Temporal Difference Learning

Reinforcement Learning

September 29, 2020

MSU

Reminder: Value functions

State value function (**V-function**): $V^{\pi}(s) := \mathbb{E}_{\mathcal{T}\sim\pi} \sum_{t\geqslant 0} \gamma^t r_t \mid s_0 = s$

State-action value function (**Q-function**): $Q^{\pi}(s,a) := \mathbb{E}_{\mathcal{T}\sim\pi} \sum_{t\geqslant 0} \gamma^t r_t \mid s_0 = s, a_0 = a$

Reminder: Value functions

State value function (**V-function**): $V^{\pi}(s) := \mathbb{E}_{\mathcal{T}\sim\pi} \sum_{t\geqslant 0} \gamma^t r_t \mid s_0 = s$

State-action value function (**Q-function**): $Q^{\pi}(s,a) := \mathbb{E}_{\mathcal{T}\sim\pi} \sum_{t\geqslant 0} \gamma^t r_t \mid s_0 = s, a_0 = a$

Optimal V-function: $V^*(s) := \max_{\pi} V^{\pi}(s)$

Optimal Q-function: $Q^*(s,a) := \max_{\pi} Q^{\pi}(s,a)$

Reminder: Value functions

State value function (**V-function**): $V^\pi(s) := \mathbb{E}_{\mathcal{T}\sim\pi} \sum_{t\geqslant 0} \gamma^t r_t \mid s_0 = s$

State-action value function (**Q-function**): $Q^\pi(s,a) := \mathbb{E}_{\mathcal{T}\sim\pi} \sum_{t\geqslant 0} \gamma^t r_t \mid s_0 = s, a_0 = a$

Optimal V-function: $V^*(s) := \max_\pi V^\pi(s)$

Optimal Q-function: $Q^*(s,a) := \max_\pi Q^\pi(s,a)$

Bellman equations: $\forall \pi, s, a$:

$$Q^\pi(s,a) := r(s,a) + \gamma \mathbb{E}_{s'\sim p(s'|s,a)} \mathbb{E}_{a'\sim\pi(a'|s')} Q^\pi(s',a')$$

$$Q^*(s,a) := r(s,a) + \gamma \mathbb{E}_{s'\sim p(s'|s,a)} \max_{a'} Q^*(s',a')$$

Reminder: Dynamic Programming

Setup: $|\mathcal{S}| \ll \infty, |\mathcal{A}| \ll \infty$
Assumption: model is known

Reminder: Dynamic Programming

Setup: $|\mathcal{S}| \ll \infty, |\mathcal{A}| \ll \infty$
Assumption: model is known

Policy Iteration

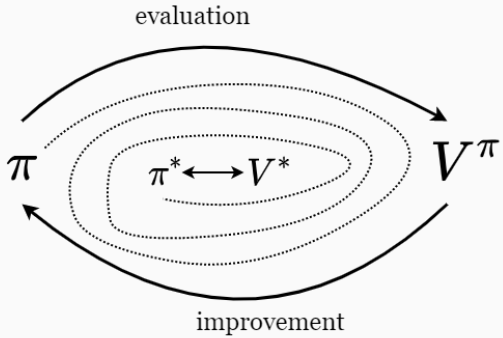
- **Policy Evaluation**:
  compute V^{π_k} for current policy π_k;
- **Policy Improvement**:
  $\pi_{k+1}(s) \leftarrow \operatorname*{argmax}_a Q^{\pi_k}(s, a)$;
- repeat;

Reminder: Dynamic Programming

Setup: $|\mathcal{S}| \ll \infty, |\mathcal{A}| \ll \infty$
Assumption: model is known

Policy Iteration

- **Policy Evaluation**:
  compute V^{π_k} for current policy π_k;
- **Policy Improvement**:
  $\pi_{k+1}(s) \leftarrow \underset{a}{\operatorname{argmax}}\, Q^{\pi_k}(s, a)$;
- repeat;



Reminder: Dynamic Programming

Setup: $|\mathcal{S}| \ll \infty, |\mathcal{A}| \ll \infty$
Assumption: model is known

Value Iteration

↻ solve Bellman optimality equation
(e. g. for V^*):

$$V^*_{k+1}(s) \leftarrow \max_a \left[r(s,a) + \gamma \mathbb{E}_{s'} V^*_k(s') \right]$$

- $\pi^*(s) \leftarrow \operatorname{argmax}_a Q^*(s,a)$ is optimal;

Reminder: Dynamic Programming

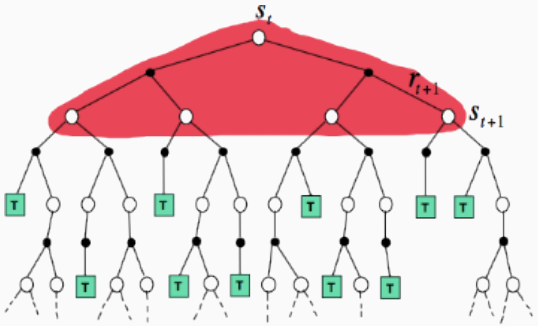
Setup: $|\mathcal{S}| \ll \infty, |\mathcal{A}| \ll \infty$
Assumption: model is known

Value Iteration

↻ solve Bellman optimality equation
(e. g. for V^*):

$$V^*_{k+1}(s) \leftarrow \max_a \left[r(s,a) + \gamma \mathbb{E}_{s'} V^*_k(s') \right]$$

- $\pi^*(s) \leftarrow \operatorname{argmax}_a Q^*(s,a)$ is optimal;



Trial and Error learning

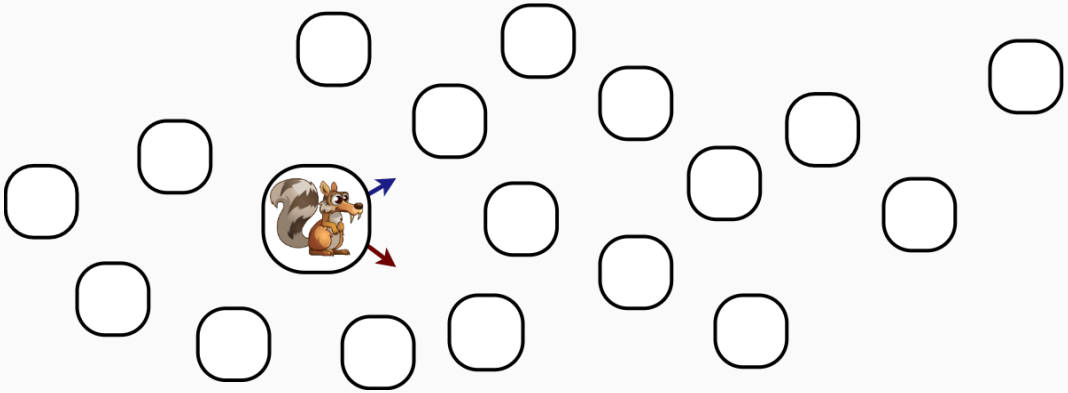
Setup: $|\mathcal{S}| \ll \infty, |\mathcal{A}| \ll \infty$

