Reinforcement Learning
Intelligent Systems Series
Lecture 4 (Part 1)

Georg Martius
Slides adapted from David Silver, Deepmind

MPI for Intelligent Systems, Tübingen, Germany

November 6, 2018

Markov Decision Processes

Markov Process (Reminder)

A Markov process is a memoryless random process, i.e. a sequence of random states $S_1, S_2, \ldots$ with the Markov property.

Reminder: Markov property

A state S_t is Markov if and only if

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

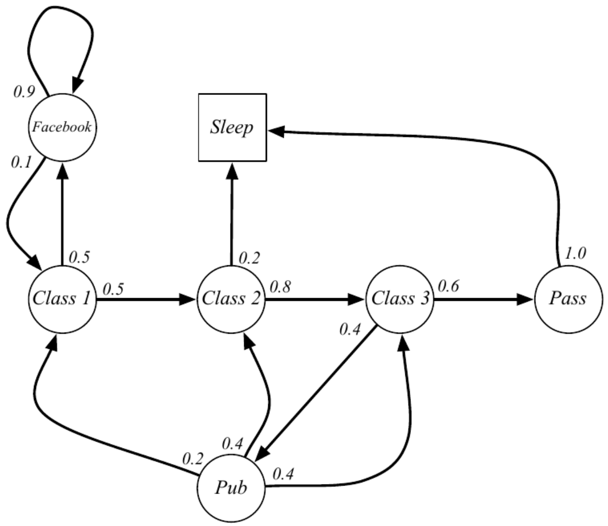
Definition (Markov Process/ Markov Chain)

A *Markov Process* (or *Markov Chain*) is a tuple $(\mathcal{S}, \mathcal{P})$

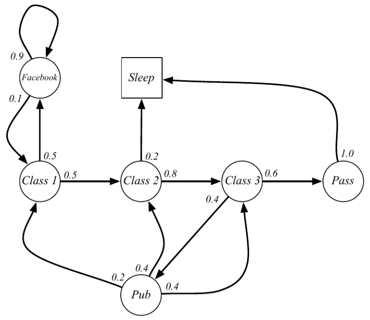
- $\mathcal{S}$ is a (finite) set of states
- $\mathcal{P}$ is a state transition 0 probability matrix,

$$P_{ss'} = P(S_{t+1} = s' \mid S_t = s)$$

Example: Student Markov Chain



Example: Student Markov Chain Transition Matrix



$$
\mathcal{P} = \begin{array}{c} \\ C1 \\ C2 \\ C3 \\ Pass \\ Pub \\ FB \\ Sleep \end{array}
\begin{array}{c}
\begin{array}{ccccccc} C1 & C2 & C3 & Pass & Pub & FB & Sleep \end{array} \\
\begin{bmatrix}
 & 0.5 & & & & 0.5 & \\
 & & 0.8 & & & & 0.2 \\
 & & & 0.6 & 0.4 & & \\
 & & & & & & 1.0 \\
0.2 & 0.4 & 0.4 & & & & \\
0.1 & & & & & 0.9 & \\
 & & & & & & 1
\end{bmatrix}
\end{array}
$$

Markov Reward Process

A Markov reward process is a Markov chain with values.

Definition (MRP)

A *Markov Reward Process* is a tuple $(\mathcal{S}, \mathcal{P}, \mathcal{R}, \gamma)$

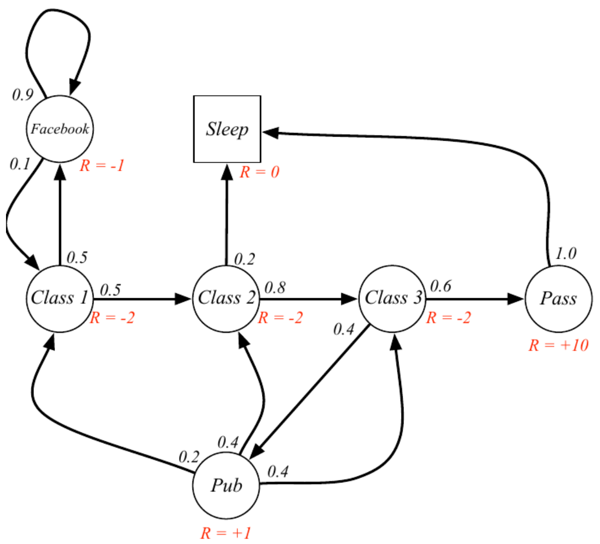
- $\mathcal{S}$ is a finite set of states
- $\mathcal{P}$ is a state transition probability matrix,

