Introduction to Markov Decision Processes

Lecturers: Diederik M. Roijers & Ann Nowé

TA: Denis Steckelmacher

AI laboratory
Vrije Universiteit Brussel

Fall, 2017

About the course

- Lectures: Mondays, 2pm
  - From now until Christmas, and from February onwards
- Grading
  - Small assignments: 10%
  - Research project: 50%
  - Test: 40%

Note

- These lectures are based on:
  - Sutton and Barto
  - Papers
  - Some of our own work

Images from http://www.irasutoya.com, Sutton and Barto's book, and my PhD thesis.

Planning and learning

- Agents
- How should a single *rational* agent interact with a *sequential decision process* to maximise its expected long-term *cumulative reward* with/without an a priori model of its environment?
- With a model: *planning*
- Without a model: *reinforcement learning*

Why planning and learning?

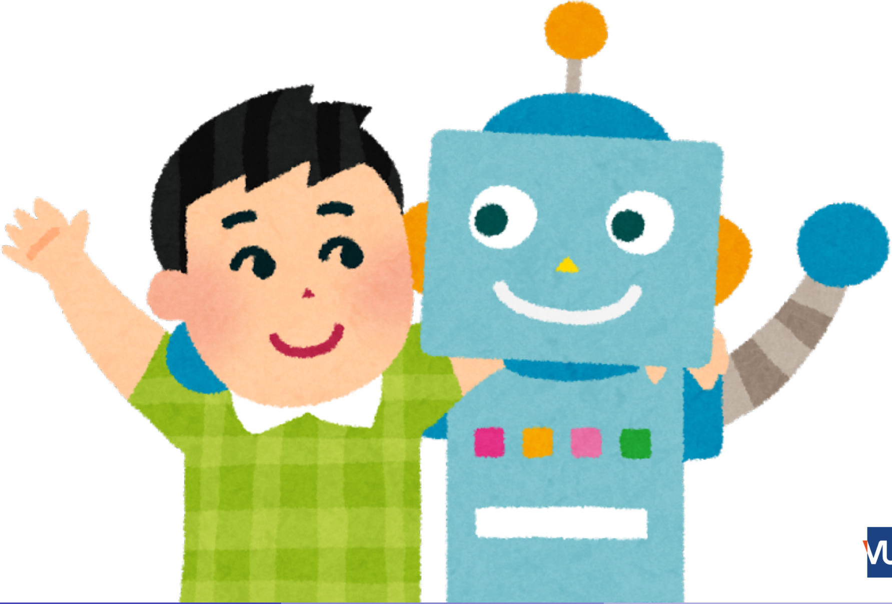
- It's hot!
- It's cool!
- It's really really useful!

Movie time

... en dan nu, een filmpje!

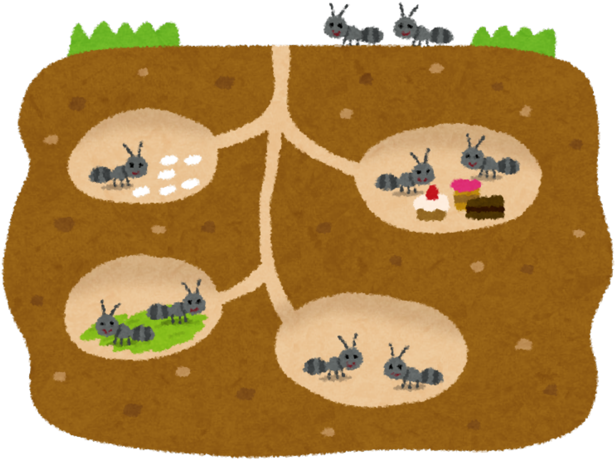
... and now, a clip!

What is an agent?



What is an agent?

What is an agent?



What is an agent?

