

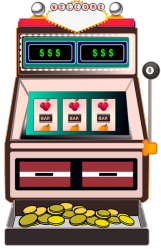
# Reinforcement learning

Tomáš Svoboda, Petr Pošík

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

April 4, 2024

# (Multi-armed) Bandits



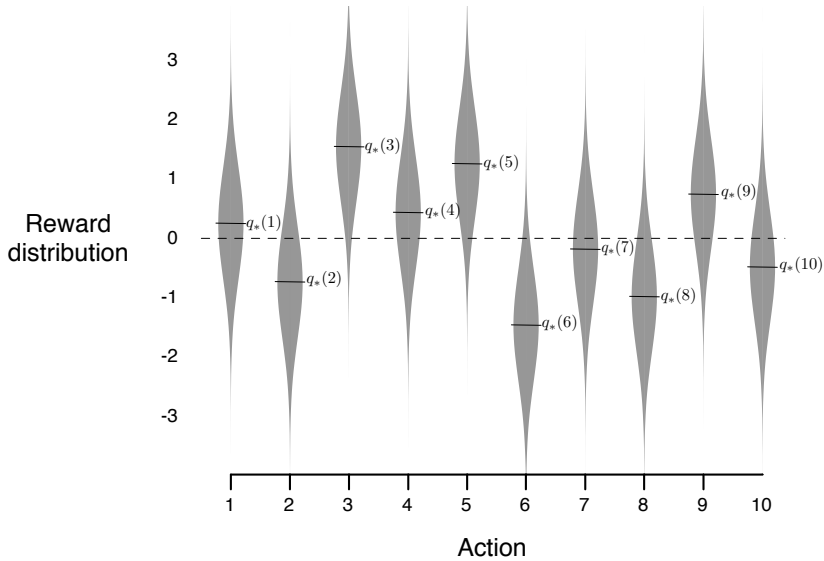
Think about not one but 10 arms you may choose to pull.

$p(s'|s, a)$  and  $r(s, a, s')$  not known!

---

Notes

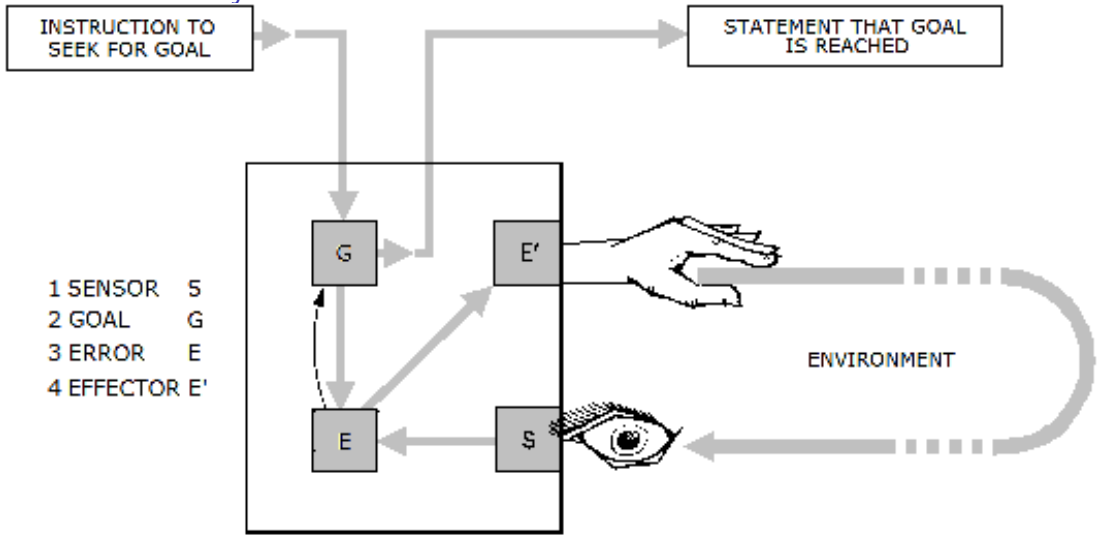
# 10 armed bandit, what arm to pull?