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

- $\mathcal{R}$ is a reward function, $\mathcal{R}_s = \mathbb{E}[R_{t+1} | S_t = s]$
- γ is a discount factor, $\gamma \in [0, 1]$

Note that the reward can be stochastic ($\mathcal{R}_s$ is in expectation)

Example: Student MRP



from David Silver

Return (end of Reminder)

Definition

The *return* G_t is the total discounted reward from time-step t.

$$G_t = R_{t+1} + \gamma R_{t+2} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

- The discount $\gamma \in [0, 1]$ devaluates future rewards: reward R after $k + 1$ time-steps is counted as $\gamma^k R$.
- Extreme cases:
  - γ close to 0 leads to immidate reward maximization only
  - γ close to 1 leads to far-sighted evaluation

Value Function

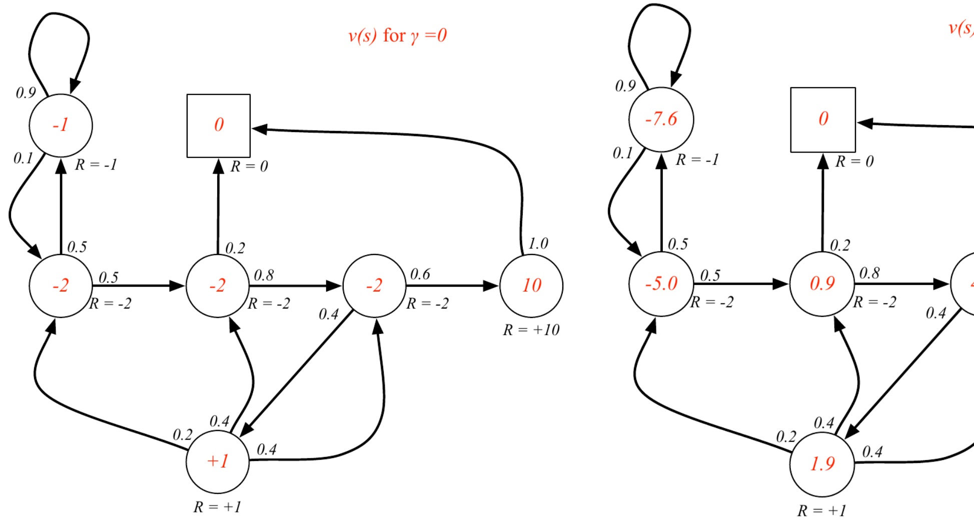
The value function describes the value of a state (in the stationary state)

Definition

The state *value function* $v(s)$ of an MRP is the expected return starting from state s

$$v(s) = \mathbb{E}[G_t | S_t = s]$$

Example: Value Function for Student MRP



from David Silver

Bellman Equation (MRP) I

Idea: Make value computation recursive by tearing apart contributions from:

- immediate reward
- and from discounted future rewards

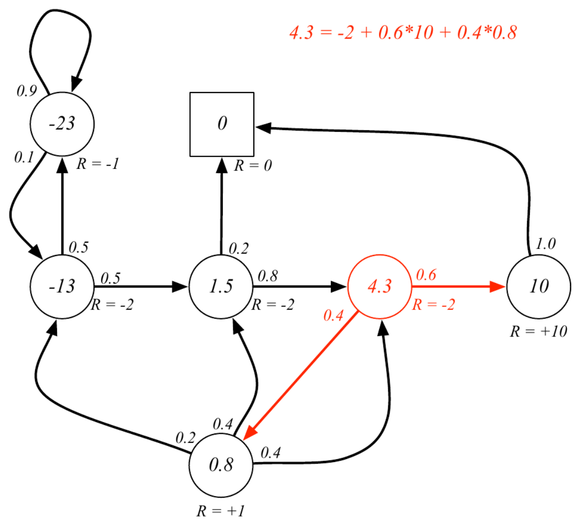
$$\begin{aligned} v(s) &= \mathbb{E}[G_t \mid S_t = s] \\ &= \mathbb{E}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots \mid S_t = s] \\ &= \mathbb{E}[R_{t+1} + \gamma G_{t+1} \mid S_t = s] \\ &= \mathbb{E}[R_{t+1} + \gamma v(S_{t+1}) \mid S_t = s] \end{aligned}$$

