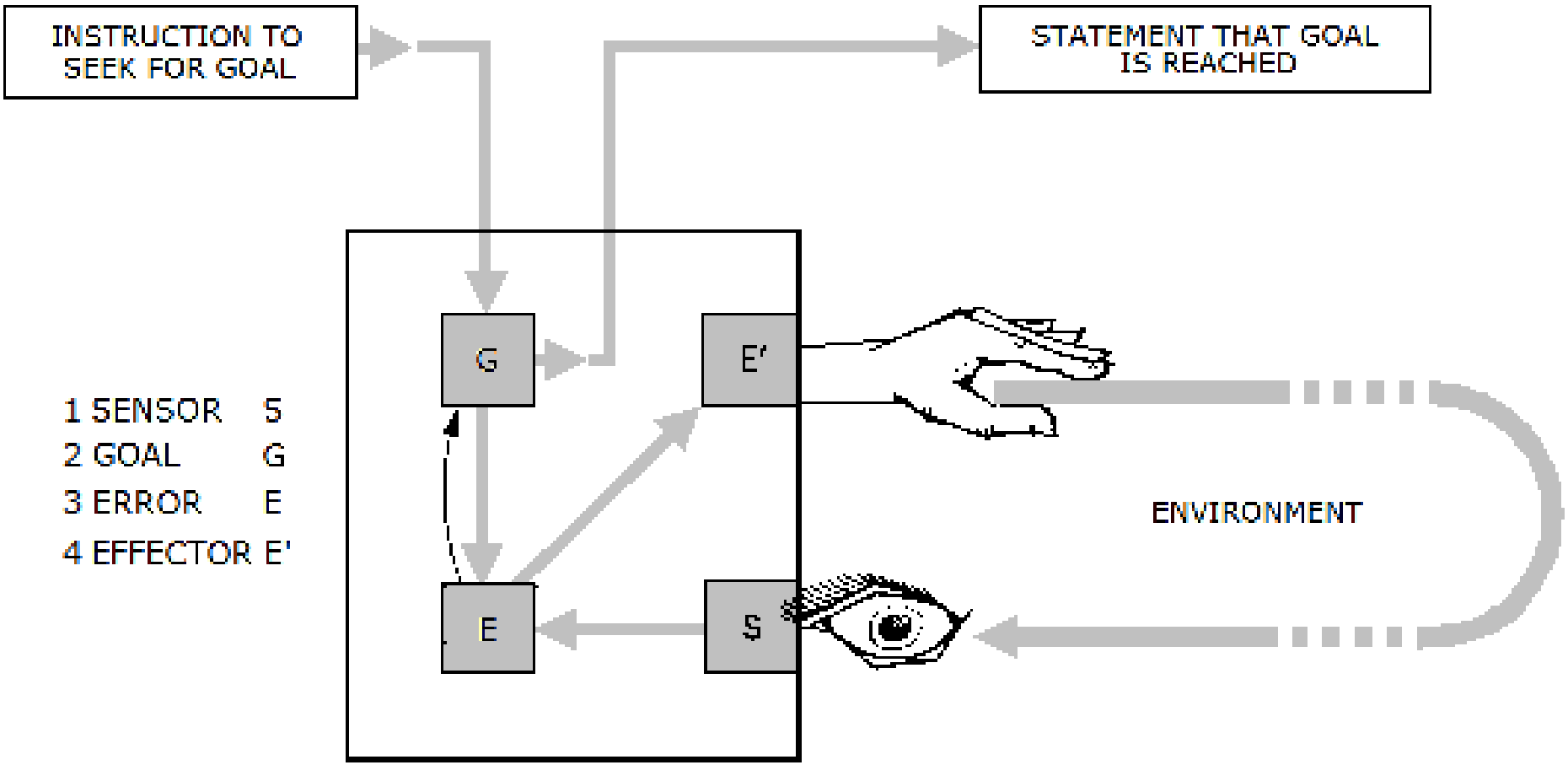
Reinforcement learning

Tomáš Svoboda

Department of Cybernetics, Vision for Robotics and Autonomous Systems,
Center for Machine Perception (CMP)

