Reinforcement Learning

Part I –Definitions

Nicolas P. Rougier
INRIA Bordeaux Sud-Ouest
Institute of Neurodegenerative Diseases

3rd Latin America Summer School in Computational Neuroscience
13-31 January 2014, Valparaiso, Chile

Learning

Supervised learning

$\rightarrow$ Correct answer is given by the supervisor

Ex: 1+2 $\rightarrow$ 4 : No ! Right answer was 3

Ex: 1+3 $\rightarrow$ 4 : Yes ! Right answer is 4

Unsupervised learning

$\rightarrow$ Nothing is said at all

Ex: 1+2 $\rightarrow$ potatoes : If you say so

Ex: 1+3 $\rightarrow$ $\sqrt{\text{table}}$: No really, I don't care

Reinforcement learning

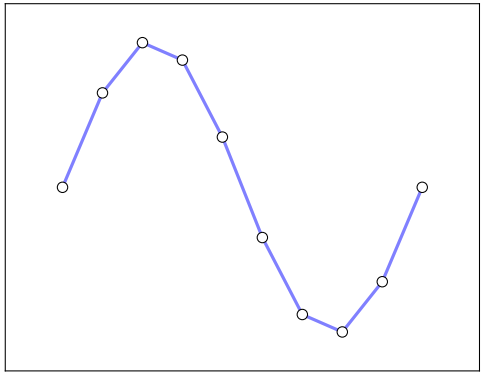
$\rightarrow$ Correctness of answer is given by the environment

Ex: 1+2 $\rightarrow$ 4 : No ! (-1)

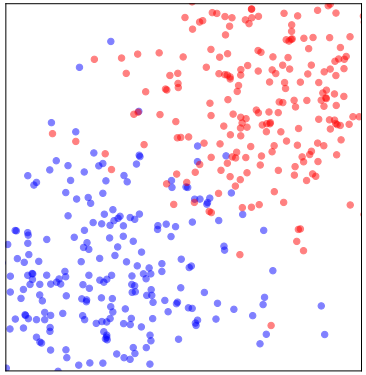
Ex: 1+2 $\rightarrow$ 3 : Yes ! (+1)

Learning

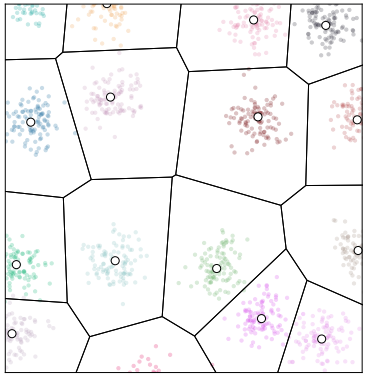
Supervised, unsupervised or reinforcement?



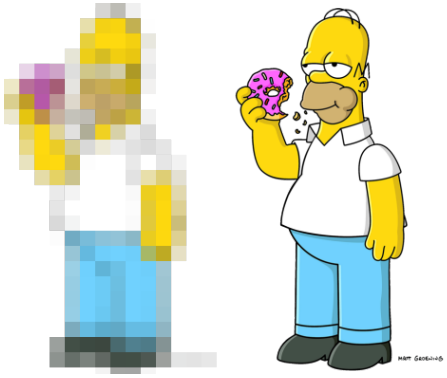
Generalization



Discrimination



Clustering



Memorization

Control

Reinforcement Learning

A tentative definition

"Of several responses made to the same situation, those which are accompanied or closely followed by satisfaction to the animal will, other things being equal, be more firmly connected with the situation, so that, when it recurs, they will be more likely to recur; those which are accompanied or closely followed by discomfort to the animal will, other things being equal, have their connections with that situation weakened, so that, when it recurs, they will be less likely to occur. The greater the satisfaction or discomfort, the greater the strengthening or weakening of the bond"

Edward L. Thorndike, *Animal Intelligence*, 1911

Reinforcement Learning is learning what to do – how to map situations to actions – so as to maximize a numerical reward signal. The learner is not told which actions to take, as in most forms of machine learning, but instead must discover which actions yield the most reward by trying them.

R.S.Sutton & A.G.Barto, *Reinforcement Learning*, 1998

Markov Decision Process (MDP)

Definition

A Markov Decision Process (MDP) is a tuple (S, A, T, R) where:

- S is a finite a set of states
- A is a finite a set of actions
- $T : S \times A \times S \to [0, 1]$ is the transition function
  $T(s, a, s')$ is the probability that action a in state s will lead to state s'
- $R : S \times A \times S \to \mathbb{R}$ is the reward function
  $R(s, a, s')$ is the immediate reward received after transition to s' from s.

A Markov Decision Process (MDP) is the description of a problem.
It is not a policy !

