Aalto University
School of Electrical Engineering

ELEC-E8125 Reinforcement Learning
Solving discrete MDPs

Ville Kyrki

1.10.2019

Today

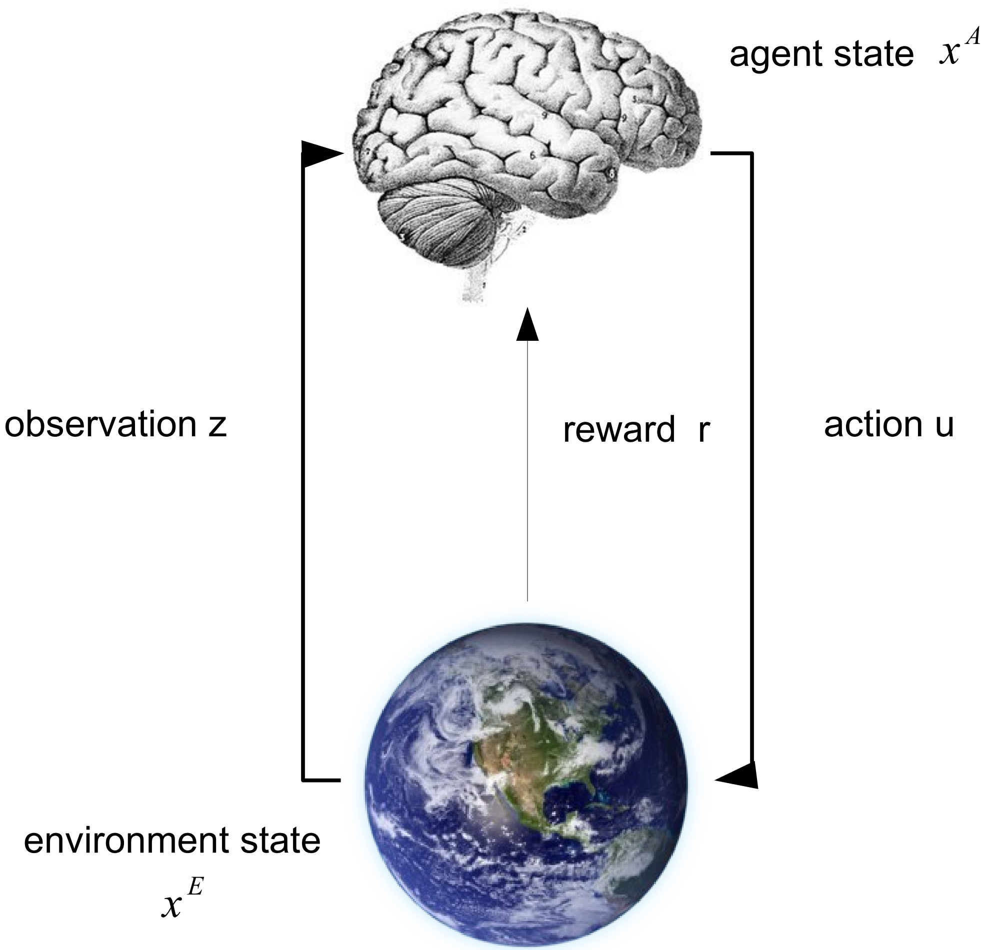
- Markov decision processes

Learning goals

- Understand MDPs and related concepts.
- Understand value functions.

- Be able to implement value iteration.

Markov decision process



MDP

Environment fully observable

$o = x^E = x^A$

Defined by dynamics

$P(x_{t+1} | x_t, u_t)$

And reward function

$r_t = r(x_{t+1}, x_t)$

Solution e.g.

$u^*_{1,\dots,T} = max_{u_1,\dots,u_T} \sum_{t=1}^{T} r_t$

Represented as policy

$u = \pi(x^A)$

Let's build this from its building blocks.

