Temporal Difference Learning

Reinforcement Learning

September 29, 2020

MSU

Reminder: Value functions

State value function (V-function): $V(s) : \mathbb{E}_T \left[\sum_{t \geq 0} {}^t r_t \mid s_0 = s \right]$

State-action value function (Q-function): $Q(s, a) : \mathbb{E}_T \left[\sum_{t \geq 0} {}^t r_t \mid s_0 = s, a_0 = a \right]$

Reminder: Value functions

State value function (V-function): $V(s) : \mathbb{E}_T \left[\sum_{t \geq 0} {}^t r_t \mid s_0 = s \right]$

State-action value function (Q-function): $Q(s, a) : \mathbb{E}_T \left[\sum_{t \geq 0} {}^t r_t \mid s_0 = s, a_0 = a \right]$

Optimal V-function: $V(s) : \max V(s)$

Optimal Q-function: $Q(s, a) : \max Q(s, a)$

Reminder: Value functions

State value function (V-function): $V^{\pi}(s) = \mathbb{E}_{\pi}\left[\sum_{t \geq 0} \gamma^t r_t \mid s_0 = s\right]$

State-action value function (Q-function): $Q^{\pi}(s, a) = \mathbb{E}_{\pi}\left[\sum_{t \geq 0} \gamma^t r_t \mid s_0 = s, a_0 = a\right]$

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

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

Bellman equations: $\forall s, a:$

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

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

Reminder: Dynamic Programming

Setup: $|S| ! 8; |A| ! 8$

Assumption: model is known

Reminder: Dynamic Programming

Setup: $|S| \; ! \; 8; |A| \; ! \; 8$
Assumption: model is known

Policy Iteration

Policy Evaluation:

compute V^{k} for current policy $_{k}$;

Policy Improvement:

$_{k \;\; 1}psq$ Đ $\underset{a}{\text{argmax}}\, Q \; ^{k}ps; aq$;

repeat;

Reminder: Dynamic Programming

Setup: $|S| < \infty$; $|A| < \infty$
Assumption: model is known

Policy Iteration

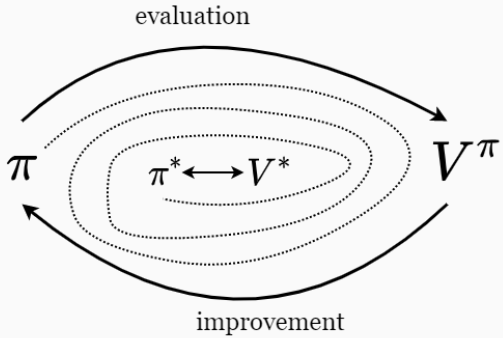
Policy Evaluation:

compute V^{π_k} for current policy π_k;

Policy Improvement:

$\pi_{k+1}(s) \doteq \arg\max_a Q^{\pi_k}(s,a)$;

repeat;



Reminder: Dynamic Programming

Setup: $|S| < \infty; |A| < \infty$
Assumption: model is known

Value Iteration

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

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

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

Reminder: Dynamic Programming

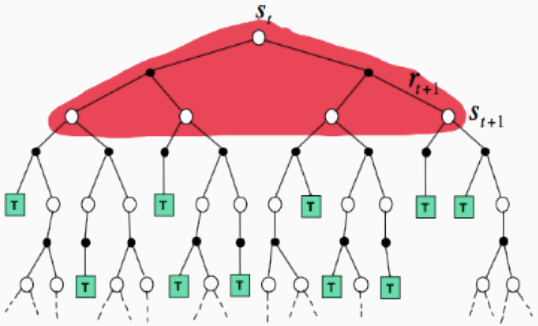
Setup: $|S| < \infty; |A| < \infty$
Assumption: model is known

