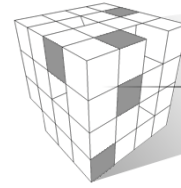




Vrije Universiteit Brussel



CoMo
Computational Modeling Lab

Reinforcement learning an introduction

Prof. Dr. Ann Nowé

Computational Modeling Group

AIlab

ai.vub.ac.be

Ann Nowé

Reinforcement Learning

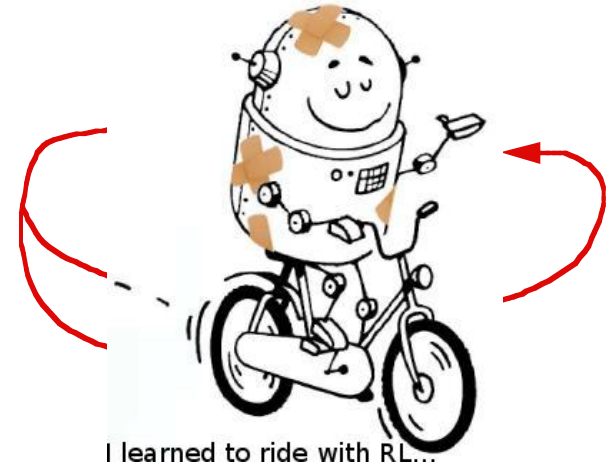
What is it?

Learning from interaction

Learning about, from, and

while interacting with an external environment

Learning what to do—how to map situations to actions—so as to maximize a numerical reward signal



Reinforcement Learning

Key features?

Learner is not told which actions to take

Trial-and-Error search

The need to **explore** and **exploit**

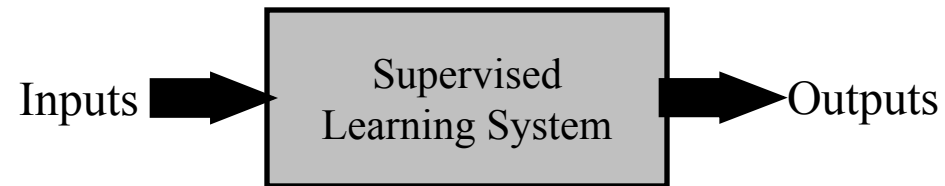
Possibility of delayed reward

- Sacrifice short-term gains for greater long-term gains

Considers the whole problem of a goal-directed agent interacting with an uncertain environment

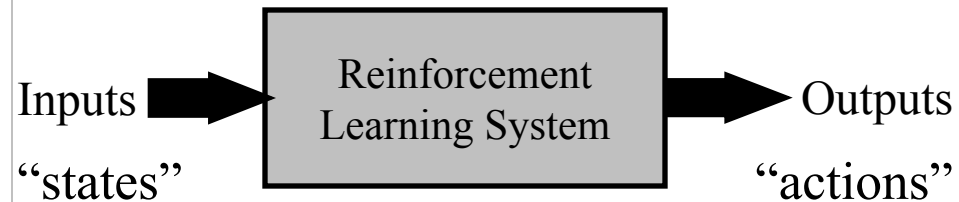
Supervised versus Unsupervised

Training Info
= desired (target) outputs



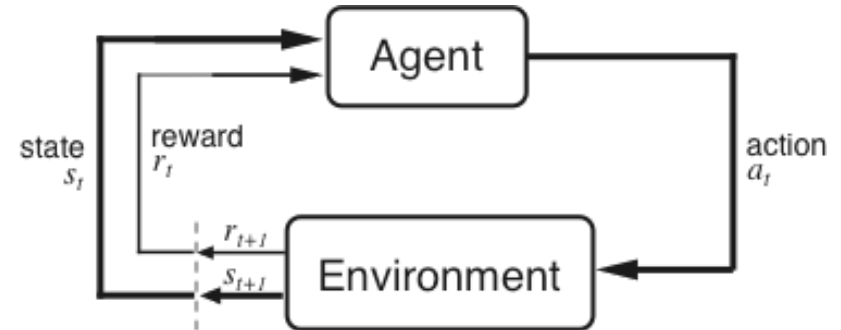
Error = (target output – actual output)

Training Info
= evaluations (“rewards” / “penalties”)



Objective: get as much reward as possible

The Agent-Environment Interface



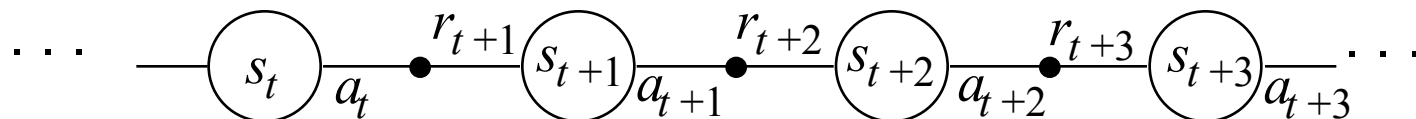
Agent and environment interact at discrete time steps: $t = 0, 1, 2, \dots$

Agent observes state at step t : $s_t \in \mathcal{S}$

produces action at step t : $a_t \in A(s_t)$

gets resulting reward: $r_{t+1} \in \mathcal{R}$

and resulting next state: s_{t+1}



Learning how to behave

Policy at step t, π_t :

a mapping from states to action probabilities

$\pi_t(s, a)$ = probability that $a_t = a$ when $s_t = s$

Reinforcement learning methods specify how the agent changes its policy as a result of experience.

Roughly, the agent's goal is to get as much reward as it can over the long run.

Goals and Rewards

A goal should specify **what** we want to achieve, not **how** we want to achieve it.

Suppose the sequence of rewards after step t is :

$$r_{t+1}, r_{t+2}, r_{t+3}, \dots$$

What do we want to maximize?

In general,

we want to maximize the **expected return**, $E\{R_t\}$ for each step t .

What's the objective?

Episodic tasks: interaction breaks naturally into episodes, e.g., plays of a game.

$$R_t = r_{t+1} + r_{t+2} + \dots + r_T$$

The diagram illustrates the components of the return R_t . The term r_{t+1} is enclosed in a box labeled "Immediate reward". A bracket groups the remaining terms $r_{t+2} + \dots + r_T$, which is enclosed in a box labeled "Long term reward".

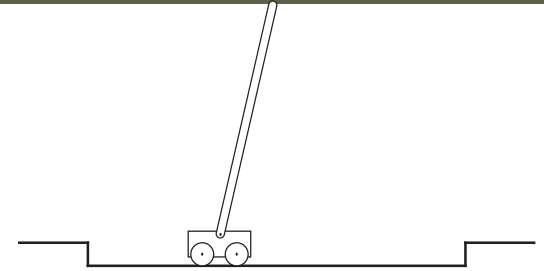
Continuing tasks: interaction does not have natural episodes.

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \dots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1},$$

where $\gamma, 0 \leq \gamma \leq 1$, is the **discount rate**.

Goals and Rewards : example

Pole-balancing:



Reward = +1 every time step before falling down,

Reward = 0 when pole fall down

OR

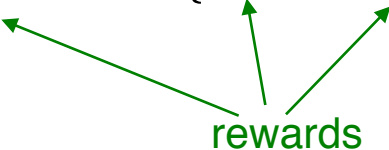
Reward = 0 every time step before falling down,

Reward = -1 when pole falls down.

and take $\gamma < 1$

The Objective Maximize Long-term Total Discounted Reward

Find a policy $\pi : s \in S \rightarrow a \in A$ (could be stochastic)
that maximizes the **value** (expected future reward) of each s :

$$\begin{aligned} V^\pi(s) &= E\{r_{t+1} + \gamma \cdot r_{t+2} + \gamma^2 \cdot r_{t+3} \dots | s_t = s, \pi\} \\ &= E\{r_{t+1}\} + \gamma \cdot E\{r_{t+2} + \gamma \cdot r_{t+3} + \dots | s_{t+1} = s, \pi\} \end{aligned}$$


rewards

Optimal Value Functions and Policies

There exist **optimal value functions** V^*

and corresponding **optimal policies** π^*

which maximize the values for all states simultaneously

$$V^*(s) = \max_{\pi} V^{\pi}(s) \quad \forall s$$

$$V(s) = \max_a Q(s, a)$$

Optimal Value Functions and Policies

There exist **optimal value functions** V^*

and corresponding **optimal policies** π^*

which maximize the values for all states simultaneously

$$V^*(s) = \max_{\pi} V^{\pi}(s) \quad \forall s$$

$$V(s) = \max_a Q(s, a)$$

Optimal Value Functions and Policies

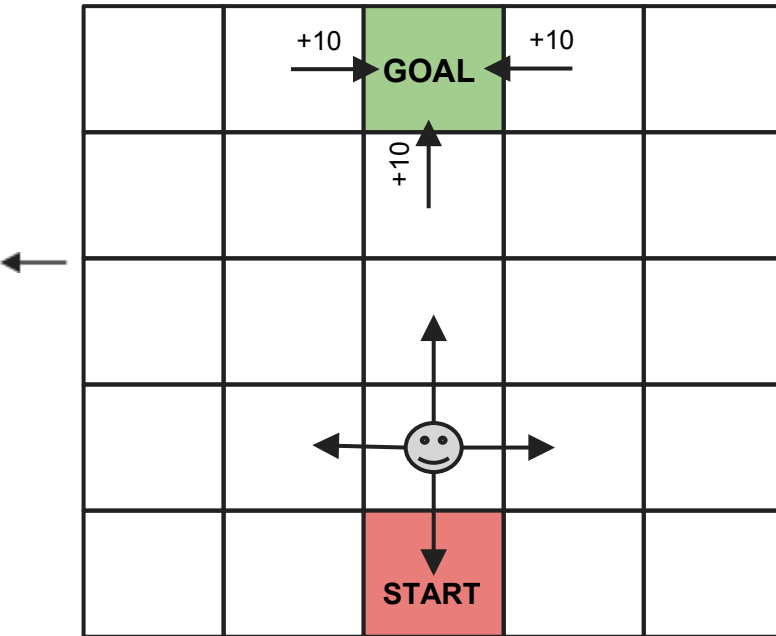
9	10	0	10	9
8.1	9	10	9	8.1
7.2	8.1	9	8.1	7.2
6.3	7.2	8.1	7.2	6.3
5.4	6.3	7.2	6.3	5.4

$V^*(s)$

→	→		←	←
↑→	↑→	↑	←↑	←↑
↑→	↑→	↑	←↑	←↑
↑→	↑→	↑	←↑	←↑
↑→	↑→	↑	←↑	←↑

$\pi^*(s)$

Optimal Value Functions and Policies



- Find shortest path to goal
- Rewards can be delayed:
 - only receive reward when reaching goal
- Unknown environment
- Consequences of an action can only be discovered by trying it and observing the result (new state s' , reward r)

- States: Location 1 ... 25
- Actions: Move N,E,S,W
- Transitions: move 1 step in selected direction (except at borders)
- Rewards: +10 if next loc == goal, 0 else

Q-Learning, basic RL algorithm

One - step Q - learning :

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right]$$

Initialize $Q(s, a)$ arbitrarily

Repeat (for each episode):

