

Temporal Difference Learning

Reinforcement Learning

September 21, 2021

MSU

Reminder

Reminder: Value functions

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

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

Reminder: Value functions

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

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

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

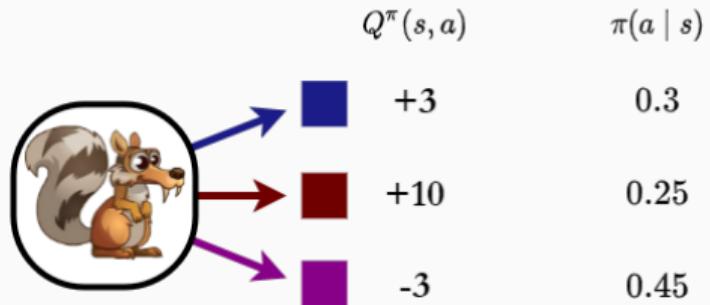
Optimal Q-function: $Q^*(s, a) := \max_\pi Q^\pi(s, a)$

Reminder: Policy Improvement

Policy Improvement:

greedy policy with respect to Q^π is not worse than π :

$$\hat{\pi}(s) = \operatorname{argmax}_a Q^\pi(s, a) \implies \hat{\pi} \geq \pi$$

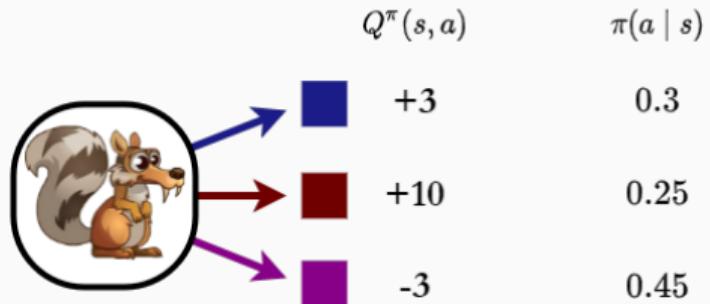


Reminder: Policy Improvement

Policy Improvement:

greedy policy with respect to Q^π is not worse than π :

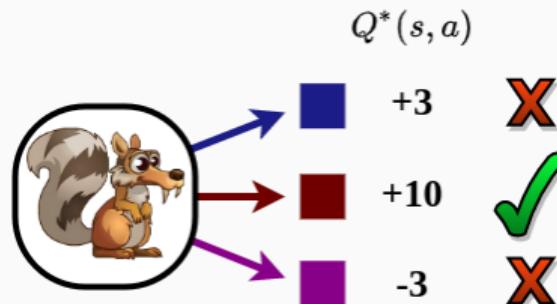
$$\hat{\pi}(s) = \operatorname{argmax}_a Q^\pi(s, a) \implies \hat{\pi} \geq \pi$$



Bellman optimality principle:

greedy policy with respect to optimal Q-function is optimal:

$$\pi^*(s) = \operatorname{argmax}_a Q^*(s, a)$$



Reminder: Bellman Equations

Bellman (expectation) equations: $\forall \pi, s, a:$

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

$$V^\pi(s) = \mathbb{E}_{a \sim \pi(a|s)} Q^\pi(s, a)$$

Reminder: Bellman Equations

Bellman (expectation) equations: $\forall \pi, s, a:$

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

$$V^\pi(s) = \mathbb{E}_{a \sim \pi(a|s)} Q^\pi(s, a)$$

Bellman (optimality) equations: $\forall s, a:$

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

$$V^*(s) = \max_a Q^*(s, a)$$

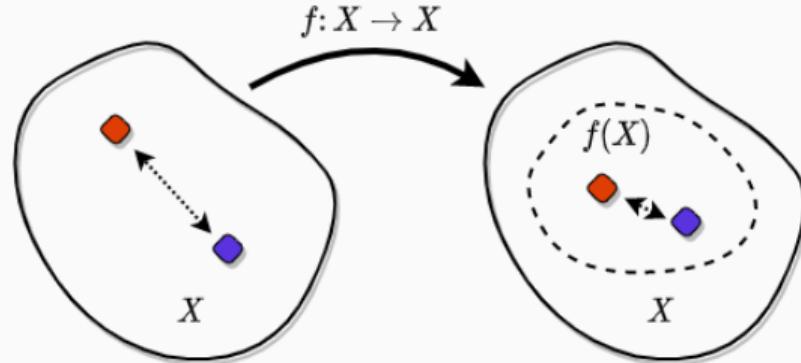
Reminder: Point iteration

Contraction Mapping

$f(x): X \rightarrow X$ is a **contraction mapping** if:

$$\forall x_1, x_2: \rho(f(x_1), f(x_2)) \leq \alpha \rho(x_1, x_2),$$

where ρ — some metric, $\alpha < 1$.



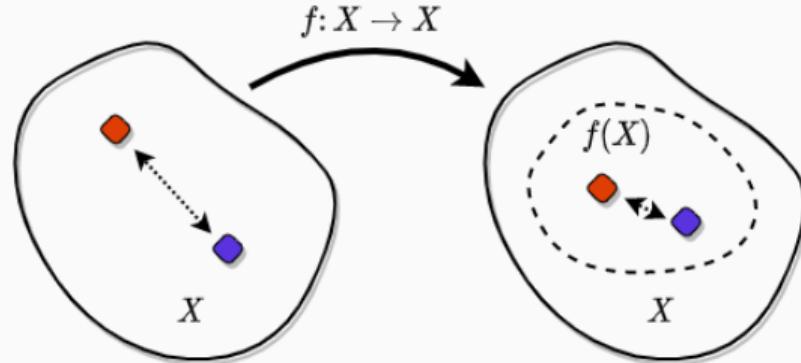
Reminder: Point iteration

Contraction Mapping

$f(x): X \rightarrow X$ is a **contraction mapping** if:

$$\forall x_1, x_2: \rho(f(x_1), f(x_2)) \leq \alpha \rho(x_1, x_2),$$

where ρ — some metric, $\alpha < 1$.