~~**Assumption:** model is known~~

Trial and Error learning

Setup: $|\mathcal{S}| \ll \infty, |\mathcal{A}| \ll \infty$

~~**Assumption:** model is known~~



Idea 1: Model-based RL

Learn model through experience

Use VI / PI with learned model to find optimal policy

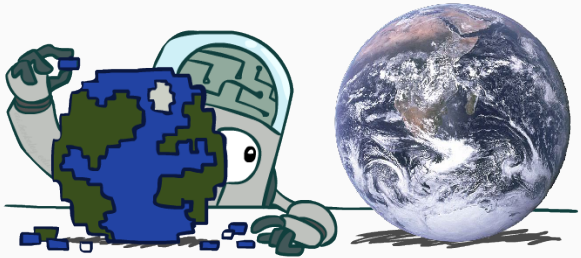
Idea 1: Model-based RL

Learn model through experience
Use VI / PI with learned model to find optimal policy

What is easier to create:

a) universe

b) brain



Idea 1: Model-based RL

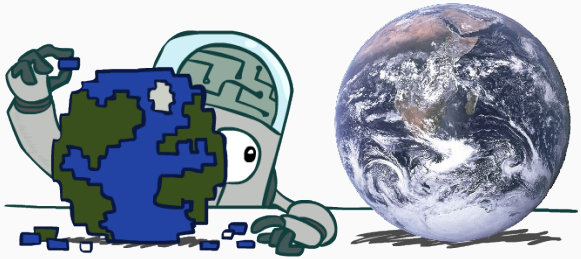
Learn model through experience
Use VI / PI with learned model to find optimal policy

What is easier to create:

a) universe

- × $\mathcal{S} \times \mathcal{A} \to P(\mathcal{S})$
- ✓ (almost) supervised learning

b) brain



Idea 1: Model-based RL

Learn model through experience
Use VI / PI with learned model to find optimal policy

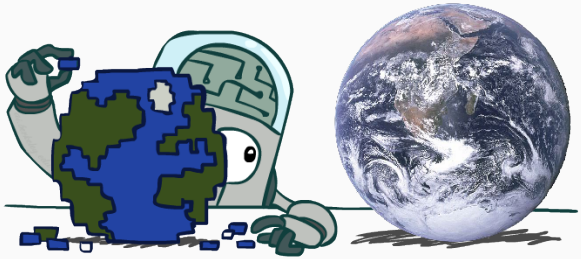
What is easier to create:

a) universe

- × $\mathcal{S} \times \mathcal{A} \to P(\mathcal{S})$
- ✓ (almost) supervised learning

b) brain

- ✓ $\mathcal{S} \to \mathcal{A}$
- × dataset is not provided :(



Idea 1: Model-based RL

Learn model through experience
Use VI / PI with learned model to find optimal policy

What is easier to create:

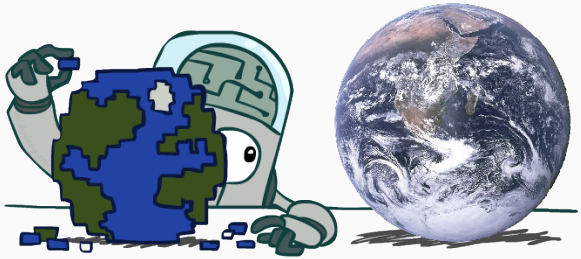
a) universe

- × $\mathcal{S} \times \mathcal{A} \to P(\mathcal{S})$
- ✓ (almost) supervised learning

b) brain

- ✓ $\mathcal{S} \to \mathcal{A}$
- × dataset is not provided :(

?!?



Idea 2: Model-free RL

Do not try to learn the model

Map states and actions to reward directly

Idea 2: Model-free RL

Do not try to learn the model
Map states and actions to reward directly

Where do we use the model?

Policy Iteration

- evaluate policy
  e. g. by solving Bellman equation:

$$V^{\pi_k}(s) = \mathbb{E}_a \left[r(s,a) + \gamma \mathbb{E}_{s'} V^{\pi_k}(s') \right]$$

- $\pi_{k+1}(s) \leftarrow \underset{a}{\operatorname{argmax}}\, Q^{\pi_k}(s,a)$;
- repeat;

Value Iteration

↻ solve Bellman optimality equation
(e. g. for V^*):

$$V^*_{k+1}(s) \leftarrow \max_a \left[r(s,a) + \gamma \mathbb{E}_{s'} V^*_k(s') \right]$$

- $\pi^*(s) \leftarrow \underset{a}{\operatorname{argmax}}\, Q^*(s,a)$ is optimal;

Idea 2: Model-free RL

Do not try to learn the model
Map states and actions to reward directly

Where do we use the model?

Policy Iteration

- evaluate policy
  e. g. by solving Bellman equation:

$$V^{\pi_k}(s) = \mathbb{E}_a \left[r(s,a) + \gamma \mathbb{E}_{s'} V^{\pi_k}(s') \right]$$

- $\pi_{k+1}(s) \leftarrow \operatorname{argmax}_a Q^{\pi_k}(s,a)$;
- repeat;

Value Iteration

↻ solve Bellman optimality equation
(e. g. for V^*):

$$V^*_{k+1}(s) \leftarrow \max_a \left[r(s,a) + \gamma \mathbb{E}_{s'} V^*_k(s') \right]$$

- $\pi^*(s) \leftarrow \operatorname{argmax}_a Q^*(s,a)$ is optimal;

Switching to Q-function

Policy Iteration

- evaluate policy:

$$Q^{\pi_k}(s,a) = r(s,a) + \gamma \mathbb{E}_{s'} \mathbb{E}_{a'} Q^{\pi_k}(s',a')$$

- $\pi_{k+1}(s) \leftarrow \operatorname*{argmax}_a Q^{\pi_k}(s,a)$;
- repeat;

Value Iteration

↻ solve Bellman optimality equation:

$$Q^*_{k+1}(s,a) \leftarrow r(s,a) + \gamma \mathbb{E}_{s'} \max_{a'} Q^*_k(s',a')$$

- $\pi^*(s) \leftarrow \operatorname*{argmax}_a Q^*(s,a)$ is optimal;

Switching to Q-function

Policy Iteration

- evaluate policy:

$$Q^{\pi_k}(s,a) = r(s,a) + \gamma \mathbb{E}_{s'} \mathbb{E}_{a'} Q^{\pi_k}(s',a')$$

- $\pi_{k+1}(s) \leftarrow \operatorname*{argmax}_a Q^{\pi_k}(s,a)$;
- repeat;

Value Iteration

↻ solve Bellman optimality equation:

$$Q^*_{k+1}(s,a) \leftarrow r(s,a) + \gamma \mathbb{E}_{s'} \max_{a'} Q^*_k(s',a')$$

- $\pi^*(s) \leftarrow \operatorname*{argmax}_a Q^*(s,a)$ is optimal;

$$x = \mathbb{E}_{s'} f(s', x), \quad x - ?$$

Switching to Q-function

Policy Iteration

- evaluate policy:

$$Q^{\pi_k}(s,a) = r(s,a) + \gamma \mathbb{E}_{s'} \mathbb{E}_{a'} Q^{\pi_k}(s',a')$$

- $\pi_{k+1}(s) \leftarrow \operatorname*{argmax}_a Q^{\pi_k}(s,a)$;
- repeat;

Value Iteration

↻ solve Bellman optimality equation:

$$Q^*_{k+1}(s,a) \leftarrow r(s,a) + \gamma \mathbb{E}_{s'} \max_{a'} Q^*_k(s',a')$$

- $\pi^*(s) \leftarrow \operatorname*{argmax}_a Q^*(s,a)$ is optimal;

$$x = \mathbb{E}_{s'} f(s', x), \quad x - ?$$

Can we derive *some* model-free PI / VI?