Initialize s

Repeat (for each step of episode):

Choose a from s using policy derived from Q (e.g., ϵ -greedy)

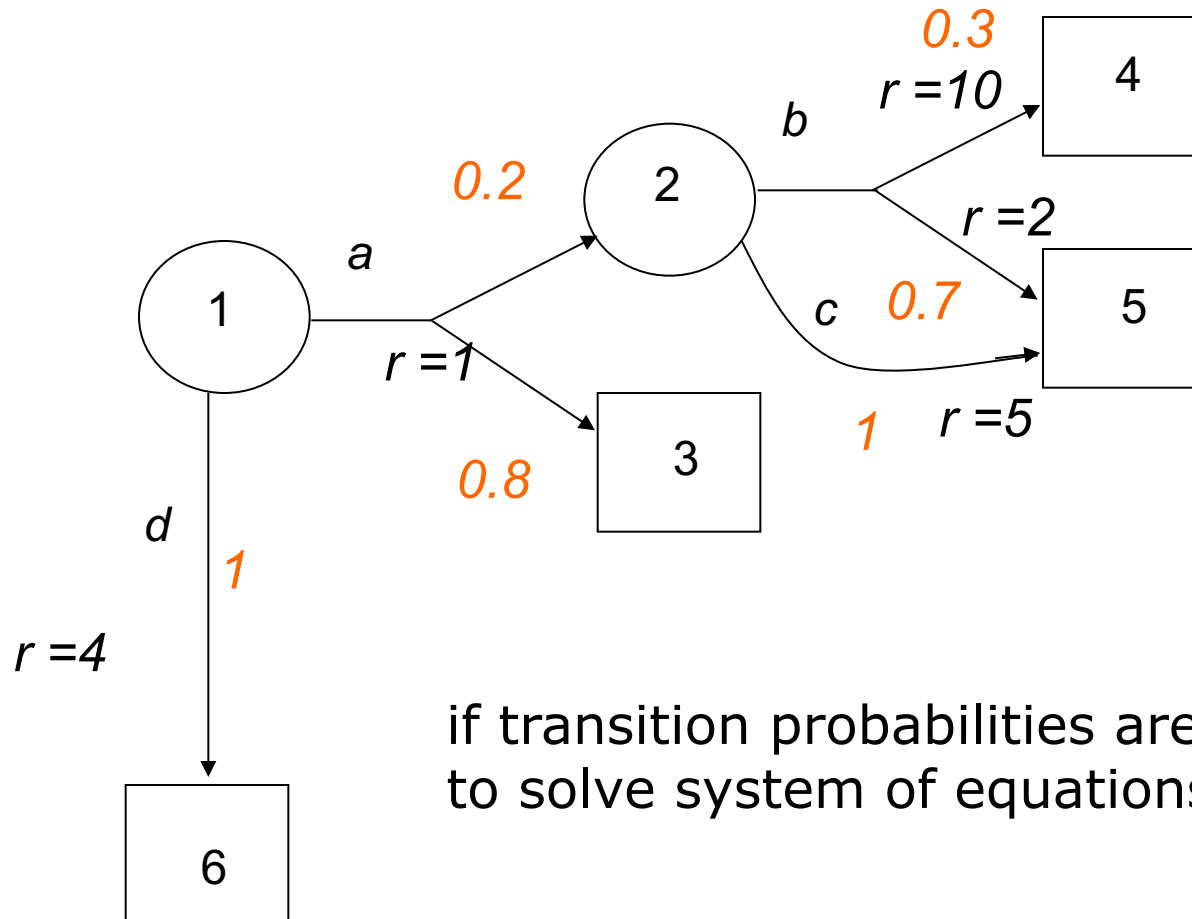
Take action a , observe r, s'

$$Q(s, a) \leftarrow Q(s, a) + \alpha [r + \gamma \max_{a'} Q(s', a') - Q(s, a)]$$

$s \leftarrow s'$;