If f is a contraction mapping:

- equation $x = f(x)$ has **unique** solution;
- we can find it using **point iteration** method

$$x_{k+1} = f(x_k),$$

starting from any x_0 .

Reminder: Policy Iteration

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

Assumption: model is known

Reminder: Policy Iteration

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: Policy Iteration

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

Assumption: model is known

Policy Iteration

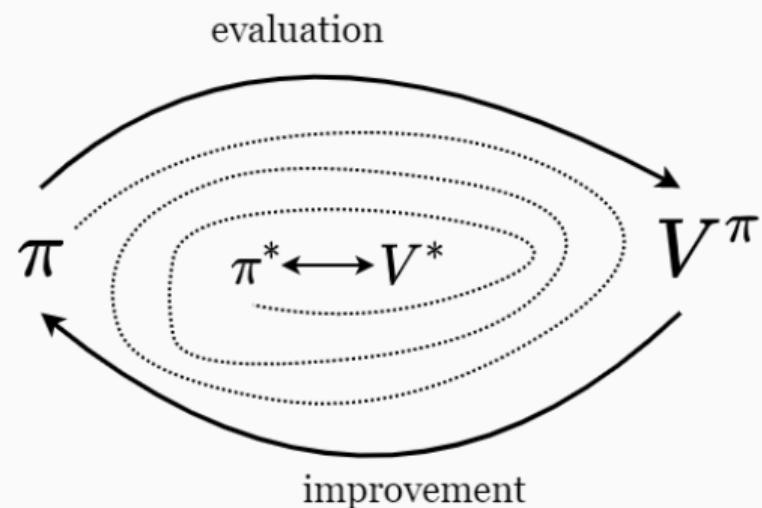
- **Policy Evaluation:**

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: Generalized Policy Iteration

Generalized Policy Iteration

Initialize V, π arbitrary, repeat:

Reminder: Generalized Policy Iteration

Generalized Policy Iteration

Initialize V, π arbitrary, repeat:

- **Policy Evaluation:** move V towards V^π (*at least a little*)
e.g. perform K iterations of point iteration method for Bellman equation

Reminder: Generalized Policy Iteration

Generalized Policy Iteration

Initialize V, π arbitrary, repeat:

- **Policy Evaluation:** move V towards V^π (*at least a little*)
e.g. perform K iterations of point iteration method for Bellman equation
- **Policy Improvement:** improve π using V (*at least a little*)
e.g. greedy policy improvement

Reminder: Generalized Policy Iteration

Generalized Policy Iteration

Initialize V, π arbitrary, repeat:

- **Policy Evaluation:** move V towards V^π (*at least a little*)
e.g. perform K iterations of point iteration method for Bellman equation
- **Policy Improvement:** improve π using V (*at least a little*)
e.g. greedy policy improvement



This always works!!!

Reminder: Generalized Policy Iteration

Generalized Policy Iteration

Initialize V, π arbitrary, repeat:

- **Policy Evaluation:** move V towards V^π (*at least a little*)
e.g. perform K iterations of point iteration method for Bellman equation
- **Policy Improvement:** improve π using V (*at least a little*)
e.g. greedy policy improvement



This always works!!!

