Aalto University
School of Electrical Engineering

ELEC-E8125 Reinforcement Learning
Solving discrete MDPs

Ville Kyrki

15.9.2020

Today

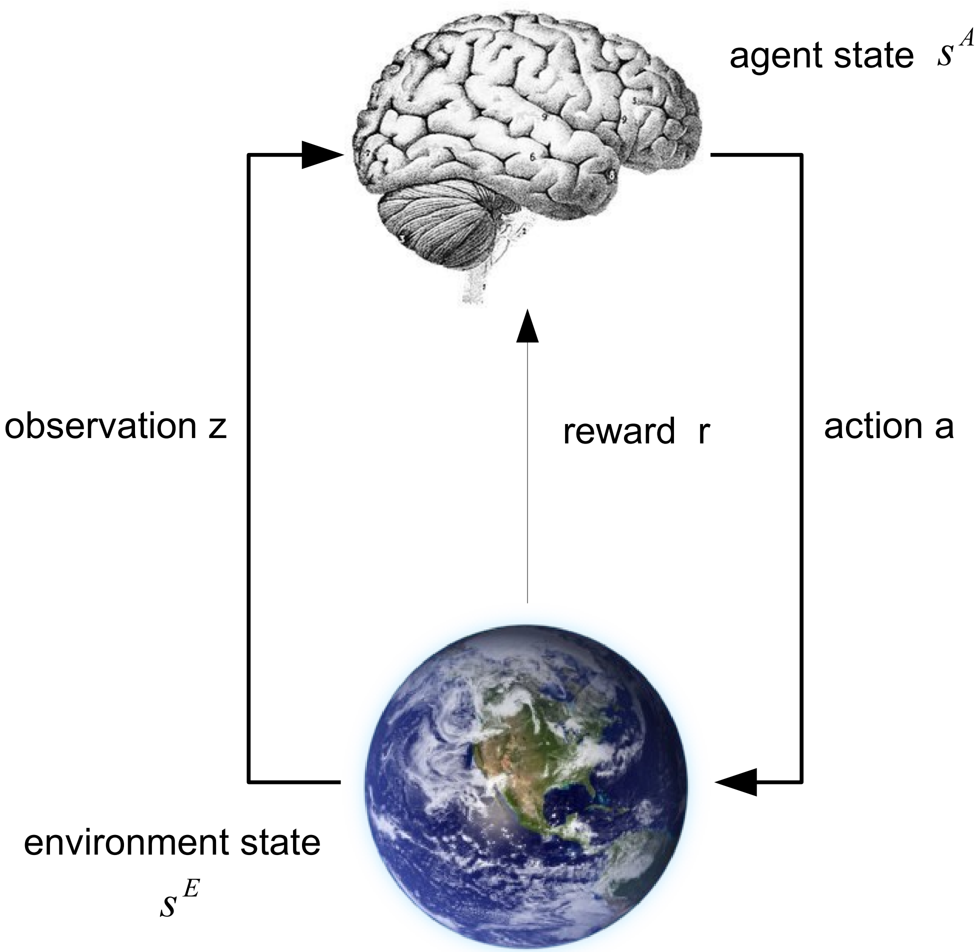
- Markov decision processes

Learning goals

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

- Be able to implement value iteration.

Markov decision process



MDP
Environment observable
$o = s^E = s^A$

Defined by dynamics
$P(s_{t+1} | s_t, a_t)$

And reward function
$r_t = r(s_{t+1}, s_t)$

Solution e.g.
$a^*_{1,\dots,T} = max_{a_1,\dots,a_T} \sum_{t=1}^{T} r_t$

Represented as policy
$a = \pi(s^A)$

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"