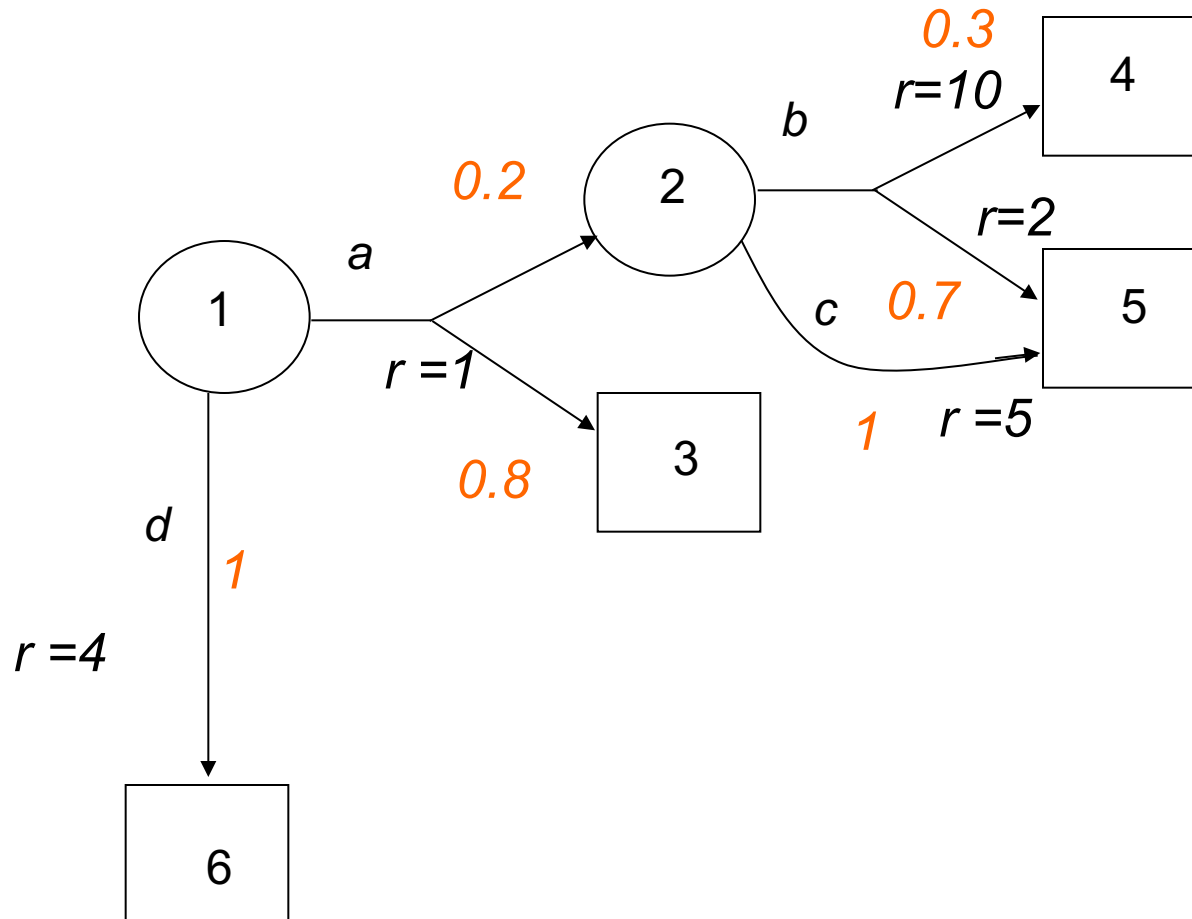
until s is terminal

A very simple MDP example

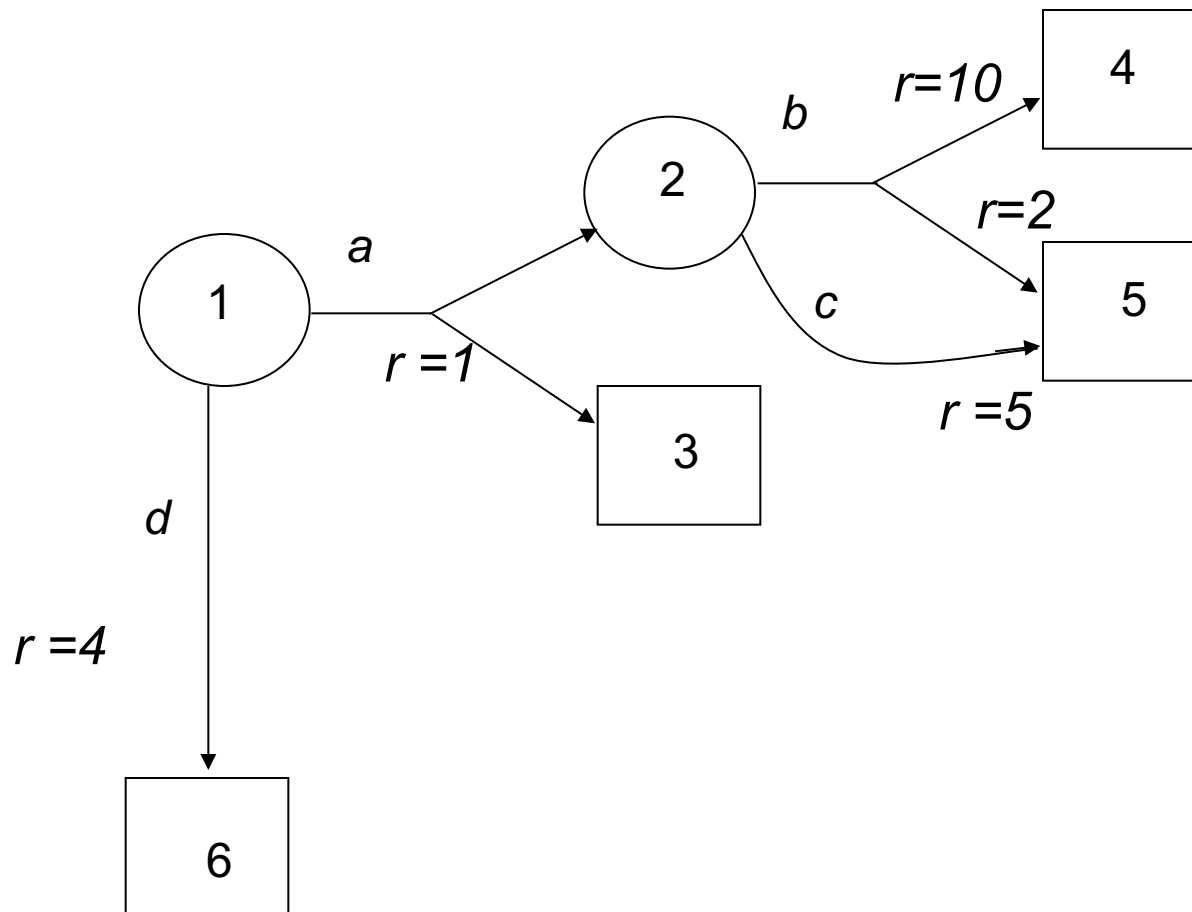


if transition probabilities are known use DP to solve system of equations

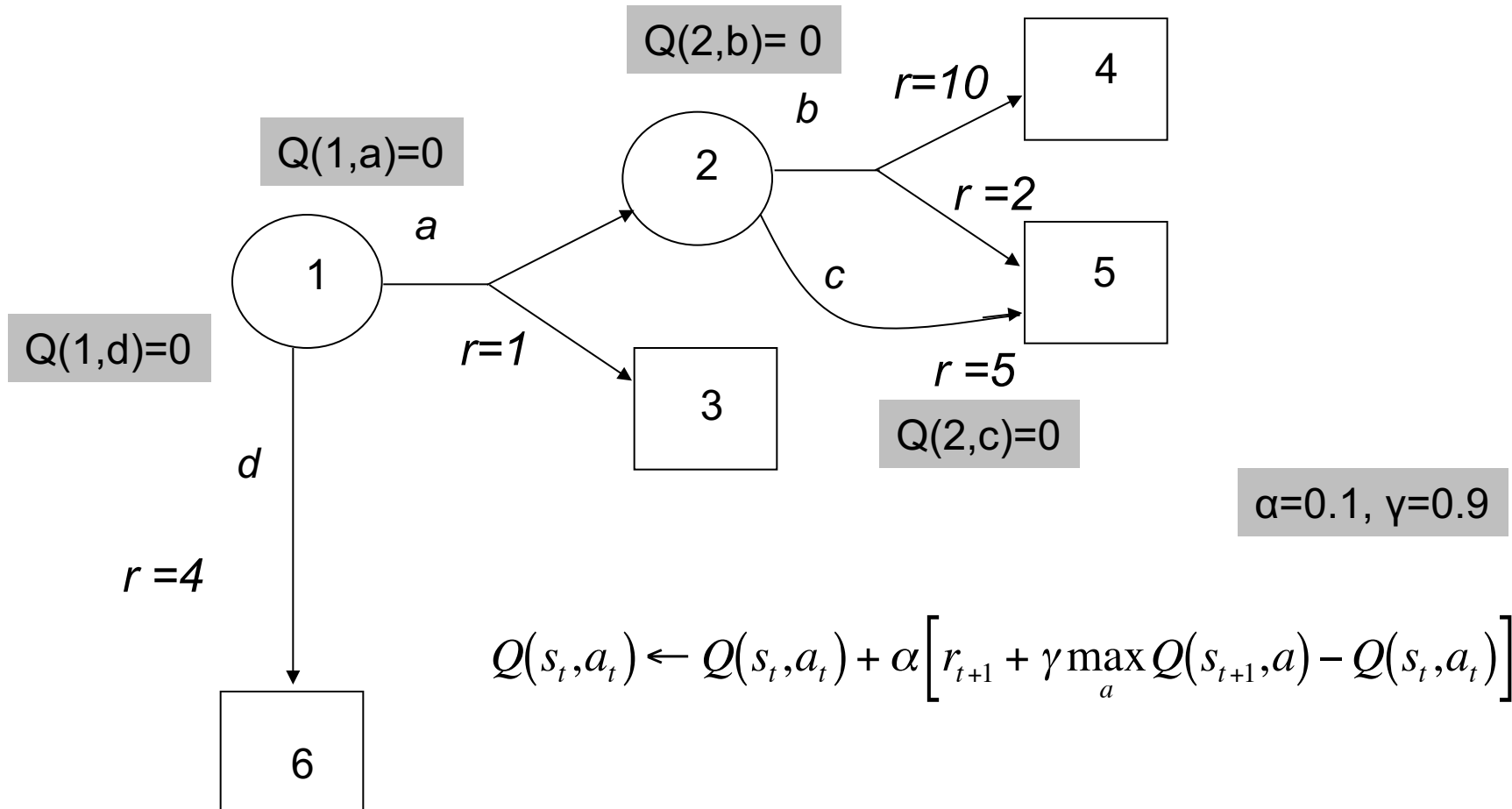
RL is model free



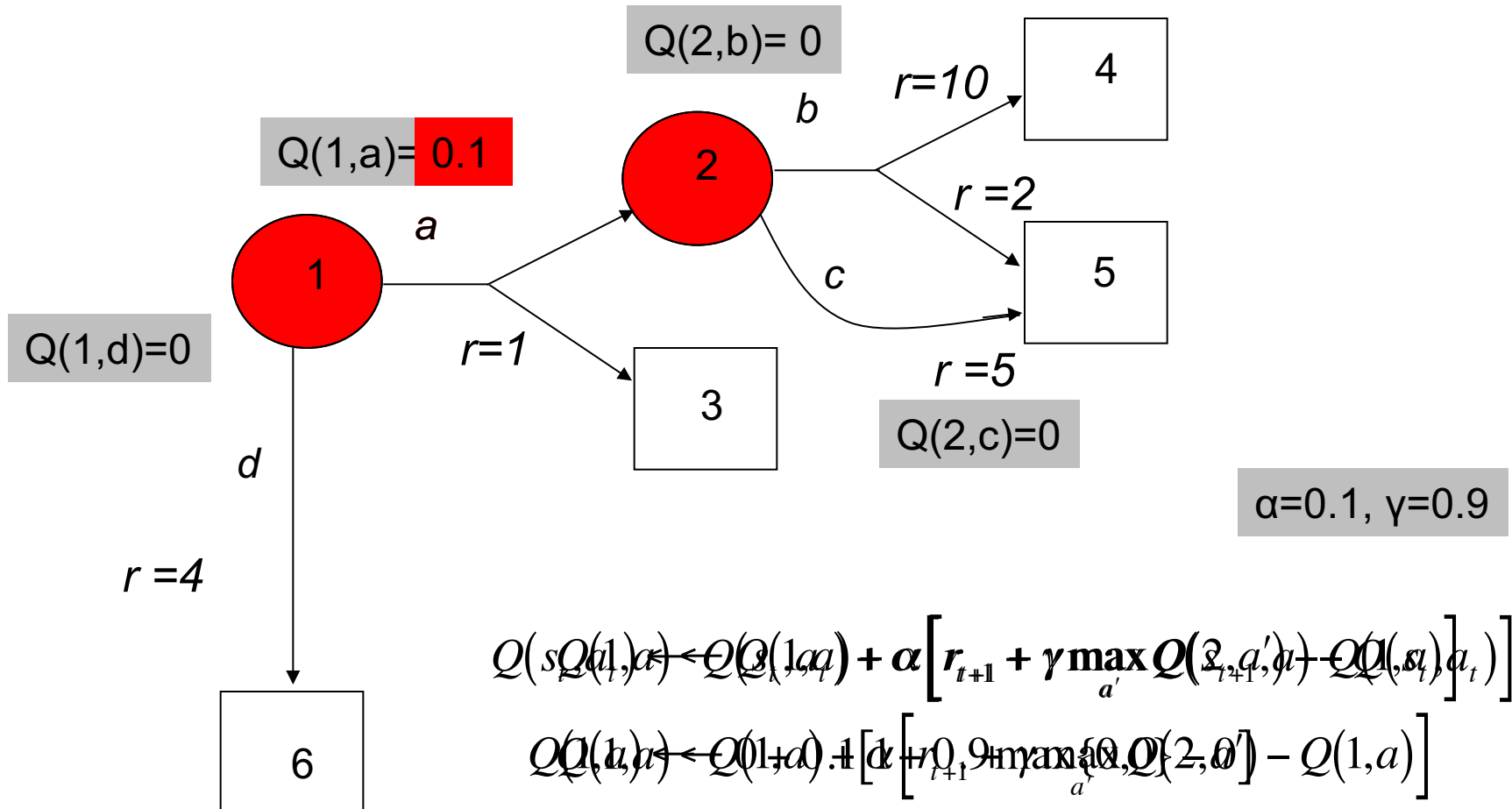
RL is model free