- An agent is "anything that can be viewed as perceiving its environment through sensors and acting upon that environment through effectors" (Russel and Norvig)
- An artificial agent typically is a computer program
  - possibly embedded in specific hardware
  - *takes actions* in an environment that changes as a result of these actions.
- An autonomous agent (Franklin and Graesser, 1996)
  - can act autonomously,
  - on a user's behalf

Intelligent Autonomous Agents

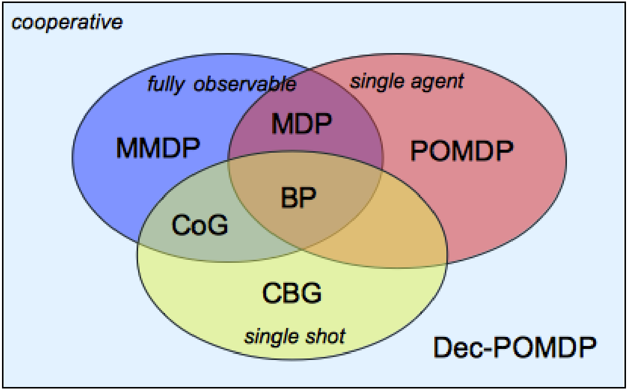
- Intelligent Autonomous Agents that
  - reason about their environment
  - reason about consequences of actions (desirability)
- Decision theory

Environments

- States
- Actions
- State transitions
- Rewards

- Sequential
- Single-agent
- Fully observable

Other models



Movie time

... en dan nu, een filmpje!

... and now, a clip!

Markov decision processes

- Formalisation of a single-agent, sequential, discrete-time, stationary environment, Markovian-observation-signal, decision problem.
- Sequential: multiple decisions over time.
- Discrete time steps: one action and state-transition per timestep
- Stationary environment: the state of the environment may change, but the dynamics do not.

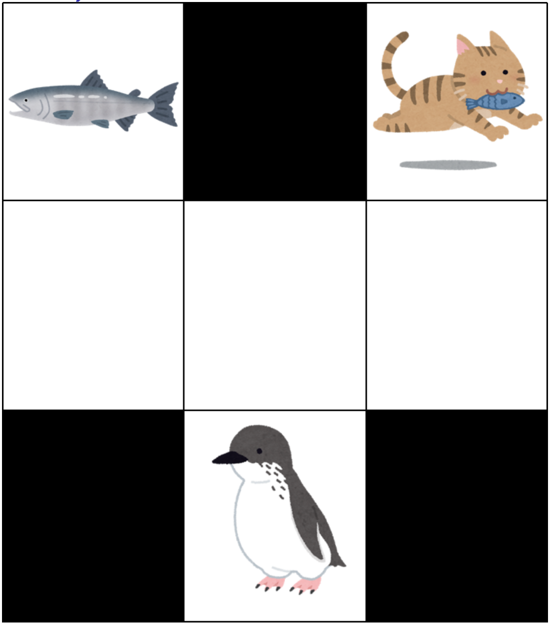
Markov decision processes

- Named after Andrey Markov (1856–1922).
- A finite MDP consists of:
  - Discrete time $t = 0, 1, 2, \dots$
  - A discrete set of states $s \in S$
  - A discrete set of actions $a \in A(s)$ for each s
  - A transition function $P^a_{ss'} = p(s'|s, a)$: probability of transitioning to state s' when taking action a at state s
  - A reward function $R^a_{ss'} = E[r|s, a, s']$: expected reward when taking action a at state s and transitioning to s'
  - A planning horizon h or discount factor γ

Markov decision processes

- Named after Andrey Markov (1856–1922).
- A finite MDP consists of:
  - Discrete time $t = 0, 1, 2, \ldots$
  - A discrete set of states $s \in S$
  - A discrete set of actions $a \in A(s)$ for each s
  - A transition function $T(s, a, s') = p(s'|s, a)$: probability of transitioning to state s' when taking action a at state s
  - A reward function $R(s, a, s') = E[r|s, a, s']$: expected reward when taking action a at state s and transitioning to s'
  - A planning horizon h or discount factor γ

A tiny (6-state) MDP



The Markov property

$$p(s_{t+1}, r_{t+1} | s_t, a_t) = p(s_{t+1}, r_{t+1} | s_t, a_t, r_t, s_{t-1}, a_{t-1}, \ldots, r_1, s_0, a_0)$$

- Therefore, the history does not yield more information about subsequent states and rewards than the current state.
- Current state is a sufficient statistic for the history.
- A Markovian/Markov state/observation signal
- Important: only need to condition on the latest observation (s_t)

Markov?

