -soft policy  
 $Q(s, a) \in \mathbb{R}$ , arbitrarily  $\forall s \in \mathcal{S}, \forall a \in \mathcal{A}$   
 $Returns(s) \leftarrow$  empty list  $\forall s \in \mathcal{S}, \forall a \in \mathcal{A}$

Loop (for each episode):

Generate episode from  $S_0, A_0$  following  $\pi$   
 $G \leftarrow 0$   
 Loop for each step of episode,  $t = T - 1, T - 2, \dots, 0$ :  
 $G \leftarrow \gamma G + R_{t+1}$   
 Unless pair  $S_t, A_t$  appears in  $S_0, A_0, \dots, S_{t-1}, A_{t-1}$ :  
 Append  $G$  to  $Returns(S_t)$   
 $Q(S_t, A_t) \leftarrow \text{average}(Returns(S_t, A_t))$   
 $A^* \leftarrow \underset{a}{\operatorname{argmax}} Q(S_t, a)$   
 $\forall a \in \mathcal{A}(S_t)$ :  

$$\pi(a | S_t) \leftarrow \begin{cases} 1 - \epsilon + \epsilon/|\mathcal{A}(S_t)| & \text{if } a = A^* \\ \epsilon/|\mathcal{A}(S_t)| & \text{if } a \neq A^* \end{cases}$$

### Off-Policy Prediction via Importance Sampling

$$\Pr\{A_t, S_{t+1}, A_{t+1}, \dots, S_T \mid S_t, A_{t:T-1} \sim \pi\} = \prod_{k=t}^{T-1} \pi(A_k \mid S_k) p(S_{k+1} \mid S_k, A_k)$$

$$\mathbb{E}[G_t \mid S_t = s] = v_b(s) \quad \mathbb{E}[\rho_{t:T-1} G_t \mid S_t = s] = v_\pi(s)$$

$$\rho_{t:T-1} = \prod_{k=t}^{T-1} \frac{\pi(A_k \mid S_k)}{b(A_k \mid S_k)}$$

$$\text{Ordinary importance sampling: } V(s) \equiv \frac{\sum_{t \in \mathcal{T}(s)} \rho G_t}{|\mathcal{T}(s)|}$$

$$\text{Weighted importance sampling: } V(s) \equiv \frac{\sum_{t \in \mathcal{T}(s)} \rho G_t}{\sum_{t \in \mathcal{T}(s)} \rho}$$

Ordinary unbiased w/ high variance; weighted is biased w/ lower variance (preferred method).

### Incremental Implementation

Suppose we have seq of returns  $G_1, G_2, \dots, G_{n-1}$  all starting from same state with random weight  $W_i$ . We wish to estimate

$$V_n \equiv \frac{\sum_{k=1}^{n-1} G_k}{\sum_{k=1}^{n-1} W_k}, \quad n \geq 2$$

We can use the following equation:

$$V_{n+1} \equiv V_n + \frac{W_n}{C_n} (G_n - V_n), \quad n \geq 1$$

where  $C_{n+1} \equiv C_n + W_{n+1}$  and  $C_0 = 0$  ( $C_n$  is sum of weights).

## 7: Off-policy MC prediction (policy evaluation) $Q \approx Q_\pi$

Input: an arbitrary target policy  $\pi$

Initialize,  $\forall s \in \mathcal{S}, a \in \mathcal{A}(s)$ :

$Q(s, a) \in \mathbb{R}$  (arbitrarily)  
 $C(s, a) \leftarrow 0$

Loop (for each episode):

$b \leftarrow$  any policy w/ coverage of  $\pi$   
 Generate an episode following  $b$ :  $S_0, A_0, R_1, \dots$   
 $G \leftarrow 0$   
 $W \leftarrow 1$

Loop for each step of episode,  $t = T - 1, T - 2, \dots, 0$  while  $W \neq 0$ :

$G \leftarrow \gamma G + R_{t+1}$   
 $C(S_t, A_t) \leftarrow C(S_t, A_t) + W$   
 $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{W}{C(S_t, A_t)} [G - Q(S_t, A_t)]$   
 $W \leftarrow W \frac{\pi(A_t \mid S_t)}{b(A_t \mid S_t)}$

## 8: Off-policy MC control $\pi \approx \pi_*$

Initialize,  $\forall s \in \mathcal{S}, a \in \mathcal{A}(s)$ :

$Q(s, a) \in \mathbb{R}$  (arbitrarily)  
 $C(s, a) \leftarrow 0$   
 $\pi(s) \leftarrow \underset{a}{\operatorname{argmax}} Q(s, a)$

Loop (for each episode):

$b \leftarrow$  any policy w/ coverage of  $\pi$   
 Generate an episode following  $b$ :  $S_0, A_0, R_1, \dots$   
 $G \leftarrow 0$   
 $W \leftarrow 1$

Loop for each step of episode,  $t = T - 1, T - 2, \dots, 0$ :