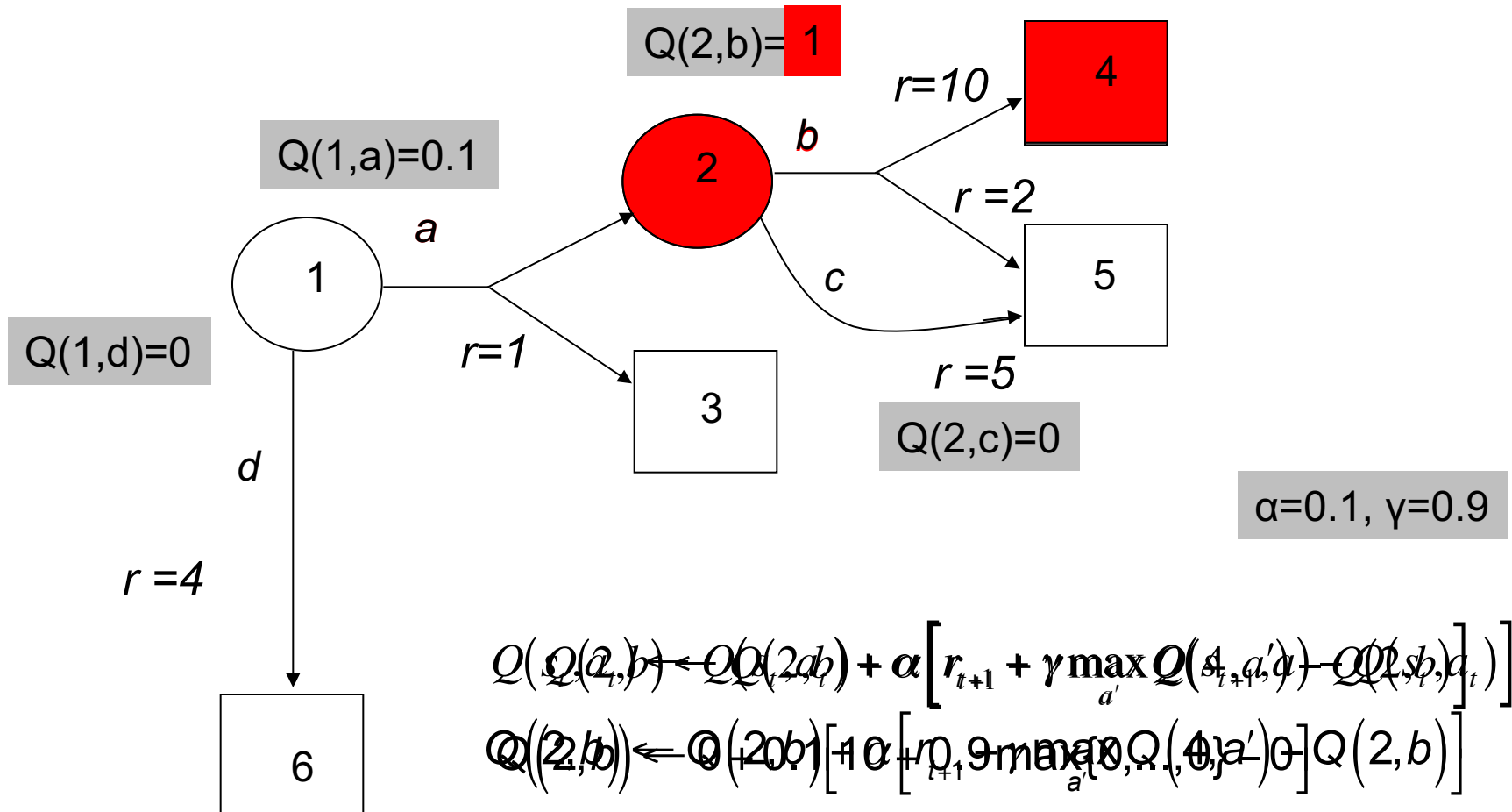
Q-learning



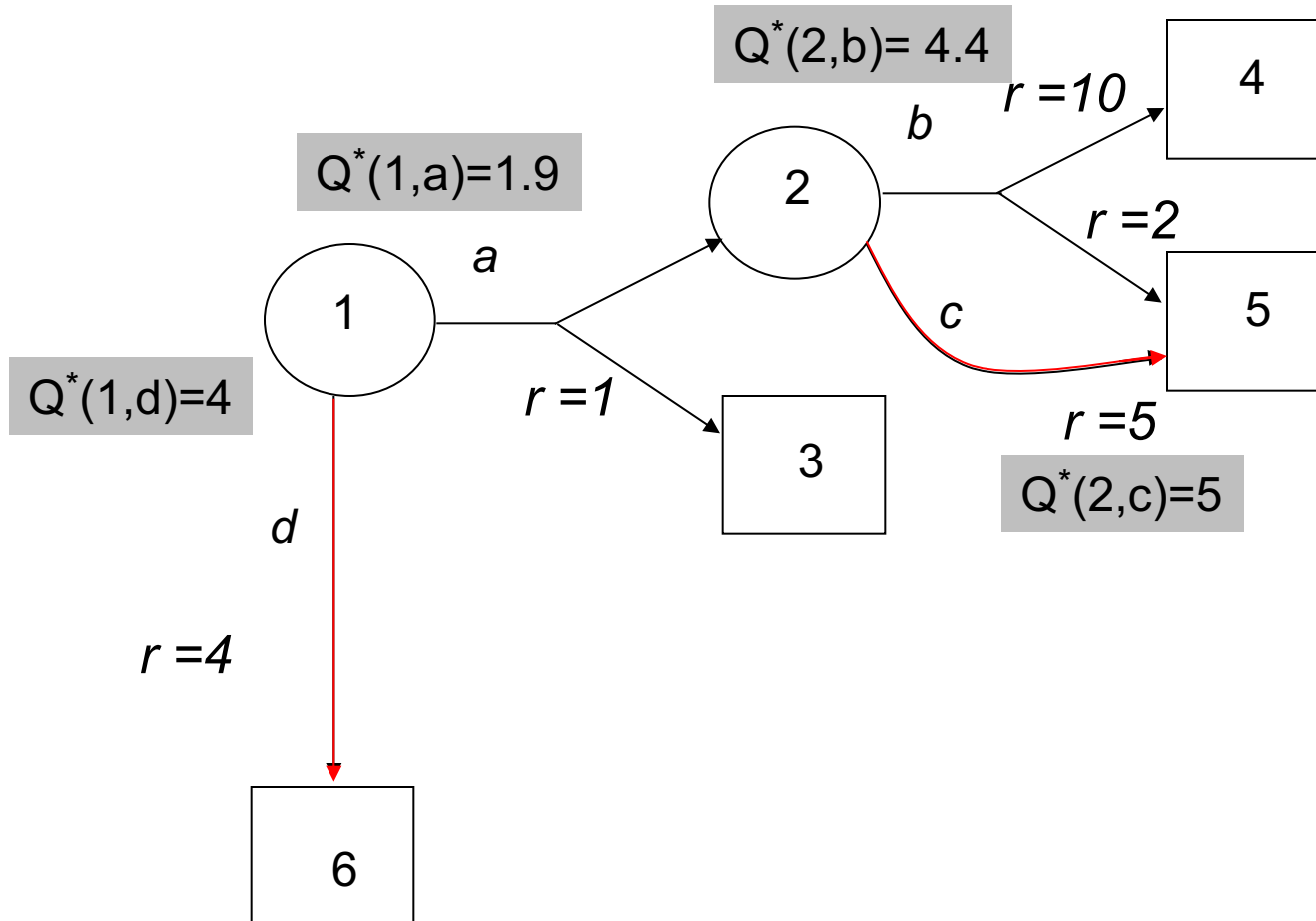
Q-learning



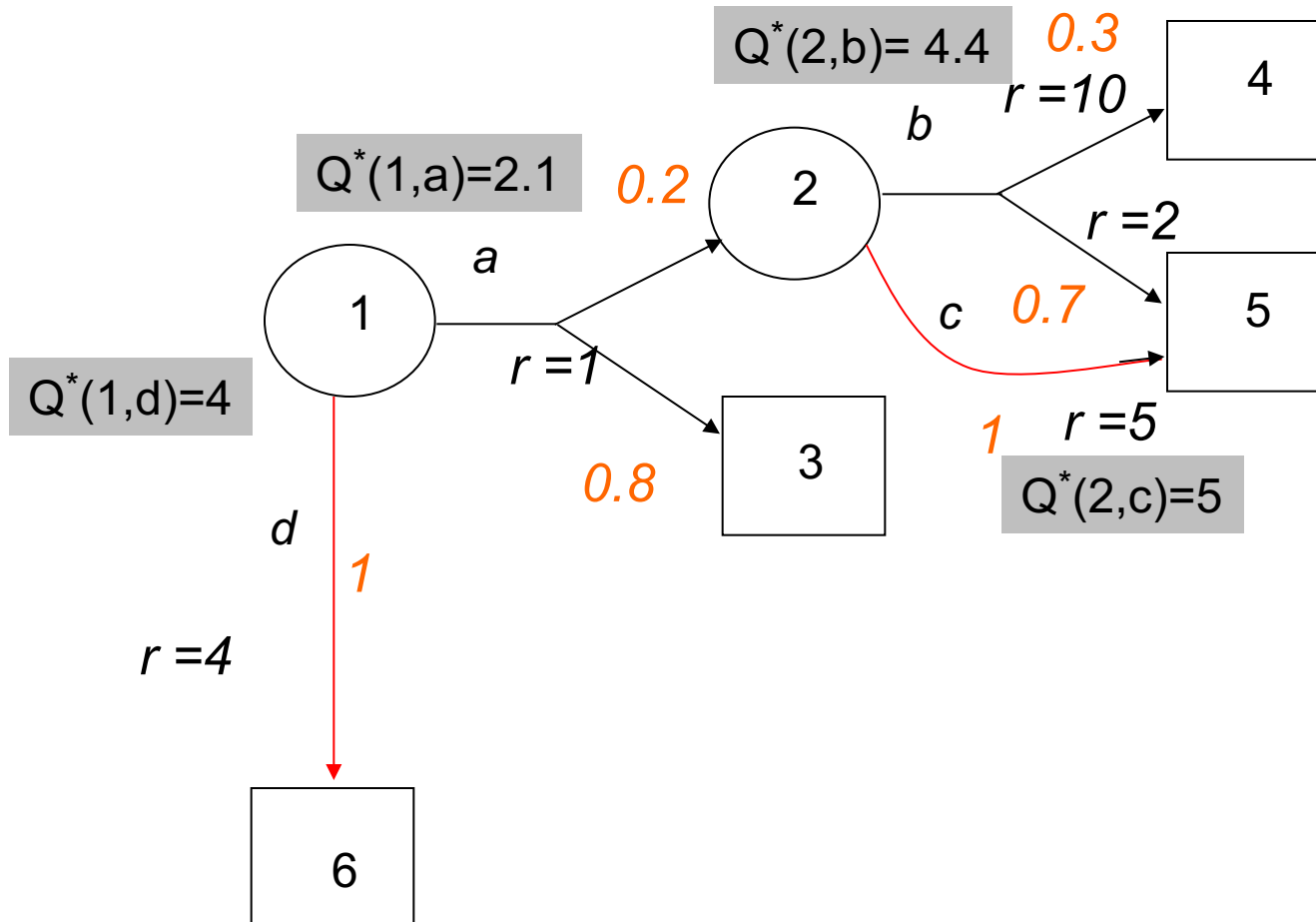
Q-learning



Q-learning



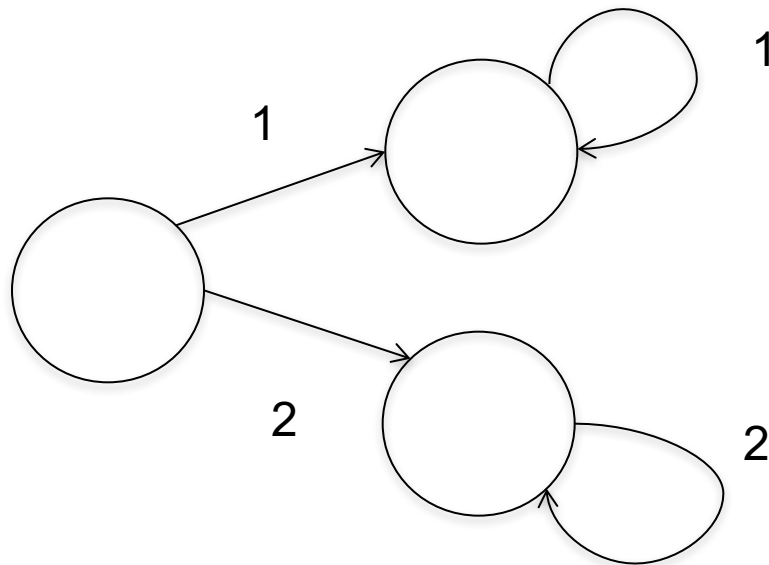
Q-learning: optimal policy



Role of γ

$\gamma < 1$ expresses that reward in the near future is more important than reward in the long term