Interesting case: $K = 1$ and greedy policy improvement

Reminder: Value Iteration

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

Assumption: model is known

Value Iteration

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

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

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

Reminder: Value Iteration

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

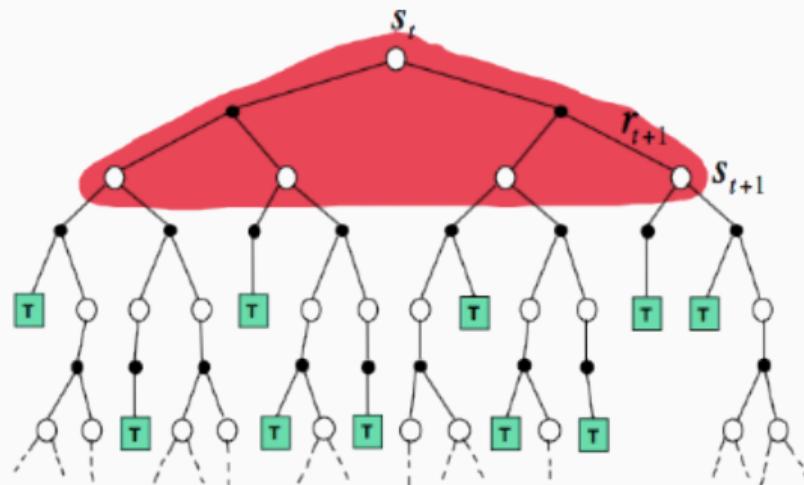
Assumption: model is known

Value Iteration

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

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

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



Model-free learning

Trial and Error learning

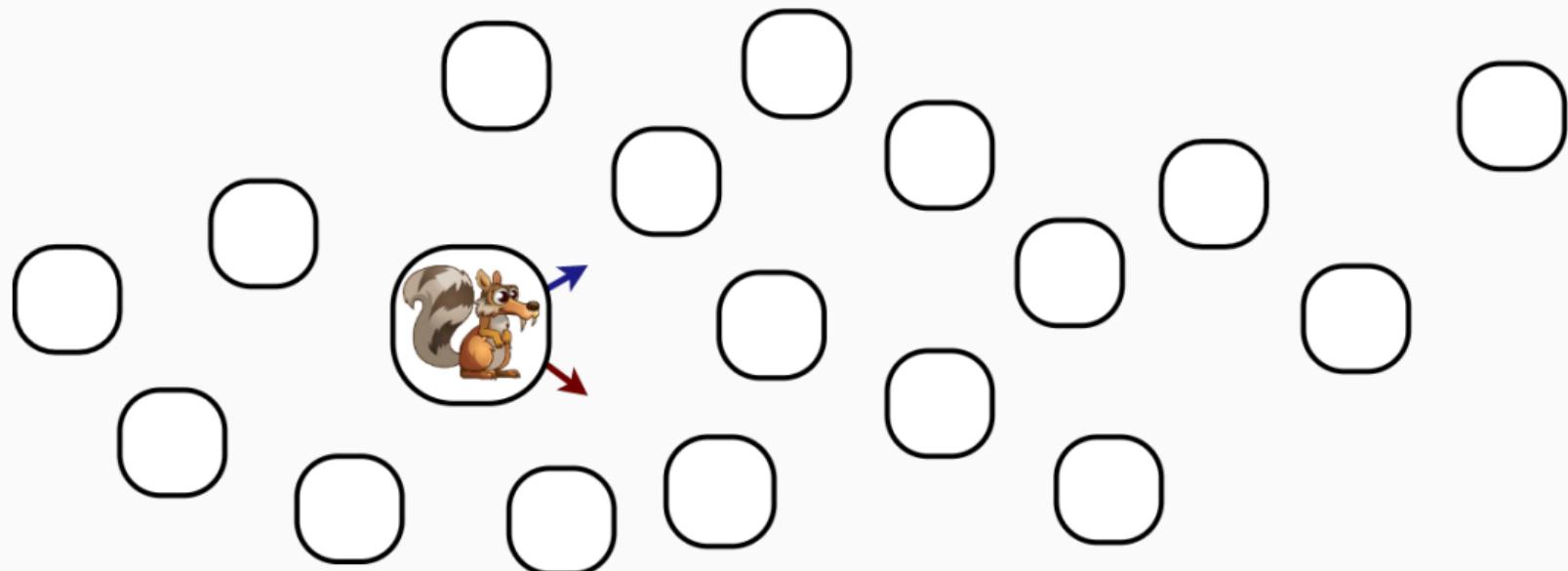
Setup: $|\mathcal{S}| \ll \infty, |\mathcal{A}| \ll \infty$

Assumption: ~~model is known~~

Trial and Error learning

Setup: $|S| \ll \infty, |A| \ll \infty$

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**

- ✗ $\mathcal{S} \times \mathcal{A} \rightarrow P(\mathcal{S})$
- ✓ (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**

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

b) **brain**

- ✓ $\mathcal{S} \rightarrow \mathcal{A}$
- × 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:

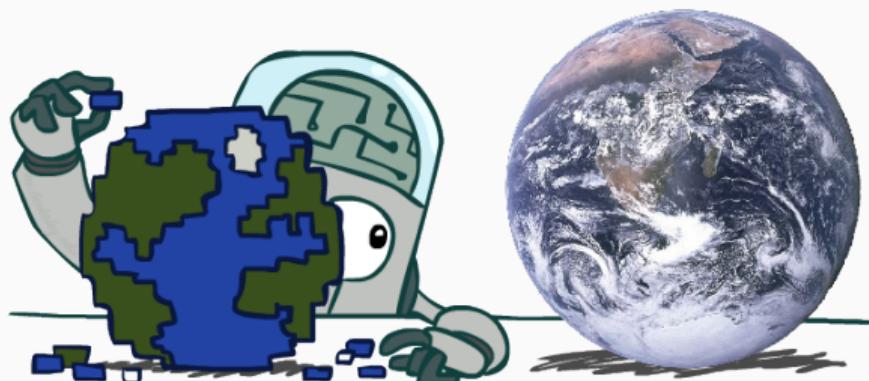
a) **universe**

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

b) **brain**

- ✓ $\mathcal{S} \rightarrow \mathcal{A}$
- × 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:

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

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

Value Iteration

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

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

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

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

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

Value Iteration

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

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

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

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

Ok, not clear how to work with this...

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

Monte Carlo Backup

$$Q^\pi(s, a) := \mathbb{E}_{\mathcal{T} \sim \pi | s_0 = s, a_0 = a} R(\mathcal{T})$$

How to solve this: $x = \mathbb{E}_{p(\varepsilon)} f(\varepsilon)$?

Monte Carlo Backup

$$Q^\pi(s, a) := \mathbb{E}_{\mathcal{T} \sim \pi | s_0 = s, a_0 = a} R(\mathcal{T})$$

How to solve this: $x = \mathbb{E}_{p(\varepsilon)} f(\varepsilon)$?



Play a lot of games using π to collect data for Monte Carlo estimations

Monte Carlo Backup

$$Q^\pi(s, a) := \mathbb{E}_{\mathcal{T} \sim \pi | s_0 = s, a_0 = a} R(\mathcal{T})$$

How to solve this: $x = \mathbb{E}_{p(\varepsilon)} f(\varepsilon)$?