## Notes

- 10 different arms
- action pulling  $k$ -th arm
- value of the action, i.e.  $q(a)$  is stochastic (Gaussian around  $q^*(a)$ )
- Playing (pulling) many times, what is the policy?

# Goal-directed system



**A SIMPLE GOAL-DIRECTED SYSTEM**

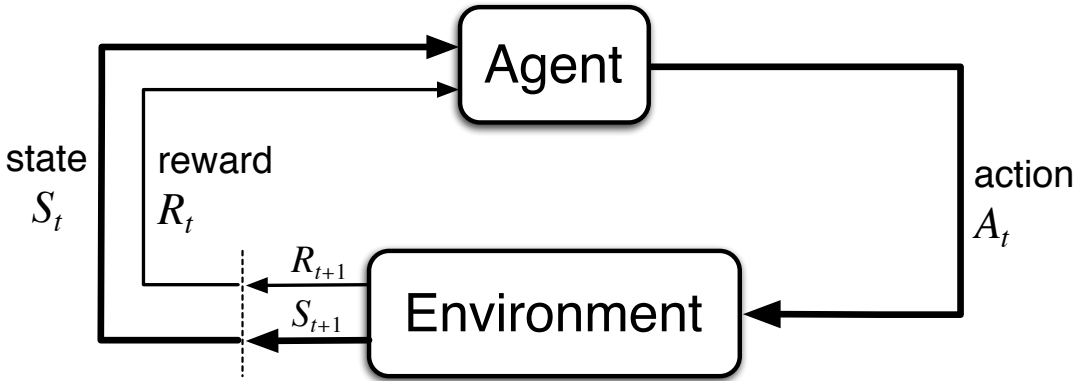
1

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

4 / 38

Notes

# Reinforcement Learning - performing actions, learning from rewards



2

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

<sup>2</sup>Scheme from [4]

5 / 38

## Autonomous Flipper Control with Safety Constraints

Martin Pecka, Vojtěch Šalanský,  
Karel Zimmermann, Tomáš Svoboda

experiments utilizing  
Constrained Relative Entropy Policy Search

**Video: Learning safe policies<sup>3</sup>**

<sup>3</sup>M. Pecka, V. Salansky, K. Zimmermann, T. Svoboda. Autonomous flipper control with safety constraints. In Intelligent Robots and Systems (IROS), 2016, [https://youtu.be/\\_oUMbBtoRcs](https://youtu.be/_oUMbBtoRcs)

6 / 38

### Notes

Policy search is a more advanced topic, only touched by this course. Later in master programme. Reinforcement learning beating humans in playing Atari games: <https://deepmind.google/discover/blog/agent57-outperforming-the-human-atari-benchmark/>

# 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.

---

## Notes

For MDPs, we know  $p, r$  for all possible states and actions.

# (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  $p(s' | s, a)$ .
- ▶ Discover (by observation) each  $r(s, a, s')$  when experienced.

Solve the learned MDP.

---

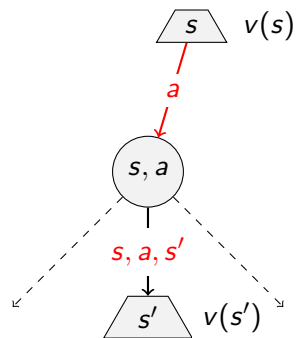
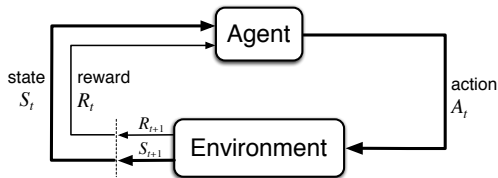
## Notes

- Where to start?
- When does it end?
- How long does it take?
- When to stop (the learning phase)?