April 11, 2018

Goal-directed system

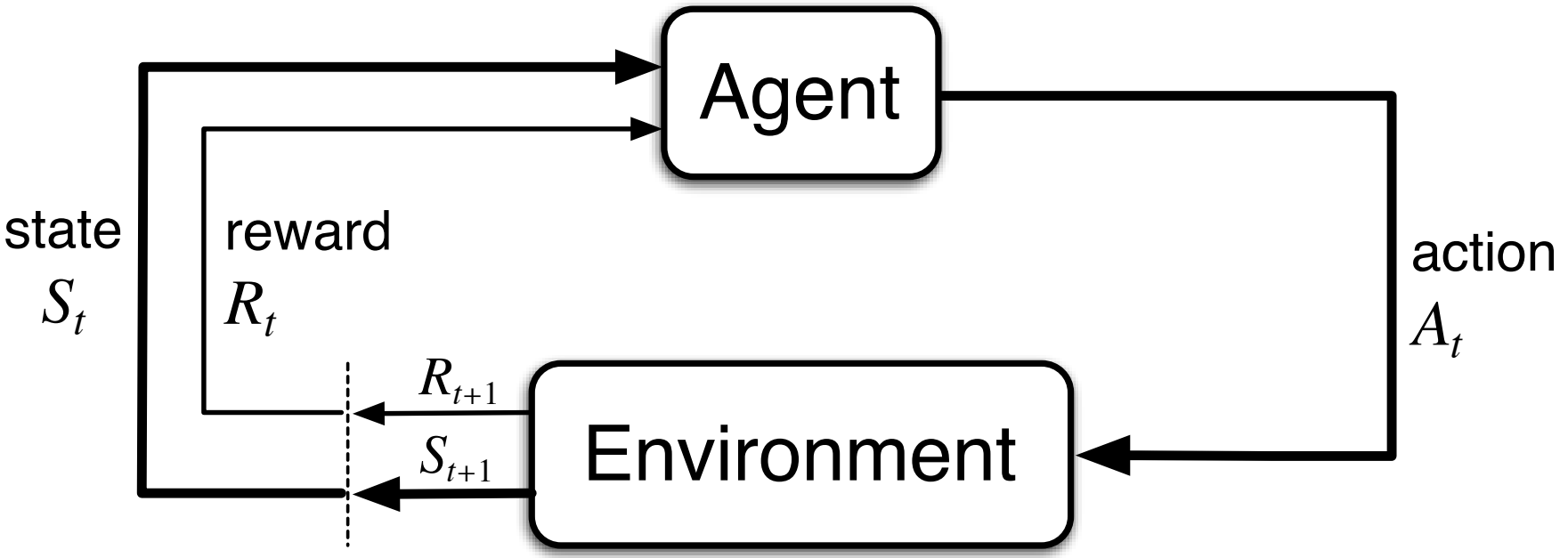


A SIMPLE GOAL-DIRECTED SYSTEM

1

[1] Figure from http://www.cybsoc.org/gcyb.htm

Reinforcement Learning



[2]

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

[2] Scheme from [3]

Examples

Video: Learning safe policies[3]

[3] M. 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 $s \in S$ (map)
- A set of actions per state. $a \in A$
- A transition model $T(s, a, s')$ or $P(s'|s, a)$ (robot)
- A reward function $R(s)$ (map, robot)

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

On-line problem:

- T and 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 were correct.