Markov Decision Process (MDP)

Example

Let's consider three states S_0, S_1, S_2 and two possible actions a_0, a_1 and the following transition and reward functions:

a_0 transition matrix: $T(*, a_0, *)$

	S_0	S_1	S_2
S_0	0.50	0.00	0.50
S_1	0.70	0.10	0.20
S_2	0.40	0.00	0.60

a_1 transition matrix: $T(*, a_1, *)$

	S_0	S_1	S_2
S_0	0.00	0.00	1.00
S_1	0.00	0.95	0.05
S_2	0.30	0.30	0.40

a_0 reward matrix: $R(*, a_0, *)$

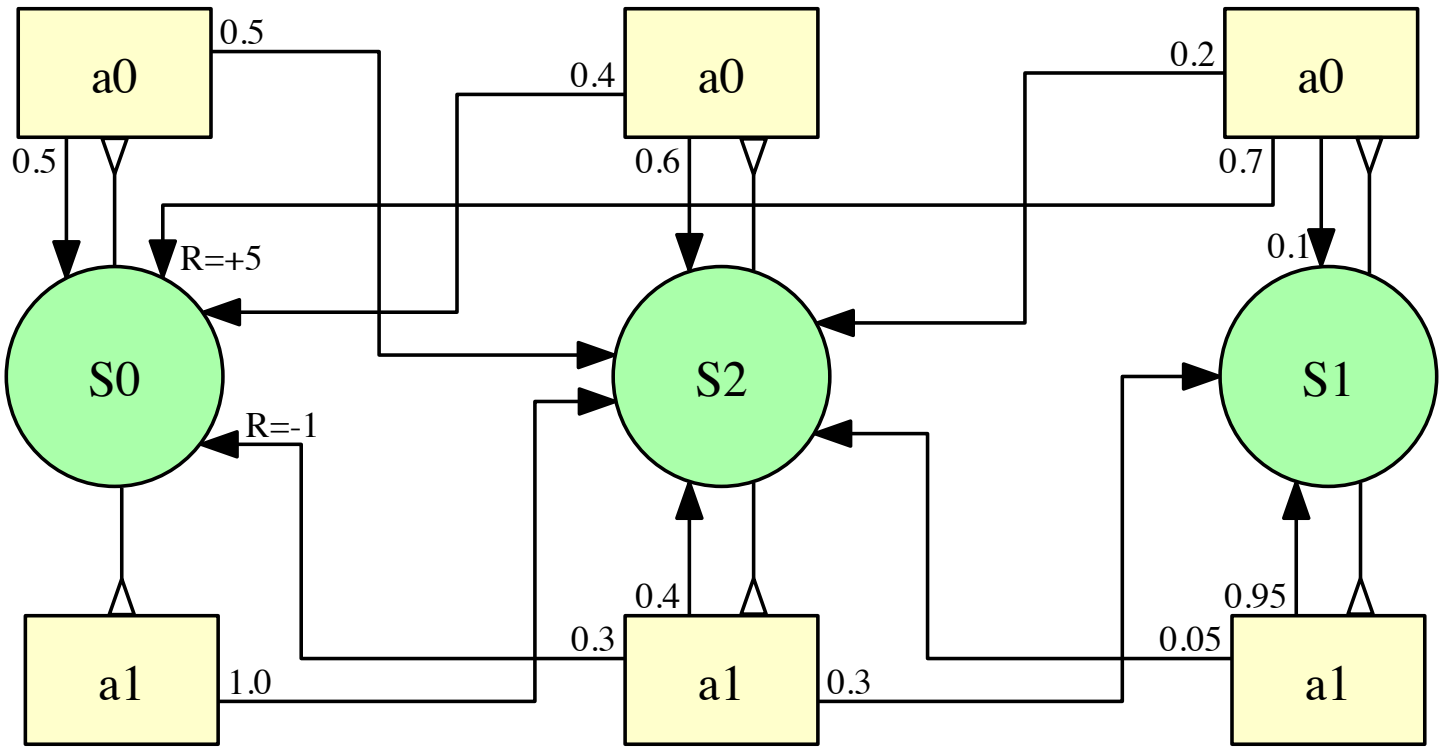
	S_0	S_1	S_2
S_0	0	0	0
S_1	5	0	0
S_2	0	0	0

a_1 reward matrix: $R(*, a_1, *)$

	S_0	S_1	S_2
S_0	0	0	0
S_1	0	0	0
S_2	−1	0	0

Markov Decision Process (MDP)