Mh... need Expectation over S_{t+1}

Use transition matrix to get probabilities of succeeding state:

$$v(s) = \mathcal{R}_s + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'} v(s')$$

Example: Bellman Equation for Student MRP



from David Silver

Bellman Equation (MRP) II

Bellman equations in matrix form:

$$v = \mathcal{R} + \gamma \mathcal{P} v$$

where $v \in \mathbb{R}^{|S|}$ and $\mathcal{R}$ are vectors

The Bellman equation can be solved directly:

$$v = (\mathbb{I} - \gamma \mathcal{P})^{-1} \mathcal{R}$$

- computational complexity is $O(|S|^3)$

Markov Decision Process

A Markov reward process has no agent, there is no influence on the system.
And MRP with an active agent forms a Markov Decision Process.

- Agent takes decision by executing *actions*
- State is Markovian

Definition (MDP)

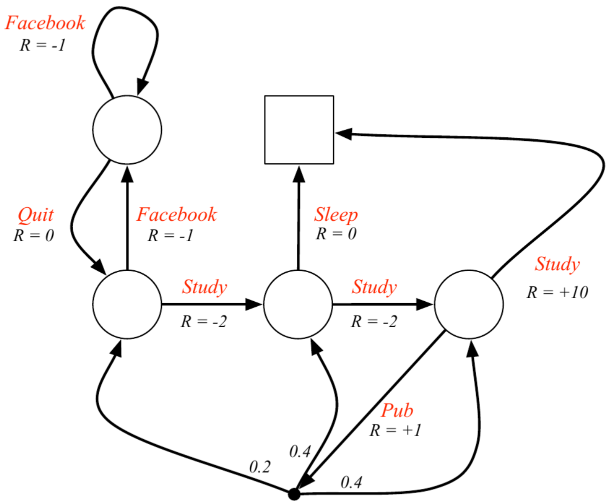
A *Markov Decision Process* is a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$

- $\mathcal{S}$ is a finite set of states
- $\mathcal{A}$ is a finite set of actions
- $\mathcal{P}$ is a state transition probability matrix,

$$\mathcal{P}^a_{ss'} = P(S_{t+1} \mid S_t, A_t = a)$$

- $\mathcal{R}$ is a reward function, $\mathcal{R}^a_s = \mathbb{E}[R_{t+1} | S_t = s, A_t = a]$
- γ is a discount factor, $\gamma \in [0, 1]$

Example: Student MDP



from David Silver

How to model decision taking?

The agent has a action function called policy.

Definition

A *policy* π is a distribution over actions given states,

$$\pi(a|s) = P(A_t = a \mid S_t = s)$$

- Since it is a Markov process the policy only depends on the current state
- Implication: policies are stationary (independent of time)

An MDP with a given policy turns into a MRP:

$$\mathcal{P}^\pi_{ss'} = \sum_{a \in \mathcal{A}} \pi(a|s) \mathcal{P}^a_{ss'}$$

$$\mathcal{R}^\pi_s = \sum_{a \in \mathcal{A}} \pi(a|s) \mathcal{R}^a_s$$

Modelling expected returns in MDP

How good is each state when we follow the policy π?

Definition

The *state-value* function $v_\pi(s)$ of an MDP is the expected return when starting from state s and following policy π.