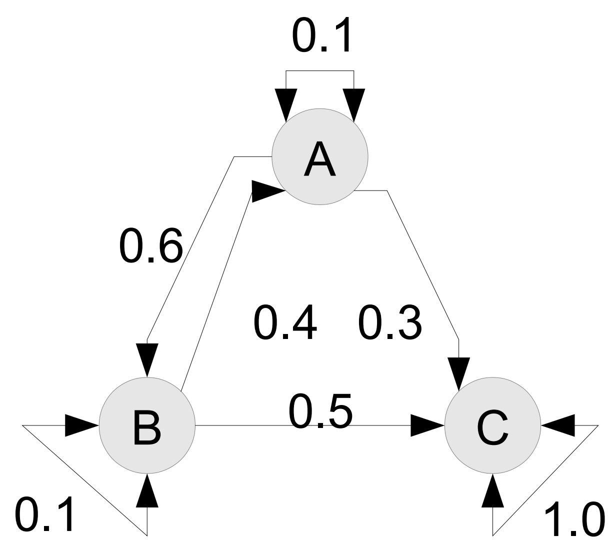
$$P\left(S_{t+1} \middle| S_t\right) = P\left(S_{t+1} \middle| S_{1}, \ldots, S_t\right)$$

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



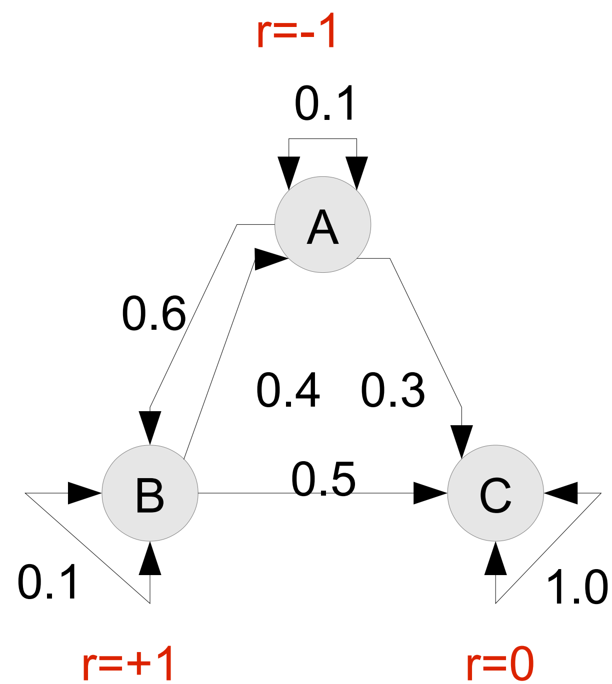
How to calculate probability of a particular episode?
Starting from A, what is the probability of A,B,C?

Markov reward process

Still no "decision"!

- Markov reward process = Markov process with rewards
- Defined by (S, *T*, *r*, γ)
  - *S*, *T* :as above
  - *r*: S $\rightarrow \Re$ 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 G*: accumulated rewards from time t

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

Why discount?

Return of (A,B,C), γ=0.9?

State value function for MRPs

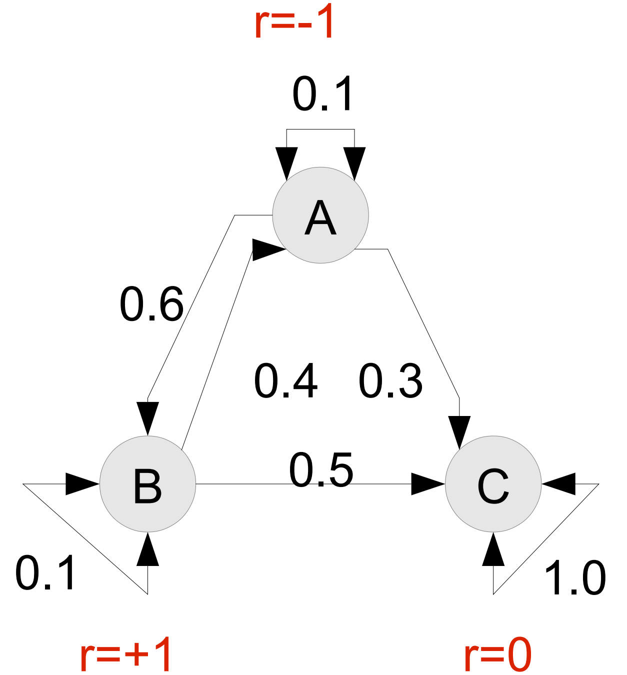
- State value function *V(s)* is expected cumulative rewards starting from state *s*