# 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 the real world, it is different (going up, down, ...)



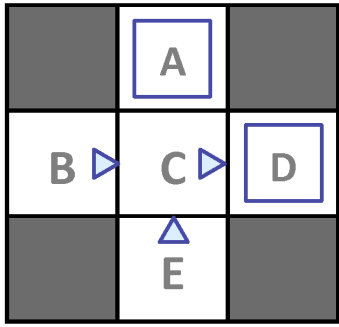
In ai-gym env.step(action) returns  $s', r(s, \text{action}, s')$ .

## Notes

In ai-gym env.step(action) returns  $s', r(s, \text{action}, s'), \dots$ . It is defined by the environment (robot simulator, system, ...) not by the (algorithms)

# Model-based learning: Grid example

## Input Policy $\pi$



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

<sup>4</sup>Figure from [1]

## Notes

## Learned Model

$$\hat{T}(s, a, s')$$

T(B, east, C) = 1.00  
T(C, east, D) = 0.75  
T(C, east, A) = 0.25  
...

$$\hat{R}(s, a, s')$$

R(B, east, C) = -1  
R(C, east, D) = -1  
R(D, exit, x) = +10  
...

# Learning transition model

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

## 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

---

### Notes

(C, east) combination performed 4 times, 3 times landed in D, once in A. Hence,  $\hat{p}(D \mid C, \text{east}) = 0.75$ .

# Learning reward function

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

## 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

---

### Notes

Whenever (C, east, D) performed, received reward was  $-1$ . Hence,  $\hat{r}(C, \text{east}, D) = -1$ .

# 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  $P(A = a)$ . Instead, we collect  $N$  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

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

13 / 38

---

## Notes

Just to avoid confusion. There are many more samples than possible ages (positive integer). Think about  $N \gg 100$ .

- Model based – eventually, we learn the correct model.
- Model free – no need for weighting; this is achieved through the frequencies of different ages within the samples (most frequent and hence most probable ages simply come up many times).

# Model-free learning

# Passive learning (evaluating given policy)

- ▶ **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  $v^\pi(s)$

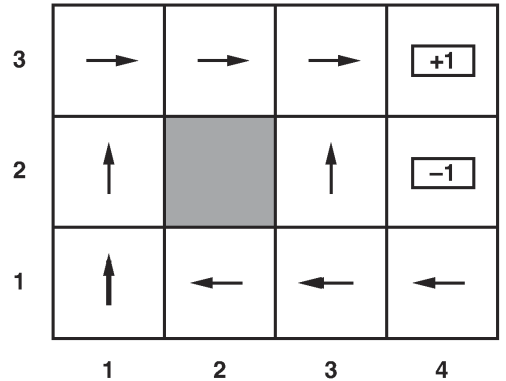


Image from [2]

---

## Notes

Executing policies - training, then learning from the observations. We want to do the policy evaluation but the necessary model is not known.

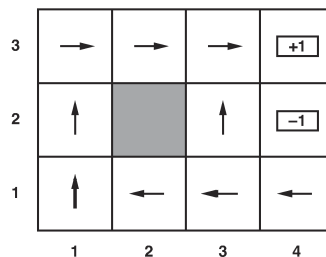
The word passive means we just follow a prescribed policy  $\pi(s)$ .

# Direct evaluation from episodes

Value of  $s$  for  $\pi$  – expected sum of discounted rewards – expected return

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

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



$(1, 1) \xrightarrow{-0.04} (1, 2) \xrightarrow{-0.04} (1, 3) \xrightarrow{-0.04} (1, 2) \xrightarrow{-0.04} (1, 3) \xrightarrow{-0.04} (2, 3) \xrightarrow{-0.04} (3, 3) \xrightarrow{-0.04} (4, 3) \xrightarrow{+1}$   
 $(1, 1) \xrightarrow{-0.04} (1, 2) \xrightarrow{-0.04} (1, 3) \xrightarrow{-0.04} (2, 3) \xrightarrow{-0.04} (3, 3) \xrightarrow{-0.04} (3, 2) \xrightarrow{-0.04} (3, 3) \xrightarrow{-0.04} (4, 3) \xrightarrow{+1}$   
 $(1, 1) \xrightarrow{-0.04} (2, 1) \xrightarrow{-0.04} (3, 1) \xrightarrow{-0.04} (3, 2) \xrightarrow{-0.04} (4, 2) \xrightarrow{-1}$