Markov?

Can we make it Markov?

The credit-assignment problem

- Sequential aspect $\rightarrow$ credit assignment problem
- Suppose an agent takes a long sequence of actions, at the end of which it receives a single positive reward?
- How can it determine to what degree each action in that sequence deserves the *credit* for the reward?

Return

- The agent's goal is to maximise the expected *return*, the sum over the rewards received.
- Three settings:
    - Finite-horizon
    - Infinite-horizon continuing
    - Infinite-horizon episodic

Return: finite-horizon

- In a finite-horizon task, the return is defined as:

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \ldots = \sum_{k=0}^{h} \gamma^k r_{t+k+1}$$

- $0 \leq \gamma \leq 1$; can be 1 only in a finite-horizon setting!

Return: infinite-horizon continuing

- In an infinite-horizon task, the return is defined as:

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

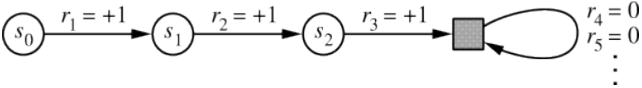
- $0 \leq \gamma < 1$
- It never stops

Return: infinite-horizon episodic

- In an infinite-horizon task, the return is defined as:

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

- $0 \leq \gamma < 1$
- When the agent reaches a terminal state, it stops
- Return after reaching a terminal state (i.e., all rewards) is 0 by definition.
- Terminal state is an *absorbing state*:



Value functions

- The *state-value function* of a policy π is:

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

- The *stateless value function* of a policy π is:

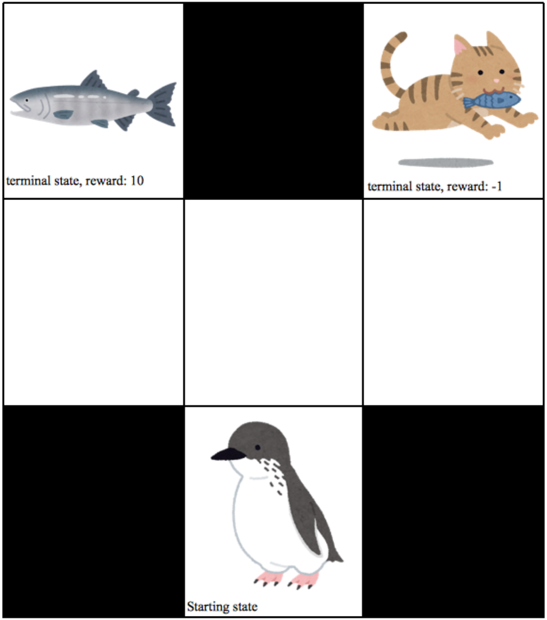
$$V^\pi = E_\pi\Big[R_t | \mu_0\Big] = E_\pi\Big[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | \mu_0\Big]$$

where μ_0 is a distribution over initial states.