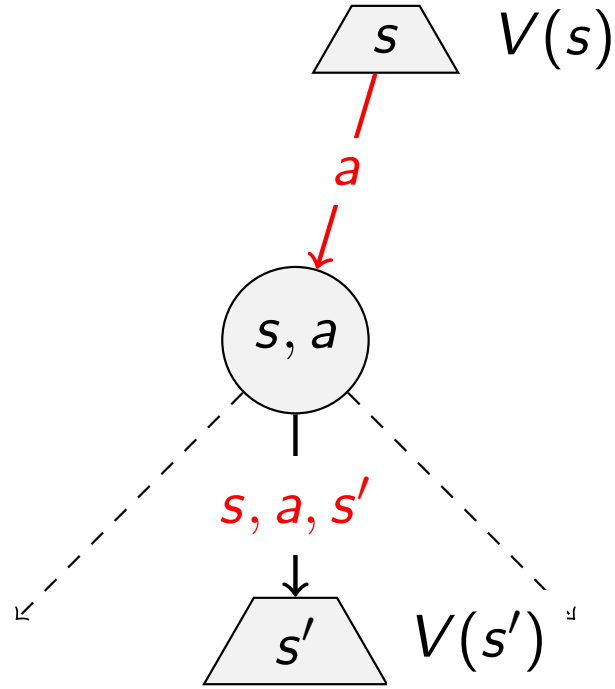
Learning MDP model:

- Try s, a, observe s', count s, a, s'.
- Normalize to get and estimate of $T(s, a, s')$.
- Discover each $R(s, a, s')$ when experience.

Solve the learned MDP.

Reward function R

- $R(s)$ - reward for dwelling in the state
- $R(s, a, s')$ - reward for going from s to 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, ...)



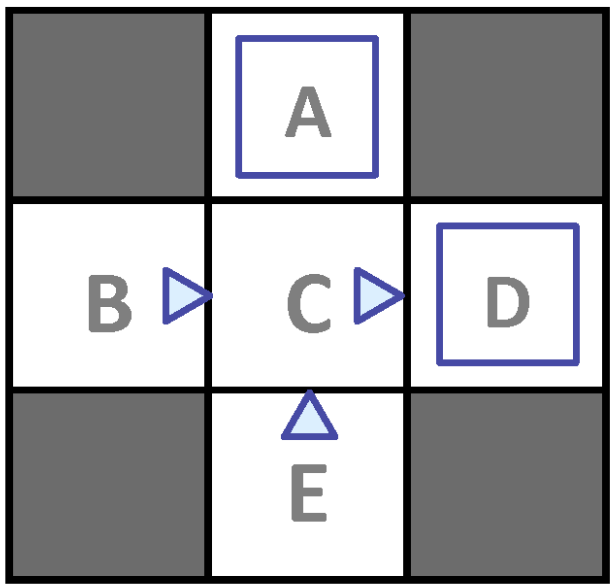
$$V(s) = R(s) + \gamma \max_a \sum_{s'} T(s, a, s') V(s')$$

$$V(s) = \max_a \sum_{s'} T(s, a, s') \left[R(s, a, s') + \gamma V(s') \right]$$

$$V(s) = \max_a \sum_{s'} P(s'|s, a) \left[R(s, a, s') + \gamma V(s') \right]$$

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

[4] Figure from [1]

Example: Expected age

Model-free learning

Passive learning

- **Input:** a fixed policy $\pi(s)$
- We want to know how good it is.
- R, T not known.
- Execute policy ...
- and learn on the way.
- **Goal:** learn the state values $V^{\pi}(s)$