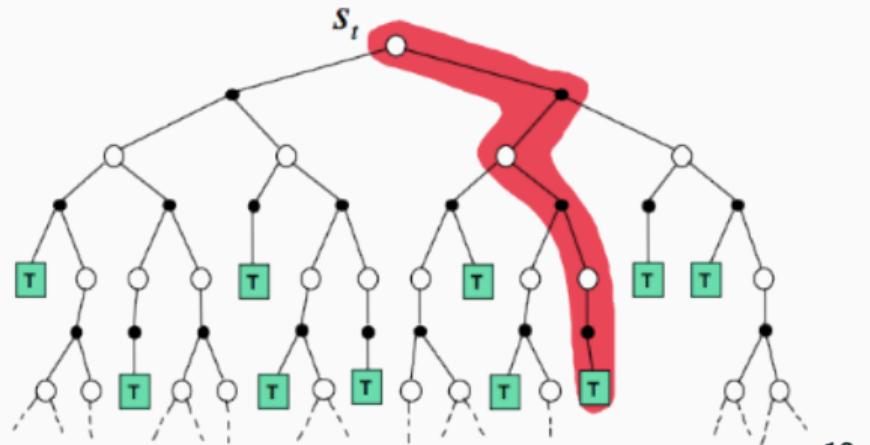


Play a lot of games using π to collect data for Monte Carlo estimations

Monte Carlo estimation:

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

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



Monte Carlo Algorithm

Monte Carlo Algorithm

Initialize policy $\pi(s)$ arbitrarily.

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

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

! requires infinite samples for each pair s, a ;

Monte Carlo Algorithm

Monte Carlo Algorithm

Initialize policy $\pi(s)$ arbitrarily.

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

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

- ! requires infinite samples for each pair s, a ;
- ✗ still needs full episodes;
- ✗ still does not utilize MDP structure;
- ✗ wastes information about state connections;
- ✗ high variance of collected samples;

Monte Carlo Algorithm

Monte Carlo Algorithm

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

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

- 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 does not utilize MDP structure;
- ✗ wastes information about state connections;
- ✗ high variance of collected samples;



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, \dots . 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, \dots . On step k :

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

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 :

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

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 :

$$\begin{aligned} m_k &:= \frac{1}{k} \sum_{i=1}^k x_i = \frac{k-1}{k} m_{k-1} + \frac{1}{k} x_k = \left\{ \alpha_k := \frac{1}{k} \right\} = \\ &= \underbrace{\left(1 - \alpha_k\right) m_{k-1} + \alpha_k x_k}_{\text{exponential smoothing}} = \end{aligned}$$

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 :

$$\begin{aligned} m_k &:= \frac{1}{k} \sum_{i=1}^k x_i = \frac{k-1}{k} m_{k-1} + \frac{1}{k} x_k = \left\{ \alpha_k := \frac{1}{k} \right\} = \\ &= \underbrace{(1 - \alpha_k) m_{k-1} + \alpha_k x_k}_{\text{exponential smoothing}} = \underbrace{m_{k-1} + \alpha_k (x_k - m_{k-1})}_{\approx \text{stoch. gradient descent}} \end{aligned}$$

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 :

$$\begin{aligned} m_k &:= \frac{1}{k} \sum_{i=1}^k x_i = \frac{k-1}{k} m_{k-1} + \frac{1}{k} x_k = \left\{ \alpha_k := \frac{1}{k} \right\} = \\ &= \underbrace{(1 - \alpha_k) m_{k-1} + \alpha_k x_k}_{\text{exponential smoothing}} = \underbrace{m_{k-1} + \alpha_k (x_k - m_{k-1})}_{\approx \text{stoch. gradient descent}} \end{aligned}$$

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 :

$$\begin{aligned} m_k &:= \frac{1}{k} \sum_{i=1}^k x_i = \frac{k-1}{k} m_{k-1} + \frac{1}{k} x_k = \left\{ \alpha_k := \frac{1}{k} \right\} = \\ &= \underbrace{(1 - \alpha_k) m_{k-1} + \alpha_k x_k}_{\text{exponential smoothing}} = \underbrace{m_{k-1} + \alpha_k (x_k - m_{k-1})}_{\approx \text{stoch. gradient descent}} \end{aligned}$$

Convergence:

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

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

Temporal Difference Idea

Solving equations

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

Temporal Difference: intuition 1

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

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

Temporal Difference: intuition 1

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

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

Temporal Difference: intuition 1

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

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

To update value for s, a we need only **transition** (s, a, r, s', a') :

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

$$Q^\pi(s, a) \leftarrow (1 - \alpha_k) Q^\pi(s, a) + \alpha_k y$$

Requirements for Temporal Difference

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

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

Requirements for Temporal Difference

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

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

s, a — considered **fixed**.

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

Requirements for Temporal Difference

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

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

s, a — considered **fixed**.

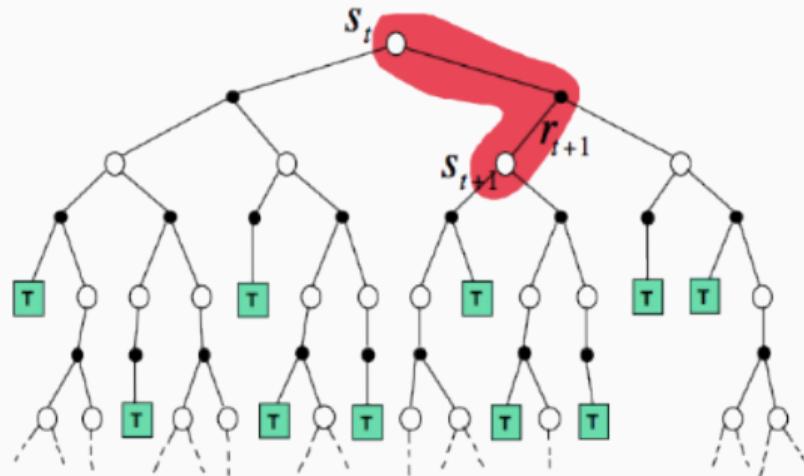
$s' \sim p(s' | s, a)$ — **random variable** from interaction experience.