## Notes

- Act according to the policy.
- When visiting a state, remember what the sum of discounted rewards (returns) turned out to be.
- Compute average of the returns.
- Each trial episode provides a sample of  $v^\pi$ .

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



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

$(1, 1)_{-0.04} \rightsquigarrow (1, 2)_{-0.04} \rightsquigarrow (1, 3)_{-0.04} \rightsquigarrow (1, 2)_{-0.04} \rightsquigarrow (1, 3)_{-0.04} \rightsquigarrow (2, 3)_{-0.04} \rightsquigarrow (3, 3)_{-0.04} \rightsquigarrow (4, 3)_{+1}$   
 $(1, 1)_{-0.04} \rightsquigarrow (1, 2)_{-0.04} \rightsquigarrow (1, 3)_{-0.04} \rightsquigarrow (2, 3)_{-0.04} \rightsquigarrow (3, 3)_{-0.04} \rightsquigarrow (3, 2)_{-0.04} \rightsquigarrow (3, 3)_{-0.04} \rightsquigarrow (4, 3)_{+1}$   
 $(1, 1)_{-0.04} \rightsquigarrow (2, 1)_{-0.04} \rightsquigarrow (3, 1)_{-0.04} \rightsquigarrow (3, 2)_{-0.04} \rightsquigarrow (4, 2)_{-1}$  .

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

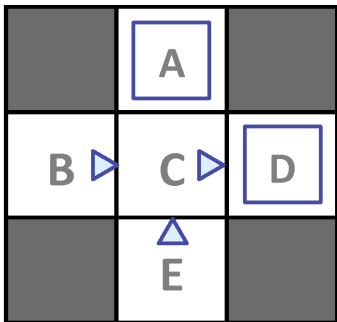
---

### Notes

- Not visited during the first episode.
- Visited once in the second, gathered return  $G = -0.04 - 0.04 + 1 = 0.92$ .
- Visited once in the third, return  $G = -0.04 - 1 = -1.04$ .
- Value, average return is  $(0.92 - 1.04)/2 = -0.06$ .

# Direct evaluation: Grid example

## Input Policy $\pi$



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

## Notes

	-10 A	
+8 B	+4 C	+10 D
	-2 E	

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

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

Let  $M$  be the number of recorded episodes.

Let  $N$  be the number of samples used to compute the averages.

What is the relation of  $M$  and  $N$ ?

- A  $N = M$
- B  $N \leq M$
- C  $N \geq M$
- D  $N$  has no relation to  $M$

## 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

### Notes

- Episode 1,  $G = -1 + 10 = 9$
- Episode 2,  $G = -1 + 10 = 9$
- Episode 3,  $G = -1 + 10 = 9$
- Episode 4,  $G = -1 - 10 = -11$
- Average return  $v(C) = (9 + 9 + 9 - 11)/4 = 4$

For first-visit variant, B is correct. For every-visit variant, D is correct.

$N$  can be lower than  $M$  (state does not have to be attended in every episode). For every-visit variant,  $N$  can be higher than  $M$  (a state can be visited several times in one episode).