Monte-Carlo Backup

How to solve this: $x = \mathbb{E}_{s'} f(s')$

Monte-Carlo Backup

How to solve this: $x = \mathbb{E}_{s'} f(s')$

Monte-Carlo estimation:

$$Q^{\pi}(s, a) \approx \frac{1}{N} \sum_{i=0}^{N} R(\mathcal{T}_i)$$

where $\mathcal{T}_i \sim \pi \mid s_0 = s, a_0 = a$

Monte-Carlo Backup

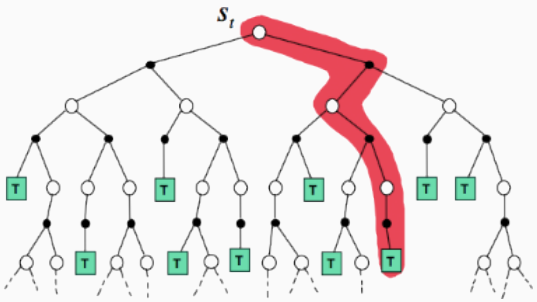
How to solve this: $x = \mathbb{E}_{s'} f(s')$

Monte-Carlo estimation:

$$Q^{\pi}(s,a) \approx \frac{1}{N} \sum_{i=0}^{N} R(\mathcal{T}_i)$$

where $\mathcal{T}_i \sim \pi \mid s_0 = s, a_0 = a$

Can we have i. i. d. samples?



Monte Carlo Algorithm

Monte Carlo Algorithm

Initialize $Q^\pi(s, a)$ and policy $\pi(s)$ arbitrarily.

for $k = 0, 1, 2 \ldots$

- play several games using π;
- estimate $Q^\pi(s, a)$ using Monte Carlo;
- $\pi(s) \leftarrow \underset{a}{\operatorname{argmax}}\, Q^\pi(s, a)$

! requires infinite samples for each pair s, a;

Monte Carlo Algorithm

Monte Carlo Algorithm

Initialize $Q^\pi(s,a)$ and policy $\pi(s)$ arbitrarily.

for $k = 0, 1, 2 \ldots$

- play several games using π;
- estimate $Q^\pi(s,a)$ using Monte Carlo;
- $\pi(s) \leftarrow \underset{a}{\operatorname{argmax}}\, Q^\pi(s,a)$

! requires infinite samples for each pair s, a;

× still needs full episodes;

× still do not utilize MDP structure;

× wastes information about state connections;

× high variance of collected samples;

× no clear theoretical guarantees;

Monte Carlo Algorithm

Monte Carlo Algorithm

Initialize $Q^\pi(s,a)$ and policy $\pi(s)$ arbitrarily.

for $k = 0, 1, 2 \ldots$

- play several games using π;
- update $Q^\pi(s,a)$ with new samples;
- $\pi(s) \leftarrow \operatorname{argmax}_a Q^\pi(s,a)$

! requires infinite samples for each pair s, a;

× still needs full episodes;

× still do not utilize MDP structure;

× wastes information about state connections;

× high variance of collected samples;

× no clear theoretical guarantees;

Reuse samples from previous policy
(for example, with smaller weight)

Closer look at Monte Carlo

Suppose we want to compute online mean m of i. i. d. samples $x_1, x_2 \ldots$. On step k:

$$m_k := \frac{1}{k} \sum_{i=1}^{k} x_i =$$

Closer look at Monte Carlo

Suppose we want to compute online mean m of i. i. d. samples $x_1, x_2 \ldots$. On step k:

$$m_k := \frac{1}{k}\sum_{i=1}^{k} x_i = \frac{k-1}{k} m_{k-1} + \frac{1}{k} x_k =$$

Closer look at Monte Carlo

Suppose we want to compute online mean m of i. i. d. samples $x_1, x_2 \ldots$. On step k:

$$m_k := \frac{1}{k}\sum_{i=1}^{k} x_i = \frac{k-1}{k} m_{k-1} + \frac{1}{k} x_k = \left(1 - \frac{1}{k}\right) m_{k-1} + \frac{1}{k} x_k =$$

Closer look at Monte Carlo

Suppose we want to compute online mean m of i. i. d. samples $x_1, x_2 \ldots$. On step k:

$$m_k := \frac{1}{k}\sum_{i=1}^{k} x_i = \frac{k-1}{k} m_{k-1} + \frac{1}{k} x_k = \left(1 - \frac{1}{k}\right) m_{k-1} + \frac{1}{k} x_k =$$

$$= \left\{\alpha_k := \frac{1}{k}\right\} = \underbrace{(1-\alpha_k)\, m_{k-1} + \alpha_k x_k}_{\text{exponential smoothing}} =$$

Closer look at Monte Carlo

Suppose we want to compute online mean m of i. i. d. samples $x_1, x_2 \ldots$. On step k:

$$m_k := \frac{1}{k}\sum_{i=1}^{k} x_i = \frac{k-1}{k} m_{k-1} + \frac{1}{k} x_k = \left(1 - \frac{1}{k}\right) m_{k-1} + \frac{1}{k} x_k =$$

$$= \left\{\alpha_k := \frac{1}{k}\right\} = \underbrace{(1-\alpha_k)\, m_{k-1} + \alpha_k x_k}_{\text{exponential smoothing}} = \underbrace{m_{k-1} + \alpha_k (x_k - m_{k-1})}_{\approx \text{ stoch. gradient descent}}$$

Closer look at Monte Carlo

Suppose we want to compute online mean m of i. i. d. samples $x_1, x_2 \ldots$. On step k:

$$m_k := \frac{1}{k}\sum_{i=1}^{k} x_i = \frac{k-1}{k} m_{k-1} + \frac{1}{k} x_k = \left(1 - \frac{1}{k}\right) m_{k-1} + \frac{1}{k} x_k =$$

$$= \left\{\alpha_k := \frac{1}{k}\right\} = \underbrace{(1-\alpha_k)\, m_{k-1} + \alpha_k x_k}_{\text{exponential smoothing}} = \underbrace{m_{k-1} + \alpha_k (x_k - m_{k-1})}_{\approx \text{ stoch. gradient descent}}$$