- The *state-action-value* of a policy π is:

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

A tiny (6-state) MDP



Bellman equation

- The definition of V^π can be rewritten recursively by making use of the transition model, yielding the *Bellman equation*:

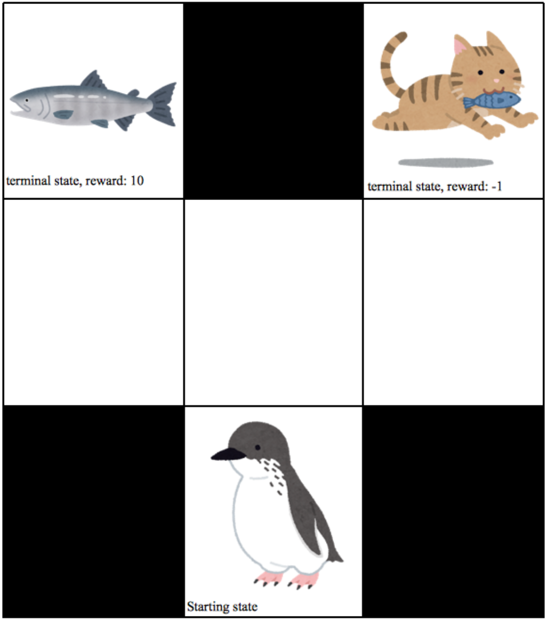
$$V^\pi(s) = \sum_a \pi(s,a) \sum_{s'} P^a_{ss'} \left[R^a_{ss'} + \gamma V^\pi(s') \right]$$

- This is a set of linear equations, one for each state, the solution of which defines the value of π
- A similar recursive definition holds for Q-values:

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

- Equations named after Richard E. Bellman (1920–1984).

A tiny (6-state) MDP



Bellman optimality equations

$$V^* = \max_{a \in A} \sum_{s'} P^a_{ss'} \left[R^a_{ss'} + \gamma V^*(s') \right]$$

$$Q^*(s,a) = \sum_{s'} P^a_{ss'} \left[R^a_{ss'} + \gamma \max_{a\ \in A} Q^*(s', a') \right]$$

Why optimal value functions are useful

An optimal policy is *greedy* with respect to V^* or Q^*:

$$\pi^*(s) \in \arg\max_a Q^*(s,a) = \arg\max_a \left[R^a_{ss'} + \gamma \sum_{s'} P^a_{ss'} V^*(s') \right]$$

Movie time

... en dan nu, een filmpje!

... and now, a clip!

Planning

- Given an MDP, find $V^*(s)/Q^*(s,a)$ and π^*

$$Q^*(s,a) = \sum_{s'} P^a_{ss'} \left[R^a_{ss'} + \gamma \max_{a\ \in A} Q^*(s',a') \right]$$

$$\pi^*(s) \in \arg\max_a Q^*(s,a) = \arg\max_a \left[R^a_{ss'} + \gamma \sum_{s'} P^a_{ss'} V^*(s') \right]$$

- Howard (1960): for any additive infinite-horizon MDP, there exists at least one deterministic stationary policy that is optimal.

- $\pi : S \rightarrow A$

Planning

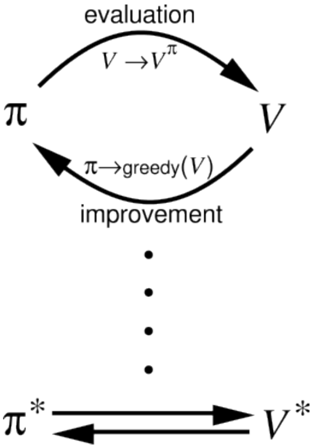
- Given an MDP, find $V^*(s)/Q^*(s,a)$ and π^*

$$Q^*(s,a) = \sum_{s'} P^a_{ss'} \left[R^a_{ss'} + \gamma \max_{a\ \in A} Q^*(s',a') \right]$$

$$\pi^*(s) \in \arg\max_a Q^*(s,a) = \arg\max_a \left[R^a_{ss'} + \gamma \sum_{s'} P^a_{ss'} V^*(s') \right]$$

- Howard (1960): for any additive infinite-horizon MDP, there exists at least one deterministic stationary policy that is optimal.
- $\pi : S \rightarrow A$
- Why does Howard's theorem not hold for finite-horizon problems?

How? A dynamic programming approach!