Value Iteration

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

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

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



Trial and Error learning

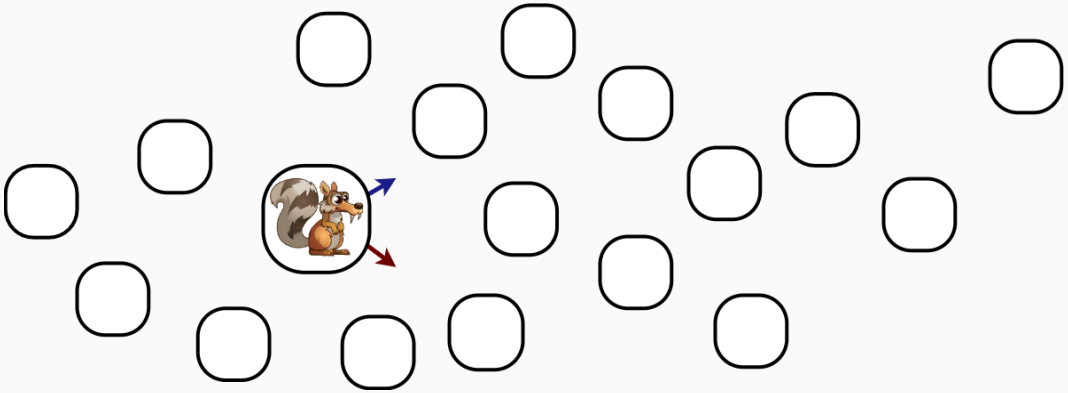
Setup: $|S| \; ! \; 8 ; |A| \; ! \; 8$

~~Assumption: model is known~~

Trial and Error learning

Setup: $|S| \; ! \; 8; |A| \; ! \; 8$

~~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

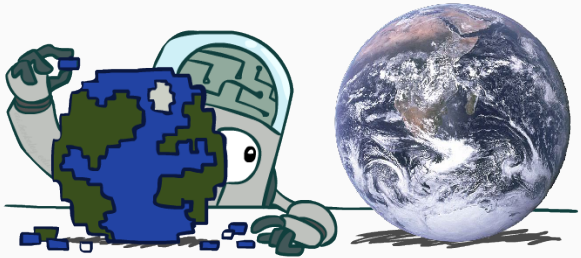
Learn model through experience

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

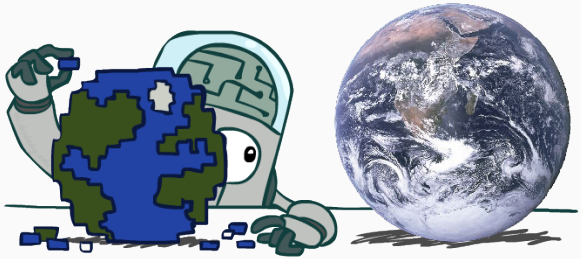
What is easier to create:

a) universe

$S \quad A \tilde{N} P_{\mathfrak{p}} S_{\mathfrak{q}}$

X (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

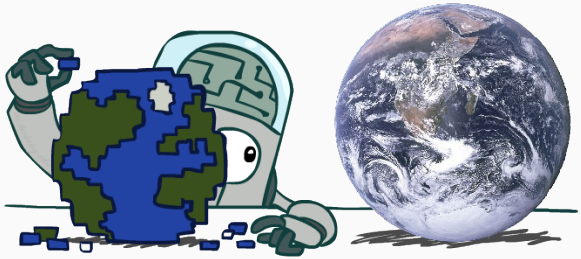
$S \quad A$ Ñ $P(S'|S,A)$

X (almost) supervised learning

b) brain

X S Ñ 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

$S \quad A$ Ñ $P(S'|S,A)$

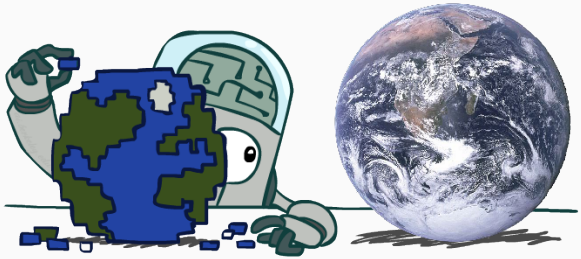
X (almost) supervised learning

b) brain

X S Ñ 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:

$$V^{k}(s) \leftarrow \mathbb{E}_{a}\, r(s,a) \quad \mathbb{E}_{s'} V^{k}(s')$$

$$k+1(s) \leftarrow \operatorname{argmax}_{a} Q^{k}(s,a);$$

repeat;

Value Iteration

÷ solve Bellman optimality equation
(e. g. for V):

$$V_{k+1}(s) \leftarrow \max_{a} r(s,a) \quad \mathbb{E}_{s'} V_{k}(s')$$

$$(s) \leftarrow \operatorname{argmax}_{a} Q(s,a) \text{ is optimal;}$$

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:

$$V^{k}(s) \leftarrow \mathbb{E}_{a}\, r(s,a) + \mathbb{E}_{s'} V^{k}(s')$$

$$k+1(s) \leftarrow \arg\max_{a} Q^{k}(s,a);$$

repeat;

Value Iteration

÷ solve Bellman optimality equation

(e. g. for V):

$$V_{k+1}(s) \leftarrow \max_{a} r(s,a) + \mathbb{E}_{s'} V_{k}(s')$$

$$(s) \leftarrow \arg\max_{a} Q(s,a) \text{ is optimal;}$$

Switching to Q-function

Policy Iteration

evaluate policy:

$$Q^{k}(s,a) \quad r(s,a) \quad \mathbb{E}_{s'}\mathbb{E}_{a'}Q^{k}(s',a')$$

$$k+1(s) \leftarrow \operatorname*{argmax}_{a} Q^{k}(s,a);$$

repeat;