Closer look at Monte Carlo

Suppose we want to compute online mean m of i. i. d. samples $x_1, x_2 \dots$. On step k:

$$m_k := \frac{1}{k}\sum_{i=1}^{k} x_i = \frac{k-1}{k} m_{k-1} + \frac{1}{k} x_k = \left(1 - \frac{1}{k}\right) m_{k-1} + \frac{1}{k} x_k =$$

$$= \left\{\alpha_k := \frac{1}{k}\right\} = \underbrace{(1-\alpha_k)\, m_{k-1} + \alpha_k x_k}_{\text{exponential smoothing}} = \underbrace{m_{k-1} + \alpha_k (x_k - m_{k-1})}_{\approx \text{ stoch. gradient descent}}$$

Convergence:

$m_k \xrightarrow[k\to\infty]{\text{w. p. 1}} \mathbb{E}x$ if learning rate satisfies **Robbins–Monro conditions**:

$$\sum_{k=1}^{+\infty} \alpha_k = +\infty, \quad \sum_{k=1}^{+\infty} \alpha_k^2 < +\infty$$

Solving equations

Equation	Method	Update
$x = f(x)$	Point Iteration	$x_{k+1} := f(x_k)$
$x = \mathbb{E}_{s'} f(s')$	Exponential smoothing	$x_{k+1} := x_k + \alpha_k \left(f(s') - x_k\right)$ $s' \sim p(s')$
$x = \mathbb{E}_{s'} f(s', x)$	Stochastic approximation	

Solving equations

Equation	Method	Update
$x = f(x)$	Point Iteration	$x_{k+1} := f(x_k)$
$x = \mathbb{E}_{s'} f(s')$	Exponential smoothing	$x_{k+1} := x_k + \alpha_k \left(f(s') - x_k\right)$ $s' \sim p(s')$
$x = \mathbb{E}_{s'} f(s', x)$	Stochastic approximation	$x_{k+1} := x_k + \alpha_k \left(f(s', x_k) - x_k\right)$ $s' \sim p(s')$

Temporal Difference: intuition

View 1: solving Bellman expectation equation using Robbins-Monro scheme:

$$Q^{\pi}(s,a) = \mathbb{E}_{s'}\underbrace{\left[r(s,a) + \gamma \mathbb{E}_{a'} Q^{\pi}(s',a')\right]}_{f(s',x)}$$

Temporal Difference: intuition

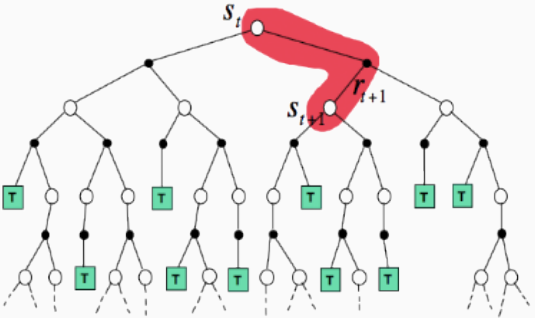
View 1: solving Bellman expectation equation using Robbins-Monro scheme:

$$Q^\pi(s,a) = \mathbb{E}_{s'}\underbrace{\left[r(s,a) + \gamma \mathbb{E}_{a'} Q^\pi(s',a')\right]}_{f(s',x)} = \mathbb{E}_{s'}\mathbb{E}_{a'}\underbrace{\left[r(s,a) + \gamma Q^\pi(s',a')\right]}_{f(s',a',x)}$$

Temporal Difference: intuition

View 1: solving Bellman expectation equation using Robbins-Monro scheme:

$$Q^\pi(s,a) = \mathbb{E}_{s'}\underbrace{\left[r(s,a) + \gamma \mathbb{E}_{a'} Q^\pi(s',a')\right]}_{f(s',x)} = \mathbb{E}_{s'}\mathbb{E}_{a'}\underbrace{\left[r(s,a) + \gamma Q^\pi(s',a')\right]}_{f(s',a',x)}$$



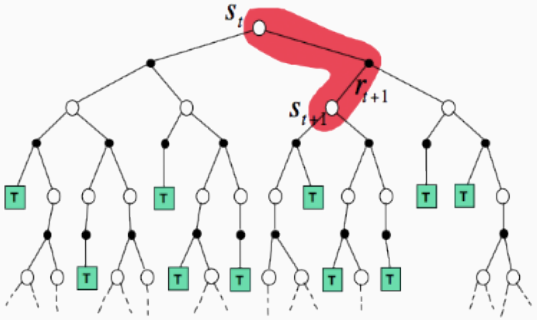
Temporal Difference: intuition

View 1: solving Bellman expectation equation using Robbins-Monro scheme:

$$Q^\pi(s,a) = \mathbb{E}_{s'}\underbrace{\left[r(s,a) + \gamma \mathbb{E}_{a'} Q^\pi(s',a')\right]}_{f(s',x)} = \mathbb{E}_{s'}\mathbb{E}_{a'}\underbrace{\left[r(s,a) + \gamma Q^\pi(s',a')\right]}_{f(s',a',x)}$$

View 2: one-step bootstrapping: approximating future rewards using current approximation.

$$r + \gamma \underbrace{\left(r' + \gamma r'' + \gamma^2 r''' + \dots\right)}_{\approx Q_k^\pi(s',a')}$$



Temporal Difference: intuition

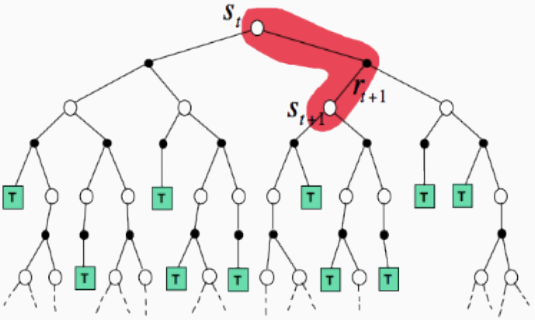
View 1: solving Bellman expectation equation using Robbins-Monro scheme:

$$Q^\pi(s,a) = \mathbb{E}_{s'} \underbrace{\left[r(s,a) + \gamma \mathbb{E}_{a'} Q^\pi(s',a')\right]}_{f(s',x)} = \mathbb{E}_{s'}\mathbb{E}_{a'} \underbrace{\left[r(s,a) + \gamma Q^\pi(s',a')\right]}_{f(s',a',x)}$$

View 2: one-step bootstrapping: approximating future rewards using current approximation.

$$r + \gamma \underbrace{\left(r' + \gamma r'' + \gamma^2 r''' + \dots\right)}_{\approx Q_k^\pi(s',a')}$$

our own predictions used as targets!



Temporal Difference

For transition (s, a, r, s', a'):

$$Q^{\pi}_{k+1}(s,a) \leftarrow Q^{\pi}_{k}(s,a) + \alpha_k \underbrace{\left(\overbrace{r + \gamma Q^{\pi}_{k}(s',a')}^{\text{Bellman target}} - Q^{\pi}_{k}(s,a) \right)}_{\text{temporal difference}}$$

Temporal Difference

For transition (s, a, r, s', a'):

$$Q^{\pi}_{k+1}(s,a) \leftarrow Q^{\pi}_{k}(s,a) + \alpha_k \underbrace{\left(\overbrace{r + \gamma Q^{\pi}_{k}(s',a')}^{\textbf{Bellman target}} - Q^{\pi}_{k}(s,a) \right)}_{\textbf{temporal difference}}$$

s, a — considered **fixed**.