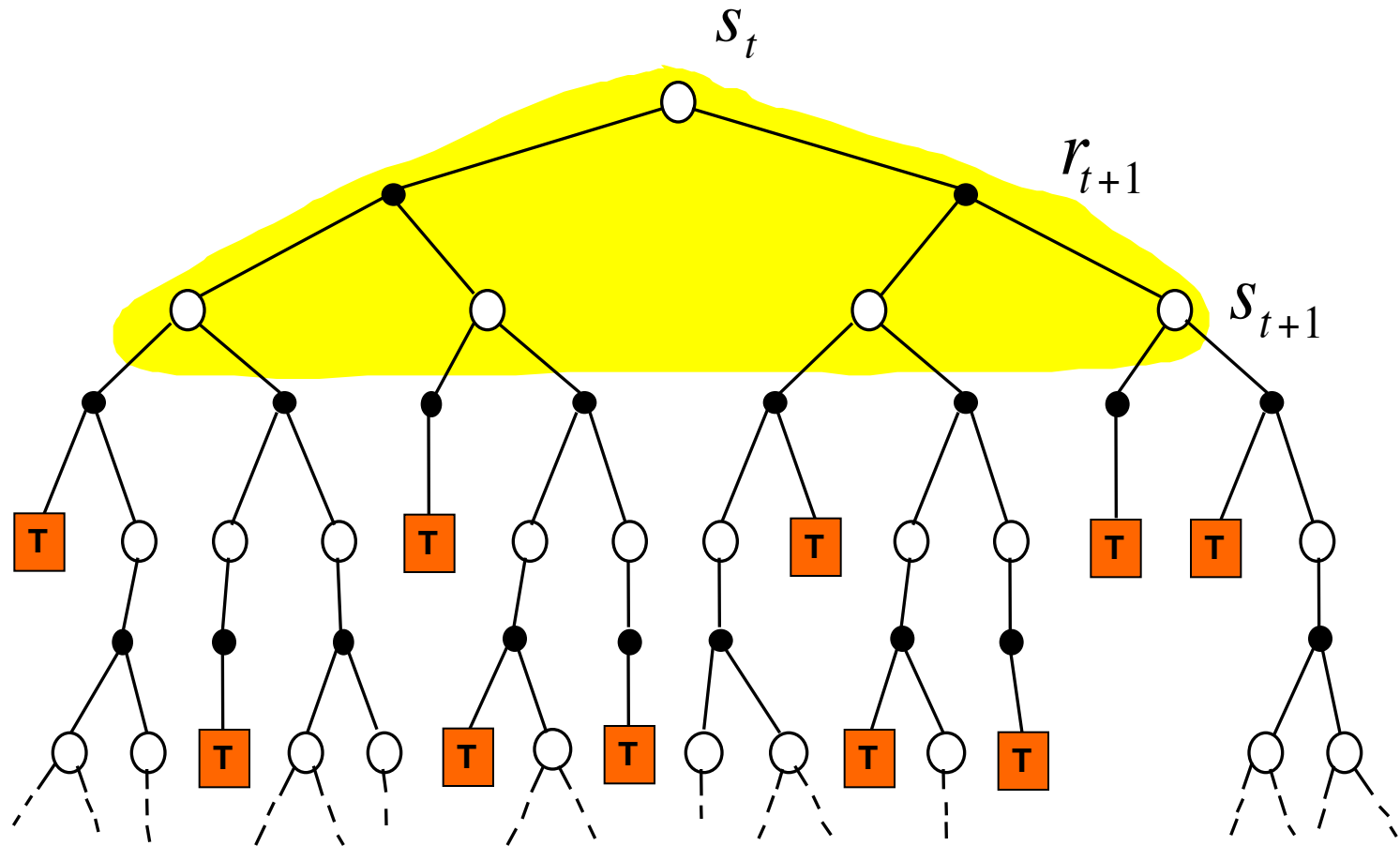
But also bounds the total expected reward



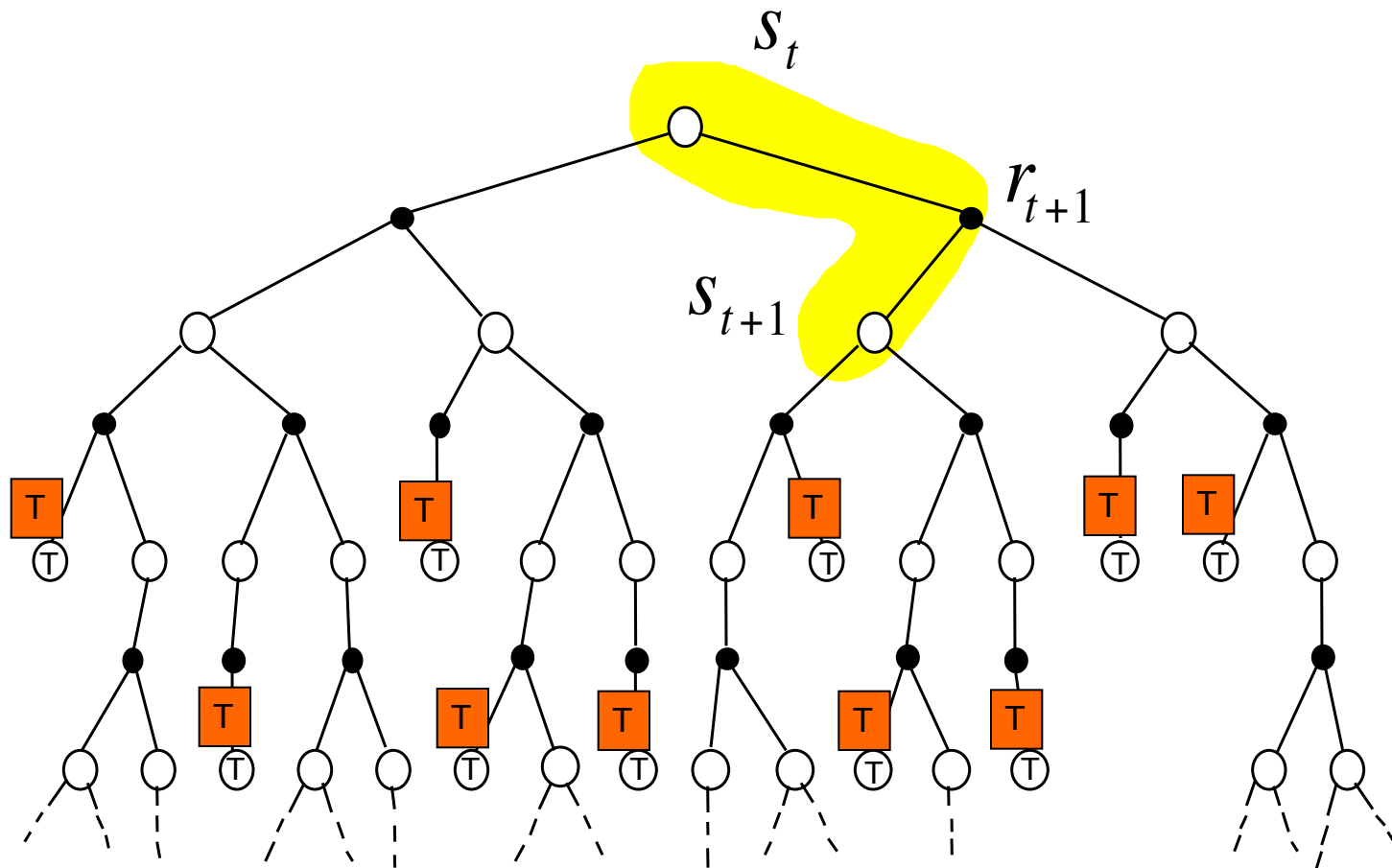
$$1 + \gamma \cdot 1 + \gamma^2 \cdot 1 + \gamma^3 + \dots$$

$$2 + \gamma \cdot 2 + \gamma^2 \cdot 2 + \gamma^3 + \dots = 2(1 + \gamma \cdot 1 + \gamma^2 \cdot 1 + \gamma^3 + \dots)$$

Backup scheme DP



Backup scheme RL



Bootstraps and Samples

Bootstrapping: update involves an estimate

- MC does not bootstrap
- DP bootstraps
- RL bootstraps

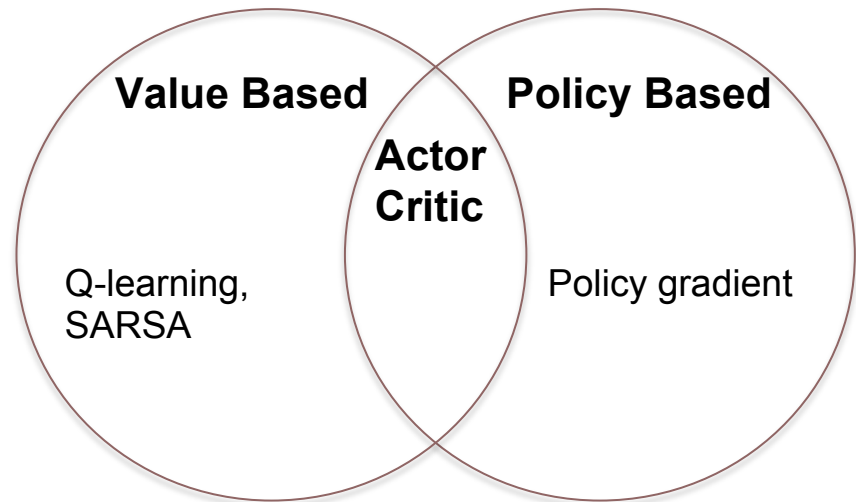
Sampling: update does not involve an expected value

- MC samples
- DP does not sample
- RL samples

MC Monte Carlo, RL Reinforcement Learning, DP Dynamic Programming

Model-free RL taxonomy

- Value Based (Critic only):
 - Learn Value Function
 - Policy is implicit (e.g. Greedy)
- Policy Based (Actor only):
 - Explicitly store Policy
 - Directly update Policy (e.g. using gradient, evolution, ...)
- Actor-Critic:
 - Learn Policy
 - Learn Value function
 - Update policy using Value Function



Q-Learning versus SARSA

One - step Q - learning :

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right]$$

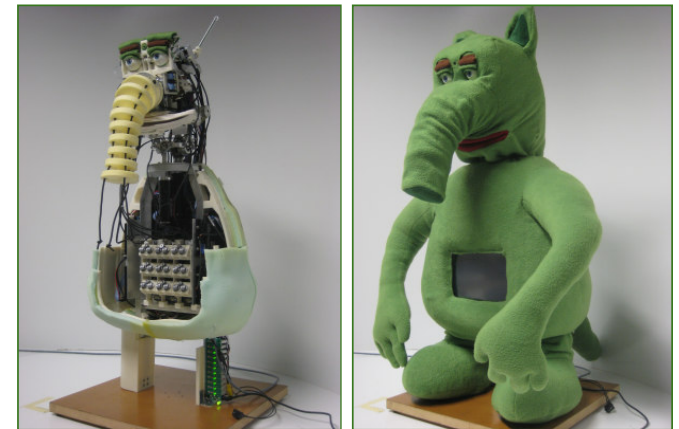
One - step SARSA

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_{t+1} + \gamma \max_a Q(s_{t+1}, a') - Q(s_t, a_t) \right]$$

With a' the actual selected action

Real life applications

- **Games**
- **Robotics**
- **Mechatronics**
- **Planning and scheduling**
- **Tutoring**
- **Routing in networks**
- **Self driving cars**
- **AlphaGO**
-



<http://probo.vub.ac.be/>

Offline learning

Ok, you've learned a value function, $Q \approx Q^*$
How do you pick actions?