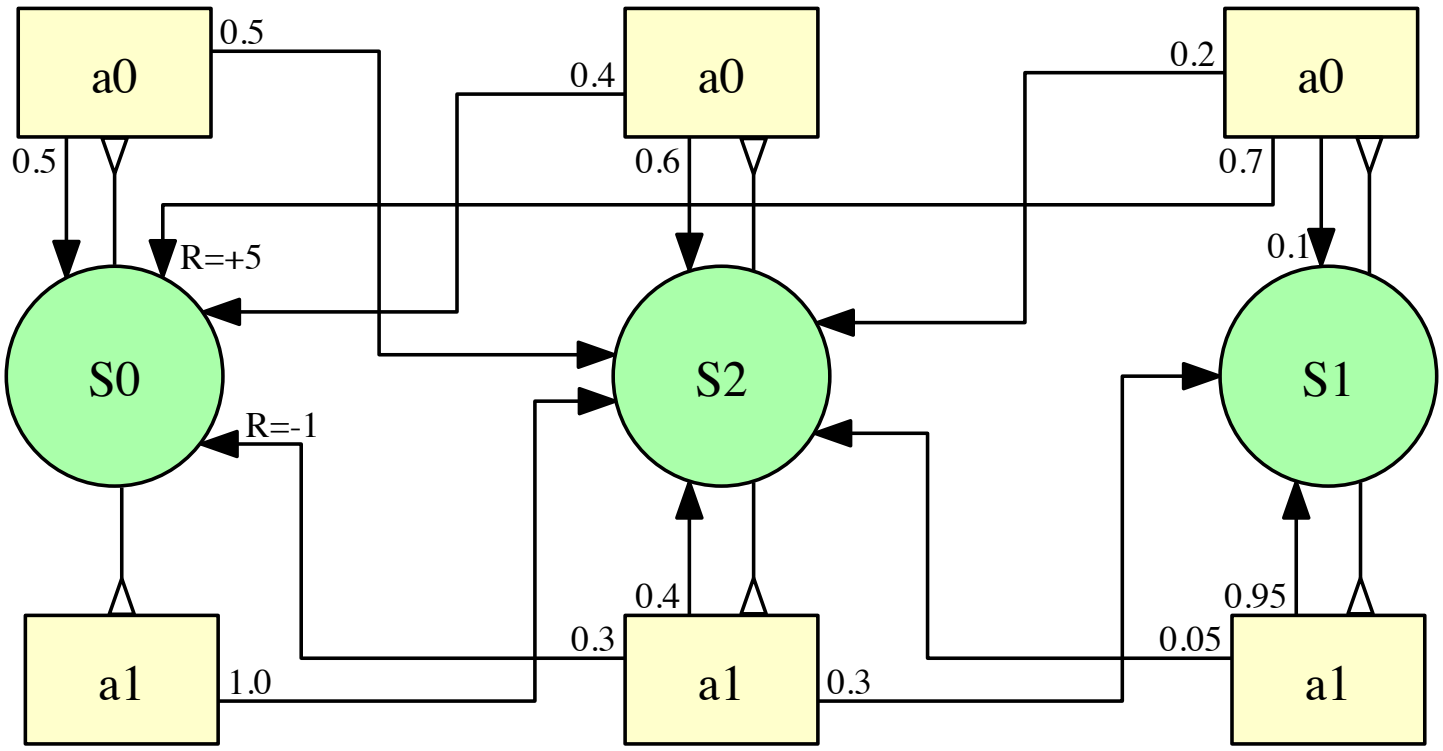
Simple example from the toy world



What is the best strategy ?

Markov Decision Process (MDP)

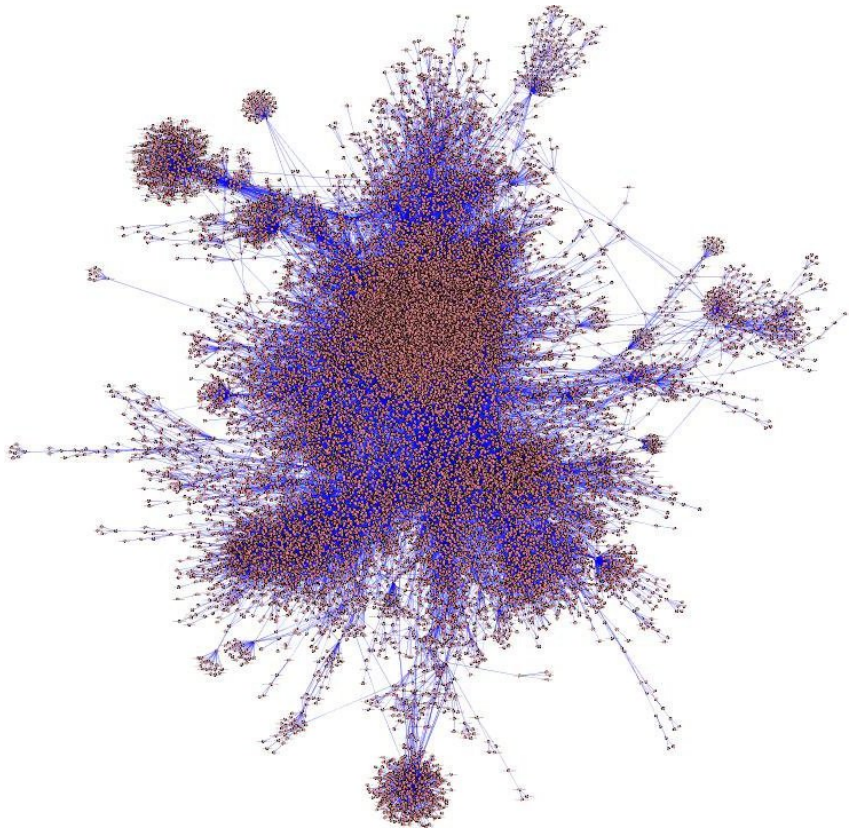
Simple example from the toy world



What is the best strategy ? $\rightarrow (S_0, a_1), (S_1, a_0), (S_2, a_1)$ (easy enough !)