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

Temporal Difference: intuition 2

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

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

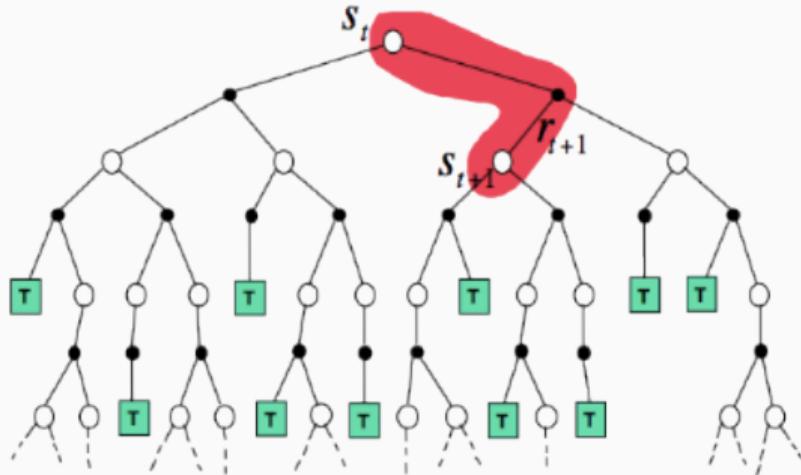


Temporal Difference: intuition 2

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

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

our own predictions used as targets!



Temporal Difference vs Monte Carlo

$$Q_{k+1}^{\pi}(s, a) \leftarrow Q_k^{\pi}(s, a) + \alpha_k (y(s, a) - Q_k^{\pi}(s, a))$$

Which is better?

Temporal Difference

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

Monte Carlo

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

Temporal Difference vs Monte Carlo

$$Q_{k+1}^{\pi}(s, a) \leftarrow Q_k^{\pi}(s, a) + \alpha_k (y(s, a) - Q_k^{\pi}(s, a))$$

Which is better?

Temporal Difference

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

- ✓ 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

$$Q_{k+1}^{\pi}(s, a) \leftarrow Q_k^{\pi}(s, a) + \alpha_k (y(s, a) - Q_k^{\pi}(s, a))$$

Which is better?

Temporal Difference

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

- ✓ 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

$$Q_{k+1}^{\pi}(s, a) \leftarrow Q_k^{\pi}(s, a) + \alpha_k (y(s, a) - Q_k^{\pi}(s, a))$$

Which is better?

Temporal Difference

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

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

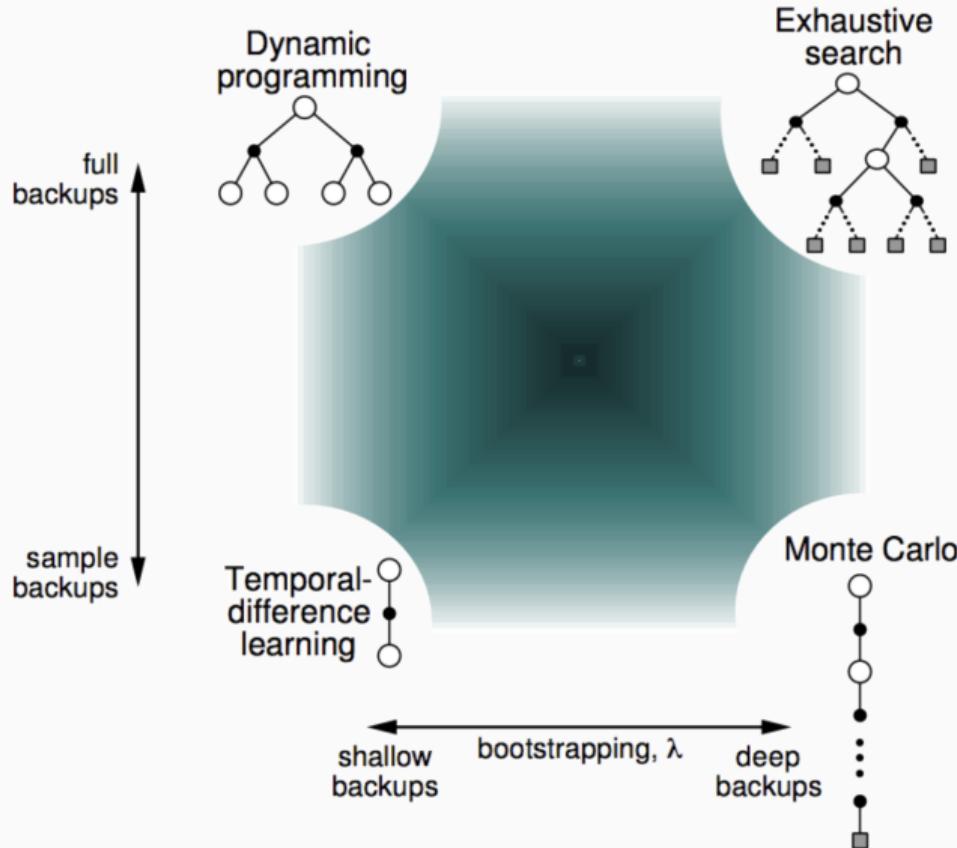
Monte Carlo

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

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

Best estimation is somewhere in between (see $\text{TD}(\lambda)$).

Types of backups



Q-learning

Value Iteration and Policy Iteration connection

What if we improve our policy after every step?



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

Value Iteration and Policy Iteration connection

What if we improve our policy after every step?



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

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

Value Iteration and Policy Iteration connection

What if we improve our policy after every step?



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

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

Value Iteration and Policy Iteration connection

What if we improve our policy after every step?



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

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

Value Iteration and Policy Iteration connection

What if we improve our policy after every step?



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

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

Value Iteration and Policy Iteration connection

What if we improve our policy after every step?



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

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

Value Iteration and Policy Iteration connection

What if we improve our policy after every step?