How? A dynamic programming approach!

- Iterately improve
  - the value estimates
  - the (deterministic stationary) policy
- Until *convergence*

Policy evaluation

- Exploit the recursive nature of the Bellman equation
- Initial value function $V_0(s)$ is chosen arbitrarily (e.g., 0 for every s)
- Turn Bellman equation into Policy evaluation update update rule:

$$V_{k+1}(s) \leftarrow \sum_a \pi(s,a) \sum_{s'} P^a_{ss'} \left[R^a_{ss'} + \gamma V_k(s') \right]$$

- Apply to every state in each iteration
- Iterate until fixed point $\lim_{k\to\infty} \forall s : V_k(s) = V_{k+1}(s)$

Policy evaluation

- Proven to converge
- $\lim_{k \to \infty} \forall s : V_k(s) = V_{k+1}(s) = V^{\pi}(s)$
- Upper bound on complexity $O(|S|^3)^\star$

*M.L. Littman, T.L. Dean, L.P. Kaelbling — On the complexity of solving Markov decision problems, UAI, 1995

Policy improvement

- Find *improvable states*: s where is a better action $a \neq \pi(s)$

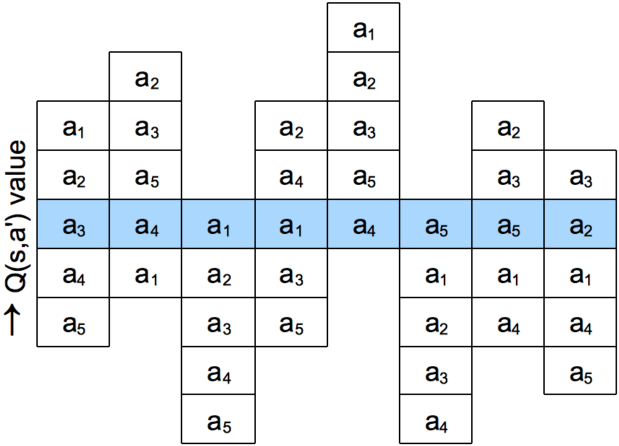
$$Q^{\pi}(s,a) = \sum_{s'} P^{a}_{ss'} \left[R^{a}_{ss'} + \gamma V^{\pi}(s') \right] > V^{\pi}(s)?$$

- *Policy improvement theorem*: changing π to take a better action (according to above equation) in one or more improvable states will increase its value:

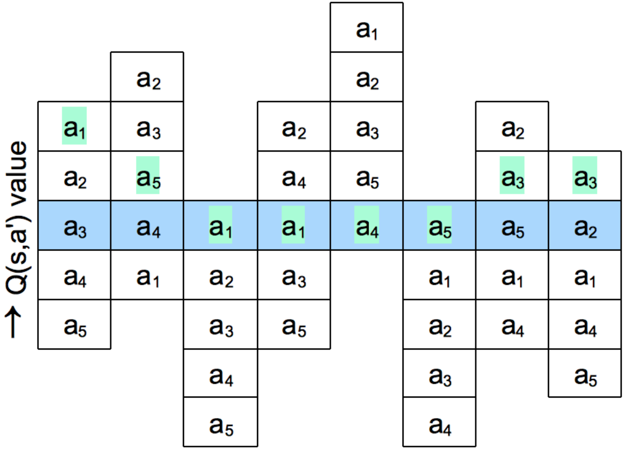
$$\forall s \in S : Q^{\pi}(s, \pi'(s)) \geq V^{\pi}(s) \Rightarrow \forall s \in S, V^{\pi'}(s) \geq V^{\pi}(s)$$

- Why does this always converge?