Markov Decision Process (MDP)

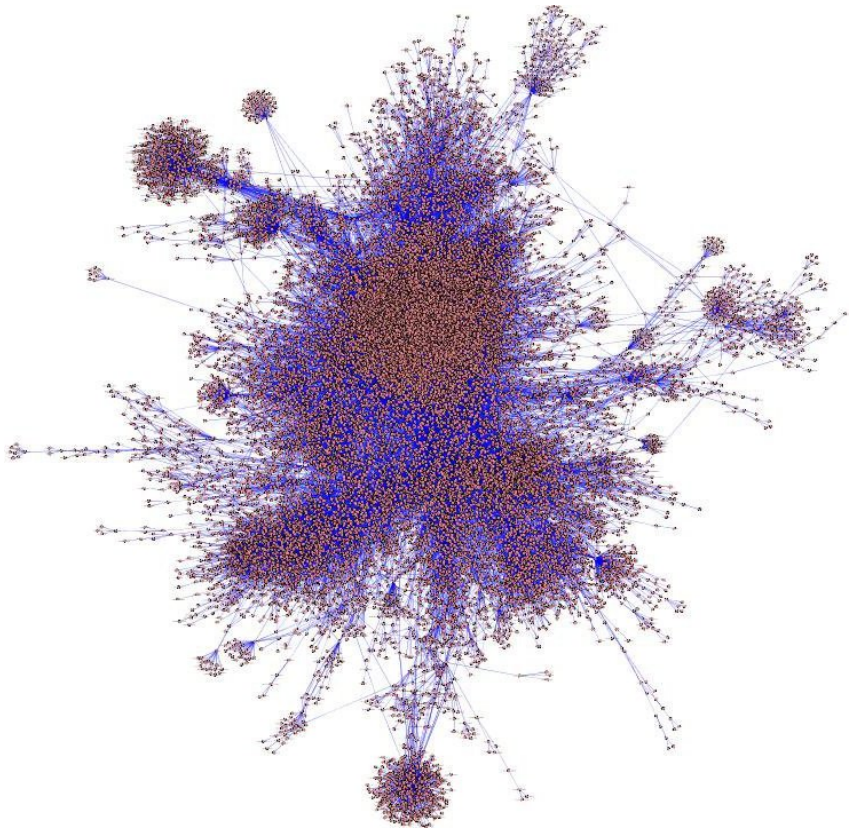
Complex example from the real world



What is the best strategy ?

Markov Decision Process (MDP)

Complex example from the real world



What is the best strategy ? Not that obvious, we need some methods.

Policy

Stochastic vs. Deterministic

A policy π specifies the action $\pi(s)$ to choose when in state s.

Deterministic policy

$\pi : S \to A$

Stochastic policy

$\pi : S \times A \to [0, 1]$

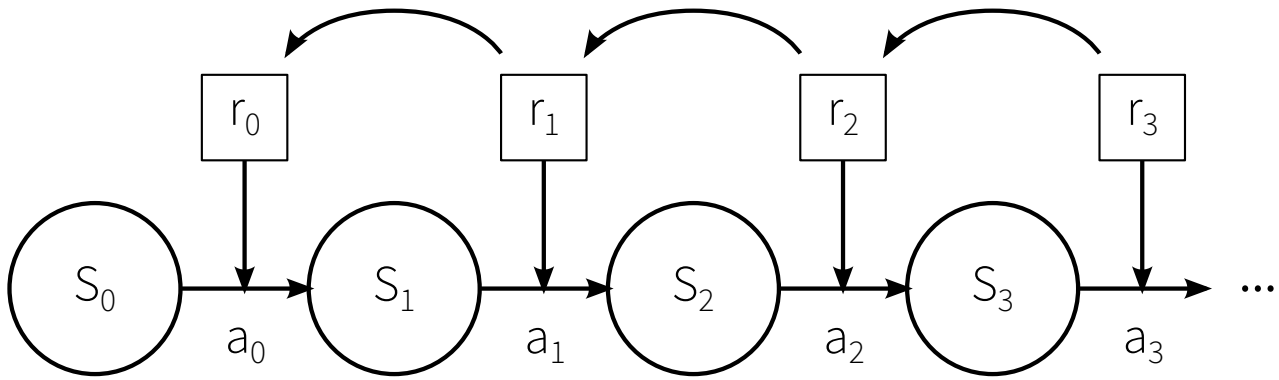
- The core problem of MDP is to find a policy π that maximize reward
- For any MDP, there exists an optimal deterministic policy π^*

Note: Once a Markov decision process is combined with a deterministic policy, this fixes the action for each state and the resulting combination is a Markov chain.

Value functions

How good is a state in terms of futures rewards ?

For a given policy π, we can valuate a state (using a scalar) such that this value expresses the sequences of future rewards. We thus need to define how to aggregate future rewards into a single value.