$s' \sim p(s' \mid s, a)$ — **random variable** from interaction experience.

Temporal Difference

For transition (s, a, r, s', a'):

$$Q^{\pi}_{k+1}(s,a) \leftarrow Q^{\pi}_{k}(s,a) + \alpha_k \underbrace{\left(\overbrace{r + \gamma Q^{\pi}_{k}(s',a')}^{\textbf{Bellman target}} - Q^{\pi}_{k}(s,a) \right)}_{\textbf{temporal difference}}$$

s, a — considered **fixed**.

$s' \sim p(s' \mid s, a)$ — **random variable** from interaction experience.

$a' \sim \pi(a' \mid s')$ — **random variable** from algorithm.

Temporal Difference vs Monte Carlo

$$Q^{\pi}_{k+1}(s,a) \leftarrow Q^{\pi}_{k}(s,a) + \alpha_k \left(y(s,a) - Q^{\pi}_{k}(s,a) \right)$$

Which is better?

Temporal Difference

$y(s,a) := r + \gamma Q^{\pi}_{k}(s',a')$

Monte Carlo

$y(s,a) := r + \gamma r' + \gamma^2 r'' + \dots$

Temporal Difference vs Monte Carlo

$$Q^{\pi}_{k+1}(s,a) \leftarrow Q^{\pi}_{k}(s,a) + \alpha_k \left(y(s,a) - Q^{\pi}_{k}(s,a) \right)$$

Which is better?

Temporal Difference

$y(s,a) := r + \gamma Q^{\pi}_{k}(s',a')$

✓ updates after each step

× slow reward propagation

Monte Carlo

$y(s,a) := r + \gamma r' + \gamma^2 r'' + \dots$

× updates at the end of the episode

✓ fast reward propagation

Temporal Difference vs Monte Carlo

$$Q^{\pi}_{k+1}(s,a) \leftarrow Q^{\pi}_{k}(s,a) + \alpha_k \left(y(s,a) - Q^{\pi}_{k}(s,a) \right)$$

Which is better?

Temporal Difference

$y(s,a) := r + \gamma Q^{\pi}_{k}(s',a')$

- ✓ updates after each step
- × slow reward propagation
- ✓ low variance
- × introduces bias

Monte Carlo

$y(s,a) := r + \gamma r' + \gamma^2 r'' + \dots$

- × updates at the end of the episode
- ✓ fast reward propagation
- × high variance
- ✓ unbiased

Temporal Difference vs Monte Carlo

$$Q^{\pi}_{k+1}(s,a) \leftarrow Q^{\pi}_{k}(s,a) + \alpha_k \left(y(s,a) - Q^{\pi}_{k}(s,a) \right)$$

Which is better?

Temporal Difference

$y(s,a) := r + \gamma Q^{\pi}_{k}(s',a')$

- ✓ updates after each step
- × slow reward propagation
- ✓ low variance
- × introduces bias

Monte Carlo

$y(s,a) := r + \gamma r' + \gamma^2 r'' + \dots$

- × updates at the end of the episode
- ✓ fast reward propagation
- × high variance
- ✓ unbiased

Best estimation is somewhere in between (see TD(λ)).

Value Iteration and Policy Iteration connection

What if we improve our policy after every step?

$$\pi_k(s) \leftarrow \underset{a}{\operatorname{argmax}}\, Q_k^{\pi}(s, a)$$

Value Iteration and Policy Iteration connection

What if we improve our policy after every step?

$$\pi_k(s) \leftarrow \underset{a}{\operatorname{argmax}}\, Q_k^{\pi}(s,a)$$

$$Q_{k+1}^{\pi}(s,a) \leftarrow Q_k^{\pi}(s,a) + \alpha_k \left(r + \gamma Q_k^{\pi}(s',a') - Q_k^{\pi}(s,a)\right) =$$

Value Iteration and Policy Iteration connection

What if we improve our policy after every step?

$$\pi_k(s) \leftarrow \underset{a}{\operatorname{argmax}}\, Q_k^\pi(s,a)$$

$$Q_{k+1}^\pi(s,a) \leftarrow Q_k^\pi(s,a) + \alpha_k \left(r + \gamma Q_k^\pi(s',a') - Q_k^\pi(s,a)\right) =$$
$$= Q_k^\pi(s,a) + \alpha_k \left(r + \gamma Q_k^\pi(s',\pi_k(s')) - Q_k^\pi(s,a)\right) =$$

Value Iteration and Policy Iteration connection

What if we improve our policy after every step?

$$\pi_k(s) \leftarrow \underset{a}{\operatorname{argmax}}\, Q_k^\pi(s,a)$$

$$\begin{aligned} Q_{k+1}^\pi(s,a) &\leftarrow Q_k^\pi(s,a) + \alpha_k \left(r + \gamma Q_k^\pi(s',a') - Q_k^\pi(s,a)\right) = \\ &= Q_k^\pi(s,a) + \alpha_k \left(r + \gamma Q_k^\pi(s',\pi_k(s')) - Q_k^\pi(s,a)\right) = \\ &= Q_k^\pi(s,a) + \alpha_k \left(r + \gamma Q_k^\pi(s', \underset{a'}{\operatorname{argmax}}\, Q_k^\pi(s',a')) - Q_k^\pi(s,a)\right) = \end{aligned}$$

Value Iteration and Policy Iteration connection

What if we improve our policy after every step?

$$\pi_k(s) \leftarrow \operatorname*{argmax}_a Q_k^\pi(s,a)$$

$$\begin{aligned}
Q_{k+1}^\pi(s,a) &\leftarrow Q_k^\pi(s,a) + \alpha_k \left(r + \gamma Q_k^\pi(s',a') - Q_k^\pi(s,a)\right) = \\
&= Q_k^\pi(s,a) + \alpha_k \left(r + \gamma Q_k^\pi(s',\pi_k(s')) - Q_k^\pi(s,a)\right) = \\
&= Q_k^\pi(s,a) + \alpha_k \left(r + \gamma Q_k^\pi(s', \operatorname*{argmax}_{a'} Q_k^\pi(s',a')) - Q_k^\pi(s,a)\right) = \\
&= Q_k^\pi(s,a) + \alpha_k \left(r + \gamma \max_{a'} Q_k^\pi(s',a') - Q_k^\pi(s,a)\right)
\end{aligned}$$

Value Iteration and Policy Iteration connection

What if we improve our policy after every step?

$$\pi_k(s) \leftarrow \operatorname*{argmax}_a Q_k^\pi(s,a)$$

$$\begin{aligned}
Q_{k+1}^\pi(s,a) &\leftarrow Q_k^\pi(s,a) + \alpha_k \left(r + \gamma Q_k^\pi(s',a') - Q_k^\pi(s,a)\right) = \\
&= Q_k^\pi(s,a) + \alpha_k \left(r + \gamma Q_k^\pi(s',\pi_k(s')) - Q_k^\pi(s,a)\right) = \\
&= Q_k^\pi(s,a) + \alpha_k \left(r + \gamma Q_k^\pi(s', \operatorname*{argmax}_{a'} Q_k^\pi(s',a')) - Q_k^\pi(s,a)\right) = \\
&= Q_k^\pi(s,a) + \alpha_k \left(r + \gamma \max_{a'} Q_k^\pi(s',a') - Q_k^\pi(s,a)\right)
\end{aligned}$$