Value Iteration

÷ solve Bellman optimality equation:

$$Q_{k+1}(s,a) \leftarrow r(s,a) \quad \mathbb{E}_{s'} \max_{a'} Q_k(s',a')$$

$$(s) \leftarrow \operatorname*{argmax}_{a} Q(s,a) \text{ is optimal;}$$

Switching to Q-function

Policy Iteration

evaluate policy:

$$Q^{k}(s,a) \leftarrow r(s,a) + \mathbb{E}_{s'}\mathbb{E}_{a'}\left[Q^{k}(s',a')\right]$$

$$\pi^{k+1}(s) \leftarrow \arg\max_{a} Q^{k}(s,a);$$

repeat;

Value Iteration

÷ solve Bellman optimality equation:

$$Q_{k+1}(s,a) \leftarrow r(s,a) + \mathbb{E}_{s'}\left[\max_{a'} Q_{k}(s',a')\right]$$

$\pi(s) \leftarrow \arg\max_{a} Q(s,a)$ is optimal;

$$x \leftarrow \mathbb{E}_{s'}\left[f(s',x)\right]; \quad x - ?$$

Switching to Q-function

Policy Iteration

evaluate policy:

$$Q^{k}(s,a) \leftarrow r(s,a) + \mathbb{E}_{s'}\mathbb{E}_{a'}Q^{k}(s',a')$$

$$k+1(s) \leftarrow \operatorname{argmax}_{a} Q^{k}(s,a);$$

repeat;

Value Iteration

÷ solve Bellman optimality equation:

$$Q_{k+1}(s,a) \leftarrow r(s,a) + \mathbb{E}_{s'}\max_{a'} Q_{k}(s',a')$$

$$(s) \leftarrow \operatorname{argmax}_{a} Q(s,a) \text{ is optimal;}$$

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

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

Monte-Carlo Backup

How to solve this: x $\mathbb{E}_{s^1} f(s^1)$

Monte-Carlo Backup

How to solve this: $x \quad \mathbb{E}_{s^1} f(s^1)$

Monte-Carlo estimation:

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

where $T_i \sim (s_0, a_0, \ldots)$

Monte-Carlo Backup

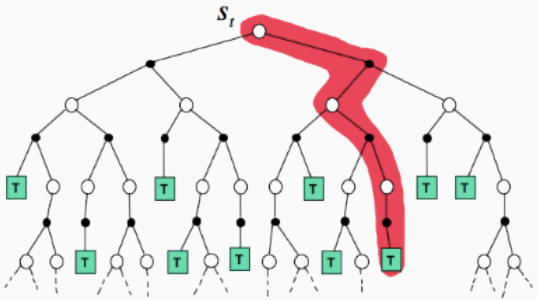
How to solve this: $x \quad E_{s^1} f(s^1)$

Monte-Carlo estimation:

$$Q(s,a) \quad \frac{1}{N} \sum_{i=0}^{N} R(T_i)$$

where $T_i \quad | s_0 \quad s; a_0 \quad a$

Can we have i. i. d. samples?



Monte Carlo Algorithm

Monte Carlo Algorithm

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

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

play several games using π;

estimate $Q(s,a)$ using Monte Carlo;

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

! requires infinite samples for each pair s, a;

Monte Carlo Algorithm

Monte Carlo Algorithm

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

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

play several games using π;

estimate $Q(s,a)$ using Monte Carlo;

$\pi(s) \leftarrow \arg\max_a Q(s,a)$

- ! 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 $Q(s,a)$ and policy $\pi(s)$ arbitrarily.

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

play several games using π;

update $Q(s,a)$ with new samples;

$\pi(s) \leftarrow \arg\max_a Q(s,a)$

- ! 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 $x_1; x_2 :::$. On step k:

$$m_k : \quad \frac{1}{k} \overset{k}{\underset{i \quad 1}{\text{ }}} x_i$$

Closer look at Monte Carlo

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

$$m_k : \quad \frac{1}{k} \sum_{i \quad 1}^{k} x_i \qquad \frac{k \quad 1}{k} m_{k \quad 1} \qquad \frac{1}{k} x_k$$

Closer look at Monte Carlo

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

$$m_k : \quad \frac{1}{k} \sum_{i \quad 1}^{k} x_i \qquad \frac{k \quad 1}{k} m_{k \quad 1} \quad \frac{1}{k} x_k \qquad 1 \quad \frac{1}{k} \quad m_{k \quad 1} \quad \frac{1}{k} x_k$$

Closer look at Monte Carlo

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

$$m_k : \frac{1}{k} \overset{k}{,} x_i \qquad \frac{k \quad 1}{k} m_{k \quad 1} \quad \frac{1}{k} x_k \qquad 1 \quad \frac{1}{k} \quad m_{k \quad 1} \quad \frac{1}{k} x_k$$

$$! \qquad \qquad)$$

$$_k : \frac{1}{k} \qquad \underbrace{1 \quad \frac{1}{k} m_{k \quad 1}}_{\text{exponential smoothing}} \quad \frac{1}{k} x_k$$