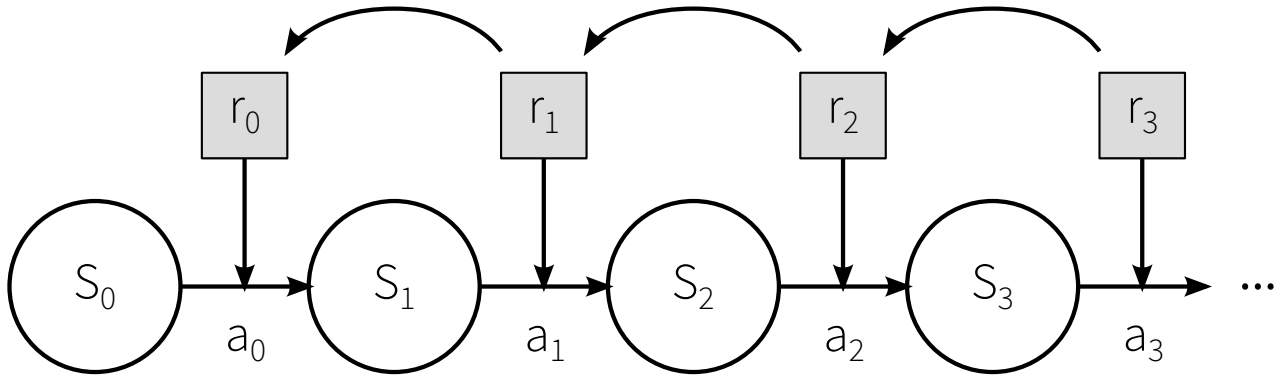


From state S_0 and following the shown sequence of actions, we can expect to receive reward r_0, then reward r_1, etc. What is this the overall expected reward (or value) $V^{\pi}(S_0)$?

Value functions

Total reward

Sum of all future rewards over a finite horizon.

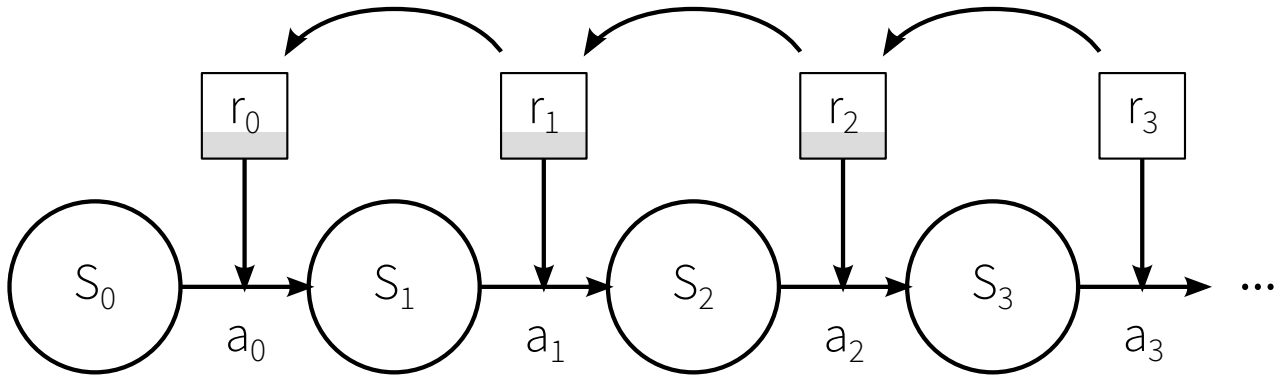


$$V^{\pi}(S_0) = r_0 + r_1 + \ldots + r_n = \sum_{i=0}^{i=n} r_i$$

Value functions

Average reward

Mean of future rewards over a finite window.

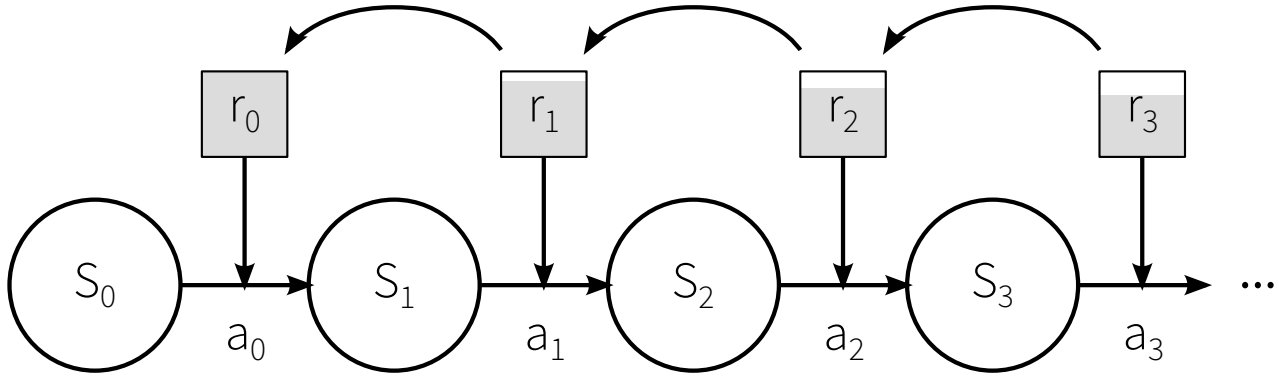


$$V^{\pi}(S_0) = \frac{r_0 + r_1 + r_2}{3}$$

Value functions

Discounted reward

Future rewards are worth less than the current reward.



$$V^{\pi}(S_0) = r_0 + \gamma r_1 + \gamma^2 r_2 + \ldots + \gamma^n r_n + \ldots = \sum_{i=0}^{+\infty} \gamma^i r_i$$

$\gamma \in [0, 1]$ is the discount factor:

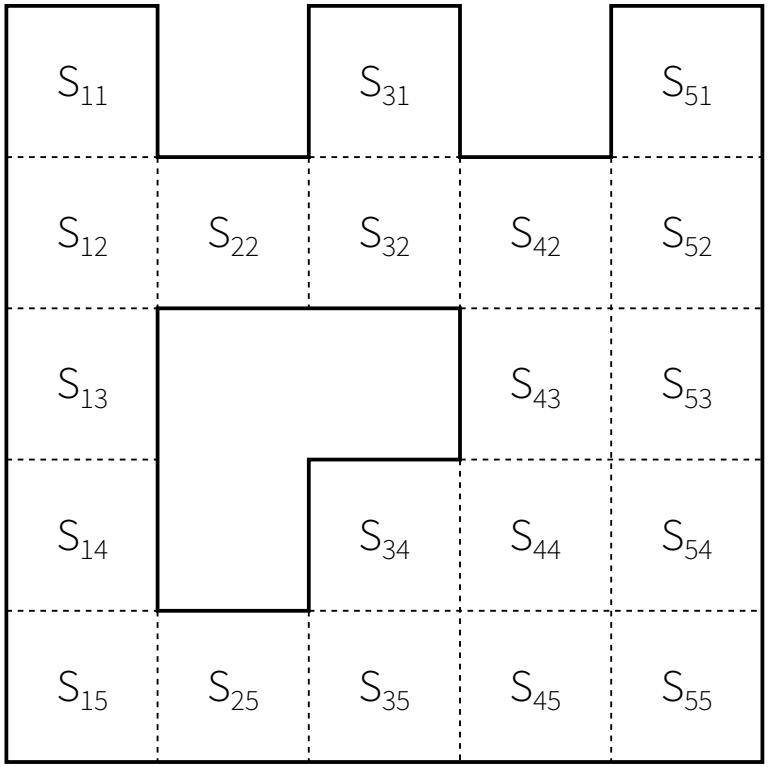
- $\gamma = 0$ means only immediate reward is important
- $\gamma = 1$ means all rewards are equally important

From now on, we'll use the discounted reward.

State-Value function

(a.k.a. Value function)

The state-value function $V^{\pi} : S \rightarrow \mathbb{R}$ represents for any state the expected future reward for policy π. It is a vector with one value per state.