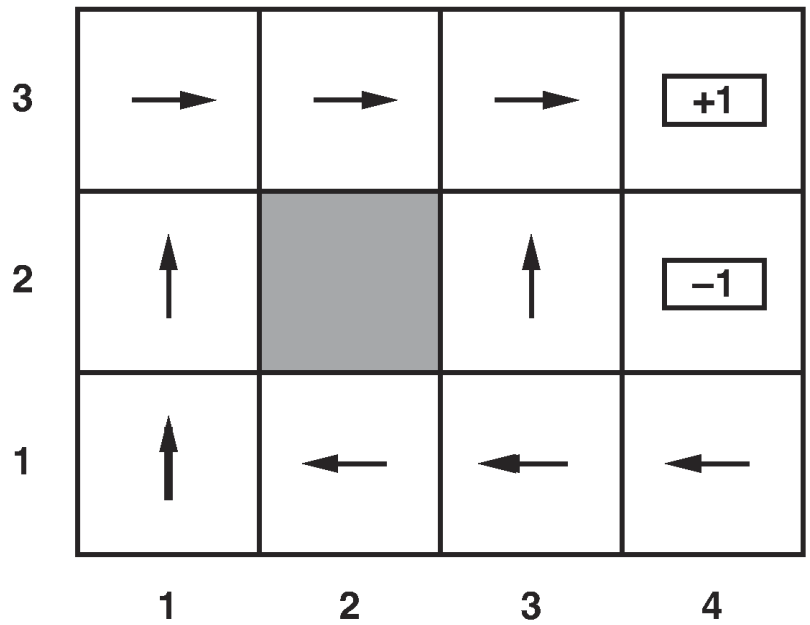
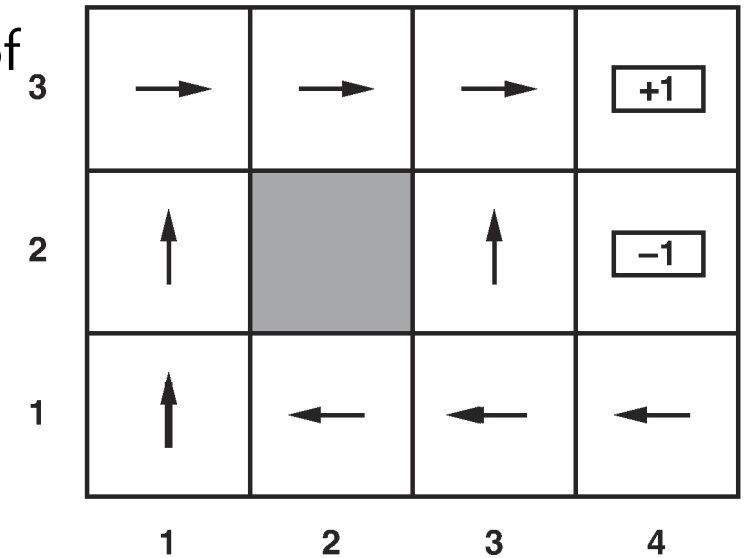


Image from [2]

Direct evaluation

Value of s for π – expected sum of discounted rewards

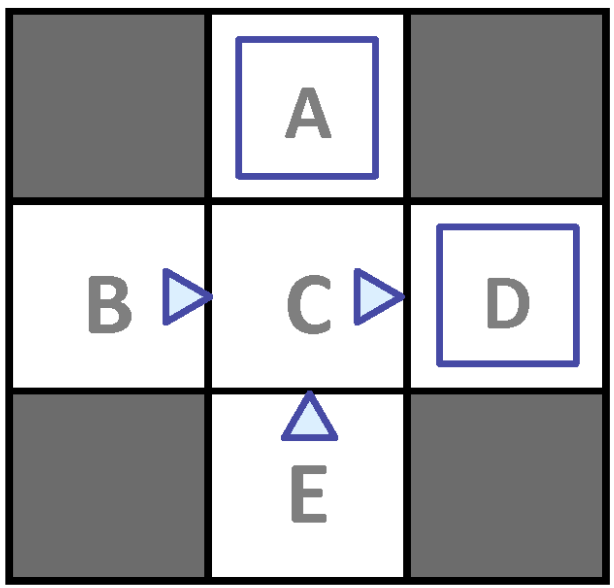
$$V^{\pi}(s) = \mathrm{E}\left[\sum_{t=0}^{\infty} \gamma^t R(S_t)\right]$$



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

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: analysis

The good:

- Simple, easy to understand and implement.
- Does not need T, R and eventually it computes the true V^π.

The bad:

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

- Each state value learned in isolation.
- State values are not independent
- $V^\pi(s) = R(s) + \gamma \sum_{s'} P(s'|s, \pi(s)) V^\pi(s')$
- $V^\pi(s) = \sum_{s'} T(s, \pi(s), s') \left[R(s, \pi(s), s') + \gamma V^\pi(s') \right]$

Policy evaluation?