Closer look at Monte Carlo

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

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

$$! \qquad \left(k : \frac{1}{k} \right)$$

$$p1 \quad q\, m_{k-1} \qquad x_k \qquad m_{k-1} \quad p x_k \qquad m_{k-1} \; q$$

exponential smoothing

stoch. gradient descent

Closer look at Monte Carlo

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

$$m_k : \frac{1}{k} \overset{k}{,} x_i \qquad \frac{k \quad 1}{k} m_{k \quad 1} \quad \frac{1}{k} x_k \qquad 1 \quad \frac{1}{k} \quad m_{k \quad 1} \quad \frac{1}{k} x_k$$

$$! \qquad \qquad)$$

$$_k : \frac{1}{k} \qquad \text{p1} \quad \text{q} m_{k \quad 1} \quad x_k \qquad m_{k \quad 1} \quad \text{p} x_k \quad m_{k \quad 1} \text{q}$$

exponential smoothing

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:

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

$$m_k = (1-\alpha) m_{k-1} + \alpha x_k = m_{k-1} - \alpha (m_{k-1} - x_k)$$

exponential smoothing — stoch. gradient descent

Convergence:

$m_k \to \mathbb{E}x$ if learning rate satisfies Robbins–Monro conditions:

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

Solving equations

Equation	Method	Update
$x \quad f(x)$	Point Iteration	$x_{k+1} : \quad f(x_k)$
$x \quad E_{s^{-1}} f(s^{-1})$	Exponential smoothing	$x_{k+1} : \quad x_k \quad {}_k(f(s^{-1}) \quad x_k) \quad s^{-1} \quad p(s^{-1})$
$x \quad E_{s^{-1}} f(s^{-1}; x)$	Stochastic approximation	

Solving equations

Equation	Method	Update
$x \quad f(x)$	Point Iteration	$x_{k+1} : \quad f(x_k)$
$x \quad E_{s^1} f(s^1)$	Exponential smoothing	$x_{k+1} : \quad x_k \quad \frac{\alpha_k}{s^1} \frac{f(s^1)}{p(s^1)} \quad x_k$
$x \quad E_{s^1} f(s^1; x)$	Stochastic approximation	$x_{k+1} : \quad x_k \quad \frac{\alpha_k}{s^1} \frac{f(s^1; x_k)}{p(s^1)} \quad x_k$

Temporal Difference: intuition

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

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

Temporal Difference: intuition

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

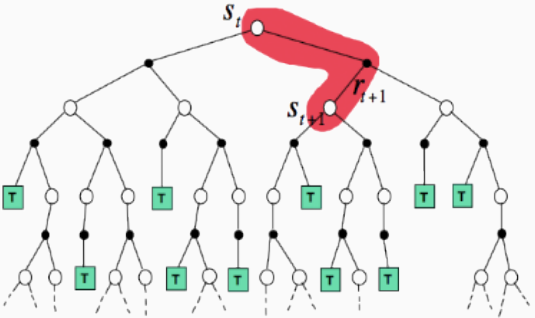
$$Q(s,a) \leftarrow \mathbb{E}_{s'}\underbrace{[r(s,a) + \gamma \mathbb{E}_{a'} Q(s',a')]}_{f(s',x)} \quad \mathbb{E}_{s'}\mathbb{E}_{a'}\underbrace{[r(s,a) + \gamma Q(s',a')]}_{f(s',a',x)}$$

Temporal Di□erence: intuition

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

$$Q(s,a) \leftarrow Q(s,a) + \underbrace{\alpha\left(r(s,a) + \gamma E_{s^1} \max_{a^1} Q(s^1,a^1) - Q(s,a)\right)}_{f(s^1;x)}$$

$$Q(s,a) \leftarrow Q(s,a) + \underbrace{\alpha\left(r(s,a) + \gamma E_{s^1} E_{a^1} \max_{a^1} Q(s^1,a^1) - Q(s,a)\right)}_{f(s^1,a^1;x)}$$



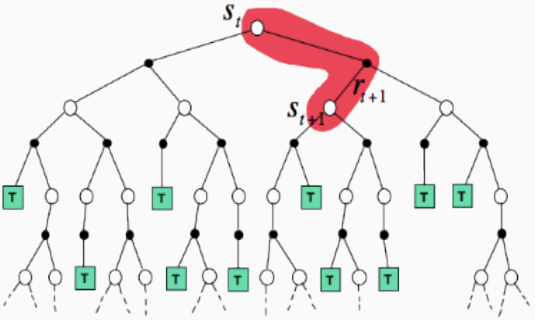
Temporal Di□erence: intuition

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

$$Q(s,a) \leftarrow \underbrace{(1-\alpha)Q(s,a)}_{} + \underbrace{\alpha\, E_{s^1}\left[r(s,a) + \gamma E_{a^1} Q(s^1,a^1)\right]}_{f(s^1;x)} \qquad \underbrace{\alpha\, E_{s^1} E_{a^1}\left[r(s,a) + \gamma Q(s^1,a^1)\right]}_{f(s^1,a^1;x)}$$