## Direct evaluation algorithm (every-visit/first-visit version)

$(1, 1) \xrightarrow{-0.04} (1, 2) \xrightarrow{-0.04} (1, 3) \xrightarrow{-0.04} (1, 2) \xrightarrow{-0.04} (1, 3) \xrightarrow{-0.04} (2, 3) \xrightarrow{-0.04} (3, 3) \xrightarrow{-0.04} (4, 3) +1$   
 $(1, 1) \xrightarrow{-0.04} (1, 2) \xrightarrow{-0.04} (1, 3) \xrightarrow{-0.04} (2, 3) \xrightarrow{-0.04} (3, 3) \xrightarrow{-0.04} (3, 2) \xrightarrow{-0.04} (3, 3) \xrightarrow{-0.04} (4, 3) +1$   
 $(1, 1) \xrightarrow{-0.04} (2, 1) \xrightarrow{-0.04} (3, 1) \xrightarrow{-0.04} (3, 2) \xrightarrow{-0.04} (4, 2) -1 .$

Input: a policy  $\pi$  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  $\pi$ :  $S_0, A_0, R_1, S_1, A_1, R_2, \dots, S_{T-1}, A_{T-1}, R_T$

$G \leftarrow 0$

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

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

Append  $G$  to  $Returns(S_t)$

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

If  $S_t$  does not appear in  $S_0, S_1, \dots, S_{t-1}$ : // Use the return for the first visit only

Append  $G$  to  $Returns(S_t)$

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

20 / 38

---

### Notes

The algorithm can be easily expanded to  $Q(S_t, A_t)$ . Instead of visiting  $S_t$  we consider visiting of a pair  $S_t, A_t$ .

# Direct evaluation: analysis

The good:

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

The bad:

$(1, 1) \xrightarrow{-0.04} (1, 2) \xrightarrow{-0.04} (1, 3) \xrightarrow{-0.04} (1, 2) \xrightarrow{-0.04} (1, 3) \xrightarrow{-0.04} (2, 3) \xrightarrow{-0.04} (3, 3) \xrightarrow{-0.04} (4, 3) +1$   
 $(1, 1) \xrightarrow{-0.04} (1, 2) \xrightarrow{-0.04} (1, 3) \xrightarrow{-0.04} (2, 3) \xrightarrow{-0.04} (3, 3) \xrightarrow{-0.04} (3, 2) \xrightarrow{-0.04} (3, 3) \xrightarrow{-0.04} (4, 3) +1$   
 $(1, 1) \xrightarrow{-0.04} (2, 1) \xrightarrow{-0.04} (3, 1) \xrightarrow{-0.04} (3, 2) \xrightarrow{-0.04} (4, 2) -1 .$

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

---

## Notes

In second trial, we visit  $(3, 2)$  for the first time. We already know that the successor  $(3, 3)$  has probably a high value but the method does not use until the end of the trial episode.

Before updating  $V(s)$  we have to wait until the training episode ends.

## (on-line) Policy evaluation?

In MDP, we did:

- ▶ Initialize the values:  $V_0^\pi(s) = 0$
- ▶ In each iteration, replace  $V$  with a one-step-look-ahead:  
$$V_{k+1}^\pi(s) \leftarrow \sum_{s'} p(s' | s, \pi(s)) [r(s, \pi(s), s') + \gamma V_k^\pi(s')]$$

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 stop, try, try, ..., and average?

Trials at time  $t$ .  $\pi(S_t) \rightarrow A_t$ , repeat  $A_t$ .

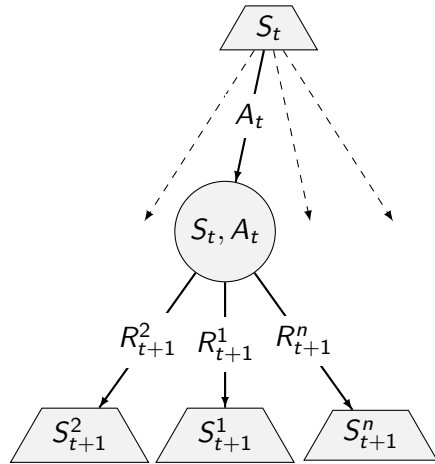
$$\text{trial}^1 = R_{t+1}^1 + \gamma V(S_{t+1}^1)$$

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

$$\vdots = \vdots$$

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

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



Problem: We cannot re-set to  $S_t$  easily.