Markov property

- "Future is independent of past given the present"
- State sequence S is Markov iff ← "if and only if"

$$P(S_{t+1}|S_t) = P(S_{t+1}|S_1, \ldots, S_t)$$

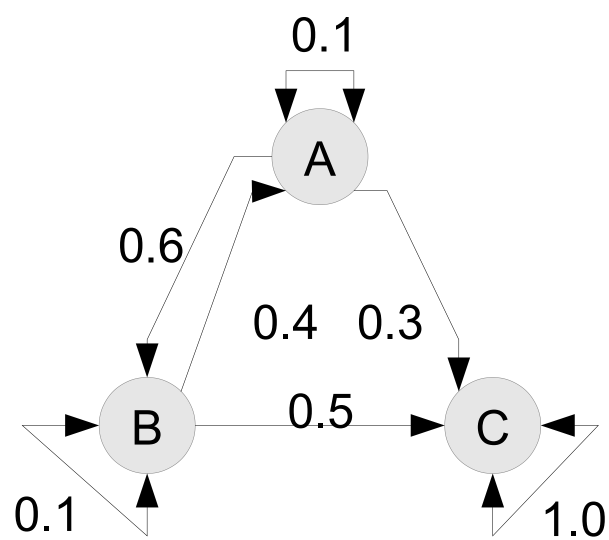
- State captures all history.
- Once state is known, history may be thrown away.

Markov process

No “decision” here!

- Markov process is a memoryless random process, i.e. random state sequence *S* with the Markov property.

- Defined as *(X,T)*
  - *X:* set of states
  - *T*: X *x X* → *[0,1]* state transition function
    - $T_t(x, x') = P(x_{t+1} = x' | x_t = x)$
    - *P* can be represented as transition probability matrix
- State sequences called *episodes*



How to calculate probability of a particular episode?
Starting from A, what is the probability of A,B,C?

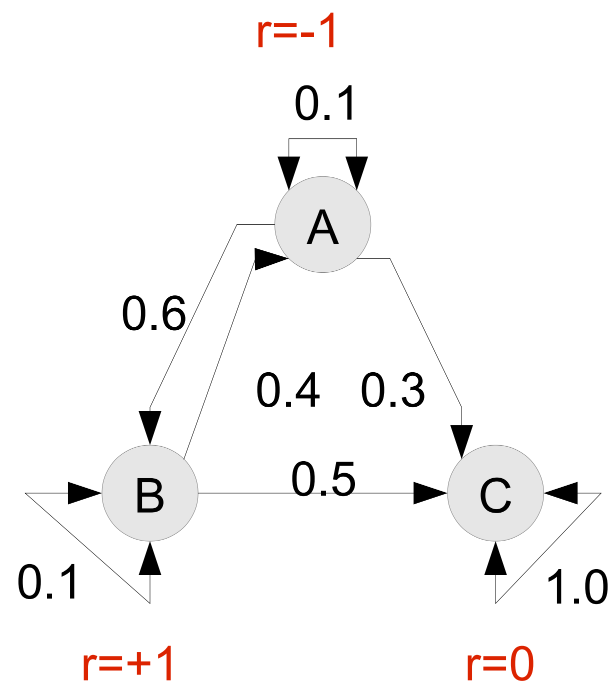
Markov reward process

Still no "decision"!

- Markov reward process = Markov process with rewards
- Defined by (X, *T, r, γ)
  - *X, T* :as above
  - *r*: X $\rightarrow \Re$ reward function
  - γ *[0,1]:* discount factor
- Accumulated rewards in finite (*H* steps) or infinite horizon

$$\sum_{t=0}^{H} \gamma^t r_t \qquad \sum_{t=0}^{\infty} \gamma^t r_t$$

- *Return R*: accumulated rewards from time t



$$R_t = \sum_{k=0}^{H} \gamma^k r_{t+k+1}$$

Why discount?

Return of (A,B,C), γ=0.9?

