

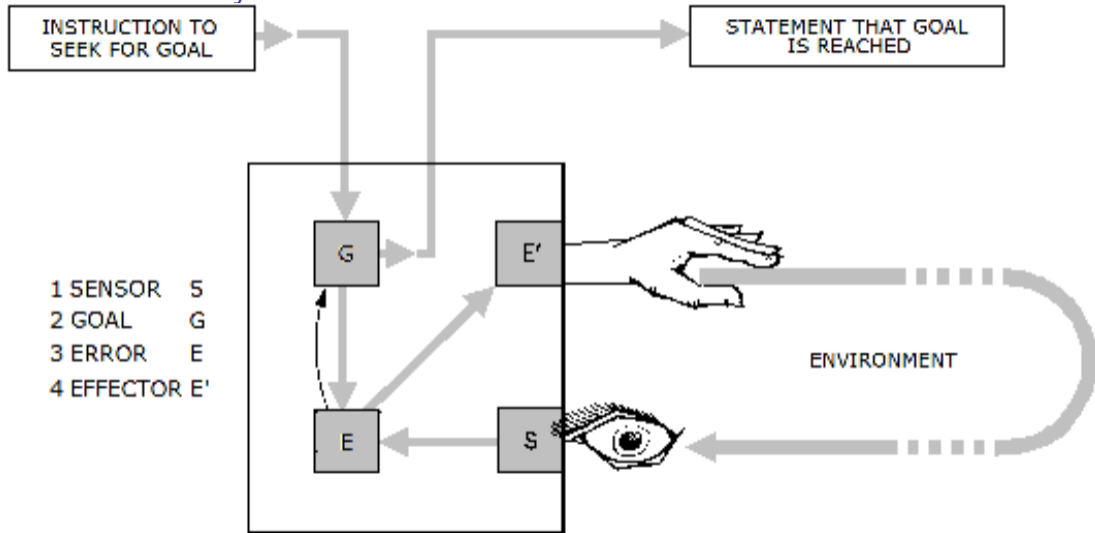
Reinforcement learning

Tomáš Svoboda

Vision for Robots and Autonomous Systems, Center for Machine Perception
Department of Cybernetics
Faculty of Electrical Engineering, Czech Technical University in Prague

April 15, 2020

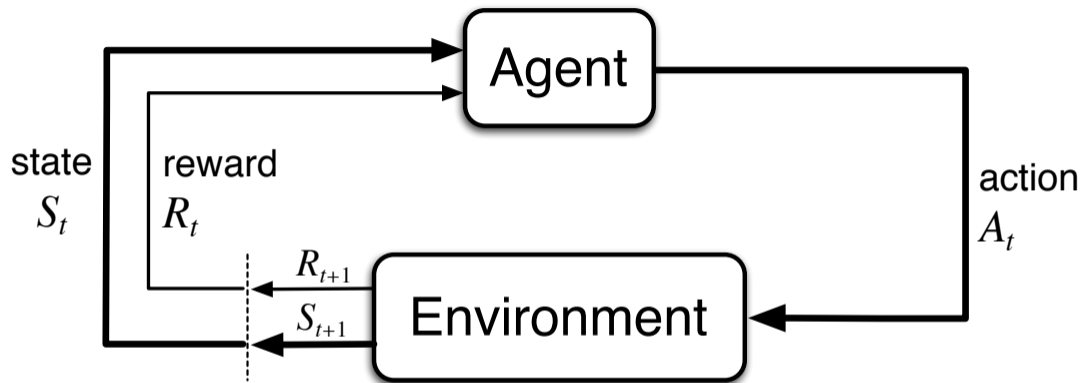
Goal-directed system



A SIMPLE GOAL-DIRECTED SYSTEM

¹Figure from <http://www.cybsoc.org/gcyb.htm>

Reinforcement Learning



2

- ▶ Feedback in form of **Rewards**
- ▶ Learn to act so as to maximize expected rewards.

²Scheme from [3]

Autonomous Flipper Control with Safety Constraints

Martin Pecka, Vojtěch Šalanský,
Karel Zimmermann, Tomáš Svoboda

experiments utilizing
Constrained Relative Entropy Policy Search

Video: Learning safe policies³

³M. Pecka, V. Salansky, K. Zimmermann, T. Svoboda. Autonomous flipper control with safety constraints. In Intelligent Robots and Systems (IROS), 2016

From off-line (MDPs) to on-line (RL)

Markov decision process – MDPs. Off-line search, we know:

- ▶ A set of states $s \in \mathcal{S}$ (map)
- ▶ A set of actions per state. $a \in \mathcal{A}$
- ▶ A transition model $T(s, a, s')$ or $p(s'|s, a)$ (robot)
- ▶ A reward function $r(s, a, s')$ (map, robot)

Looking for the optimal policy $\pi(s)$. We can plan/search before the robot enters the environment.

On-line problem:

- ▶ Transition model p and reward function r not known.
- ▶ Agent/robot must act and learn from experience.

From off-line (MDPs) to on-line (RL)

Markov decision process – MDPs. Off-line search, we know:

- ▶ A set of states $s \in \mathcal{S}$ (map)
- ▶ A set of actions per state. $a \in \mathcal{A}$
- ▶ A transition model $T(s, a, s')$ or $p(s'|s, a)$ (robot)
- ▶ A reward function $r(s, a, s')$ (map, robot)

Looking for the optimal policy $\pi(s)$. We can plan/search before the robot enters the environment.

On-line problem:

- ▶ Transition model p and reward function r not known.
- ▶ Agent/robot must act and learn from experience.

(Transition) Model-based learning

The main idea: Do something and:

- ▶ Learn an approximate model from experiences.
- ▶ Solve as if the model was correct.

Learning MDP model:

- ▶ In s try a , observe s' , count (s, a, s') .
- ▶ Normalize to get an estimate of $p(s' | a, s)$.
- ▶ Discover (by observation) each $r(s, a, s')$ when experienced.

Solve the learned MDP.

(Transition) Model-based learning

The main idea: Do something and:

- ▶ Learn an approximate model from experiences.
- ▶ Solve as if the model was correct.

Learning MDP model:

- ▶ In s try a , observe s' , count (s, a, s') .
- ▶ Normalize to get an estimate of $p(s' | a, s)$.
- ▶ Discover (by observation) each $r(s, a, s')$ when experienced.

Solve the learned MDP.

(Transition) Model-based learning

The main idea: Do something and:

- ▶ Learn an approximate model from experiences.
- ▶ Solve as if the model was correct.

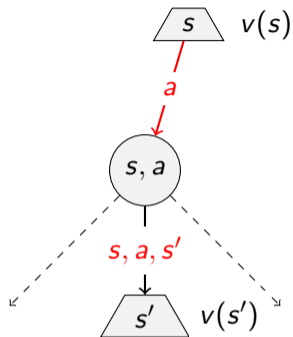
Learning MDP model:

- ▶ In s try a , **observe** s' , count (s, a, s') .
- ▶ Normalize to get an estimate of $p(s' | a, s)$.
- ▶ Discover (by observation) each $r(s, a, s')$ when experienced.

Solve the learned MDP.