$G \leftarrow \gamma G + R_{t+1}$   
 $C(S_t, A_t) \leftarrow C(S_t, A_t) + W$   
 $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{W}{C(S_t, A_t)} [G - Q(S_t, A_t)]$   
 $\pi(S_t) \leftarrow \underset{a}{\operatorname{argmax}} Q(S_t, a)$   
 If  $A_t \neq \pi(S_t)$  then exit inner Loop  
 $W \leftarrow W \frac{1}{b(A_t \mid S_t)}$

## Temporal-Difference Learning

TD(0) update:  $V(S_t) \leftarrow V(S_t) + \alpha [R_{t+1} + \gamma V(S_{t+1}) - V(S_t)]$

TD error:  $\delta_t \equiv R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$

## 9: Tabular TD(0) for estimating $v_\pi$

Input: policy  $\pi$  to be evaluated

Algorithm parameter: step size  $\alpha \in (0, 1]$

Initialize  $V(s)$ ,  $\forall s \in \mathcal{S}$  arbitrarily except  $V(\text{terminal}) = 0$

Loop (for each episode):

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

MC error:  $G_t - V(S_t) = \sum_{k=t}^{T-1} \gamma^{k-t} S_t$

## Sarsa: On-Policy TD Control

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)]$$

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

Algorithm parameter: step size  $\alpha \in (0, 1]$ , small  $\epsilon > 0$   
Initialize  $Q(s, a)$ ,  $\forall s \in \mathcal{S}, a \in \mathcal{A}(s)$  arbitrarily except  
 $Q(\text{terminal}, \cdot) = 0$

Loop (for each episode):

Initialize  $S$

Choose  $A$  from  $S$  using policy derived from  $Q$ :

Loop for each step of episode:

Take action  $A$ , observe  $R, S'$

Choose  $A'$  from  $S'$  using policy derived from  $Q$

$$Q(S, A) \leftarrow Q(S, A) + \alpha[R + \gamma Q(S', A') - Q(S, A)]$$

$S \leftarrow S'; A \leftarrow A'$

until  $S$  is terminal

## Q-Learning: Off-Policy TD Control

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

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

Algorithm parameter: step size  $\alpha \in (0, 1]$ , small  $\epsilon > 0$   
Initialize  $Q(s, a)$ ,  $\forall s \in \mathcal{S}, a \in \mathcal{A}(s)$  arbitrarily except  
 $Q(\text{terminal}, \cdot) = 0$

Loop (for each episode):

Initialize  $S$

Choose  $A$  from  $S$  using policy derived from  $Q$ :

Loop for each step of episode:

Choose  $A$  from  $S$  using policy derived from  $Q$

Take action  $A$ , observe  $R, S'$

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

$S \leftarrow S'$

until  $S$  is terminal

Double  $Q$ -learning addresses maximization bias problem. Instead of single  $Q(S, A)$  update, with 0.5 probability, choose one of:

$$\begin{cases} Q_1(S, A) + \alpha \left( R + \gamma Q_2(S', \arg\max_a Q_1(S', a)) - Q_1(S, A) \right) \\ Q_2(S, A) + \alpha \left( R + \gamma Q_1(S', \arg\max_a Q_2(S', a)) - Q_2(S, A) \right) \end{cases}$$

Can adapt this for Sarsa and expected Sarsa updates.

## Expected Sarsa

$$Q(S_t, A_t)$$

$$\leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma \mathbb{E}_\pi[Q(S_{t+1}, A_{t+1}) \mid S_{t+1}] - Q(S_t, A_t)]$$

$$\leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma \sum_a \pi(a \mid S_{t+1}) Q(S_{t+1}, a) - Q(S_t, A_t)]$$

Eliminates variance due to random selection of  $A_{t+1}$  from Sarsa.

## Models and Planning

model  $\xrightarrow{\text{planning}}$  policy

### 12: Random-sample one-step tabular Q-planning

Loop:

1. Select a state,  $S \in \mathcal{S}$  and action,  $A \in \mathcal{A}(S)$ , at random
2. Send  $S, A$  to a sample model and obtain sample next reward  $R$  and sample next state  $S'$
3. Apply one-step tabular Q-learning to  $S, A, R, S'$ :  

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

## Dyna-Q

### 13: Tabular Dyna-Q

Initialize  $Q(s, a)$  and  $Model(s, a)$ ,  $\forall s \in \mathcal{S}, a \in \mathcal{A}(s)$

Loop:

1.  $S \leftarrow$  current (nonterminal) state
2.  $A \leftarrow \epsilon$ -greedy( $S, Q$ )
3. Take action  $A$ ; observe resultant reward  $R$  and state  $S'$
4.  $Q(S, A) \leftarrow Q(S, A) + \alpha[R + \gamma \max_a Q(S', a) - Q(S, A)]$
5.  $Model(S, A) \leftarrow R, S'$  (assuming deterministic enviro)
6. Loop repeat  $n$  times:  
 $S \leftarrow$  random previously observed state  
 $A \leftarrow$  random action previously taken in  $S$   
 $R, S' \leftarrow Model(S, A)$   
 $Q(S, A) \leftarrow Q(S, A) + \alpha[R + \gamma \max_a Q(S', a) - Q(S, A)]$

Dyna-Q+ has reward of  $r + \kappa\sqrt{\tau}$ , where  $\kappa$  is a constant and  $\tau$  is the number of time steps since a given transition.