Greedy Action Selection:

Always select the action that looks best:

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

Online learning

Try to perform well during learning,
but still converge to the optimal action

Balance exploration versus exploitation

N- armbandits : minimise regret.

$$\text{Re } gret = t \times r(a^*) - \sum_{i=1}^t r_i$$

Online learning

Try to perform well during learning,
but still converge to the optimal action

Balance exploration versus exploitation

N- armbandits : minimise regret.

$$\text{Regret} = \underbrace{t \times r(a^*)}_{\text{Expected total reward for optimal action}} - \underbrace{\sum_{i=1}^t r_i}_{\text{Actual reward received}}$$

Expected total reward for optimal action

Online learning

Try to perform well during learning,
but still converge to the optimal action

Balance exploration versus exploitation

N- armbandits : minimise regret.

$$\text{Regret} = t \times r(a^*) - \sum_{i=1}^t r_i$$

Expected total reward for optimal action

Total reward received

The n -Armed Bandit Problem

a_1

a_2

a_3

....

a_n

The n -Armed Bandit Problem

	a_1	a_2	a_3	a_n
Expected Values	5	6	4		7

The n -Armed Bandit Problem

a_1 a_2 a_3 a_n

Expected
Values

5 — 6 — 4 — — — 7

The n -Armed Bandit Problem

a_1 a_2 a_3 a_n

Expected
Values

5 — 6 — 4 — 7

Trial1

4

The n -Armed Bandit Problem

a_1 a_2 a_3 a_n

Expected
Values

5 — 6 — 4 — 7

Trial1

4

Trial2

7

The n -Armed Bandit Problem

a_1 a_2 a_3 a_n

Expected
Values

5 — 6 — 4 — — — 7

Trial1

4

Trial2

7

Trial3

4

The n -Armed Bandit Problem

	a_1	a_2	a_3	a_n
Expected Values	5	6	4		7
Trial1	4				
Trial2		7			
Trial3			4		
....					
Trialn					6

The n -Armed Bandit Problem

	a_1	a_2	a_3	a_n
Expected Values	5	6	4		7
Trial1	4				
Trial2		7			
Trial3			4		
....					
Trialn					6
Trialn+1	?	?	?		?

Commonly used exploration techniques in RL

ϵ -Greedy action selection: $a_t = \begin{cases} a_t^* & \text{with probability } 1 - \epsilon \\ \text{random action} & \text{with probability } \epsilon \end{cases}$

Boltzmann distribution action selection: $\frac{e^{Q_t(a)/\tau}}{\sum_{b=1}^n e^{Q_t(b)/\tau}}$

τ very large \rightarrow random action selection

τ very small \rightarrow greedy action selection

10-Armed Testbed

$n = 10$ possible actions

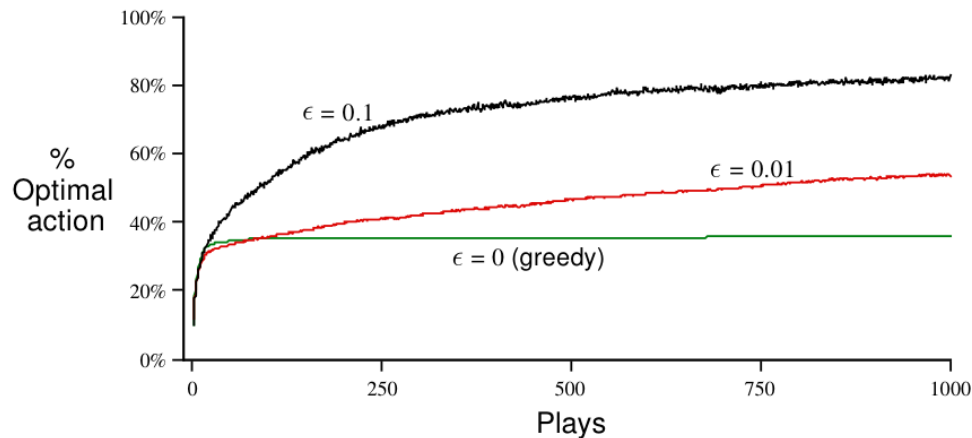
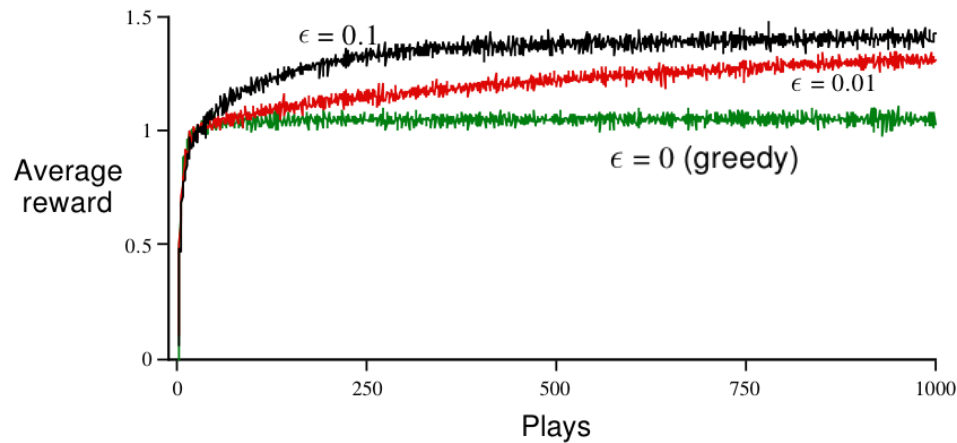
Each $Q^*(a)$ is chosen randomly from a normal distribution: $\eta(0, 1)$

each r_t is also normal: $\eta(Q^*(a_t), 1)$

1000 plays

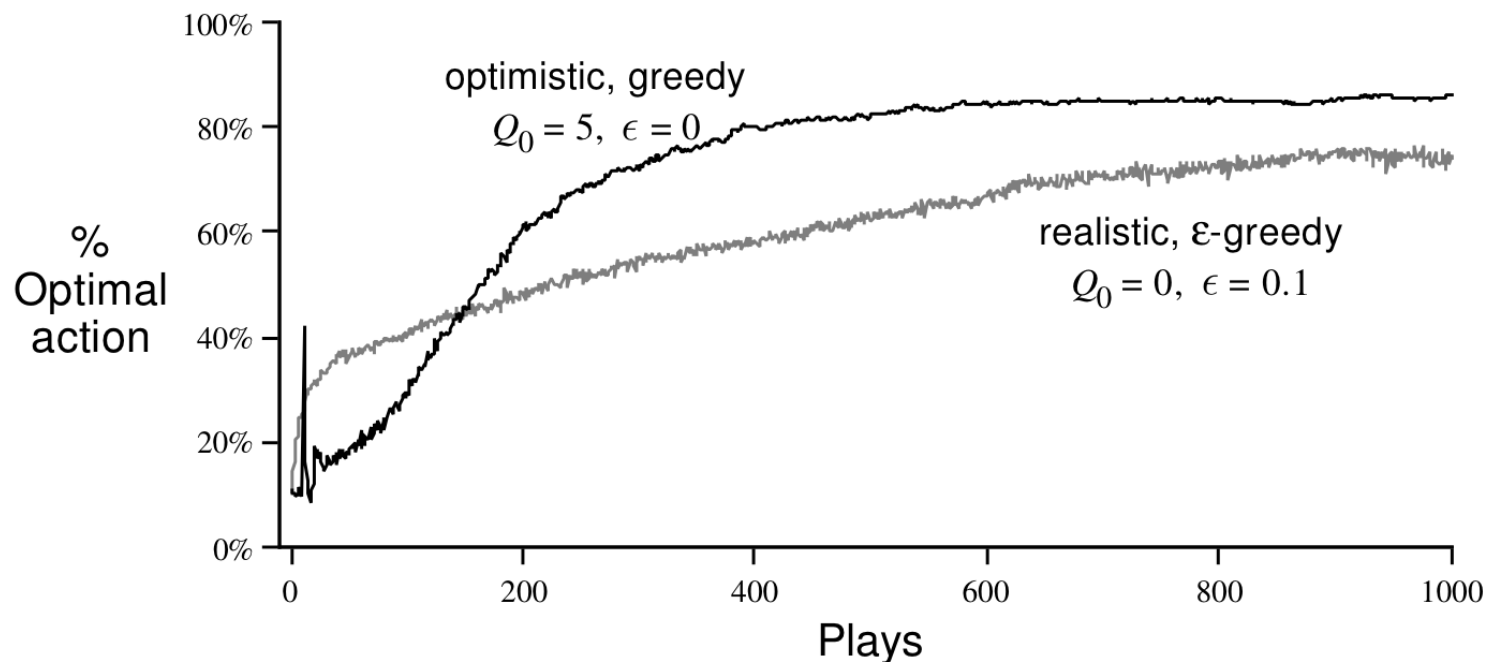
repeat the whole thing 2000 times and average the results

ϵ -Greedy Methods on the 10-Armed Testbed



Optimistic Initial Values

All methods so far depend on $Q_0(a)$, i.e., they are biased. Suppose instead we initialize the action values optimistically, i.e., on the 10-armed testbed, use $Q_0(a) = 5$ for all a



Some convergence issues

Most RL algorithms are proven to converge in Markovian environments

But RL can also be applied to non-Markovian environments, provided careful exploration

The environment can be Non-Markovian to the RL learner due to :

- limited sensors (POMDP)
- discretisation of state space
- use function approximators
- other agents in environment

The Markov Property

A state should summarize past sensations so as to retain all “essential” information, which is the information needed in order to behave optimally, i.e., it should have the Markov Property:

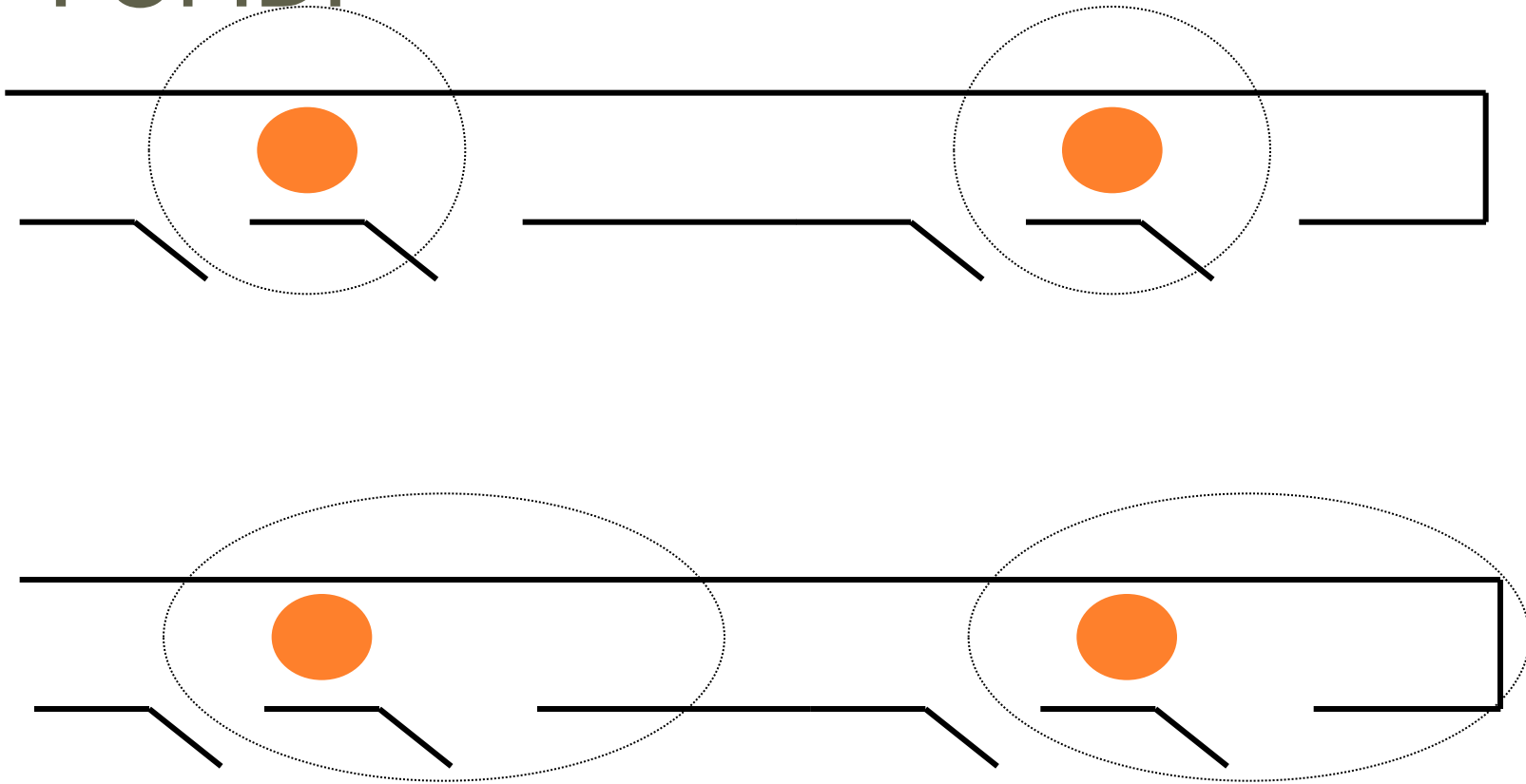
$$\Pr\{s_{t+1} = s', r_{t+1} = r \mid s_t, a_t, r_t, s_{t-1}, a_{t-1}, \dots, r_1, s_0, a_0\} = \Pr\{s_{t+1} = s', r_{t+1} = r \mid s_t, a_t\}$$