Reward function $r(s, a, s')$

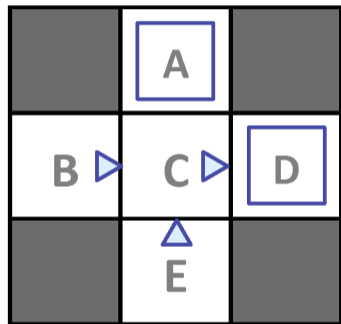
- ▶ $r(s, a, s')$ - reward for taking a in s and landing in s' .
- ▶ In Grid world we assumed $r(s, a, s')$ to be the same everywhere.
- ▶ In a real world it is different (going up, down, ...)



In ai-gym `env.step(action)` returns $s', r(s, action, s')$.

Model-based learning: Grid example

Input Policy π



Assume: $\gamma = 1$

Observed Episodes (Training)

Episode 1

B, east, C, -1
C, east, D, -1
D, exit, x, +10

Episode 2

B, east, C, -1
C, east, D, -1
D, exit, x, +10

Episode 3

E, north, C, -1
C, east, D, -1
D, exit, x, +10

Episode 4

E, north, C, -1
C, east, A, -1
A, exit, x, -10

4

⁴Figure from [1]

Learning transition model

$p(D \mid \text{east}, C) = ?$

Episode 1

B, east, C, -1
C, east, D, -1
D, exit, x, +10

Episode 2

B, east, C, -1
C, east, D, -1
D, exit, x, +10

Episode 3

E, north, C, -1
C, east, D, -1
D, exit, x, +10

Episode 4

E, north, C, -1
C, east, A, -1
A, exit, x, -10

Learning reward function

$r(C, \text{east}, D) = ?$

Episode 1

B, east, C, -1
C, east, D, -1
D, exit, x, +10

Episode 2

B, east, C, -1
C, east, D, -1
D, exit, x, +10

Episode 3

E, north, C, -1
C, east, D, -1
D, exit, x, +10

Episode 4

E, north, C, -1
C, east, A, -1
A, exit, x, -10

Model based vs model-free: Expected age $E[A]$

Random variable age A .

$$E[A] = \sum_a P(A = a)a$$

We do not know $P(A = a)$, collecting N samples $[a_1, a_2, \dots, a_N]$.

Model based

$$\hat{P}(a) = \frac{\text{num}(a)}{N}$$
$$E[A] \approx \sum_a \hat{P}(a)a$$

Model free

$$E[A] \approx \frac{1}{N} \sum_i a_i$$

Model based vs model-free: Expected age $E[A]$

Random variable age A .

$$E[A] = \sum_a P(A = a)a$$

We do not know $P(A = a)$, collecting N samples $[a_1, a_2, \dots, a_N]$.

Model based

$$\hat{P}(a) = \frac{\text{num}(a)}{N}$$
$$E[A] \approx \sum_a \hat{P}(a)a$$

Model free

$$E[A] \approx \frac{1}{N} \sum_i a_i$$

Model based vs model-free: Expected age $E[A]$

Random variable age A .

$$E[A] = \sum_a P(A = a)a$$

We do not know $P(A = a)$, collecting N samples $[a_1, a_2, \dots, a_N]$.

Model based

$$\hat{P}(a) = \frac{\text{num}(a)}{N}$$

$$E[A] \approx \sum_a \hat{P}(a)a$$

Model free

$$E[A] \approx \frac{1}{N} \sum_i a_i$$

Model based vs model-free: Expected age $E[A]$

Random variable age A .

$$E[A] = \sum_a P(A = a)a$$

We do not know $P(A = a)$, collecting N samples $[a_1, a_2, \dots, a_N]$.

Model based

$$\hat{P}(a) = \frac{\text{num}(a)}{N}$$
$$E[A] \approx \sum_a \hat{P}(a)a$$

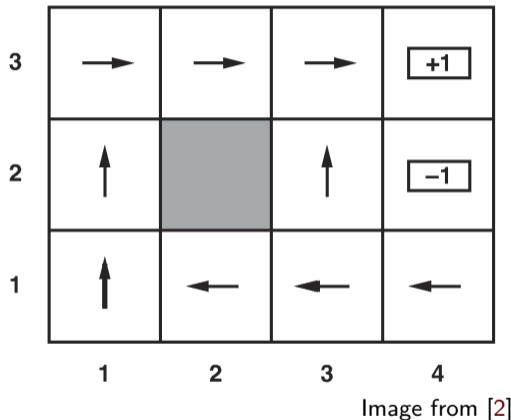
Model free

$$E[A] \approx \frac{1}{N} \sum_i a_i$$

Model-free learning

Passive learning

- ▶ **Input:** a fixed policy $\pi(s)$
- ▶ We want to know how good it is.
- ▶ r, p not known.
- ▶ Execute policy ...
- ▶ and learn on the way.
- ▶ **Goal:** learn the state values $v^\pi(s)$

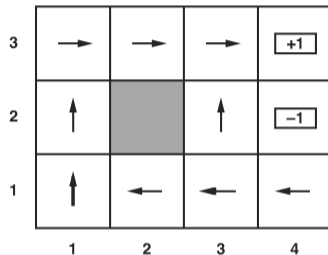


Direct evaluation from episodes

Value of s for π – expected sum of discounted rewards – expected return

$$v^\pi(S_t) = E \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \right]$$

$$v^\pi(S_t) = E[G_t]$$

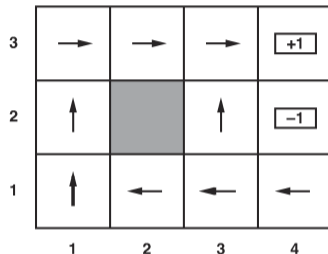


Direct evaluation from episodes

Value of s for π – expected sum of discounted rewards – expected return

$$v^\pi(S_t) = E \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \right]$$

$$v^\pi(S_t) = E[G_t]$$



$(1, 1) \cdot 0.04 \rightsquigarrow (1, 2) \cdot 0.04 \rightsquigarrow (1, 3) \cdot 0.04 \rightsquigarrow (1, 2) \cdot 0.04 \rightsquigarrow (1, 3) \cdot 0.04 \rightsquigarrow (2, 3) \cdot 0.04 \rightsquigarrow (3, 3) \cdot 0.04 \rightsquigarrow (4, 3) + 1$

$(1, 1) \cdot 0.04 \rightsquigarrow (1, 2) \cdot 0.04 \rightsquigarrow (1, 3) \cdot 0.04 \rightsquigarrow (2, 3) \cdot 0.04 \rightsquigarrow (3, 3) \cdot 0.04 \rightsquigarrow (3, 2) \cdot 0.04 \rightsquigarrow (3, 3) \cdot 0.04 \rightsquigarrow (4, 3) + 1$

$(1, 1) \cdot 0.04 \rightsquigarrow (2, 1) \cdot 0.04 \rightsquigarrow (3, 1) \cdot 0.04 \rightsquigarrow (3, 2) \cdot 0.04 \rightsquigarrow (4, 2) - 1$.