$$v_\pi(s) = \mathbb{E}[G_t | S_t = s]$$

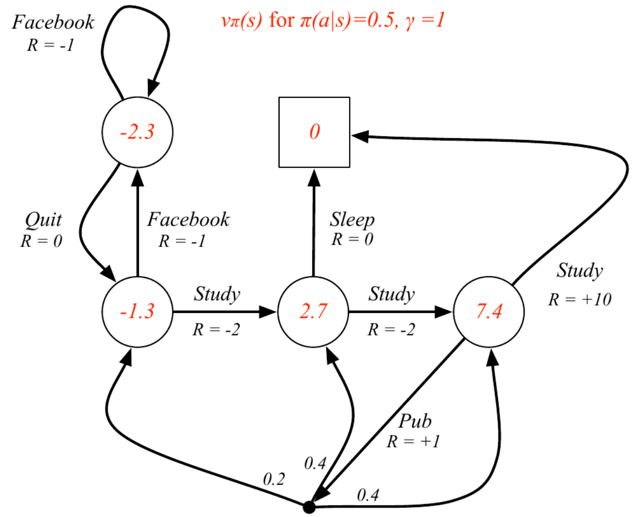
Should we change the policy?
How much does choosing a different action change the value?

Definition

The *action-value* function $q_\pi(s, a)$ of an MDP is the expected return when starting from state s, taking action a, and then following policy π.

$$q_\pi(s, a) = \mathbb{E}[G_t | S_t = s, A_t = a]$$

Example: State-Value function for Student MDP



from David Silver

Bellman Expectation Equation

Recall: Bellman Equation: decompose expected reward into immediate reward plus discounted value of successor state,

$$v_\pi(s) = \mathbb{E}_\pi[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s]$$

The action-value function can be similarly decomposed,

$$q_\pi(s, a) = \mathbb{E}_\pi[R_{t+1} + \gamma q_\pi(S_{t+1}, A_{t+1}) | S_t = s, A_t = a]$$

Bellman Equation: joined update of v_π and q_π

Value function can be derived from q_π:

$$v_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) q_\pi(s, a)$$

. . . and q can be computed from transition model

$$q_\pi(s, a) = \mathcal{R}_s^a + \gamma \sum_{s' \in S} \mathcal{P}_{ss'}^a v_\pi(s')$$

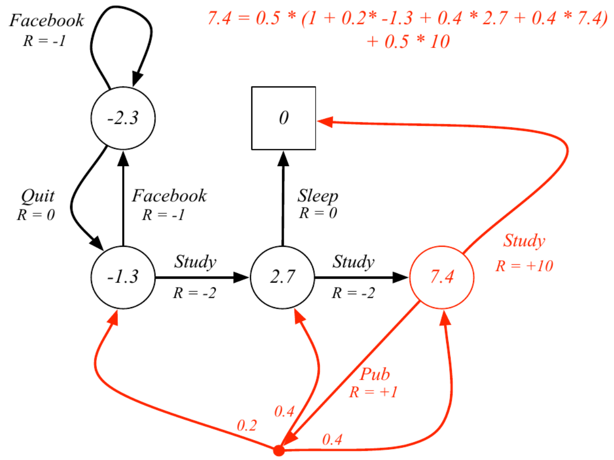
Substituting q in v:

$$v_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) \left(\mathcal{R}_s^a + \gamma \sum_{s' \in S} \mathcal{P}_{ss'}^a v_\pi(s') \right)$$

Substituting v in q:

$$q_\pi(s, a) = \mathcal{R}_s^a + \gamma \sum_{s' \in S} \mathcal{P}_{ss'}^a \sum_{a' \in \mathcal{A}} \pi(a'|s') q_\pi(s', a')$$

Example: Bellman update for v in Student MDP



from David Silver

$\pi(a|s) = 0.5$

Explicit solution for v_π

Since a policy induces a MRP v_π can be diretly computed (as before)

$$v = (\mathbb{I} - \gamma \mathcal{P}^\pi)^{-1} \mathcal{R}^\pi$$

But do we want v_π?

We want to find the optimal policy and its value function!

Optimal Value Function

Definition

The *optimal state-value function* $v_*(s)$ is the maximum value function over all policies