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

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

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

Convergence of temporal difference

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 temporal difference

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:

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

Convergence of temporal difference

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:

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

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:

- $s'_{k,s,a} \sim p(s' | s, a)$;
- 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 $Q_k^*(s, a) \xrightarrow[k \rightarrow \infty]{\text{w. p. 1}} Q^*(s, a)$

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

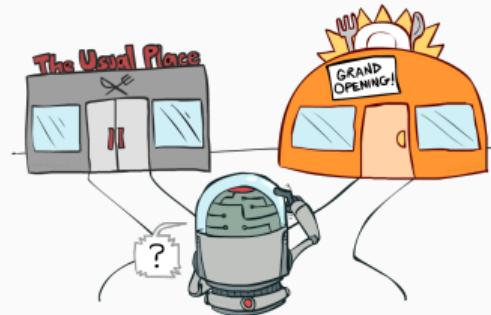


Analogy 2

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 $1 - \varepsilon$

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

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 $1 - \varepsilon$

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

Decaying exploration:

$$\varepsilon_k \rightarrow 0, \quad \varepsilon_k > 0$$

Persistent exploration:

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

Q-learning

Q-learning (online)

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

observe s_0 ;

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

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

Q-learning

Q-learning (online)

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

observe s_0 ;

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

- take action $a_k \sim \varepsilon\text{-greedy}(Q^*(s_k, a))$;
- 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 $k = 0, 1, 2 \dots$

- take action $a_k \sim \varepsilon\text{-greedy}(Q^*(s_k, a))$;
- 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 $k = 0, 1, 2 \dots$

- take action $a_k \sim \varepsilon\text{-greedy}(Q^*(s_k, a))$;
- 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, $\mathcal{D} = \emptyset$;

Q-learning

Q-learning (online)

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

observe s_0 ;

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

- take action $a_k \sim \varepsilon\text{-greedy}(Q^*(s_k, a))$;
- 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, $\mathcal{D} = \emptyset$;

observe s_0 ;

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

- take action $a_k \sim \varepsilon\text{-greedy}(Q^*(s_k, a))$;
- 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 $k = 0, 1, 2 \dots$

- take action $a_k \sim \varepsilon\text{-greedy}(Q^*(s_k, a))$;
- 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, $\mathcal{D} = \emptyset$;

observe s_0 ;

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

- take action $a_k \sim \varepsilon\text{-greedy}(Q^*(s_k, a))$;
- 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} ;
- $y := r + \gamma \max_{a'} Q^*(s', a')$;
- $Q^*(s, a) \leftarrow (1 - \alpha_k)Q^*(s, a) + \alpha_k y$

Behavior and target policies

The policy μ interacting with environment (collecting data) is called **behavior policy**;

Behavior and target policies

The policy μ interacting with environment (collecting data) is called **behavior policy**;

The trained policy π that will be outputted by algorithm is called **target policy**;

Behavior and target policies

The policy μ interacting with environment (collecting data) is called **behavior policy**;

The trained policy π that will be outputted by algorithm is called **target policy**;

If we learn Q^π using samples from $\mu \equiv \pi$, we are **on-policy**. If $\mu \neq \pi$, we are **off-policy**.

Behavior and target policies

The policy μ interacting with environment (collecting data) is called **behavior policy**;

The trained policy π that will be outputted by algorithm is called **target policy**;

If we learn Q^π using samples from $\mu \equiv \pi$, we are **on-policy**. If $\mu \neq \pi$, we are **off-policy**.

Q-learning

off-policy

(arbitrary μ can be used)

Monte Carlo

on-policy

($\mu \equiv \pi$ is required)

Vanilla version: $\mu := \varepsilon$ -greedy($Q^*(s, a)$)

Experience replay is useless!