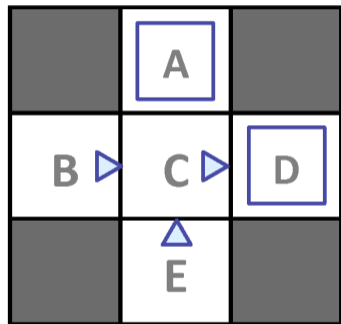
Direct evaluation from episodes, $v^\pi(S_t) = E[G_t]$, $\gamma = 1$

$(1, 1)_{-.04} \rightsquigarrow (1, 2)_{-.04} \rightsquigarrow (1, 3)_{-.04} \rightsquigarrow (1, 2)_{-.04} \rightsquigarrow (1, 3)_{-.04} \rightsquigarrow (2, 3)_{-.04} \rightsquigarrow (3, 3)_{-.04} \rightsquigarrow (4, 3)_{+1}$
 $(1, 1)_{-.04} \rightsquigarrow (1, 2)_{-.04} \rightsquigarrow (1, 3)_{-.04} \rightsquigarrow (2, 3)_{-.04} \rightsquigarrow (3, 3)_{-.04} \rightsquigarrow (3, 2)_{-.04} \rightsquigarrow (3, 3)_{-.04} \rightsquigarrow (4, 3)_{+1}$
 $(1, 1)_{-.04} \rightsquigarrow (2, 1)_{-.04} \rightsquigarrow (3, 1)_{-.04} \rightsquigarrow (3, 2)_{-.04} \rightsquigarrow (4, 2)_{-1}$.

What is $v(3, 2)$ after these episodes?

Direct evaluation: Grid example

Input Policy π



Assume: $\gamma = 1$

Observed Episodes (Training)

Episode 1

B, east, C, -1
C, east, D, -1
D, exit, x, +10

Episode 2

B, east, C, -1
C, east, D, -1
D, exit, x, +10

Episode 3

E, north, C, -1
C, east, D, -1
D, exit, x, +10

Episode 4

E, north, C, -1
C, east, A, -1
A, exit, x, -10

Direct evaluation: Grid example, $\gamma = 1$

What is $v(C)$ after the 4 episodes?

Episode 1

B, east, C, -1
C, east, D, -1
D, exit, x, +10

Episode 2

B, east, C, -1
C, east, D, -1
D, exit, x, +10

Episode 3

E, north, C, -1
C, east, D, -1
D, exit, x, +10

Episode 4

E, north, C, -1
C, east, A, -1
A, exit, x, -10

Direct evaluation algorithm

$(1, 1)_{-.04} \rightsquigarrow (1, 2)_{-.04} \rightsquigarrow (1, 3)_{-.04} \rightsquigarrow (1, 2)_{-.04} \rightsquigarrow (1, 3)_{-.04} \rightsquigarrow (2, 3)_{-.04} \rightsquigarrow (3, 3)_{-.04} \rightsquigarrow (4, 3)_{+1}$
 $(1, 1)_{-.04} \rightsquigarrow (1, 2)_{-.04} \rightsquigarrow (1, 3)_{-.04} \rightsquigarrow (2, 3)_{-.04} \rightsquigarrow (3, 3)_{-.04} \rightsquigarrow (3, 2)_{-.04} \rightsquigarrow (3, 3)_{-.04} \rightsquigarrow (4, 3)_{+1}$
 $(1, 1)_{-.04} \rightsquigarrow (2, 1)_{-.04} \rightsquigarrow (3, 1)_{-.04} \rightsquigarrow (3, 2)_{-.04} \rightsquigarrow (4, 2)_{-1}$.

Input: a policy π to be evaluated

Initialize:

$V(s) \in \mathbb{R}$, arbitrarily, for all $s \in \mathcal{S}$

$Returns(s) \leftarrow$ an empty list, for all $s \in \mathcal{S}$

Loop forever (for each episode):

Generate an episode following π : $S_0, A_0, R_1, S_1, A_1, R_2, \dots, S_{T-1}, A_{T-1}, R_T$

$G \leftarrow 0$

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

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

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

Append G to $Returns(S_t)$

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

Direct evaluation: analysis

The good:

- ▶ Simple, easy to understand and implement.
- ▶ Does not need p, r and eventually it computes the true v^π .

The bad:

- ▶ Each state value learned in isolation.
- ▶ State values are not independent
- ▶
$$v^\pi(s) = \sum_{s'} p(s' | s, \pi(s)) [r(s, \pi(s), s') + \gamma v^\pi(s')]$$

Direct evaluation: analysis

The good:

- ▶ Simple, easy to understand and implement.
- ▶ Does not need p, r and eventually it computes the true v^π .

The bad:

$(1, 1)_{-.04} \rightsquigarrow (1, 2)_{-.04} \rightsquigarrow (1, 3)_{-.04} \rightsquigarrow (1, 2)_{-.04} \rightsquigarrow (1, 3)_{-.04} \rightsquigarrow (2, 3)_{-.04} \rightsquigarrow (3, 3)_{-.04} \rightsquigarrow (4, 3)_{+1}$
 $(1, 1)_{-.04} \rightsquigarrow (1, 2)_{-.04} \rightsquigarrow (1, 3)_{-.04} \rightsquigarrow (2, 3)_{-.04} \rightsquigarrow (3, 3)_{-.04} \rightsquigarrow (3, 2)_{-.04} \rightsquigarrow (3, 3)_{-.04} \rightsquigarrow (4, 3)_{+1}$
 $(1, 1)_{-.04} \rightsquigarrow (2, 1)_{-.04} \rightsquigarrow (3, 1)_{-.04} \rightsquigarrow (3, 2)_{-.04} \rightsquigarrow (4, 2)_{-1}$.

- ▶ Each state value learned in isolation.
- ▶ State values are not independent
- ▶ $v^\pi(s) = \sum_{s'} p(s' | s, \pi(s)) [r(s, \pi(s), s') + \gamma v^\pi(s')]$

Direct evaluation: analysis

The good:

- ▶ Simple, easy to understand and implement.
- ▶ Does not need p, r and eventually it computes the true v^π .

The bad:

$(1, 1)_{-.04} \rightsquigarrow (1, 2)_{-.04} \rightsquigarrow (1, 3)_{-.04} \rightsquigarrow (1, 2)_{-.04} \rightsquigarrow (1, 3)_{-.04} \rightsquigarrow (2, 3)_{-.04} \rightsquigarrow (3, 3)_{-.04} \rightsquigarrow (4, 3)_{+1}$
 $(1, 1)_{-.04} \rightsquigarrow (1, 2)_{-.04} \rightsquigarrow (1, 3)_{-.04} \rightsquigarrow (2, 3)_{-.04} \rightsquigarrow (3, 3)_{-.04} \rightsquigarrow (3, 2)_{-.04} \rightsquigarrow (3, 3)_{-.04} \rightsquigarrow (4, 3)_{+1}$
 $(1, 1)_{-.04} \rightsquigarrow (2, 1)_{-.04} \rightsquigarrow (3, 1)_{-.04} \rightsquigarrow (3, 2)_{-.04} \rightsquigarrow (4, 2)_{-1}$.