$$v_*(s) = \max_\pi v_\pi(s)$$

Definition

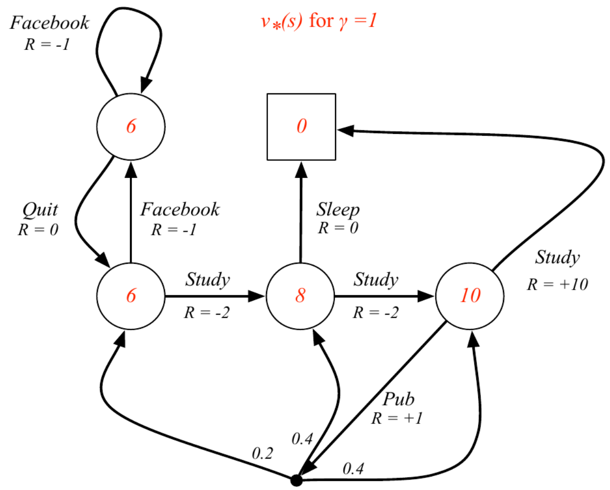
The *optimal action-value function* $q_*(s, a)$ is the maximum action-value function over all policies

$$q_*(s, a) = \max_\pi q_\pi(s, a)$$

What does it mean?

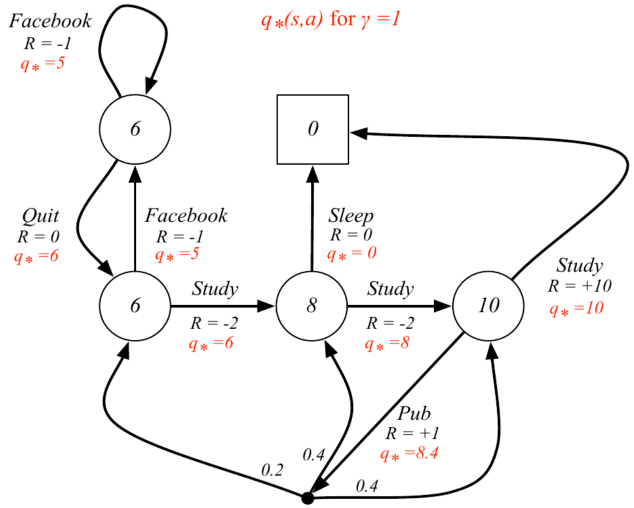
- v_* specifies the best possible performance in an MDP
- Knowing v_* solves the MDP (how? we will see. . .)

Example: Optimal Value Function v_* in Student MDP



from David Silver

Example: Optimal State Function q_* in Student MDP



from David Silver

Optimal Policy

Actually solving the MDP means we have also the optimal policy.
Define a partial ordering over policies

$$\pi \geq \pi' \text{ if } v_\pi(s) \geq v_{\pi'}(s), \forall s$$

Theorem

For any Markov Decision Process

- *There exists an optimal policy π_* that is better than or equal to all other policies, $\pi_* \geq \pi, \forall \pi$*
- *All optimal policies achieve the optimal state-value function, $v_{\pi_*}(s) = v_*(s)$*
- *All optimal policies achieve the optimal action-value function, $q_{\pi_*}(s,a) = q_*(s,a)$*

Finding an Optimal Policy

Given the optimal action-value function q_*: the optimal policy is given by maximizing it.

$$\pi_*(a|s) = [\![a = \arg\max_{a \in \mathcal{A}} q_*(s,a)]\!]$$

$[\![\cdot]\!]$ is Iverson bracket: 1 if *true*, otherwise 0.

- There is always a deterministic optimal policy for any MDP
- If we know $q_*(s,a)$, we immediately have the optimal policy (greedy)

Bellman Equation for optimal value functions

Also for the optimal value functions we can use Bellmans optimality equations:

$$v_*(s) = \max_{a \in \mathcal{A}} q_*(s,a)$$

$$q_*(s,a) = \mathcal{R}_s^a + \gamma \sum_{s' \in S} \mathcal{P}_{ss'}^a v_*(s')$$

Substituting q in v:

$$v_*(s) = \max_{a \in \mathcal{A}} \left(\mathcal{R}_s^a + \gamma \sum_{s' \in S} \mathcal{P}_{ss'}^a v_*(s') \right)$$

Substituting v in q:

$$q_*(s,a) = \mathcal{R}_s^a + \gamma \sum_{s' \in S} \mathcal{P}_{ss'}^a \max_{a' \in \mathcal{A}} q_*(s',a')$$

Solving the Bellman Optimality Equation

Bellman Optimality Equation is non-linear

- No closed form solution (in general)
- Many iterative solution methods
  - Value Iteration
  - Policy Iteration
  - Q-learning
  - SARSA