$$V(s)=E[G_t|s_t=s]$$

- Value function can be defined by Bellman equation

$$V(s)=E[G_t|s_t=s]$$

$$V(s)=E[r_{t+1}+\gamma V(s_{t+1})|s_t=s]$$

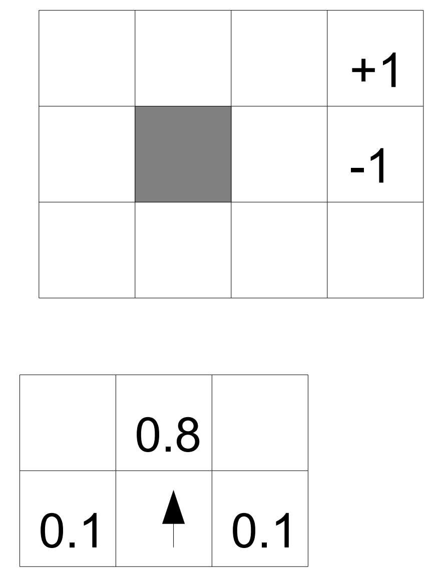


What is the value function for γ=0?

Markov decision process (MDP)

- Markov decision process defined by (S, *A*, *T*, *R*, γ *)*
  - *S*, γ : as above
  - *A:* set of actions (inputs)
  - *T: S x A x S* $\rightarrow$ *[0,1]*
    $T_t(s,a,s')=P(s_{t+1}=s'|s_t=s,a_t=a)$
  - *R: S x A x S* $\rightarrow \Re$ reward function
    $r_t(s,u,s')=r(s_{t+1}=s',s_t=s,a_t=a)$

- Goal: Find policy π*(s)* that maximizes cumulative rewards.



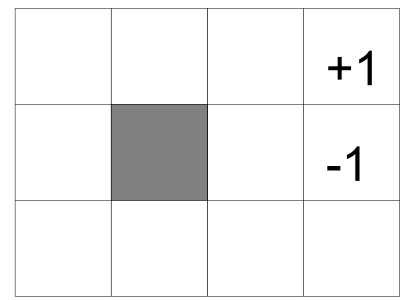
Grid world example!

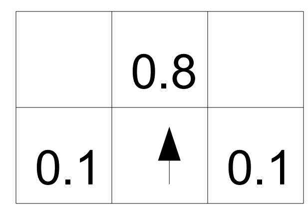
Policy

- Deterministic policy $\pi(S): S \rightarrow A$ is mapping from states to actions.

- Stochastic policy $\pi(a|s): S,A \rightarrow [0,1]$ is a distribution over actions given states.

- Optimal policy $\pi^*(s)$ 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

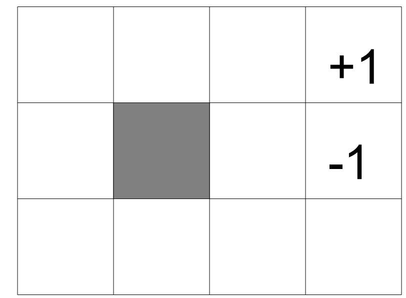
- *State-value function* of an MDP is expected return starting from state *s* and following policy π.

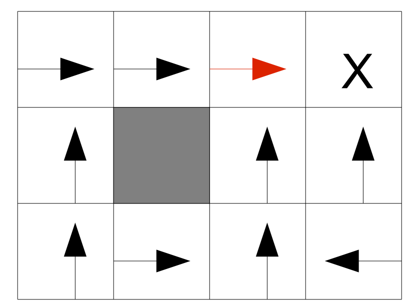
$$V_\pi(s) = E_\pi[G_t | s_t = s]$$

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

$$V_\pi(s) = E_\pi[r_t + \gamma V_\pi(s_{t+1}) | s_t = s]$$

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



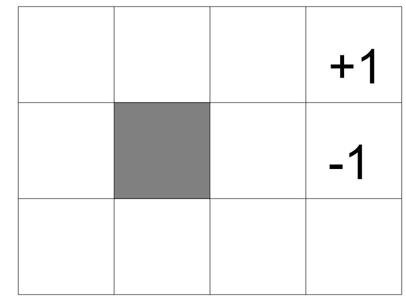


What is value function here?