- ▶ Each state value learned in isolation.
- ▶ State values are not independent
- ▶ $v^\pi(s) = \sum_{s'} p(s' | s, \pi(s)) [r(s, \pi(s), s') + \gamma v^\pi(s')]$

Policy evaluation?

In each round, replace V with a one-step-look-ahead

$$V_0^\pi(s) = 0$$

$$V_{k+1}^\pi(s) \leftarrow \sum_{s'} p(s' | s, \pi(s)) [r(s, \pi(s), s') + \gamma V_k^\pi(s')]$$

Problem: both $p(s' | s, \pi(s))$ and $r(s, \pi(s), s')$ unknown!

Policy evaluation?

In each round, replace V with a one-step-look-ahead

$$V_0^\pi(s) = 0$$

$$V_{k+1}^\pi(s) \leftarrow \sum_{s'} p(s' | s, \pi(s)) [r(s, \pi(s), s') + \gamma V_k^\pi(s')]$$

Problem: both $p(s' | s, \pi(s))$ and $r(s, \pi(s), s')$ unknown!

Use samples for evaluating policy?

MDP (p, r known) : Update V estimate by a weighted average:

$$V_{k+1}^{\pi}(s) \leftarrow \sum_{s'} p(s' | s, \pi(s)) [r(s, \pi(s), s') + \gamma V_k^{\pi}(s')]$$

What about try and average? Trials at time t

$$\text{trial}^1 = R_{t+1}^1 + \gamma V(S_{t+1}^1)$$

$$\text{trial}^2 = R_{t+1}^2 + \gamma V(S_{t+1}^2)$$

$$\vdots = \vdots$$

$$\text{trial}^n = R_{t+1}^n + \gamma V(S_{t+1}^n)$$

$$V(S_t) \leftarrow \frac{1}{n} \sum_i \text{trial}^i$$

Use samples for evaluating policy?

MDP (p, r known) : Update V estimate by a weighted average:

$$V_{k+1}^{\pi}(s) \leftarrow \sum_{s'} p(s' | s, \pi(s)) [r(s, \pi(s), s') + \gamma V_k^{\pi}(s')]$$

What about try and average? Trials at time t

$$\text{trial}^1 = R_{t+1}^1 + \gamma V(S_{t+1}^1)$$

$$\text{trial}^2 = R_{t+1}^2 + \gamma V(S_{t+1}^2)$$

$$\vdots = \vdots$$

$$\text{trial}^n = R_{t+1}^n + \gamma V(S_{t+1}^n)$$

$$V(S_t) \leftarrow \frac{1}{n} \sum_i \text{trial}^i$$

Temporal-difference value learning

$(1, 1)_{-.04} \rightsquigarrow (1, 2)_{-.04} \rightsquigarrow (1, 3)_{-.04} \rightsquigarrow (1, 2)_{-.04} \rightsquigarrow (1, 3)_{-.04} \rightsquigarrow (2, 3)_{-.04} \rightsquigarrow (3, 3)_{-.04} \rightsquigarrow (4, 3)_{+1}$
 $(1, 1)_{-.04} \rightsquigarrow (1, 2)_{-.04} \rightsquigarrow (1, 3)_{-.04} \rightsquigarrow (2, 3)_{-.04} \rightsquigarrow (3, 3)_{-.04} \rightsquigarrow (3, 2)_{-.04} \rightsquigarrow (3, 3)_{-.04} \rightsquigarrow (4, 3)_{+1}$
 $(1, 1)_{-.04} \rightsquigarrow (2, 1)_{-.04} \rightsquigarrow (3, 1)_{-.04} \rightsquigarrow (3, 2)_{-.04} \rightsquigarrow (4, 2)_{-1}$.

$$\gamma = 1$$

From first trial (episode): $V(2, 3) = 0.92$, $V(1, 3) = 0.84$, ...

In second episode, going from $S_t = (1, 3)$ to $S_{t+1} = (2, 3)$ with reward $R_{t+1} = -0.04$, hence:

$$V(1, 3) = R_{t+1} + V(2, 3) = -0.04 + 0.92 = 0.88$$

- ▶ First estimate 0.84 is a bit lower than 0.88. $V(S_t)$ is different than $R_{t+1} + \gamma V(S_{t+1})$
- ▶ Update: $V(S_t) \leftarrow V(S_t) + \alpha \left([R_{t+1} + \gamma V(S_{t+1})] - V(S_t) \right)$
- ▶ α is the learning rate.
- ▶ $V(S_t) \leftarrow (1 - \alpha)V(S_t) + \alpha$ (new sample)

Temporal-difference value learning

$(1, 1)_{-.04} \rightsquigarrow (1, 2)_{-.04} \rightsquigarrow (1, 3)_{-.04} \rightsquigarrow (1, 2)_{-.04} \rightsquigarrow (1, 3)_{-.04} \rightsquigarrow (2, 3)_{-.04} \rightsquigarrow (3, 3)_{-.04} \rightsquigarrow (4, 3)_{+1}$

$(1, 1)_{-.04} \rightsquigarrow (1, 2)_{-.04} \rightsquigarrow (1, 3)_{-.04} \rightsquigarrow (2, 3)_{-.04} \rightsquigarrow (3, 3)_{-.04} \rightsquigarrow (3, 2)_{-.04} \rightsquigarrow (3, 3)_{-.04} \rightsquigarrow (4, 3)_{+1}$

$(1, 1)_{-.04} \rightsquigarrow (2, 1)_{-.04} \rightsquigarrow (3, 1)_{-.04} \rightsquigarrow (3, 2)_{-.04} \rightsquigarrow (4, 2)_{-1}$.

$$\gamma = 1$$

From first trial (episode): $V(2, 3) = 0.92$, $V(1, 3) = 0.84, \dots$

In second episode, going from $S_t = (1, 3)$ to $S_{t+1} = (2, 3)$ with reward $R_{t+1} = -0.04$, hence:

$$V(1, 3) = R_{t+1} + V(2, 3) = -0.04 + 0.92 = 0.88$$

- ▶ First estimate 0.84 is a bit lower than 0.88. $V(S_t)$ is different than $R_{t+1} + \gamma V(S_{t+1})$
- ▶ Update: $V(S_t) \leftarrow V(S_t) + \alpha \left([R_{t+1} + \gamma V(S_{t+1})] - V(S_t) \right)$
- ▶ α is the learning rate.
- ▶ $V(S_t) \leftarrow (1 - \alpha)V(S_t) + \alpha$ (new sample)

Temporal-difference value learning

$(1, 1)_{-.04} \rightsquigarrow (1, 2)_{-.04} \rightsquigarrow (1, 3)_{-.04} \rightsquigarrow (1, 2)_{-.04} \rightsquigarrow (1, 3)_{-.04} \rightsquigarrow (2, 3)_{-.04} \rightsquigarrow (3, 3)_{-.04} \rightsquigarrow (4, 3)_{+1}$
 $(1, 1)_{-.04} \rightsquigarrow (1, 2)_{-.04} \rightsquigarrow (1, 3)_{-.04} \rightsquigarrow (2, 3)_{-.04} \rightsquigarrow (3, 3)_{-.04} \rightsquigarrow (3, 2)_{-.04} \rightsquigarrow (3, 3)_{-.04} \rightsquigarrow (4, 3)_{+1}$
 $(1, 1)_{-.04} \rightsquigarrow (2, 1)_{-.04} \rightsquigarrow (3, 1)_{-.04} \rightsquigarrow (3, 2)_{-.04} \rightsquigarrow (4, 2)_{-1}$.