for all s', r , and histories $s_t, a_t, r_t, s_{t-1}, a_{t-1}, \dots, r_1, s_0, a_0$.

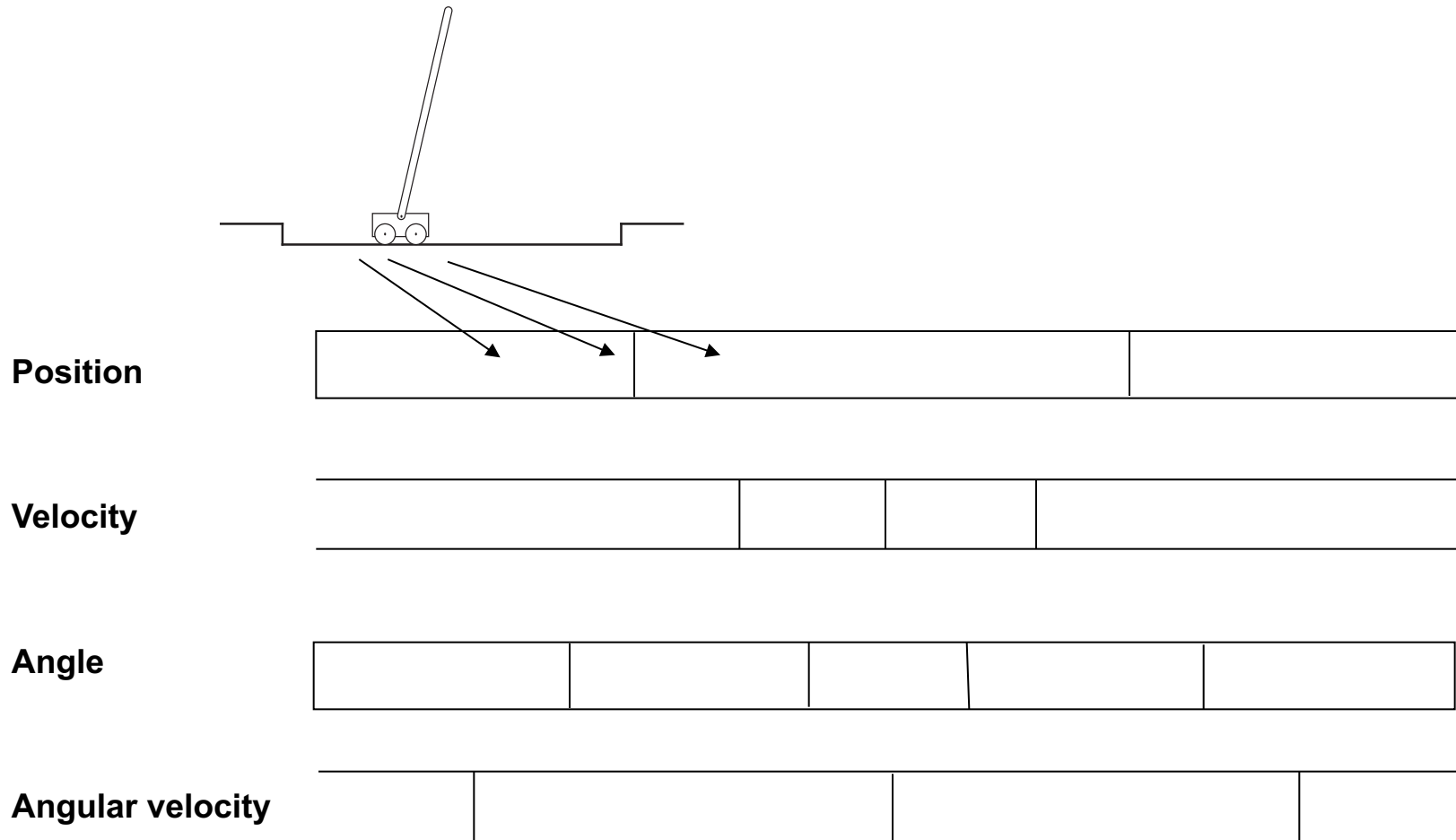
Knowing how you got is this state is not giving you extra information