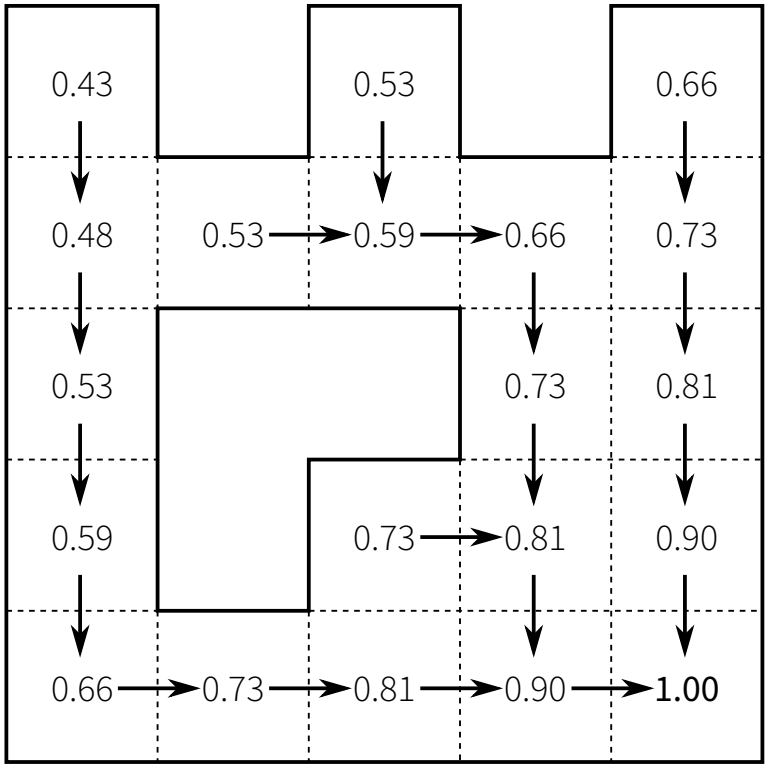


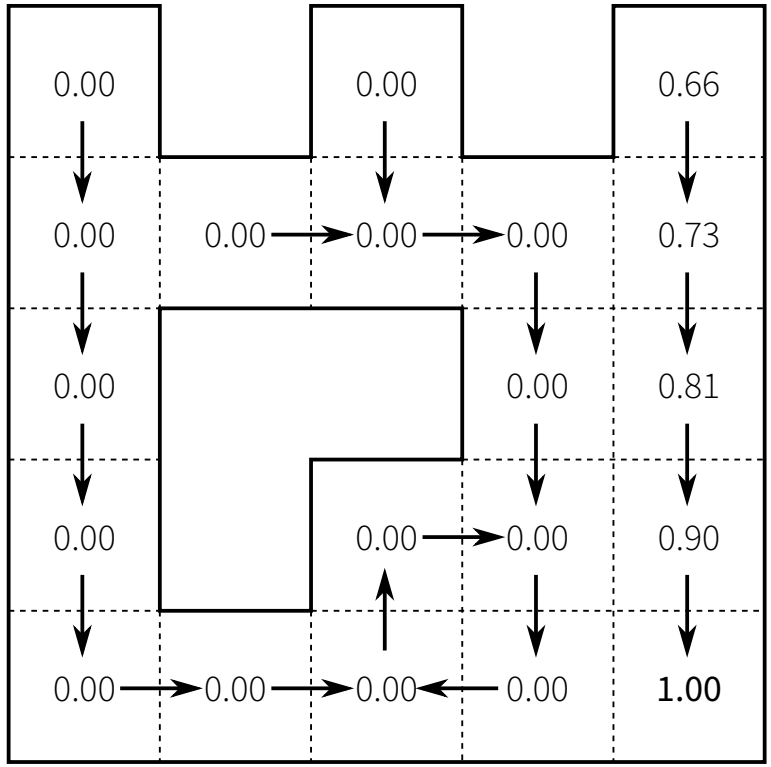
State	*Value*	*Reward*
S_{11}	V^{π}_{11}	0
S_{12}	V^{π}_{12}	0
S_{22}	V^{π}_{22}	0
S_{13}	V^{π}_{13}	0
…	…	…
S_{55}	V^{π}_{55}	1

State-Value function

(a.k.a. Value function)

The state-value function $V^{\pi} : S \rightarrow \mathbb{R}$ represents for any state the expected future reward for policy π. It is a vector with one value per state.

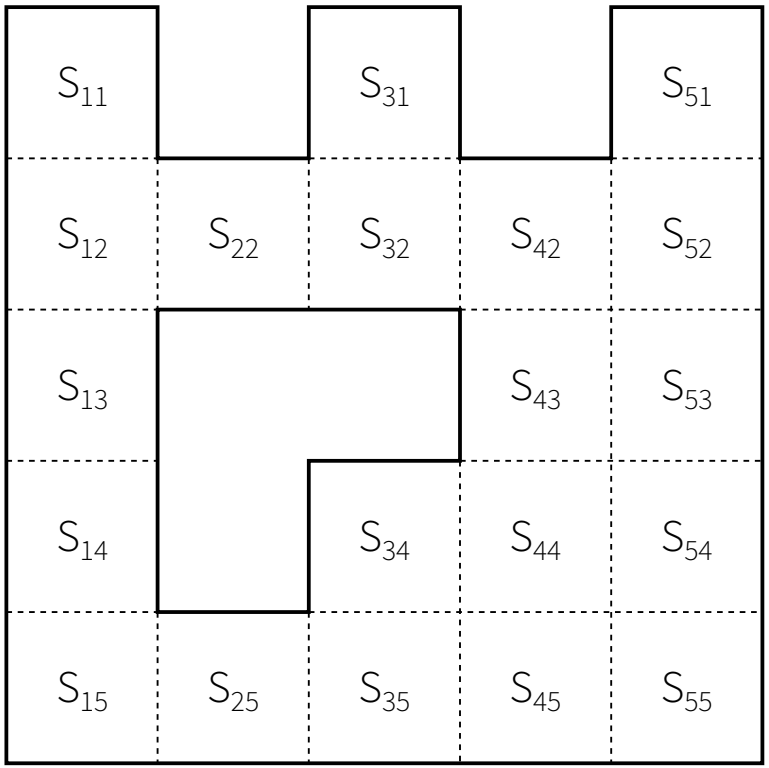




Action-Value function

(a.k.a. Q-function)

The action-value function $Q^{\pi} : S \times A \rightarrow \mathbb{R}$ represents the expected future reward after taking the action a and then following policy π. It is a matrix with one value per state and action.