$$\gamma = 1$$

From first trial (episode): $V(2, 3) = 0.92$, $V(1, 3) = 0.84, \dots$

In second episode, going from $S_t = (1, 3)$ to $S_{t+1} = (2, 3)$ with reward $R_{t+1} = -0.04$, hence:

$$V(1, 3) = R_{t+1} + V(2, 3) = -0.04 + 0.92 = 0.88$$

- ▶ First estimate 0.84 is a bit lower than 0.88. $V(S_t)$ is different than $R_{t+1} + \gamma V(S_{t+1})$
- ▶ Update: $V(S_t) \leftarrow V(S_t) + \alpha \left([R_{t+1} + \gamma V(S_{t+1})] - V(S_t) \right)$
- ▶ α is the learning rate.
- ▶ $V(S_t) \leftarrow (1 - \alpha)V(S_t) + \alpha$ (new sample)

Temporal-difference value learning

$(1, 1)_{-.04} \rightsquigarrow (1, 2)_{-.04} \rightsquigarrow (1, 3)_{-.04} \rightsquigarrow (1, 2)_{-.04} \rightsquigarrow (1, 3)_{-.04} \rightsquigarrow (2, 3)_{-.04} \rightsquigarrow (3, 3)_{-.04} \rightsquigarrow (4, 3)_{+1}$
 $(1, 1)_{-.04} \rightsquigarrow (1, 2)_{-.04} \rightsquigarrow (1, 3)_{-.04} \rightsquigarrow (2, 3)_{-.04} \rightsquigarrow (3, 3)_{-.04} \rightsquigarrow (3, 2)_{-.04} \rightsquigarrow (3, 3)_{-.04} \rightsquigarrow (4, 3)_{+1}$
 $(1, 1)_{-.04} \rightsquigarrow (2, 1)_{-.04} \rightsquigarrow (3, 1)_{-.04} \rightsquigarrow (3, 2)_{-.04} \rightsquigarrow (4, 2)_{-1}$.

$$\gamma = 1$$

From first trial (episode): $V(2, 3) = 0.92$, $V(1, 3) = 0.84, \dots$

In second episode, going from $S_t = (1, 3)$ to $S_{t+1} = (2, 3)$ with reward $R_{t+1} = -0.04$, hence:

$$V(1, 3) = R_{t+1} + V(2, 3) = -0.04 + 0.92 = 0.88$$

- ▶ First estimate 0.84 is a bit lower than 0.88. $V(S_t)$ is different than $R_{t+1} + \gamma V(S_{t+1})$
- ▶ Update: $V(S_t) \leftarrow V(S_t) + \alpha \left([R_{t+1} + \gamma V(S_{t+1})] - V(S_t) \right)$
- ▶ α is the learning rate.
- ▶ $V(S_t) \leftarrow (1 - \alpha)V(S_t) + \alpha$ (new sample)

Temporal-difference value learning

$(1, 1)_{-.04} \rightsquigarrow (1, 2)_{-.04} \rightsquigarrow (1, 3)_{-.04} \rightsquigarrow (1, 2)_{-.04} \rightsquigarrow (1, 3)_{-.04} \rightsquigarrow (2, 3)_{-.04} \rightsquigarrow (3, 3)_{-.04} \rightsquigarrow (4, 3)_{+1}$
 $(1, 1)_{-.04} \rightsquigarrow (1, 2)_{-.04} \rightsquigarrow (1, 3)_{-.04} \rightsquigarrow (2, 3)_{-.04} \rightsquigarrow (3, 3)_{-.04} \rightsquigarrow (3, 2)_{-.04} \rightsquigarrow (3, 3)_{-.04} \rightsquigarrow (4, 3)_{+1}$
 $(1, 1)_{-.04} \rightsquigarrow (2, 1)_{-.04} \rightsquigarrow (3, 1)_{-.04} \rightsquigarrow (3, 2)_{-.04} \rightsquigarrow (4, 2)_{-1}$.

$$\gamma = 1$$

From first trial (episode): $V(2, 3) = 0.92$, $V(1, 3) = 0.84, \dots$

In second episode, going from $S_t = (1, 3)$ to $S_{t+1} = (2, 3)$ with reward $R_{t+1} = -0.04$, hence:

$$V(1, 3) = R_{t+1} + V(2, 3) = -0.04 + 0.92 = 0.88$$

- ▶ First estimate 0.84 is a bit lower than 0.88. $V(S_t)$ is different than $R_{t+1} + \gamma V(S_{t+1})$
- ▶ Update: $V(S_t) \leftarrow V(S_t) + \alpha \left([R_{t+1} + \gamma V(S_{t+1})] - V(S_t) \right)$
- ▶ α is the learning rate.
- ▶ $V(S_t) \leftarrow (1 - \alpha)V(S_t) + \alpha (\text{new sample})$

Temporal-difference value learning

$(1, 1)_{-.04} \rightsquigarrow (1, 2)_{-.04} \rightsquigarrow (1, 3)_{-.04} \rightsquigarrow (1, 2)_{-.04} \rightsquigarrow (1, 3)_{-.04} \rightsquigarrow (2, 3)_{-.04} \rightsquigarrow (3, 3)_{-.04} \rightsquigarrow (4, 3)_{+1}$
 $(1, 1)_{-.04} \rightsquigarrow (1, 2)_{-.04} \rightsquigarrow (1, 3)_{-.04} \rightsquigarrow (2, 3)_{-.04} \rightsquigarrow (3, 3)_{-.04} \rightsquigarrow (3, 2)_{-.04} \rightsquigarrow (3, 3)_{-.04} \rightsquigarrow (4, 3)_{+1}$
 $(1, 1)_{-.04} \rightsquigarrow (2, 1)_{-.04} \rightsquigarrow (3, 1)_{-.04} \rightsquigarrow (3, 2)_{-.04} \rightsquigarrow (4, 2)_{-1}$.

$$\gamma = 1$$

From first trial (episode): $V(2, 3) = 0.92$, $V(1, 3) = 0.84, \dots$

In second episode, going from $S_t = (1, 3)$ to $S_{t+1} = (2, 3)$ with reward $R_{t+1} = -0.04$, hence:

$$V(1, 3) = R_{t+1} + V(2, 3) = -0.04 + 0.92 = 0.88$$

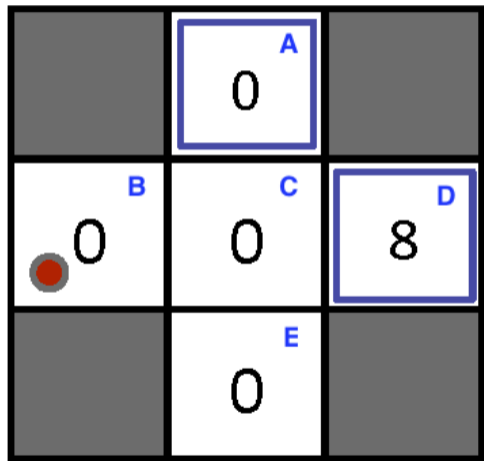
- ▶ First estimate 0.84 is a bit lower than 0.88. $V(S_t)$ is different than $R_{t+1} + \gamma V(S_{t+1})$
- ▶ Update: $V(S_t) \leftarrow V(S_t) + \alpha \left([R_{t+1} + \gamma V(S_{t+1})] - V(S_t) \right)$
- ▶ α is the learning rate.
- ▶ $V(S_t) \leftarrow (1 - \alpha)V(S_t) + \alpha (\text{new sample})$