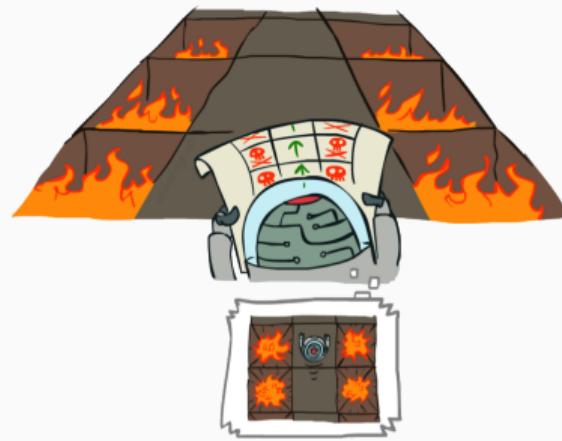
Q-learning is off-policy

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



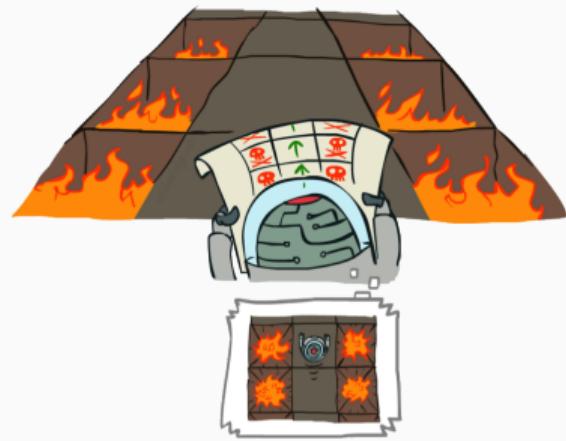
SARSA

Expectation

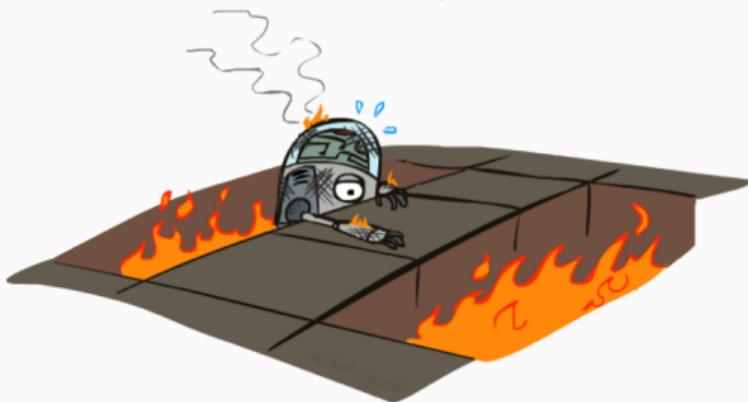


Safe RL

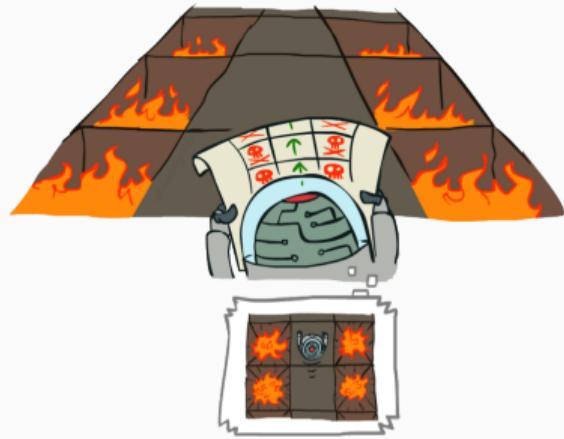
Expectation



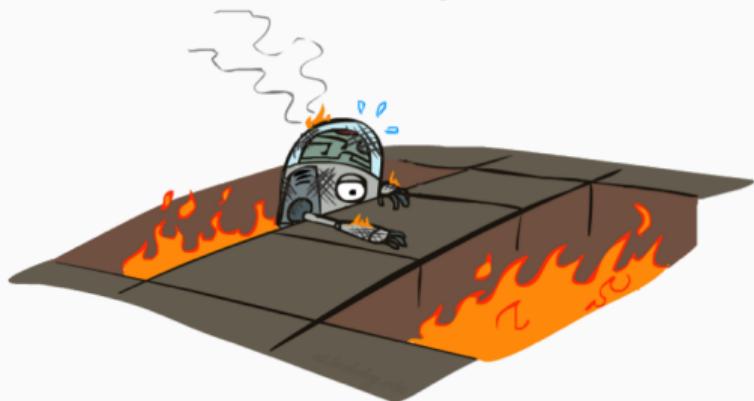
Reality



Expectation



Reality



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

$$\forall s, a: \pi(a | s) \geq \frac{\varepsilon}{|\mathcal{A}|}$$



Family of « ε -soft policies»: what changes

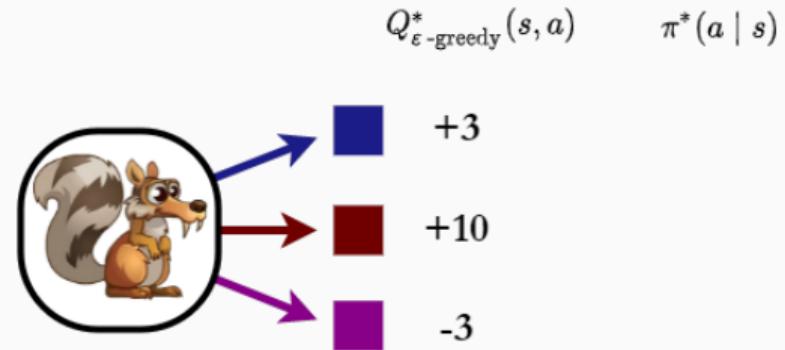
Policy Evaluation: nothing changes, Q^π is the same.

Family of « ε -soft policies»: what changes

Policy Evaluation: nothing changes, Q^π is the same.

Optimal value function:

$$Q_{\varepsilon\text{-greedy}}^*(s, a) := \max_{\pi \in \varepsilon\text{-soft}} Q^\pi(s, a)$$



Family of « ε -soft policies»: what changes

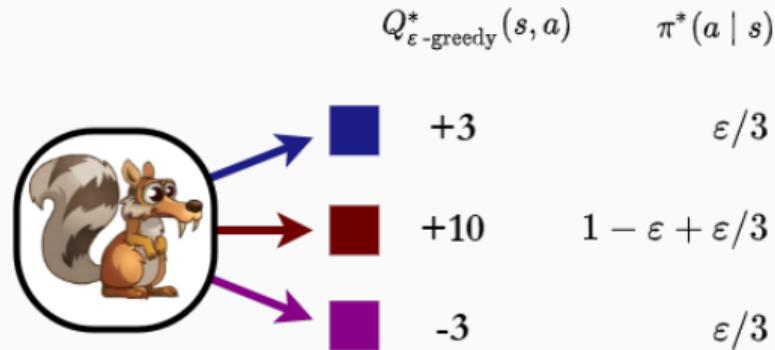
Policy Evaluation: nothing changes, Q^π is the same.