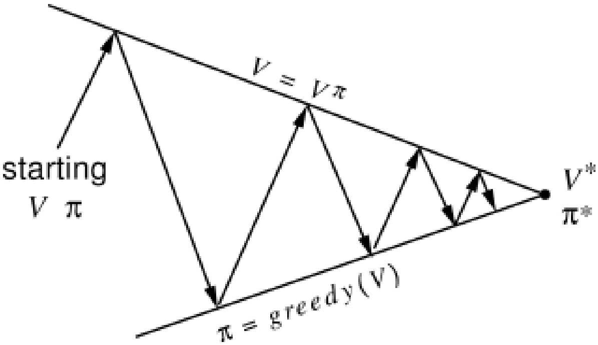
Policy improvement illustration



Policy improvement illustration



Policy iteration



Policy iteration

- Begin with arbitrary policy
- Repeat:
  - Policy evaluation (PE) (until convergence)
  - Policy improvement (PI) (on one or more states)
- PI
  - On all improvable states $\rightarrow$ Howard's (1960) policy iteration
  - On one improvable state $\rightarrow$ Simple policy iteration
  - Mansour and Singh's (1999) Randomised PI
  - (Recursive) Batch-switching PI [Kalyanakrishnan, Mall, and Goyal (2016), Gupta and Kalyanakrishnan (2017)]

Policy iteration

- Begin with arbitrary policy
- Repeat:
  - Policy evaluation (PE) (until convergence)
  - Policy improvement (PI) (on one or more states)
- PI
  - On all improvable states → Howard's (1960) policy iteration
  - On one improvable state → Simple policy iteration
  - Mansour and Singh's (1999) Randomised PI
  - (Recursive) Batch-switching PI [Kalyanakrishnan, Mall, and Goyal (2016), Gupta and Kalyanakrishnan (2017)]
  - In practice, Howard's PI seems to be most effective

Value iteration

- We do not have to wait for policy evaluation to complete improving the policy
- Value iteration (VI) integrates evaluation and improvement in one update rule:

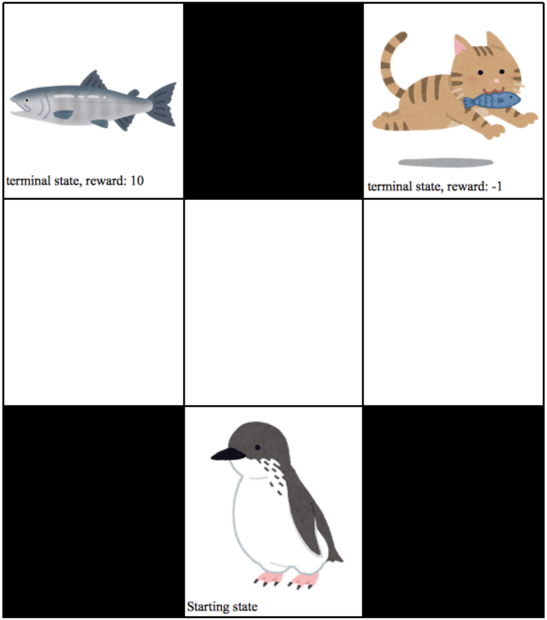
$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} P^a_{ss'} \left[R^a_{ss'} + \gamma V_k(s') \right]$$

- This can also be written:

$$V_{k+1}(s) \leftarrow \max_a Q_{k+1}(s,a),$$
$$Q_{k+1}(s,a) \leftarrow \sum_{s'} P^a_{ss'} \left[R^a_{ss'} + \gamma V_k(s') \right]$$

- Guaranteed to converge to $V^*(s)$ and π^*

A tiny (6-state) MDP



Efficiency of dynamic programming

- An MDP has $|A|^{|S|}$ deterministic stationary policies
- Worst-case computational complexity of DP is polynomial in $|S|$, $|A|$, and $\frac{1}{1-\gamma} \log\left(\frac{1}{1-\gamma}\right)^\star$
- MDP planning can also be done with *linear programming*
  - Better runtime guarantees, but impractical for large MDPs

*M.L. Littman, T.L. Dean, L.P. Kaelbling — On the complexity of solving Markov decision problems, UAI, 1995,