Exponential moving average

$$\bar{x}_n = (1 - \alpha)\bar{x}_{n-1} + \alpha x_n$$

Example: TD Value learning

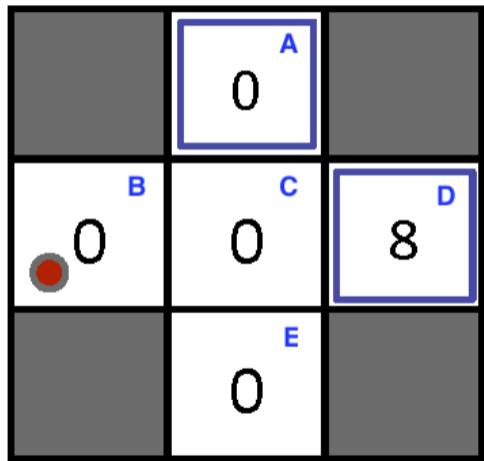
$$V(S_t) \leftarrow V(S_t) + \alpha(R_{t+1} + \gamma V(S_{t+1}) - V(S_t))$$



- ▶ Values represent initial $V(s)$
- ▶ Assume: $\gamma = 1, \alpha = 0.5, \pi(s) \Rightarrow$
 - ▶ $(B, \rightarrow, C), -2, \Rightarrow V(B)?$
 - ▶ $(C, \rightarrow, D), -2, \Rightarrow V(C)?$

Example: TD Value learning

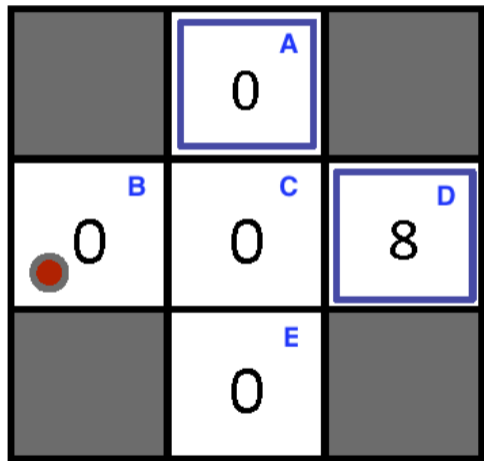
$$V(S_t) \leftarrow V(S_t) + \alpha(R_{t+1} + \gamma V(S_{t+1}) - V(S_t))$$



- ▶ Values represent initial $V(s)$
- ▶ Assume: $\gamma = 1, \alpha = 0.5, \pi(s) \Rightarrow$
- ▶ $(B, \rightarrow, C), -2, \Rightarrow V(B)?$
- ▶ $(C, \rightarrow, D), -2, \Rightarrow V(C)?$

Example: TD Value learning

$$V(S_t) \leftarrow V(S_t) + \alpha(R_{t+1} + \gamma V(S_{t+1}) - V(S_t))$$



- ▶ Values represent initial $V(s)$
- ▶ Assume: $\gamma = 1, \alpha = 0.5, \pi(s) \Rightarrow$
- ▶ $(B, \rightarrow, C), -2, \Rightarrow V(B)?$
- ▶ $(C, \rightarrow, D), -2, \Rightarrow V(C)?$

Temporal difference value learning: algorithm

Input: the policy π to be evaluated

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

Initialize $V(s)$, for all $s \in \mathcal{S}^+$, arbitrarily except that $V(\text{terminal}) = 0$

Loop for each episode:

 Initialize S

 Loop for each step of episode:

$A \leftarrow$ action given by π for S

 Take action A , observe R, S'

$V(S) \leftarrow V(S) + \alpha[R + \gamma V(S') - V(S)]$

$S \leftarrow S'$

 until S is terminal

What is wrong with the temporal difference Value learning?

The Good: Model-free value learning through mimicking Bellman updates

The Bad: How to turn values into a (new) policy?

$$\blacktriangleright \pi(s) = \arg \max_a \sum_{s'} p(s' | s, a) [r(s, a, s') + \gamma V(s')]$$

What is wrong with the temporal difference Value learning?

The Good: Model-free value learning through mimicking Bellman updates

The Bad: How to turn values into a (new) policy?

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

What is wrong with the temporal difference Value learning?

The Good: Model-free value learning through mimicking Bellman updates

The Bad: How to turn values into a (new) policy?

$$\blacktriangleright \pi(s) = \arg \max_a \sum_{s'} p(s' | s, a) [r(s, a, s') + \gamma V(s')]$$

Active reinforcement learning

Reminder: V , Q -value iteration for MDPs

Value/Utility iteration (depth limited evaluation):

▶ Start: $V_0(s) = 0$

▶ In each step update V by looking one step ahead:

$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} p(s' | s, a) [r(s, a, s') + \gamma V_k(s')]$$

Q values more useful (think about updating π)

▶ Start: $Q_0(s, a) = 0$

▶ In each step update Q by looking one step ahead:

$$Q_{k+1}(s, a) \leftarrow \sum_{s'} p(s' | s, a) \left[r(s, a, s') + \gamma \max_{a'} Q_k(s', a') \right]$$

Q-learning

$$\text{MDP update: } Q_{k+1}(s, a) \leftarrow \sum_{s'} p(s' | s, a) \left[r(s, a, s') + \gamma \max_{a'} Q_k(s', a') \right]$$

Learn Q values as the robot/agent goes (temporal difference)

- ▶ Drive the robot and fetch rewards (s, a, s', R)
- ▶ We know old estimates $Q(s, a)$ (and $Q(s', a')$), if not, initialize.
- ▶ A new trial/sample estimate at time t
 $\text{trial} = R_{t+1} + \gamma \max_a Q(S_{t+1}, a)$
- ▶ α update
 $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(\text{trial} - Q(S_t, A_t))$
or (the same)
 $Q(S_t, A_t) \leftarrow (1 - \alpha)Q(S_t, A_t) + \alpha \text{trial}$

In each step Q approximates the optimal q^* function.

Q-learning

$$\text{MDP update: } Q_{k+1}(s, a) \leftarrow \sum_{s'} p(s' | s, a) \left[r(s, a, s') + \gamma \max_{a'} Q_k(s', a') \right]$$

Learn Q values as the robot/agent goes (temporal difference)

- ▶ Drive the robot and fetch rewards (s, a, s', R)
- ▶ We know old estimates $Q(s, a)$ (and $Q(s', a')$), if not, initialize.
- ▶ A new trial/sample estimate at time t
 $\text{trial} = R_{t+1} + \gamma \max_a Q(S_{t+1}, a)$
- ▶ α update
 $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(\text{trial} - Q(S_t, A_t))$
or (the same)
 $Q(S_t, A_t) \leftarrow (1 - \alpha)Q(S_t, A_t) + \alpha \text{trial}$

In each step Q approximates the optimal q^* function.