In each round, replace V with a one-step-look-ahead
$V_0^\pi(s) = 0$
$V_{k+1}^\pi(s) \leftarrow \sum_{s'} T(s, \pi(s), s') \left[R(s, \pi(s), s') + \gamma V_k^\pi(s') \right]$

Problem: both $T(s, \pi(s), s')$ and $R(s, \pi(s), s')$ unknown!

Use samples for evaluating policy?

MDP (T, R known) : Update V estimate by a weighted average:
$V^{\pi}_{k+1}(s) \leftarrow \sum_{s'} T(s, \pi(s), s') \big[R(s, \pi(s), s') + \gamma \, V^{\pi}_{k}(s') \big]$

What about try and average?

$$
\begin{aligned}
\text{trial}_1 &= R(s, \pi(s), s'_1) + \gamma \, V^{\pi}_{k}(s'_1) \\
\text{trial}_2 &= R(s, \pi(s), s'_2) + \gamma \, V^{\pi}_{k}(s'_2) \\
\vdots &= \vdots \\
\text{trial}_n &= R(s, \pi(s), s'_n) + \gamma \, V^{\pi}_{k}(s'_n)
\end{aligned}
$$

$$
V^{\pi}_{k+1}(s) \leftarrow \frac{1}{n} \sum_i \text{trial}_i
$$

Temporal-difference value learning

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

$\gamma = 1$
From first trial (episode): $V(2,3) = 0.92$, $V(1,3) = 0.84, \ldots$
In second episod, going from $(1,3)$ to $(2,3)$ with reward -0.04, hence:

$$V(1,3) = R((1,3), \pi(1,3), (2,3)) + V(2,3) = -0.04 + 0.92 = 0.88$$

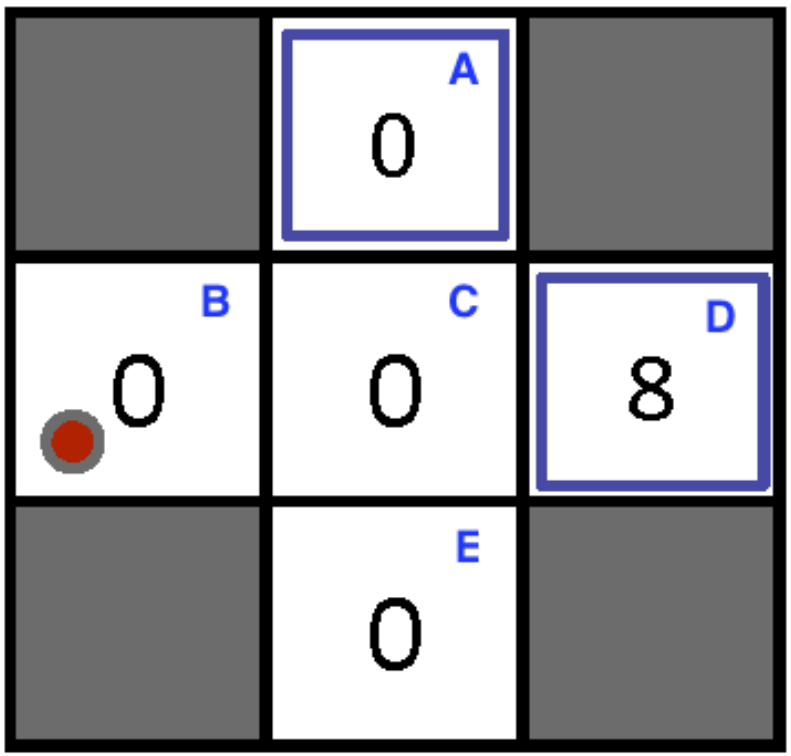
- First estimate 0.84 is a bit lower than 0.88. $V(s)$ is different than $R(s, \pi(s), s') + \gamma V(s')$
- Update: $V(s) \leftarrow V(s) + \alpha(R(s, \pi(s), s') + \gamma V(s') - V(s))$
- α is the learning rate.
- $V_{k+1}(s) \leftarrow (1-\alpha)V_k(s) + \alpha\,(\text{new sample})$

Exponential moving average

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

Example: TD Value learning

$$V(s) \leftarrow V(s) + \alpha(R(s, \pi(s), s') + \gamma V(s') - V(s))$$



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

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?