Optimal value function:

$$Q_{\varepsilon\text{-greedy}}^*(s, a) := \max_{\pi \in \varepsilon\text{-soft}} Q^\pi(s, a)$$

Optimal policy:

$$\pi^*(a | s) = \varepsilon\text{-greedy}(Q_{\varepsilon\text{-greedy}}^*(s, a))$$



Family of « ε -soft policies»: what changes

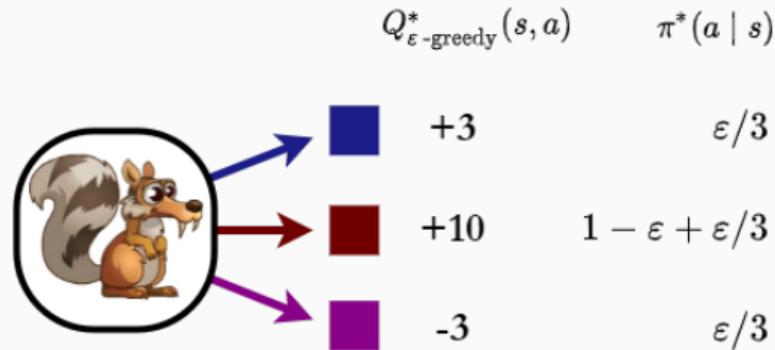
Policy Evaluation: nothing changes, Q^π is the same.

Optimal value function:

$$Q_{\varepsilon\text{-greedy}}^*(s, a) := \max_{\pi \in \varepsilon\text{-soft}} Q^\pi(s, a)$$

Optimal policy:

$$\pi^*(a | s) = \varepsilon\text{-greedy}(Q_{\varepsilon\text{-greedy}}^*(s, a))$$



ε -soft Policy Improvement

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

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

Q-learning vs SARSA

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

a' must be taken from **target** policy

Q-learning vs SARSA

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

a' must be taken from **target** policy

	Q-learning	SARSA
Target policy:	$\underset{a}{\operatorname{argmax}} Q_k(s, a)$	$\varepsilon\text{-greedy}(Q_k(s, a))$
Behavior policy:	$\varepsilon\text{-greedy}(Q_k(s, a))$	$\varepsilon\text{-greedy}(Q_k(s, a))$

Q-learning vs SARSA

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

a' must be taken from **target** policy

	Q-learning	SARSA
Target policy:	$\underset{a}{\operatorname{argmax}} Q_k(s, a)$	$\varepsilon\text{-greedy}(Q_k(s, a))$
Behavior policy:	$\varepsilon\text{-greedy}(Q_k(s, a))$	$\varepsilon\text{-greedy}(Q_k(s, a))$

Q-learning vs SARSA

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

a' must be taken from **target** policy

	Q-learning	SARSA
Target policy:	$\underset{a}{\operatorname{argmax}} Q_k(s, a)$	$\varepsilon\text{-greedy}(Q_k(s, a))$
Behavior policy:	$\varepsilon\text{-greedy}(Q_k(s, a))$	$\varepsilon\text{-greedy}(Q_k(s, a))$
Convergence:	$\pi(s) = \underset{a}{\operatorname{argmax}} Q^\pi(s, a)$	$\pi \equiv \varepsilon\text{-greedy}(Q^\pi(s, a))$

SARSA

SARSA

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

observe s_0 , select a_0 randomly;

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

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

- **on-policy** by construction

SARSA

SARSA

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

observe s_0 , select a_0 randomly;

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

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

- **on-policy** by construction
- same as Q-learning if $\varepsilon = 0$
- converges to Q^* if $\varepsilon_k \rightarrow 0$

SARSA

SARSA

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

observe s_0 , select a_0 randomly;

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

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

- **on-policy** by construction
- same as Q-learning if $\varepsilon = 0$
- converges to Q^* if $\varepsilon_k \rightarrow 0$

Expected-SARSA:

$$y := r_k + \gamma \mathbb{E}_{a_{k+1}} Q^\pi(s_{k+1}, a_{k+1})$$

SARSA is on-policy

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

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

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

SARSA is on-policy

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

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

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

SARSA is on-policy

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

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

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

«Off-policy» SARSA (not really sarsa now)

Can we learn Q^π for *given* π off-policy?

«Off-policy» SARSA (not really sarsa now)

Can we learn Q^π for *given* π off-policy?

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

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

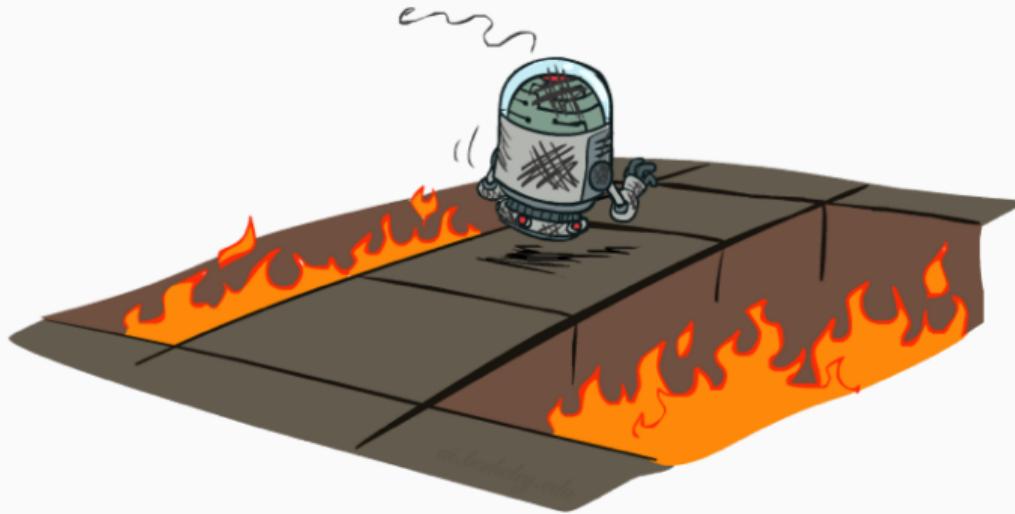
«Off-policy» SARSA (not really sarsa now)

Can we learn Q^π for given π off-policy?

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

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

Data source	Target 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;