State value function for MRPs

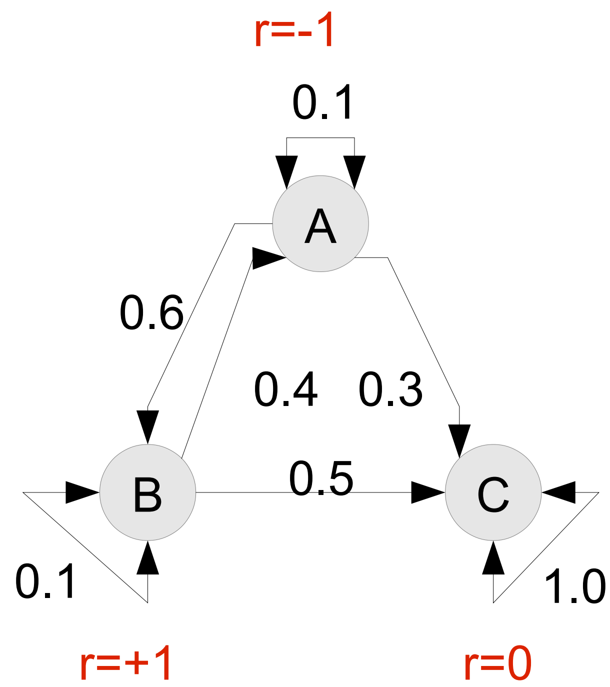
- State value function *V(x)* is expected cumulative rewards starting from state *x*

$$V(x) = E[R_t | x_t = x]$$

- Value function can be defined by Bellman equation

$$V(x) = E[R_t | x_t = x]$$

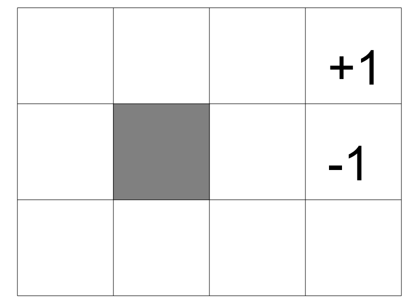
$$V(x) = E[r_{t+1} + \gamma V(x_{t+1}) | x_t = x]$$

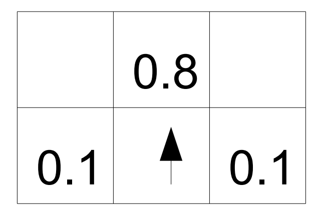


What is the value function for γ=0?

Markov decision process (MDP)

- Markov decision process defined by (X, *U*, *T*, *R*, γ)
  - *X*, γ : as above
  - *U:* set of actions (inputs)
  - *T: X x U x X* $\rightarrow$ *[0,1]*
    $$T_t(x, u, x') = P(x_{t+1} = x' | x_t = x, u_t = u)$$
  - *R: X x U x X* $\rightarrow \Re$ reward function
    $$r_t(x, u, x') = r(x_{t+1} = x', x_t = x, u_t = u)$$





- Goal: Find policy $\pi(x)$ that maximizes cumulative rewards.

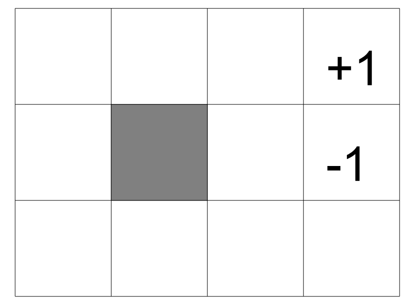
Grid world example!

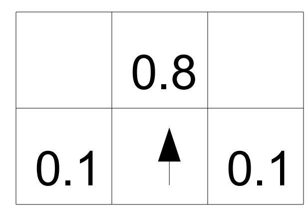
Policy

- Deterministic policy $\pi(X){:}X \rightarrow U$ is mapping from states to actions.

- Stochastic policy $\pi(u|x){:}\ X,U \rightarrow [0,1]$ is a distribution over actions given states.