- $\pi(s) = \arg\max_a \sum_{s'} T(s, a, s') \left[R(s, a, s') + \gamma V(s')\right]$

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:
  $V_{k+1}(s) \leftarrow \max_a \sum_{s'} T(s, a, s') \left[R(s, a, s') + \gamma V_k(s')\right]$

Q values more useful (think about updating π)

- Start: $Q_0(s, a) = 0$
- In each step update Q by looking one step ahead:
  $$Q_{k+1}(s, a) \leftarrow \sum_{s'} T(s, a, s') \left[R(s, a, s') + \gamma \max_{a'} Q_k(s', a')\right]$$

Q-learning

$$Q_{k+1}(s,a) \leftarrow \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma \max_{a'} Q_k(s',a') \right]$$

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

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

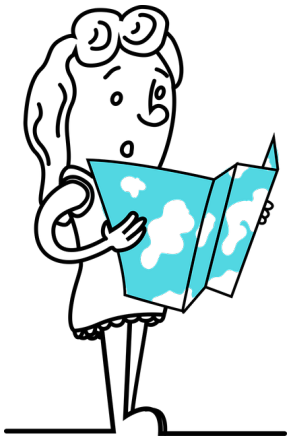
From Q-learning to Q-learning agent

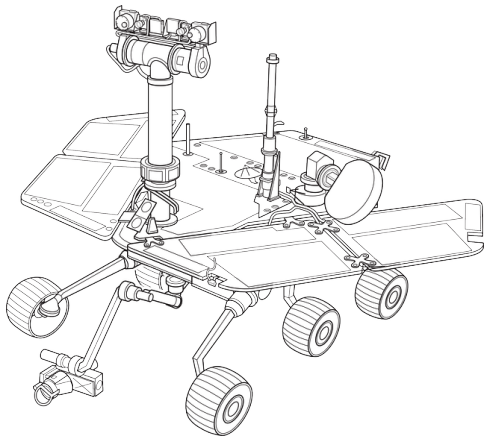
- Drive the robot and fetch: $s, a, s', R(s, a, s')$
- We know old estimates $Q(s, a)$ (and $Q(s', a')$), if not, initialize.
- A new trial/sample estimate: $\text{trial} = R(s, a, s') + \gamma \max_{a'} Q_i(s', a')$
- α update: $Q(s, a) \leftarrow Q(s, a) + \alpha(\text{trial} - Q(s, a))$

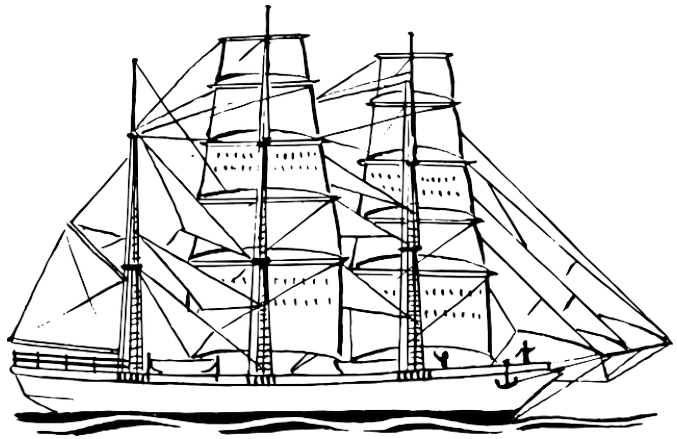
Technicalities for the Q-learning agent

- 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?
- ...

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] with slightly different notation, though.

[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.