Value Iteration and Policy Iteration connection

What if we improve our policy after every step?

$$\pi_k(s) \leftarrow \operatorname*{argmax}_a Q_k^\pi(s,a)$$

$$\begin{aligned}
Q_{k+1}^\pi(s,a) &\leftarrow Q_k^\pi(s,a) + \alpha_k \left(r + \gamma Q_k^\pi(s',a') - Q_k^\pi(s,a)\right) = \\
&= Q_k^\pi(s,a) + \alpha_k \left(r + \gamma Q_k^\pi(s',\pi_k(s')) - Q_k^\pi(s,a)\right) = \\
&= Q_k^\pi(s,a) + \alpha_k \left(r + \gamma Q_k^\pi(s', \operatorname*{argmax}_{a'} Q_k^\pi(s',a')) - Q_k^\pi(s,a)\right) = \\
&= Q_k^\pi(s,a) + \alpha_k \left(r + \gamma \max_{a'} Q_k^\pi(s',a') - Q_k^\pi(s,a)\right)
\end{aligned}$$

View 1: Policy Iteration with policy improvement after *each* policy evaluation update;
View 2: solving Bellman optimality equation (model-free Value Iteration);

Convergence of Q-learning updates

Let $Q_0^*(s, a)$ be initialized arbitrary, and for every s, a the following update is used:

$$Q_{k+1}^*(s, a) \leftarrow Q_k^*(s, a) + \alpha_k(s, a) \left(r(s, a) + \gamma \max_{a'} Q_k^*(s'_{k,s,a}, a') - Q_k^*(s, a) \right)$$

Convergence of Q-learning updates

Let $Q_0^*(s, a)$ be initialized arbitrary, and for every s, a the following update is used:

$$Q_{k+1}^*(s, a) \leftarrow Q_k^*(s, a) + \alpha_k(s, a) \left(r(s, a) + \gamma \max_{a'} Q_k^*(s'_{k,s,a}, a') - Q_k^*(s, a) \right)$$

Then, if:

- $s'_{k,s,a} \sim p(s' \mid s, a)$;

Convergence of Q-learning updates

Let $Q_0^*(s, a)$ be initialized arbitrary, and for every s, a the following update is used:

$$Q_{k+1}^*(s, a) \leftarrow Q_k^*(s, a) + \alpha_k(s, a)\left(r(s, a) + \gamma \max_{a'} Q_k^*(s'_{k,s,a}, a') - Q_k^*(s, a)\right)$$

Then, if:

- $s'_{k,s,a} \sim p(s' \mid s, a)$;
- with probability 1 for every s, a learning rate $\alpha_k(s, a) \in [0, 1]$ satisfies Robbins-Monro conditions:

$$\sum_{k=0}^{\infty} \alpha_k(s, a) = +\infty \quad \sum_{k=0}^{\infty} \alpha_k^2(s, a) < +\infty$$

Convergence of Q-learning updates

Let $Q_0^*(s, a)$ be initialized arbitrary, and for every s, a the following update is used:

$$Q_{k+1}^*(s, a) \leftarrow Q_k^*(s, a) + \alpha_k(s, a) \left(r(s, a) + \gamma \max_{a'} Q_k^*(s'_{k,s,a}, a') - Q_k^*(s, a) \right)$$

Then, if:

- $s'_{k,s,a} \sim p(s' \mid s, a)$;
- with probability 1 for every s, a learning rate $\alpha_k(s, a) \in [0, 1]$ satisfies Robbins-Monro conditions:

$$\sum_{k=0}^{\infty} \alpha_k(s, a) = +\infty \quad \sum_{k=0}^{\infty} \alpha_k^2(s, a) < +\infty$$

then $Q_k^*(s, a) \xrightarrow[k \to \infty]{\text{w. p. } 1} Q^*(s, a)$

Exploration vs Exploitation

Analogy 1

Global optimization

Exploration vs Exploitation

Analogy 1

Global optimization

Exploration:

random search

Exploration vs Exploitation

Analogy 1

Global optimization

Exploration:

random search

Exploitation:

local optimization

Exploration vs Exploitation

Analogy 1
Global optimization

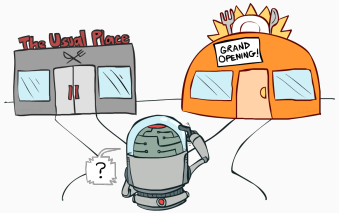
Exploration:
random search

Exploitation:
local optimization

Analogy 2
Multi-armed bandits
(regret minimization)

Exploration:
try something new

Exploitation:
try your favourite



Exploration vs Exploitation

Analogy 1
Global optimization

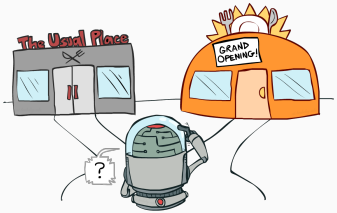
Exploration:
random search

Exploitation:
local optimization

Analogy 2
Multi-armed bandits
(regret minimization)

Exploration:
try something new

Exploitation:
try your favourite



Analogy 3
General MDP

Exploration:
explore the environment

Exploitation:
solve the task

ε-greedy exploration

Do something random *sometimes*

Let's call policy ε-greedy with respect to some Q-function, if it behaves:

randomly with probability ε

greedy with probability $1 - \varepsilon$

ε-greedy exploration

Do something random *sometimes*

Let's call policy ε-greedy with respect to some Q-function, if it behaves:

randomly with probability ε

greedy with probability $1 - \varepsilon$

$$\pi(a \mid s) = \begin{cases} 1 - \varepsilon + \frac{\varepsilon}{|\mathcal{A}|} & \text{when } a = \operatorname*{argmax}_a Q(s, a) \\ \frac{\varepsilon}{|\mathcal{A}|} & \text{otherwise} \end{cases}$$

ε-greedy exploration

Do something random *sometimes*

Let's call policy ε-greedy with respect to some Q-function, if it behaves:

randomly with probability ε

greedy with probability $1 - \varepsilon$

$$\pi(a \mid s) = \begin{cases} 1 - \varepsilon + \frac{\varepsilon}{|\mathcal{A}|} & \text{when } a = \operatorname*{argmax}_a Q(s, a) \\ \frac{\varepsilon}{|\mathcal{A}|} & \text{otherwise} \end{cases}$$

Decaying exploration:

$$\varepsilon_k \to 0, \quad \varepsilon_k > 0$$

Persistent exploration:

$$\varepsilon_k = \text{const}(k)$$

Q-learning

Q-learning (online)

Initialize $Q^*(s, a)$ arbitrarily;

observe s_0;

for $k = 0, 1, 2 \ldots$

- take action $a_k \sim \varepsilon$-greedy$(Q^*(s_k, a))$;
- observe r_k, s_{k+1};

Q-learning

Q-learning (online)