Partially observable decision problem

POMDP

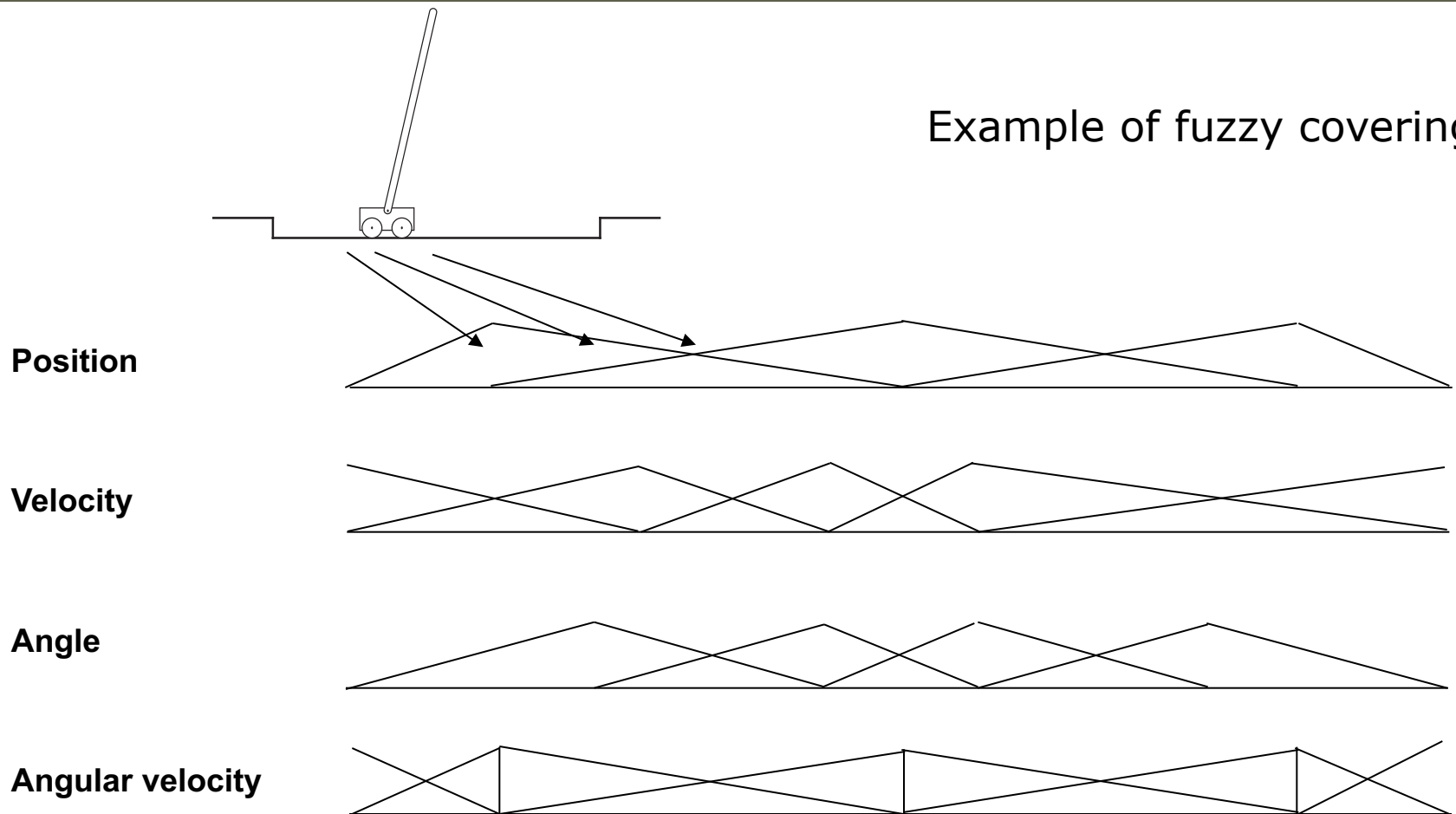


Non-makovian due to discretisation

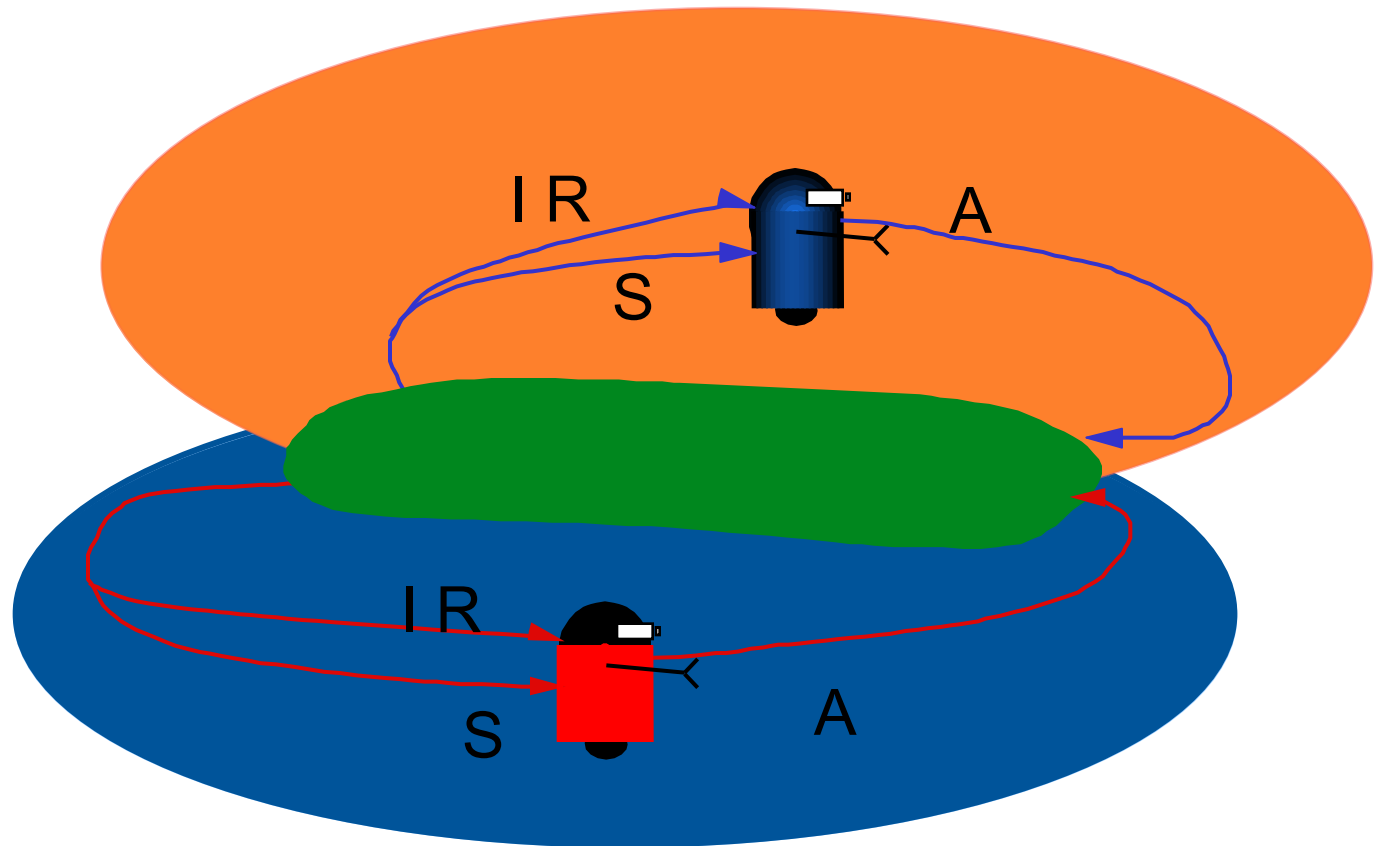


“Less” Non-makovian due to “better function approximation

Example of fuzzy covering



Non-makovian due to other agents



Some convergence issues

Q-learning is guaranteed to converge in a Markovian setting

Tsitsiklis J.N. *Asynchronous Stochastic Approximation and Q-learning*.
Machine Learning, Vol. 16:185-202, 1994.

Proof by Tsitsiklis

On the convergence of Q-learning

Stochastic approximation :

$$q_i(t+1) \leftarrow q_i(t) + \alpha_i(t) [F_i(q^i(t)) - q_i(t) + w_i(t)]$$

$$q = (q_1, \dots, q_n)$$

Q(s,a)

Noise term

"Learning factor"

q vector, but with possibly outdated components

Contraction mapping

$$\sum_{0 \leq t \leq \infty} \alpha_i(t) = \infty$$

$$\sum_{0 \leq t \leq \infty} \alpha_i^2(t) < \infty$$

$$\|F(q(t)) - Q^*\| < \|q(t) - Q^*\|$$

$$q^i(t) = (q_1(\tau_1^i(t)), \dots, q_n(\tau_n^i(t))),$$

with $0 \leq \tau_j^i(t) \leq t$

Proof by Tsitsiklis, cont.

On the convergence of Q-learning

One - step Q - learning :

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right]$$

Can be written as :

$$\begin{aligned} Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha & \left[E \left[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) \right] - Q(s_t, a_t) \right. \\ & \left. + \left(\left(r_{t+1} + \gamma \max_a Q(s_{t+1}, a) \right) - E \left[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) \right] \right) \right] \end{aligned}$$

Proof by Tsitsiklis, cont.

Relating Q-learning to stochastic approximation

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[E \left[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) \right] - Q(s_t, a_t) + \left(\left(r_{t+1} + \gamma \max_a Q(s_{t+1}, a) \right) - E \left[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) \right] \right) \right]$$

i^{th} component

Can vary in time

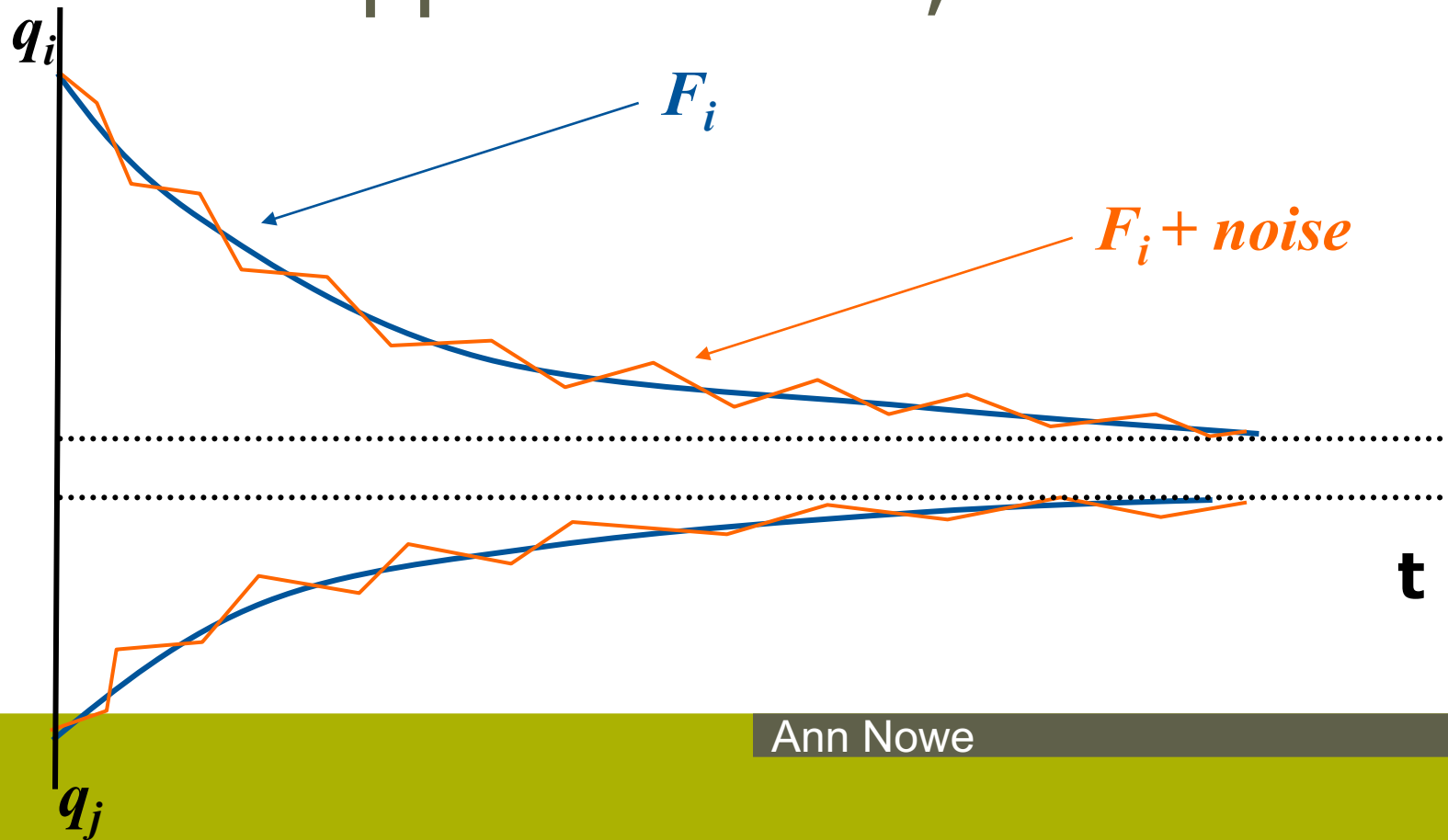
Contraction mapping
Bellman operator

Noise term

$$Q_i(t+1) \leftarrow Q_i(t) + \alpha_i(t) \left[F_i(Q^i(t)) - Q_i(t) + w_i(t) \right]$$

Proof by Tsitsiklis, cont.

Stochastic approximation, as a vector



Wrap up

Reinforcement learner told what to do, not how to do it.

Proven to converge in Markovian environments.

The more non-Markovian , the more careful the exploration must be.

When to use RL?

When no model of the environment is available or too difficult to obtain or not flexible.

However domain knowledge can be easily incorporated

$V(s)$ and $Q(s,a)$ often expressed by function approximation(e.g.Neural networks, fuzzy coverings)

Extensions for practical applications

Representations of states

- Discrete versus continuous (tile coding, fuzzy, NN, radial basis functions)
- Fixed versus adaptive discretisation

Take advantage of asynchronous updates

- Propagate interesting information more quickly:
- Prioritized sweeping, eligibility traces

Incorporate domain knowledge

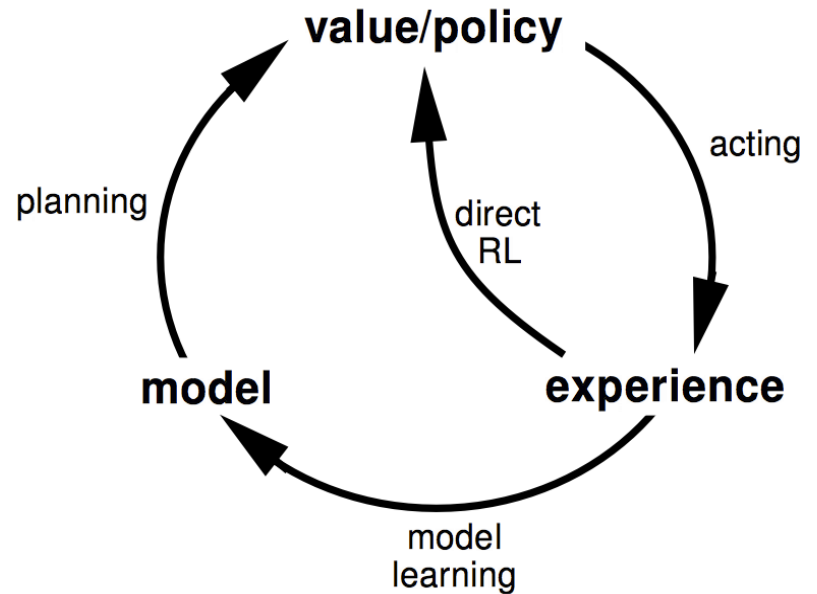
- Initialise policy
- Steer exploration
- Combine with model information (planning)

Planning

Two uses of real experience:

- ♦ **model learning**: to improve the model
- ♦ **direct RL**: to directly improve the value function and policy

Improving value function and/or policy via a model is sometimes called **indirect RL** or **model-based RL** or **planning**.



RL research @ COMO

Multi-agent reinforcement learning

Multi-type ant systems

Distributed scheduling

Distributed control

Multi-criteria

More on RL?

Visit: ai.vub.ac.be

Thanks for your attention!