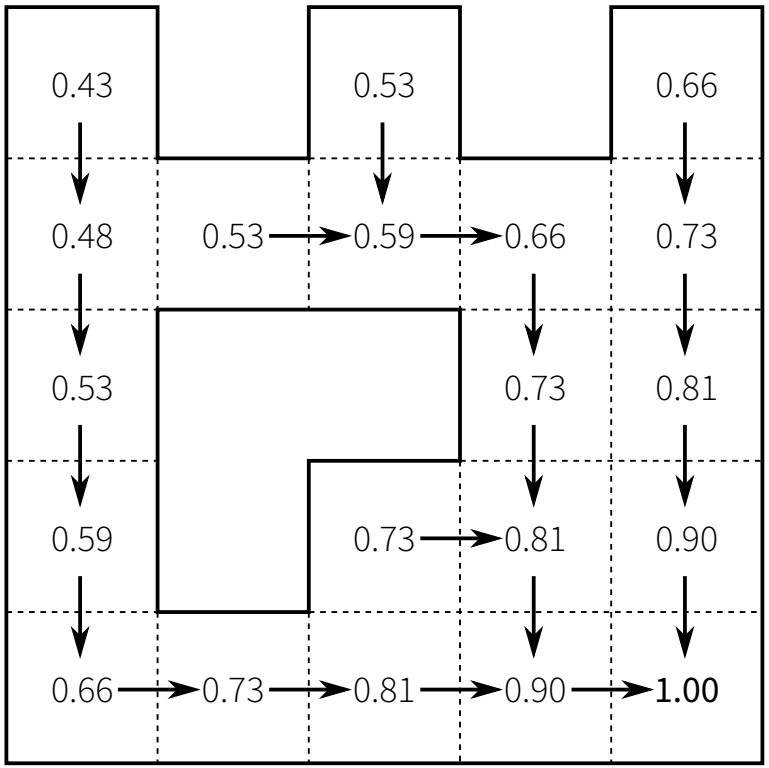


	$\leftarrow$	$\rightarrow$	$\uparrow$	$\downarrow$
S_{11}	$Q_{11}^{\leftarrow}$	$Q_{11}^{\rightarrow}$	$Q_{11}^{\uparrow}$	$Q_{11}^{\downarrow}$
...				
S_{54}	$Q_{54}^{\leftarrow}$	$Q_{54}^{\rightarrow}$	$Q_{54}^{\uparrow}$	$Q_{54}^{\downarrow}$
S_{45}	$Q_{45}^{\leftarrow}$	$Q_{45}^{\rightarrow}$	$Q_{45}^{\uparrow}$	$Q_{45}^{\downarrow}$
S_{55}	$Q_{55}^{\leftarrow}$	$Q_{55}^{\rightarrow}$	$Q_{55}^{\uparrow}$	$Q_{55}^{\downarrow}$

Action-Value function

(a.k.a. Q-function)

The action-value function $Q^\pi : S \times A \to \mathbb{R}$ represents the expected future reward after taking the action a and then following policy π. It is a matrix with one value per state and action.



	$\leftarrow$	$\rightarrow$	$\uparrow$	$\downarrow$
S_{11}	0.43	0.43	0.43	0.48
...				
S_{54}	0.81	0.90	0.81	1.00
S_{45}	0.81	1.00	0.81	0.90
S_{55}	0.90	1.00	0.90	1.00

Value functions

Formal definition using discounted reward

State-Value function for policy π

$$V^{\pi}(s) = E_{\pi}\{R_t | s_t = s\} = E_{\pi}\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s\right\}$$

Action-Value function for policy π

$$Q^{\pi}(s,a) = E_{\pi}\{R_t | s_t = s, a_t = a\} = E_{\pi}\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s, a_t = a\right\}$$

Bellman equation

$$V^{\pi}(s) = \sum_{a} \pi(s,a) \sum_{s'} \mathcal{P}^{a}_{ss'} \left[\mathcal{R}^{a}_{ss'} + \gamma V^{\pi}(s')\right]$$

with

$$\mathcal{P}^{a}_{ss'} = P_r \left\{s_{t+1} = s' | s_t = s, a_t = a\right\}$$
$$\mathcal{R}^{a}_{ss'} = E \left\{r_{t+1} | s_t = s, a_t = a, s_{t+1} = s'\right\}$$

Optimality

Optimal policy

For finite MDPs, there exists an optimal policy π^* such that
$\forall \pi, \forall s \in S, V^{\pi^*}(s) \geq V^{\pi}(s)$

Optimal state-value function

The optimal state-value function is defined as: $V^*(s) = \max_{\pi} V^{\pi}(s)$

Optimal action-value function

The optimal action-value function is defined as: $Q^*(s, a) = \max_{\pi} Q^{\pi}(s, a)$

Bellman Optimality equation

Bellman optimality equation for V^*

$$
\begin{aligned}
V^*(s) &= \max_a E\left\{r_{t+1} + \gamma V^*(s_{t+1}) | s_t =, a_t = a\right\} \\
&= \max_a \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma V^*(s')\right]
\end{aligned}
$$

Bellman optimality equation for Q^*

$$
\begin{aligned}
Q^*(s) &= E\left\{r_{t+1} + \gamma \max_{a'} Q^*(s_{t+1}, a') | s_t =, a_t = a\right\} \\
&= \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma \max_{a'} Q^*(s', a')\right]
\end{aligned}
$$

Exercise 1: Recycling Robot

(from Sutton & Barto, 1998)

A mobile robot has the job of collecting empty soda cans in an office environment. It has sensors for detecting cans, and an arm and gripper that can pick them up and place them in an onboard bin; it runs on a rechargeable battery. The robot's control system has components for interpreting sensory information, for navigating, and for controlling the arm and gripper. High-level decisions about how to search for cans are made by a reinforcement learning agent based on the current charge level of the battery. This agent has to decide whether the robot should

1. actively search for a can for a certain period of time
2. remain stationary and wait for someone to bring it a can
3. head back to its home base to recharge its battery

This decision has to be made either periodically or whenever certain events occur, such as finding an empty can. The agent therefore has three actions, and its state is determined by the state of the battery. The rewards might be zero most of the time, but then become positive when the robot secures an empty can, or large and negative if the battery runs all the way down.

Exercise 1: Recycling Robot

(from Sutton & Barto, 1998)

- Identify states and actions
- Write the transition table
- Draw the corresponding MDP
- Find the best policy

Exercise 2: Path finding

(Value iteration)

Consider the following maze. We want to find the shortest path going from the entrance (top-left) to the exit (bottom-right).

1. Find a path by hand
2. How to formalize the problem ? (actions/states/rewards)
3. How to automatize the finding ?
4. Write a program that find a path