23 / 38

### Notes

It looks promising. Unfortunately, we cannot do it that way. After an action, the robot is in a next state and cannot go back to the very same state where it was before. Energy was consumed and some actions may be irreversible; think about falling into a hole. We have to utilize the  $s, a, s'$  experience anytime when performed/visited.

# Temporal-difference value learning

$(1, 1) \xrightarrow{-0.04} (1, 2) \xrightarrow{-0.04} (1, 3) \xrightarrow{-0.04} (1, 2) \xrightarrow{-0.04} (1, 3) \xrightarrow{-0.04} (2, 3) \xrightarrow{-0.04} (3, 3) \xrightarrow{-0.04} (4, 3) \xrightarrow{+1}$   
 $(1, 1) \xrightarrow{-0.04} (1, 2) \xrightarrow{-0.04} (1, 3) \xrightarrow{-0.04} (2, 3) \xrightarrow{-0.04} (3, 3) \xrightarrow{-0.04} (3, 2) \xrightarrow{-0.04} (3, 3) \xrightarrow{-0.04} (4, 3) \xrightarrow{+1}$   
 $(1, 1) \xrightarrow{-0.04} (2, 1) \xrightarrow{-0.04} (3, 1) \xrightarrow{-0.04} (3, 2) \xrightarrow{-0.04} (4, 2) \xrightarrow{-1}$

$\gamma = 1$

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 ( $\alpha \times$  difference):  $V(S_t) \leftarrow V(S_t) + \alpha \left( [R_{t+1} + \gamma V(S_{t+1})] - V(S_t) \right)$
- ▶  $\alpha$  is the learning rate.
- ▶  $V(S_t) \leftarrow (1 - \alpha)V(S_t) + \alpha(\text{new sample})$

24 / 38

---

## Notes

Trial episode: acting, observing, until it stops (in a terminal state or by a limit).

We visit  $S(1, 3)$  twice during the first episode. Its value estimate is the average of two returns.

Note the main difference. In *Direct evaluation*, we had to wait until the end of the episode, compute  $G_t$  for each  $t$  on the way, and then we update  $V(S_t)$ . We can do it  $\alpha$  incrementally

$$V(S_t) \leftarrow V(S_t) + \alpha \left( G_t - V(S_t) \right)$$

In *TD learning*, we update as we go.



# Exponential moving average

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

What does it remember about the past? Try to derive:

$$\bar{x}_n = f(\alpha, x_n, x_{n-1}, x_{n-2}, x_{n-3}, \dots)$$

---

## Notes

Recursively inserting we end up with

$$\bar{x}_n = \alpha \left[ x_n + (1 - \alpha)x_{n-1} + (1 - \alpha)^2 x_{n-2} + \dots \right]$$

We already know the sum of geometric series for  $r < 1$

$$1 + r + r^2 + r^3 + \dots = \frac{1}{1 - r}$$

Putting  $r = 1 - \alpha$ , we see that

$$\frac{1}{\alpha} = 1 + (1 - \alpha) + (1 - \alpha)^2 + \dots$$

And hence:

$$\bar{x}_n = \frac{x_n + (1 - \alpha)x_{n-1} + (1 - \alpha)^2 x_{n-2} + \dots}{1 + (1 - \alpha) + (1 - \alpha)^2 + (1 - \alpha)^3 + \dots}$$

a weighted average that exponentially forgets about the past.