Initialize $Q^*(s, a)$ arbitrarily;

observe s_0;

for $k = 0, 1, 2 \ldots$

- take action $a_k \sim \varepsilon$-greedy$(Q^*(s_k, a))$;
- observe r_k, s_{k+1};
- $y := r_k + \gamma \max_{a_{k+1}} Q^*(s_{k+1}, a_{k+1})$;

Q-learning

Q-learning (online)

Initialize $Q^*(s, a)$ arbitrarily;

observe s_0;

for $k = 0, 1, 2 \ldots$

- take action $a_k \sim \varepsilon$-greedy$(Q^*(s_k, a))$;
- observe r_k, s_{k+1};
- $y := r_k + \gamma \max_{a_{k+1}} Q^*(s_{k+1}, a_{k+1})$;
- $Q^*(s_k, a_k) \leftarrow (1 - \alpha_k) Q^*(s_k, a_k) + \alpha_k y$

Q-learning

Q-learning (online)

Initialize $Q^*(s, a)$ arbitrarily;

observe s_0;

for $k = 0, 1, 2 \ldots$

- take action $a_k \sim \varepsilon$-greedy$(Q^*(s_k, a))$;
- observe r_k, s_{k+1};
- $y := r_k + \gamma \max_{a_{k+1}} Q^*(s_{k+1}, a_{k+1})$;
- $Q^*(s_k, a_k) \leftarrow (1 - \alpha_k) Q^*(s_k, a_k) + \alpha_k y$

Q-learning (with replay buffer)

Initialize $Q^*(s, a)$ arbitrarily, $\mathcal{D} = \varnothing$;

Q-learning

Q-learning (online)

Initialize $Q^*(s, a)$ arbitrarily;

observe s_0;

for $k = 0, 1, 2 \ldots$

- take action $a_k \sim \varepsilon$-greedy$(Q^*(s_k, a))$;
- observe r_k, s_{k+1};
- $y := r_k + \gamma \max_{a_{k+1}} Q^*(s_{k+1}, a_{k+1})$;
- $Q^*(s_k, a_k) \leftarrow (1 - \alpha_k) Q^*(s_k, a_k) + \alpha_k y$

Q-learning (with replay buffer)

Initialize $Q^*(s, a)$ arbitrarily, $\mathcal{D} = \varnothing$;

observe s_0;

for $k = 0, 1, 2 \ldots$

- take action $a_k \sim \varepsilon$-greedy$(Q^*(s_k, a))$;
- observe r_k, s_{k+1};
- store (s_k, a_k, r_k, s_{k+1}) in $\mathcal{D}$;

Q-learning

Q-learning (online)

Initialize $Q^*(s, a)$ arbitrarily;

observe s_0;

for $k = 0, 1, 2 \ldots$

- take action $a_k \sim \varepsilon$-greedy$(Q^*(s_k, a))$;
- observe r_k, s_{k+1};
- $y := r_k + \gamma \max_{a_{k+1}} Q^*(s_{k+1}, a_{k+1})$;
- $Q^*(s_k, a_k) \leftarrow (1 - \alpha_k) Q^*(s_k, a_k) + \alpha_k y$

Q-learning (with replay buffer)

Initialize $Q^*(s, a)$ arbitrarily, $\mathcal{D} = \varnothing$;

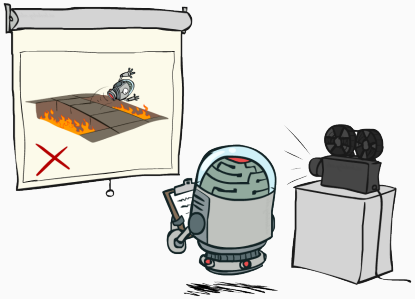
observe s_0;

for $k = 0, 1, 2 \ldots$

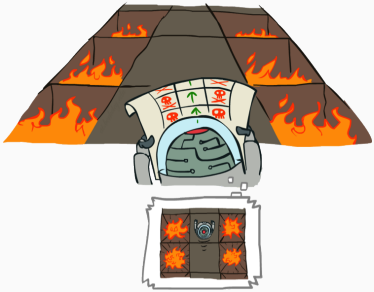
- take action $a_k \sim \varepsilon$-greedy$(Q^*(s_k, a))$;
- observe r_k, s_{k+1};
- store (s_k, a_k, r_k, s_{k+1}) in $\mathcal{D}$;
- sample (s, a, r, s') from $\mathcal{D}$;
- $y := r + \gamma \max_{a'} Q^*(s', a')$;
- $Q^*(s, a) \leftarrow (1 - \alpha_k) Q^*(s, a) + \alpha_k y$

Q-learning is off-policy

Data source	Interaction policy	Convergence
Online experience	Infinite visitation	Q^*
Expert data	-	Q^*
Experience replay	Infinite visitation	Q^*

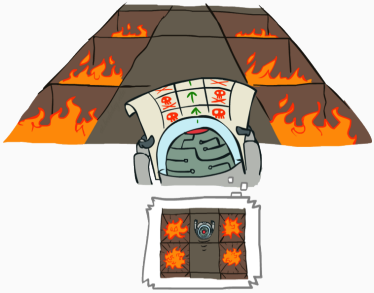


Expectation

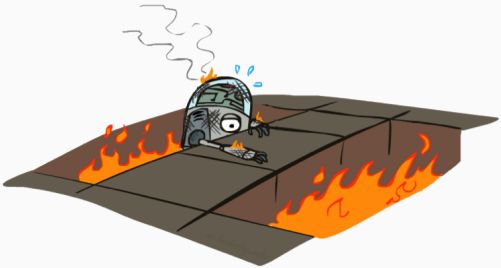


Safe RL

Expectation

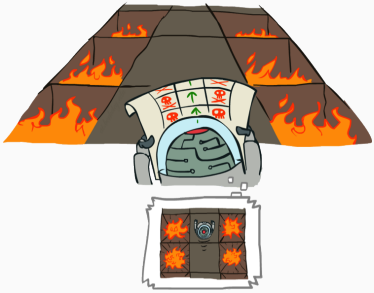


Reality

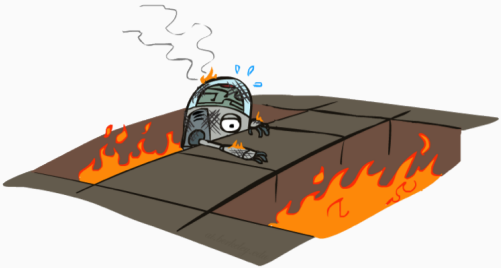


Safe RL

Expectation



Reality



Consider **the family of «ε-soft policies»**:

$$\forall s, a\colon \pi(a \mid s) \geqslant \frac{\varepsilon}{|\mathcal{A}|}$$

Q-learning vs SARSA

ε-soft Policy Improvement

In the family of «ε-soft policies» ε-greedy is policy improvement:

$$\tilde{\pi} := \varepsilon\text{-greedy}(Q^{\pi}(s,a)) \quad \Rightarrow \quad \tilde{\pi} \geq \pi$$