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

$$r + \underbrace{\gamma r^1 + \gamma^2 r^2 + \gamma^3 r^3 + \ldots}_{Q_k(s^1,a^1)}$$



Temporal Difference: intuition

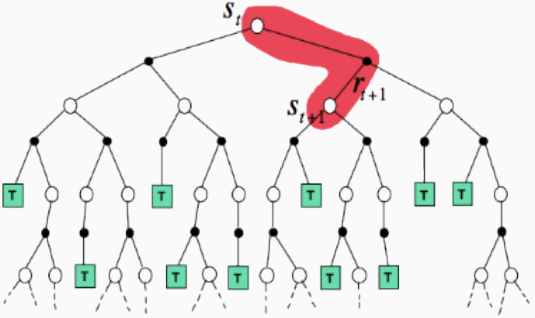
View 1: solving Bellman expectation equation using Robbins-Monro scheme:

$$Q(s,a) \leftarrow \underbrace{r(s,a) + \gamma E_{s'} \max_{a'} Q(s',a')}_{f(s',x)}$$

$$Q(s,a) \leftarrow \underbrace{r(s,a) + \gamma E_{s'} E_{a'} Q(s',a')}_{f(s',a',x)}$$

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

$$r \leftarrow \underbrace{r^1 + r^2 + \gamma r^3 + \dots}_{Q_k(s',a')}$$



our own predictions used as targets!

Temporal Di□erence

For transition ps; a; r; s^1; a^{1}q:

$$Q_{k\ \ 1}(s,a) \leftarrow Q_k(s,a) + \alpha_k \underbrace{\overbrace{\left[r + Q_k(s^1,a^1)\right]}^{\text{Bellman target}} - Q_k(s,a)}_{\text{temporal di□erence}}$$

Temporal Di□erence

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

$$Q_{k+1}(s,a) \leftarrow Q_k(s,a) + \alpha_k \underbrace{\Big[\overbrace{r + \gamma Q_k(s',a')}^{\text{Bellman target}} - Q_k(s,a)\Big]}_{\text{temporal di□erence}}$$

(s, a) — considered fixed.

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

Temporal Di□erence

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

$$Q_{k+1}(s,a) \leftarrow Q_k(s,a) + \alpha_k \underbrace{\overbrace{\left[r + Q_k(s',a')\right]}^{\text{Bellman target}} - Q_k(s,a)}_{\text{temporal di□erence}}$$

(s, a) — considered fixed.

(s', a') — random variable from interaction experience.

a' — random variable from algorithm.

Temporal Di□erence vs Monte Carlo

$$Q_{k\ \ 1}ps;aq \text{ Đ } Q_k ps;aq \qquad {}_k pyps;aq \quad Q_k ps;aqq$$

Which is better?

Temporal Di□erence

$yps;aq : \ r \quad Q_k ps^1;a^1q$

Monte Carlo

$yps;aq : \ r \quad r^1 \quad {}^2r^2 \quad :::$

Temporal Di□erence vs Monte Carlo

$$Q_{k\ \ 1}ps;aq \text{ Đ } Q_k ps;aq \qquad {}_k pyps;aq \quad Q_k ps;aqq$$

Which is better?

Temporal Di□erence

$yps;aq : \quad r \qquad Q_k ps^1;a^1q$

X updates after each step

slow reward propagation

Monte Carlo

$yps;aq : \quad r \qquad r^1 \qquad {}^2r^2 \qquad :::$

updates at the end of the episode

X fast reward propagation

Temporal Di□erence vs Monte Carlo

$$Q_{k\ \ 1}ps;aq \text{ Đ } Q_k ps;aq \qquad {}_k pyps;aq \quad Q_k ps;aqq$$

Which is better?

Temporal Di□erence

$yps;aq : \quad r \qquad Q_k ps^1;a^1q$

X updates after each step

slow reward propagation

X low variance

introduces bias

Monte Carlo

$yps;aq : \quad r \qquad r^1 \qquad {}^2r^2 \qquad :::$

updates at the end of the episode

X fast reward propagation

high variance

X unbiased

Temporal Di□erence vs Monte Carlo

$$Q_{k\ \ 1}ps;aq \text{ Đ } Q_k ps;aq \qquad {}_k p\mathit{y}ps;aq \quad Q_k ps;aqq$$

Which is better?

Temporal Di□erence

$yps;aq : \quad r \qquad Q_k ps^1;a^1q$

X updates after each step

slow reward propagation

X low variance

introduces bias

Monte Carlo

$yps;aq : \quad r \qquad r^1 \qquad {}^2r^2 \qquad :::$

updates at the end of the episode

X fast reward propagation

high variance

X unbiased

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

Value Iteration and Policy Iteration connection

What if we improve our policy after every step?