# Example: TD Value learning

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

	<span style="color: blue;">A</span> 0	
<span style="color: blue;">B</span> 0	<span style="color: blue;">C</span> 0	<span style="color: blue;">D</span> 8
	<span style="color: blue;">E</span> 0	

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

## Notes

### States

	A	
B	C	D
	E	

Assume:  $\gamma = 1, \alpha = 1/2$

### Observed Transitions

B, east, C, -2
C, east, D, -2

	0	
0	0	8
	0	

	0	
-1	0	8
	0	

	0	
-1	3	8
	0	

$$V^\pi(s) \leftarrow (1 - \alpha)V^\pi(s) + \alpha [R(s, \pi(s), s') + \gamma V^\pi(s')]$$

## Temporal difference value learning: algorithm

Input: the policy  $\pi$  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  $\pi$  for  $S$

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

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

$S \leftarrow S'$

    until  $S$  is terminal

# What is wrong with the temporal difference Value learning?

**The Good:** Model-free value learning by mimicking Bellman updates.

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

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

$$\blacktriangleright \pi(s) = \arg \max_a Q(s, a)$$

---

## Notes

Learn Q-values, not V-values, and make the action selection model-free too!

# Q-learning

---

## Notes

So far we walked as prescribed by a  $\pi(s)$  because we did not know how to act better.

## 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  $\pi$ )

- ▶ Start:  $Q_0(s, a) = 0$
- ▶ In each step update  $Q$  by looking one step ahead:  
$$Q_{k+1}(s, a) \leftarrow \sum_{s'} p(s' | s, a) \left[ r(s, a, s') + \gamma \max_{a'} Q_k(s', a') \right]$$

---

### Notes

Draw the  $(s)$ - $(s,a)$ - $(s')$ - $(s',a')$  tree. It will be also handy when discussing exploration vs. exploitation – where to drive next.

# Q-learning (one episode of)

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

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  $t$

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

- ▶  $\alpha$  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.

---

## Notes

There are alternatives how to compute the trial value. SARSA method takes  $Q(S_{t+1}, A_{t+1})$  directly, not the max. More next week.

# Q-learning: algorithm (repeating episodes, until terminal or exhausted)

step size  $0 < \alpha \leq 1$

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

**repeat** episodes:

    initialize  $S$

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

        choose  $A$  from  $\mathcal{A}(S)$

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

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

$S \leftarrow S'$

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

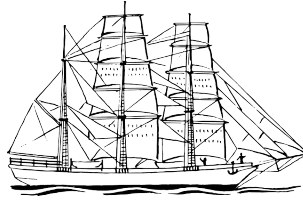
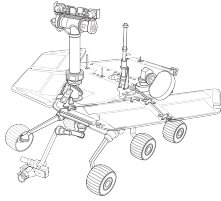
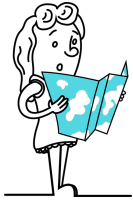
- ▶ How to represent the Q-function?
- ▶ What is the value for terminal?  $Q(s, \text{Exit})$  or  $Q(s, \text{None})$
- ▶ How to drive? Where to drive next? Does it change over the course?

---

### Notes

Q-function for a discrete, finite problem? But what about continuous space or discrete but a very large one?  
Use the  $(s)-(s,a)-(s')-(s',a')$  tree to discuss the next-action selection.

# 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 a new one?
- ▶ Go to bussiness or study a demanding program?
- ▶ ...

# How to explore?

## Random ( $\epsilon$ -greedy):

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

## Problems with randomness?

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

---

## Notes

- We can think about lowering  $\epsilon$  as the learning progresses.
- Favor unexplored states - be optimistic - exploration functions -  $f(u, n) = u + k/n$ , where  $u$  is the value estimated, and  $n$  is the visit count, and  $k$  is the training/simulation episode.

# What we have learned

- ▶ Agent/robot may learn by acting and getting rewards
- ▶ Model based vs. model-free methods
- ▶ Direct learning vs. temporal-difference learning
- ▶ From learning state values to Q-learning

# References I

Further reading: Chapter 21 of [2] (chapter 23 of [3]). More detailed discussion in [4], 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] Stuart Russell and Peter Norvig.  
*Artificial Intelligence: A Modern Approach*.  
Prentice Hall, 4th edition, 2021.

## References II

- [4] Richard S. Sutton and Andrew G. Barto.  
*Reinforcement Learning; an Introduction*.  
MIT Press, 2nd edition, 2018.  
<http://www.incompleteideas.net/book/the-book-2nd.html>.