Action-value function

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

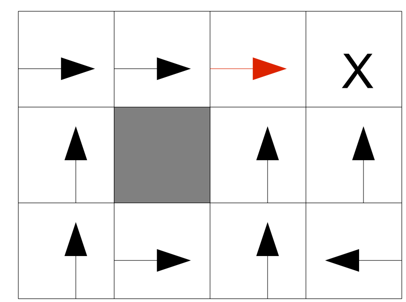
$$Q_\pi(s,a)=E_\pi[G_t|s_t=s,a_t=a]$$



- Using Bellman expectation equation

$$Q_\pi(s,a)=E_\pi[r_t+\gamma Q_\pi(s_{t+1},a_{t+1}|s_t=s,a_t=a)]$$

$$Q_\pi(s,a)=\sum_{s'} T(s,a,s')r(s,a,s')$$
$$+\gamma\sum_{s'} T(s,a,s')Q_\pi(s',\pi(s'))$$



Optimal value function

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

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

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

$$Q^*(s,a) = max_\pi Q_\pi(s,a)$$

- 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

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

- Optimal action for optimal state-value function

$$\pi^*(s) = arg\,max_a E_{s'}[r(s,a,s') + \gamma V^*(s')]$$

$$\pi^*(s) = arg\,max_a \sum_{s'} T(s,a,s')\left(r(s,a,s') + \gamma V^*(s')\right)$$

Value iteration

Do you notice that this is an expectation?

- Starting from $V_0^*(s) = 0 \quad \forall s$

  iterate

$$V_{i+1}^*(s) = max_a \sum_{s'} T(s, a, s')\left(r(s, a, s') + \gamma V_i^*(s')\right)$$

  until convergence.

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

Compare to

$$G^*(s) = min_a\left\{l(s, a) + G^*(f(s, a))\right\}$$

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 *s*
  - Update $V_{k+1}(s)$ from $V_k(s')$
  - Repeat

From slide 11.

$$V_{k+1}(s) = \sum_{s'} T(s, \pi(s), s')\big(r(s, \pi(s), s') + \gamma V_k(s')\big)$$

$$V_{k+1}(s) = \sum_a \pi(a|s) \cdot \sum_{s'} T(s, a, s')\big(r(s, a, s') + \gamma V_k(s')\big)$$

Aalto University
School of Electrical Engineering

Note: Starting point can be random policy.

V

$k = 0$

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

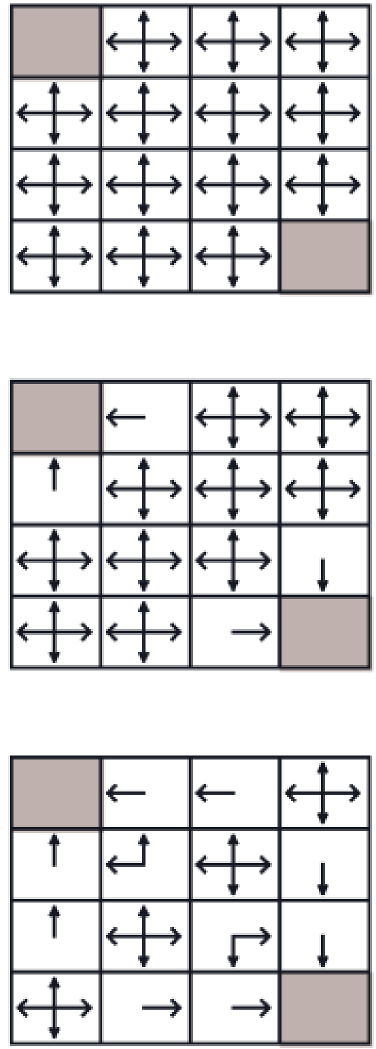
$k = 1$

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

Greedy policy



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

- Complexity when applied to action-value function $O(|A|^2|S|^2)$ 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