- Optimal policy $\pi^*(x)$ is a policy that is better or equal than any other policy (in terms of cumulative rewards)
  - There always exists a deterministic optimal policy for a MDP.





What is grid world optimal policy!

MDP value function

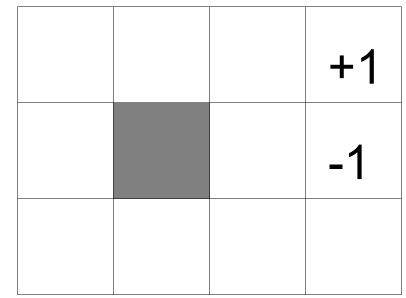
- *State-value function* of an MDP is expected return starting from state *s* and following policy π.

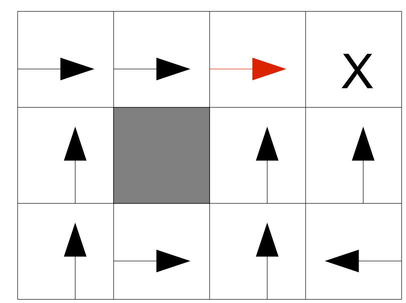
$$V_\pi(x) = E_\pi[R_t | x_t = x]$$

- Can be decomposed into immediate and future components using Bellman expectation equation

$$V_\pi(x) = E_\pi[r_t + \gamma V_\pi(x_{t+1}) | x_t = x]$$

$$V_\pi(x) = \sum_{x'} T(x, \pi(x), x') r(x, \pi(x), x') + \gamma \sum_{x'} T(x, \pi(x), x') V_\pi(x')$$



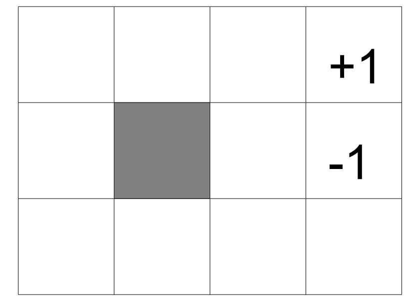


What is value function here?

Action-value function

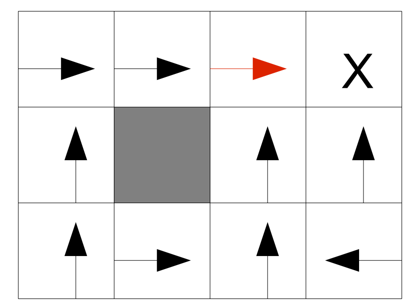
- *Action-value function Q* is expected return starting from state *s*, taking action *a*, and then following policy π.

$$Q_\pi(x,u)=E_\pi[R_t|x_t=x,u_t=u]$$



- Using Bellman expectation equation

$$Q_\pi(x,u)=E_\pi[r_t+\gamma Q_\pi(x_{t+1},u_{t+1}|x_t=x,u_t=u)]$$
$$Q_\pi(x,u)=\sum_{x'} T(x,u,x')r(x,u,x')$$
$$+\gamma\sum_{x'} T(x,u,x')Q_\pi(x',\pi(x'))$$



Optimal value function

- Optimal state-value function is maximum value function over all policies.

$$V^*(x) = max_\pi V_\pi(x)$$

- Optimal action-value function is maximum action-value function over all policies.

$$Q^*(x, u) = max_\pi Q_\pi(x, u)$$

- All optimal policies achieve optimal state- and action-value functions.

What is the optimal action if we know Q^*?
What about V^*?

Optimal policy vs optimal value function

- Optimal policy for optimal action-value function

$$\pi^*(x) = arg\,max_u Q^*(x, u)$$

- Optimal action for optimal state-value function

$$\pi^*(x) = arg\,max_u E_{x'}[r(x, u, x') + \gamma V^*(x')]$$
$$\pi^*(x) = arg\,max_u \sum_{s'} T(x, u, x')\big(r(x, u, x') + \gamma V^*(x')\big)$$

Value iteration

Do you notice that this is an expectation?