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

Value Iteration and Policy Iteration connection

What if we improve our policy after every step?

$$\pi_k(s) = \arg\max_a Q_k(s,a)$$

$$Q_{k+1}(s,a) = Q_k(s,a) + \alpha \left[r + \gamma Q_k(s',a') - Q_k(s,a) \right]$$

Value Iteration and Policy Iteration connection

What if we improve our policy after every step?

$$\pi_k(s) = \arg\max_a Q_k(s,a)$$

$$Q_{k+1}(s,a) = Q_k(s,a) + \alpha\left[r + Q_k(s',a') - Q_k(s,a)\right]$$

$$Q_k(s,a) + \alpha\left[r + Q_k(s',\pi_k(s')) - Q_k(s,a)\right]$$

Value Iteration and Policy Iteration connection

What if we improve our policy after every step?

$$\pi_k(s) \leftarrow \arg\max_a Q_k(s,a)$$

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

$$Q_k(s,a) + \alpha \left[r + Q_k(s', \pi_k(s')) - Q_k(s,a) \right]$$

$$Q_k(s,a) + \alpha \left[r + Q_k(s', \arg\max_{a'} Q_k(s',a')) - Q_k(s,a) \right]$$

Value Iteration and Policy Iteration connection

What if we improve our policy after every step?

$$\pi_k(s) \leftarrow \arg\max_a Q_k(s,a)$$

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

$$Q_k(s,a) + \alpha \left[r + Q_k(s',\pi_k(s')) - Q_k(s,a) \right]$$

$$Q_k(s,a) + \alpha \left[r + Q_k(s', \arg\max_{a'} Q_k(s',a')) - Q_k(s,a) \right]$$

$$Q_k(s,a) + \alpha \left[r + \max_{a'} Q_k(s',a') - Q_k(s,a) \right]$$

Value Iteration and Policy Iteration connection

What if we improve our policy after every step?

$$\pi_k(s) \leftarrow \arg\max_a Q_k(s,a)$$

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

$$Q_k(s,a) + \alpha\left[r + Q_k(s',\pi_k(s')) - Q_k(s,a)\right]$$

$$Q_k(s,a) + \alpha\left[r + Q_k\left(s', \arg\max_{a'} Q_k(s',a')\right) - Q_k(s,a)\right]$$

$$Q_k(s,a) + \alpha\left[r + \max_{a'} Q_k(s',a') - Q_k(s,a)\right]$$

Value Iteration and Policy Iteration connection

What if we improve our policy after every step?

$$\pi_k(s) \leftarrow \arg\max_a Q_k(s,a)$$

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

$$Q_k(s,a) + \alpha \left[r + Q_k(s',\pi_k(s')) - Q_k(s,a) \right]$$

$$Q_k(s,a) + \alpha \left[r + Q_k(s', \arg\max_{a'} Q_k(s',a')) - Q_k(s,a) \right]$$

$$Q_k(s,a) + \alpha \left[r + \max_{a'} Q_k(s',a') - Q_k(s,a) \right]$$

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 di□erence

Convergence of Q-learning updates

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

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

Convergence of temporal di□erence

Convergence of Q-learning updates

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

$$Q_{k+1}(s, a) \leftarrow Q_k(s, a) + \alpha_k(s, a)\left[r(s, a) + \max_{a'} Q_k(s'_{k;s;a}, a') - Q_k(s, a)\right]$$

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:

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

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:

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

Convergence of temporal di□erence

Convergence of Q-learning updates

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

$$Q_{k+1}(s,a) \leftarrow Q_k(s,a) + \alpha_k(s,a)\left[r(s,a) + \max_{a'} Q_k(s'_{k;s,a},a') - Q_k(s,a)\right]$$

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:

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

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

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

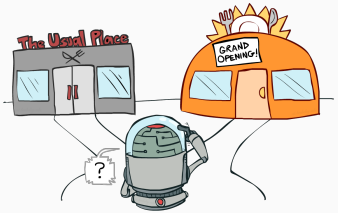
Exploration:
random search

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

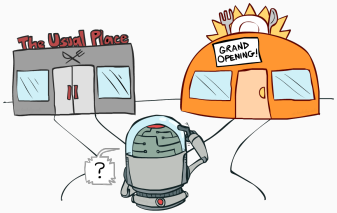
Exploration:
random search

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

″-greedy exploration

Do something random *sometimes*

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

randomly with probability ″

greedy with probability 1 ″

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 $1 - \varepsilon$

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

″-greedy exploration

Do something random *sometimes*

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

randomly with probability ″

greedy with probability 1 ″

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

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 ε-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 ε-greedy$(Q(s_k, a))$;

observe r_k, s_{k+1};

$y = r_k + \max_{a_{k+1}} Q(s_{k+1}, a_{k+1})$;