Q-learning vs SARSA

ε-soft Policy Improvement

In the family of «ε-soft policies» ε-greedy is policy improvement:

$$\tilde{\pi} := \varepsilon\text{-greedy}(Q^{\pi}(s,a)) \quad \Rightarrow \quad \tilde{\pi} \geq \pi$$

$$Q_{k+1}(s,a) \leftarrow Q_k(s,a) + \alpha_k \left(r(s,a) + \gamma Q_k(s', a') - Q_k(s,a)\right)$$

Q-learning

a' taken from **target** policy

$$\pi_k(s) := \operatorname*{argmax}_a Q_k(s,a)$$

SARSA

a' taken from **behavior** policy

$$\pi_k := \varepsilon\text{-greedy}(Q_k(s,a))$$

Q-learning vs SARSA

ε-soft Policy Improvement

In the family of «ε-soft policies» ε-greedy is policy improvement:

$$\tilde{\pi} := \varepsilon\text{-greedy}(Q^{\pi}(s,a)) \quad \Rightarrow \quad \tilde{\pi} \geq \pi$$

$$Q_{k+1}(s,a) \leftarrow Q_k(s,a) + \alpha_k \left(r(s,a) + \gamma Q_k(s', a') - Q_k(s,a)\right)$$

Q-learning

a' taken from **target** policy

$$\pi_k(s) := \operatorname*{argmax}_a Q_k(s,a)$$

SARSA

a' taken from **behavior** policy

$$\pi_k := \varepsilon\text{-greedy}(Q_k(s,a))$$

SARSA convergence properties

SARSA converges to optimal ε-soft policy, satisfying

$$\pi \equiv \varepsilon\text{-greedy}(Q^{\pi}(s,a))$$

SARSA

SARSA

Initialize $Q^\pi(s, a)$ arbitrarily.

observe s_0, select a_0 randomly;
for $k = 0, 1, 2 \ldots$

- take action a_k, observe r_k, s_{k+1};
- sample $a_{k+1} \sim \varepsilon$-greedy$(Q^\pi(s_{k+1}, a))$
- $y := r_k + \gamma Q^\pi(s_{k+1}, a_{k+1})$;
- $Q^\pi(s_k, a_k) \leftarrow (1 - \alpha_k) Q^\pi(s_k, a_k) + \alpha_k y$

SARSA

SARSA

Initialize $Q^\pi(s, a)$ arbitrarily.

observe s_0, select a_0 randomly;
for $k = 0, 1, 2 \ldots$

- take action a_k, observe r_k, s_{k+1};
- sample $a_{k+1} \sim \varepsilon$-greedy$(Q^\pi(s_{k+1}, a))$
- $y := r_k + \gamma Q^\pi(s_{k+1}, a_{k+1})$;
- $Q^\pi(s_k, a_k) \leftarrow (1 - \alpha_k) Q^\pi(s_k, a_k) + \alpha_k y$

Expected-SARSA

Initialize $Q^\pi(s, a)$ arbitrarily.

observe s_0;
for $k = 0, 1, 2 \ldots$

- sample $a_k \sim \varepsilon$-greedy$(Q^\pi(s_k, a))$;
- take action a_k, observe r_k, s_{k+1};
- $y := r_k + \gamma \mathbb{E}_{a_{k+1}} Q^\pi(s_{k+1}, a_{k+1})$;
- $Q^\pi(s_k, a_k) \leftarrow (1 - \alpha_k) Q^\pi(s_k, a_k) + \alpha_k y$;

SARSA is on-policy

For transition (s, a, r, s', a'):

$$Q^{\pi}_{k+1}(s, a) \leftarrow Q^{\pi}_{k}(s, a) + \alpha_k \left(r + \gamma Q^{\pi}_{k}(s', a') - Q^{\pi}_{k}(s, a)\right)$$

Data source	**Interaction policy**	**Convergence**
Online experience	Decaying exploration	Q^*
	Persistent exploration	$Q^*_{\varepsilon\text{-greedy}}$
Expert data (a' taken from buffer)	-	

SARSA is on-policy

For transition (s, a, r, s', a'):

$$Q^{\pi}_{k+1}(s, a) \leftarrow Q^{\pi}_{k}(s, a) + \alpha_k \left(r + \gamma Q^{\pi}_{k}(s', a') - Q^{\pi}_{k}(s, a)\right)$$

Data source	**Interaction policy**	**Convergence**
Online experience	Decaying exploration	Q^*
	Persistent exploration	$Q^*_{\varepsilon\text{-greedy}}$
Expert data (a' taken from buffer)	-	$Q^{\pi_{\text{expert}}}$
Experience replay (a' taken from buffer)		

SARSA is on-policy

For transition (s, a, r, s', a'):

$$Q^{\pi}_{k+1}(s, a) \leftarrow Q^{\pi}_{k}(s, a) + \alpha_k \left(r + \gamma Q^{\pi}_{k}(s', a') - Q^{\pi}_{k}(s, a)\right)$$

Data source	**Interaction policy**	**Convergence**
Online experience	Decaying exploration	Q^*
	Persistent exploration	$Q^*_{\varepsilon\text{-greedy}}$
Expert data (a' taken from buffer)	-	$Q^{\pi_{\text{expert}}}$
Experience replay (a' taken from buffer)	(arbitrary)	divergence

«Fixing» SARSA (not really sarsa now)

For transition (s, a, r, s') generate $a' \sim \pi(a' \mid s')$:

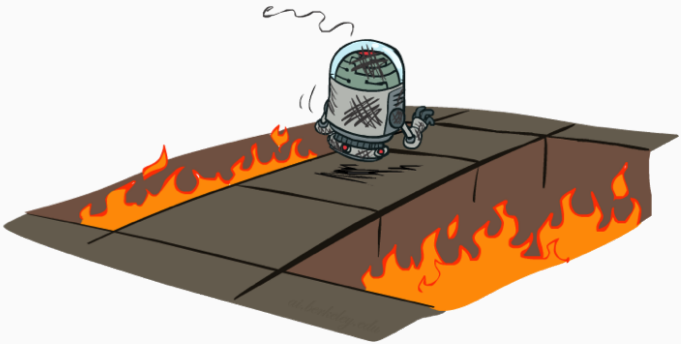
$$Q^{\pi}_{k+1}(s, a) \leftarrow Q^{\pi}_{k}(s, a) + \alpha_k \left(r + \gamma Q^{\pi}_{k}(s', a') - Q^{\pi}_{k}(s, a)\right)$$

«Fixing» SARSA (not really sarsa now)

For transition (s, a, r, s') generate $a' \sim \pi(a' \mid s')$:

$$Q^{\pi}_{k+1}(s, a) \leftarrow Q^{\pi}_{k}(s, a) + \alpha_k \left(r + \gamma Q^{\pi}_{k}(s', a') - Q^{\pi}_{k}(s, a)\right)$$

Data source	Interaction policy	Convergence
(any, but a' generated online)	Decaying exploration	Q^*
	Persistent exploration	$Q^*_{\varepsilon\text{-greedy}}$



Literature

- Sutton, Barto — Reinforcement Learning, an Introduction, ch. 5-6;
- Watkins, Dayan — Technical Note, Q-learning;

Pictures from Berkeley CS 188 | Introduction to Artificial Intelligence;