- Starting from $V_0^*(x) = 0 \quad \forall\, x$
  iterate

$$V_{i+1}^*(x) = max_u \sum_{x'} T(x, u, x') \left(r(x, u, x') + \gamma V_i^*(x') \right)$$

  until convergence.

- Value iteration converges to *V*(x)*.

Compare to

$$G^*(x) = min_u \left\{ l(x, u) + G^*(f(x, u)) \right\}$$

from last week!

Iterative policy evaluation

- Problem: Evaluate value of policy π.
- Solution: Iterate Bellman expectation back-ups.
- $V_1 \rightarrow V_2 \rightarrow \ldots \rightarrow V_\pi$
- Using synchronous back-ups:
  - For all states *x*
  - Update $V_{k+1}(x)$ from $V_k(x')$
  - Repeat

From slide 11.

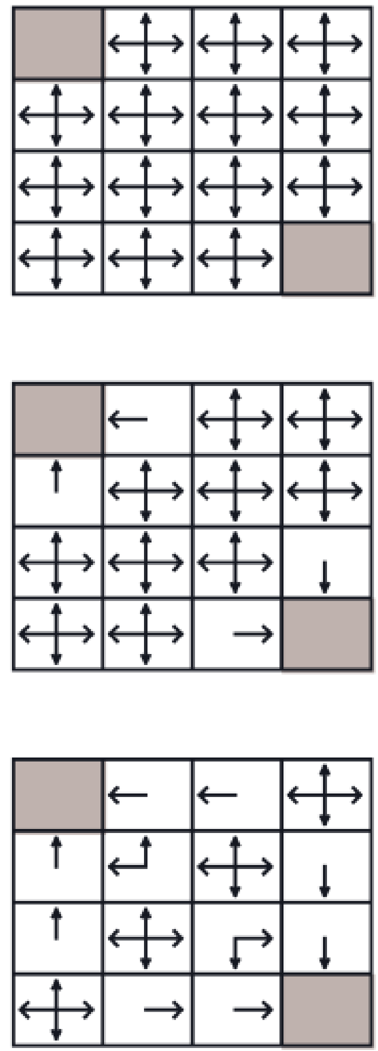
$$V_{k+1}(x) = \sum_{x'} T(x, \pi(x), x')\left(r(x, \pi(x), x') + \gamma V_k(x')\right)$$

$$V_{k+1}(x) = \sum_u \pi(u|x) \cdot \sum_{x'} T(x, u, x')\left(r(x, u, x') + \gamma V_k(x')\right)$$

Note: Starting point can be random policy.

V

Greedy policy



$k = 0$

0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0

$k = 1$

0.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	0.0

$k = 2$

0.0	-1.7	-2.0	-2.0
-1.7	-2.0	-2.0	-2.0
-2.0	-2.0	-2.0	-1.7
-2.0	-2.0	-1.7	0.0

	1	2	3
4	5	6	7
8	9	10	11
12	13	14	

r=-1 for all actions

Policy improvement and policy iteration

- Given a policy π, it can be improved by
  - Evaluating V_{π}
  - Forming a new policy by acting greedily with respect to V_{π}

- This always improves the policy.

- Iterating multiple times called *policy* iteration.
  - Converges to optimal policy.

Computational limits – Value iteration

- Complexity $O(|U||X|^2)$ per iteration.
- Effective up to medium size problems (millions of states).

- Complexity when applied to action-value function $O(|U|^2|X|^2)$ per iteration.

Summary

- Markov decision processes represent environments with uncertain dynamics.
- Deterministic optimal policies can be found using state-value or action-value functions.
- Dynamic programming is used in value iteration and policy iteration algorithms.

Next week: From MDPs to RL

- Readings
  - SB Ch. 5-5.4, 5.6, 6-6.5