Q-learning

Q-learning (online)

Initialize $Q(s, a)$ arbitrarily;

observe s_0;

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

take action a_k ε-greedy$(Q(s_k, a))$;

observe r_k, s_{k+1};

$y = r_k + \max_{a_{k+1}} Q(s_{k+1}, a_{k+1})$;

$Q(s_k, a_k) \leftarrow (1 - \alpha_k) Q(s_k, a_k) + \alpha_k y$

Q-learning

Q-learning (online)

Initialize $Q\ ps; aq$ arbitrarily;

observe s_0;

for $k \quad 0; 1; 2 : : :$

take action $a_k \quad "$-greedy$pQ\ ps_k; aqq$;

observe $r_k; s_{k\ 1}$;

$y : \quad r_k \quad \max_{a_{k\ 1}} Q\ ps_{k\ 1}; a_{k\ 1}q$;

$Q\ ps_k; a_kq \text{ Đ } p1 \quad {}_kqQ\ ps_k; a_kq \quad {}_ky$

Q-learning (with replay buffer)

Initialize $Q\ ps; aq$ arbitrarily, $D \quad H$;

Q-learning

Q-learning (online)

Initialize $Q(s, a)$ arbitrarily;

observe s_0;
for $k = 0, 1, 2, \ldots$
  take action a_k ε-greedy$(Q(s_k, a))$;
  observe r_k, s_{k+1};
  $y = r_k + \max_{a_{k+1}} Q(s_{k+1}, a_{k+1})$;
  $Q(s_k, a_k) \leftarrow (1 - \alpha_k) Q(s_k, a_k) + \alpha_k y$

Q-learning (with replay buffer)

Initialize $Q(s, a)$ arbitrarily, $D \leftarrow \emptyset$;

observe s_0;
for $k = 0, 1, 2, \ldots$
  take action a_k ε-greedy$(Q(s_k, a))$;
  observe r_k, s_{k+1};
  store (s_k, a_k, r_k, s_{k+1}) in D;

Q-learning

Q-learning (online)

Initialize $Q(s, a)$ arbitrarily;

observe s_0;
for $k = 0, 1, 2, \ldots$
  take action a_k ε-greedy$(Q(s_k, a))$;
  observe r_k, s_{k+1};
  $y \leftarrow r_k + \gamma \max_{a_{k+1}} Q(s_{k+1}, a_{k+1})$;
  $Q(s_k, a_k) \leftarrow (1 - \alpha) Q(s_k, a_k) + \alpha y$

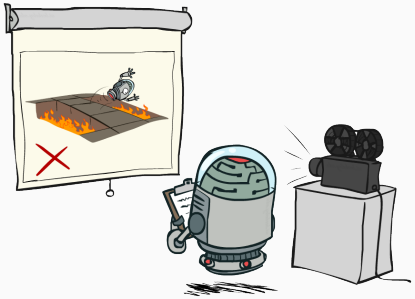
Q-learning (with replay buffer)

Initialize $Q(s, a)$ arbitrarily, $D \leftarrow \emptyset$;

observe s_0;
for $k = 0, 1, 2, \ldots$
  take action a_k ε-greedy$(Q(s_k, a))$;
  observe r_k, s_{k+1};
  store (s_k, a_k, r_k, s_{k+1}) in D;
  sample (s, a, r, s') from D;
  $y \leftarrow r + \gamma \max_{a'} Q(s', a')$;
  $Q(s, a) \leftarrow (1 - \alpha) Q(s, a) + \alpha y$

Q-learning is o□-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»:

$$\pi_\varepsilon(s,a) = \pi(a|s) \cdot (1-\varepsilon) + \frac{\varepsilon}{|A|}$$

"-soft Policy Improvement

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

$$\sim : \quad \text{"-greedy}pQ \quad ps; aqq \quad \tilde{n} \quad \sim \copyright$$

Q-learning vs SARSA

"-soft Policy Improvement

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

$$\sim : \quad \text{"-greedy}(Q\ ps; aqq \quad \tilde{n} \quad \sim \copyright$$

$$Q_{k\ 1}ps; aq \text{ Đ } Q_k ps; aq \qquad {}_k\ rps; aq \qquad Q_k ps^1; a^1q \qquad Q_k ps; aq$$

Q-learning

a^1 taken from target policy

$${}_k psq : \quad \operatorname*{argmax}_a Q_k ps; aq$$

SARSA

a^1 taken from behavior policy

$${}_k : \quad \text{"-greedy}(Q_k ps; aqq$$

Q-learning vs SARSA

"-soft Policy Improvement

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

$$\sim : \quad \text{"-greedy}(Q \ ps; aqq \quad \tilde{n} \quad \sim \copyright$$

$$Q_{k \ 1}ps; aq \ Ð \ Q_k ps; aq \qquad _k \ rps; aq \qquad Q_k ps^1; a^1q \qquad Q_k ps; aq$$