Q-learning

$$\text{MDP update: } Q_{k+1}(s, a) \leftarrow \sum_{s'} p(s' | s, a) \left[r(s, a, s') + \gamma \max_{a'} Q_k(s', a') \right]$$

Learn Q values as the robot/agent goes (temporal difference)

- ▶ Drive the robot and fetch rewards (s, a, s', R)
- ▶ We know old estimates $Q(s, a)$ (and $Q(s', a')$), if not, initialize.

▶ A new trial/sample estimate at time t

$$\text{trial} = R_{t+1} + \gamma \max_a Q(S_{t+1}, a)$$

▶ α update

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(\text{trial} - Q(S_t, A_t))$$

or (the same)

$$Q(S_t, A_t) \leftarrow (1 - \alpha)Q(S_t, A_t) + \alpha \text{ trial}$$

In each step Q approximates the optimal q^* function.

Q-learning

$$\text{MDP update: } Q_{k+1}(s, a) \leftarrow \sum_{s'} p(s' | s, a) \left[r(s, a, s') + \gamma \max_{a'} Q_k(s', a') \right]$$

Learn Q values as the robot/agent goes (temporal difference)

- ▶ Drive the robot and fetch rewards (s, a, s', R)
- ▶ We know old estimates $Q(s, a)$ (and $Q(s', a')$), if not, initialize.
- ▶ A new trial/sample estimate at time t

$$\text{trial} = R_{t+1} + \gamma \max_a Q(S_{t+1}, a)$$

- ▶ α update
 $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(\text{trial} - Q(S_t, A_t))$
or (the same)
 $Q(S_t, A_t) \leftarrow (1 - \alpha)Q(S_t, A_t) + \alpha \text{trial}$

In each step Q approximates the optimal q^* function.

Q-learning

$$\text{MDP update: } Q_{k+1}(s, a) \leftarrow \sum_{s'} p(s' | s, a) \left[r(s, a, s') + \gamma \max_{a'} Q_k(s', a') \right]$$

Learn Q values as the robot/agent goes (temporal difference)

- ▶ Drive the robot and fetch rewards (s, a, s', R)
- ▶ We know old estimates $Q(s, a)$ (and $Q(s', a')$), if not, initialize.
- ▶ A new trial/sample estimate at time t

$$\text{trial} = R_{t+1} + \gamma \max_a Q(S_{t+1}, a)$$

- ▶ α update
 $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(\text{trial} - Q(S_t, A_t))$
or (the same)
 $Q(S_t, A_t) \leftarrow (1 - \alpha)Q(S_t, A_t) + \alpha \text{trial}$

In each step Q approximates the optimal q^* function.

Q-learning

$$\text{MDP update: } Q_{k+1}(s, a) \leftarrow \sum_{s'} p(s' | s, a) \left[r(s, a, s') + \gamma \max_{a'} Q_k(s', a') \right]$$

Learn Q values as the robot/agent goes (temporal difference)

- ▶ Drive the robot and fetch rewards (s, a, s', R)
- ▶ We know old estimates $Q(s, a)$ (and $Q(s', a')$), if not, initialize.
- ▶ A new trial/sample estimate at time t

$$\text{trial} = R_{t+1} + \gamma \max_a Q(S_{t+1}, a)$$

- ▶ α update
 $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(\text{trial} - Q(S_t, A_t))$
or (the same)
 $Q(S_t, A_t) \leftarrow (1 - \alpha)Q(S_t, A_t) + \alpha \text{trial}$

In each step Q approximates the optimal q^* function.

Q-learning: algorithm

step size $0 < \alpha \leq 1$

initialize $Q(s, a)$ for all $s \in \mathcal{S}, a \in \mathcal{S}(s)$

repeat episodes:

 initialize S

for for each step of episode: **do**

 choose A from S

 take action A , observe R, S'

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

$S \leftarrow S'$

end for until S is terminal

until Time is up, ...

From Q-learning to Q-learning agent

- ▶ Drive the robot and fetch rewards. (s, a, s', R)
- ▶ We know old estimates $Q(s, a)$ (and $Q(s', a')$), if not, initialize.
- ▶ A new trial/sample estimate: $\text{trial} = R_{t+1} + \gamma \max_a Q(S_{t+1}, a)$
- ▶ α update: $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(\text{trial} - Q(S_t, A_t))$

Technicalities for the Q-learning agent

- ▶ How to represent Q -function?
- ▶ What is the value for terminal? $Q(s, \text{Exit})$ or $Q(s, \text{None})$
- ▶ How to drive? Where to drive next? Does it change over the course?

From Q-learning to Q-learning agent

- ▶ Drive the robot and fetch rewards. (s, a, s', R)
- ▶ We know old estimates $Q(s, a)$ (and $Q(s', a')$), if not, initialize.
- ▶ A new trial/sample estimate: $\text{trial} = R_{t+1} + \gamma \max_a Q(S_{t+1}, a)$
- ▶ α update: $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(\text{trial} - Q(S_t, A_t))$

Technicalities for the Q-learning agent

- ▶ How to represent Q -function?
- ▶ What is the value for terminal? $Q(s, \text{Exit})$ or $Q(s, \text{None})$
- ▶ How to drive? Where to drive next? Does it change over the course?

From Q-learning to Q-learning agent

- ▶ Drive the robot and fetch rewards. (s, a, s', R)
- ▶ We know old estimates $Q(s, a)$ (and $Q(s', a')$), if not, initialize.
- ▶ A new trial/sample estimate: $\text{trial} = R_{t+1} + \gamma \max_a Q(S_{t+1}, a)$
- ▶ α update: $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(\text{trial} - Q(S_t, A_t))$

Technicalities for the Q-learning agent

- ▶ How to represent Q-function?
 - ▶ What is the value for terminal? $Q(s, \text{Exit})$ or $Q(s, \text{None})$
 - ▶ How to drive? Where to drive next? Does it change over the course?

From Q-learning to Q-learning agent

- ▶ Drive the robot and fetch rewards. (s, a, s', R)
- ▶ We know old estimates $Q(s, a)$ (and $Q(s', a')$), if not, initialize.
- ▶ A new trial/sample estimate: $\text{trial} = R_{t+1} + \gamma \max_a Q(S_{t+1}, a)$
- ▶ α update: $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(\text{trial} - Q(S_t, A_t))$

Technicalities for the Q-learning agent

- ▶ How to represent Q-function?
- ▶ What is the value for terminal? $Q(s, \text{Exit})$ or $Q(s, \text{None})$
- ▶ How to drive? Where to drive next? Does it change over the course?