Q-learning

a^1 taken from target policy

$$_k psq : \quad \operatorname{argmax}_a Q_k ps; aq$$

SARSA

a^1 taken from behavior policy

$$_k : \quad \text{"-greedy}(Q_k ps; aqq$$

SARSA convergence properties

SARSA converges to optimal "-soft policy, satisfying

$$\text{"-greedy}(Q \ ps; aqq$$

SARSA

SARSA

Initialize $Q(s, a)$ 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} ε-greedy$(Q(s_{k+1}, a))$

$y = r_k + Q(s_{k+1}, a_{k+1})$;

$Q(s_k, a_k) \leftarrow (1 - \alpha_k) Q(s_k, a_k) + \alpha_k y$

SARSA

SARSA

Initialize $Q(s, a)$ 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} ε-greedy$(Q(s_{k+1}, a))$;
  $y \leftarrow r_k + Q(s_{k+1}, a_{k+1})$;
  $Q(s_k, a_k) \leftarrow (1 - \alpha_k) Q(s_k, a_k) + \alpha_k y$

Expected-SARSA

Initialize $Q(s, a)$ arbitrarily.

observe s_0;
for $k = 0, 1, 2, \ldots$
  sample a_k ε-greedy$(Q(s_k, a))$;
  take action a_k, observe r_k, s_{k+1};
  $y \leftarrow r_k + \mathbb{E}_{a_{k+1}} Q(s_{k+1}, a_{k+1})$;
  $Q(s_k, a_k) \leftarrow (1 - \alpha_k) Q(s_k, a_k) + \alpha_k y$;

SARSA is on-policy

For transition $(s; a; r; s^1; a^1)$:

$$Q_{k+1}(s; a) \leftarrow Q_k(s; a) + \alpha_k \left(r + Q_k(s^1; a^1) - Q_k(s; a) \right)$$

Data source	Interaction policy	Convergence
Online experience	Decaying exploration	Q^*
	Persistent exploration	Q^* -greedy
Expert data (a^1 taken from buffer)	-	

SARSA is on-policy

For transition $ps; a; r; s^1; a^1q$:

$$Q_{k\ \ 1}ps; aq \text{ Đ } Q_k ps; aq \qquad {}_k pr \qquad Q_k ps^1; a^1q \qquad Q_k ps; aqq$$

Data source	Interaction policy	Convergence
Online experience	Decaying exploration Persistent exploration	Q Q_{ε}-greedy
Expert data (a^1 taken from bu□er)	-	Q^{expert}
Experience replay (a^1 taken from bu□er)		

SARSA is on-policy

For transition $ps; a; r; s^1; a^1q$:

$$Q_{k\ \ 1}ps; aq \text{ Đ } Q_k ps; aq \qquad {}_k pr \qquad Q_k ps^1; a^1q \qquad Q_k ps; aqq$$

Data source	Interaction policy	Convergence
Online experience	Decaying exploration	Q
	Persistent exploration	$Q_{\text{"-greedy}}$
Expert data (a^1 taken from bu□er)	-	Q^{expert}
Experience replay (a^1 taken from bu□er)	(arbitrary)	divergence

«Fixing» SARSA (not really sarsa now)

For transition (s, a, r, s^1) generate a^1 $(a^1 \mid s^1)$:

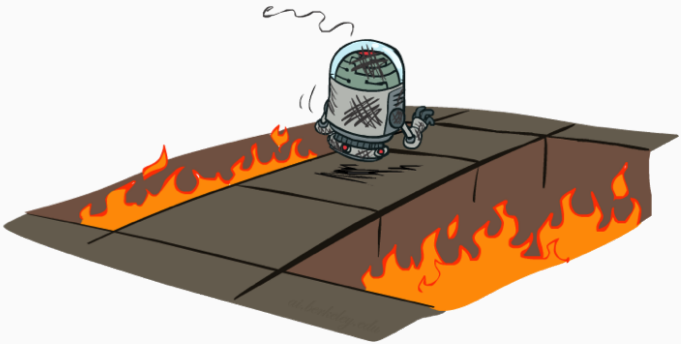
$$Q_{k\ 1}(s, a) \leftarrow Q_k(s, a) + \alpha r \quad Q_k(s^1, a^1) \quad Q_k(s, a))$$

«Fixing» SARSA (not really sarsa now)

For transition $ps; a; r; s^1q$ generate a^1 $pa^1 \mid s^1q$:

$$Q_{k\ \ 1}ps; aq \text{ Đ } Q_k ps; aq \qquad {}_k pr \qquad Q_k ps^1; a^1q \qquad Q_k ps; aqq$$

Data source	Interaction policy	Convergence
(any, but a^1 generated online)	Decaying exploration	Q
	Persistent exploration	$Q_{\text{„ -greedy}}$



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;