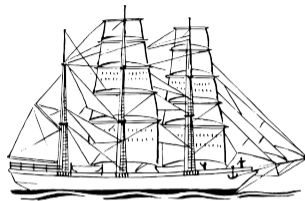
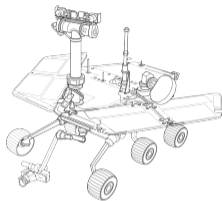
From Q-learning to Q-learning agent

- ▶ Drive the robot and fetch rewards. (s, a, s', R)
- ▶ We know old estimates $Q(s, a)$ (and $Q(s', a')$), if not, initialize.
- ▶ A new trial/sample estimate: $\text{trial} = R_{t+1} + \gamma \max_a Q(S_{t+1}, a)$
- ▶ α update: $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(\text{trial} - Q(S_t, A_t))$

Technicalities for the Q-learning agent

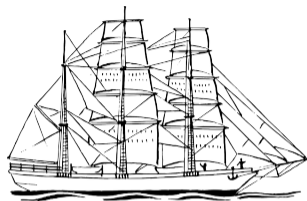
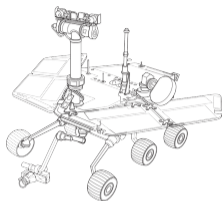
- ▶ How to represent Q-function?
- ▶ What is the value for terminal? $Q(s, \text{Exit})$ or $Q(s, \text{None})$
- ▶ How to drive? Where to drive next? Does it change over the course?

Exploration vs Exploitation



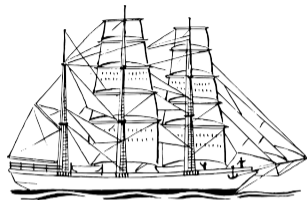
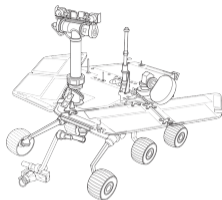
- ▶ Drive the known road or try a new one?
- ▶ Go to the university menza or try a nearby restaurant?
- ▶ Use the SW (operating system) I know or try new one?
- ▶ Go to bussiness or study a demanding program?
- ▶ ...

Exploration vs Exploitation



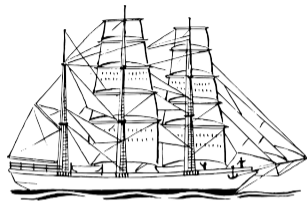
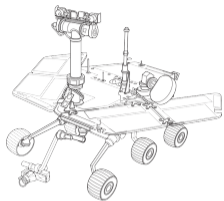
- ▶ Drive the known road or try a new one?
- ▶ Go to the university menza or try a nearby restaurant?
- ▶ Use the SW (operating system) I know or try new one?
- ▶ Go to bussiness or study a demanding program?
- ▶ ...

Exploration vs Exploitation



- ▶ Drive the known road or try a new one?
- ▶ Go to the university menza or try a nearby restaurant?
- ▶ Use the SW (operating system) I know or try new one?
- ▶ Go to bussiness or study a demanding program?
- ▶ ...

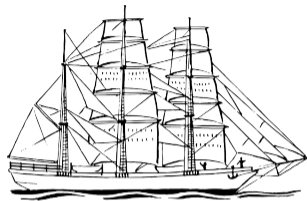
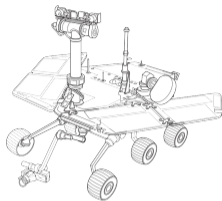
Exploration vs Exploitation



- ▶ Drive the known road or try a new one?
- ▶ Go to the university menza or try a nearby restaurant?
- ▶ Use the SW (operating system) I know or try new one?
- ▶ Go to bussiness or study a demanding program?



Exploration vs Exploitation



- ▶ Drive the known road or try a new one?
- ▶ Go to the university menza or try a nearby restaurant?
- ▶ Use the SW (operating system) I know or try new one?
- ▶ Go to bussiness or study a demanding program?
- ▶ ...

How to explore?

Random (ϵ -greedy):

- ▶ Flip a coin every step.
- ▶ With probability ϵ , act randomly.
- ▶ With probability $1 - \epsilon$, use the policy.

Problems with randomness?

- ▶ Keeps exploring forever.
- ▶ Should we keep ϵ fixed (over learning)?
- ▶ ϵ same everywhere?

How to explore?

Random (ϵ -greedy):

- ▶ Flip a coin every step.
- ▶ With probability ϵ , act randomly.
- ▶ With probability $1 - \epsilon$, use the policy.

Problems with randomness?

- ▶ Keeps exploring forever.
- ▶ Should we keep ϵ fixed (over learning)?
- ▶ ϵ same everywhere?

How to explore?

Random (ϵ -greedy):

- ▶ Flip a coin every step.
- ▶ With probability ϵ , act randomly.
- ▶ With probability $1 - \epsilon$, use the policy.

Problems with randomness?

- ▶ Keeps exploring forever.
- ▶ Should we keep ϵ fixed (over learning)?
- ▶ ϵ same everywhere?

How to explore?

Random (ϵ -greedy):

- ▶ Flip a coin every step.
- ▶ With probability ϵ , act randomly.
- ▶ With probability $1 - \epsilon$, use the policy.

Problems with randomness?

- ▶ Keeps exploring forever.
- ▶ Should we keep ϵ fixed (over learning)?
- ▶ ϵ same everywhere?

How to explore?

Random (ϵ -greedy):

- ▶ Flip a coin every step.
- ▶ With probability ϵ , act randomly.
- ▶ With probability $1 - \epsilon$, use the policy.

Problems with randomness?

- ▶ Keeps exploring forever.
- ▶ Should we keep ϵ fixed (over learning)?
- ▶ ϵ same everywhere?

How to explore?

Random (ϵ -greedy):

- ▶ Flip a coin every step.
- ▶ With probability ϵ , act randomly.
- ▶ With probability $1 - \epsilon$, use the policy.

Problems with randomness?

- ▶ Keeps exploring forever.
- ▶ Should we keep ϵ fixed (over learning)?
- ▶ ϵ same everywhere?

How to explore?

Random (ϵ -greedy):

- ▶ Flip a coin every step.
- ▶ With probability ϵ , act randomly.
- ▶ With probability $1 - \epsilon$, use the policy.

Problems with randomness?

- ▶ Keeps exploring forever.
- ▶ Should we keep ϵ fixed (over learning)?
- ▶ ϵ same everywhere?

How to explore?

Random (ϵ -greedy):

- ▶ Flip a coin every step.
- ▶ With probability ϵ , act randomly.
- ▶ With probability $1 - \epsilon$, use the policy.

Problems with randomness?

- ▶ Keeps exploring forever.
- ▶ Should we keep ϵ fixed (over learning)?
- ▶ ϵ same everywhere?

References

Further reading: Chapter 21 of [2]. More detailed discussion in [3], chapters 5 and 6.

[1] Dan Klein and Pieter Abbeel.

UC Berkeley CS188 Intro to AI – course materials.

<http://ai.berkeley.edu/>.

Used with permission of Pieter Abbeel.

[2] Stuart Russell and Peter Norvig.

Artificial Intelligence: A Modern Approach.

Prentice Hall, 3rd edition, 2010.

<http://aima.cs.berkeley.edu/>.

[3] Richard S. Sutton and Andrew G. Barto.

Reinforcement Learning; an Introduction.

MIT Press, 2nd edition, 2018.

<http://www.incompleteideas.net/book/bookdraft2018jan1.pdf>.