Reinforcement Learning
Intelligent Systems Series
Lecture 4 (Part 1)

Georg Martius
Slides adapted from David Silver, Deepmind

MPI for Intelligent Systems, Tübingen, Germany

November 9, 2018
Markov Decision Processes
A Markov process is a memoryless random process, i.e. a sequence of random states $S_1, S_2, \ldots$ with the Markov property.
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

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

- $S$ 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

\[
P = \begin{bmatrix}
C1 & C2 & C3 & Pass & Pub & FB & Sleep \\
C1 & 0.5 & & & & & 0.2 \\
C2 & & 0.8 & & & 0.4 & \\
C3 & & & 0.6 & 0.4 & & \\
Pass & & & & 1.0 & & \\
Pub & 0.2 & 0.4 & 0.4 & & & \\
FB & & 0.2 & 0.4 & & 0.9 & \\
Sleep & & & 0.4 & & 1 & \\
\end{bmatrix}
\]
A Markov reward process is a Markov chain with values.

**Definition (MRP)**

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

- \(S\) is a finite set of states
- \(P\) is a state transition probability matrix,
  \[ P(S_{t+1} \mid S_t) = P(S_{t+1} \mid S_1, \ldots, S_t) \]
- \(R\) is a reward function,
  \[ R_s = \mathbb{E}[R_{t+1} \mid S_t = s] \]
- \(\gamma\) is a discount factor, \(\gamma \in [0, 1]\)

Note that the reward can be stochastic (\(R_s\) is in expectation)
A Markov reward process is a Markov chain with values.

**Definition (MRP)**

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

- \(S\) is a finite set of states
- \(P\) is a state transition probability matrix,
  \[
P(S_{t+1} \mid S_t) = P(S_{t+1} \mid S_1, \ldots, S_t)
\]
- \(R\) is a reward function, \(R_s = \mathbb{E}[R_{t+1} \mid S_t = s]\)
- \(\gamma\) is a discount factor, \(\gamma \in [0, 1]\)

Note that the reward can be stochastic (\(R_s\) is in expectation)
Example: Student MRP

from David Silver
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 $\gamma \in [0, 1]$ devalues future rewards: reward $R$ after $k+1$ time-steps is counted as $\gamma^k R$.

Extreme cases:
- $\gamma$ close to 0 leads to immediate reward maximization only
- $\gamma$ close to 1 leads to far-sighted evaluation
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:
  - $\gamma$ close to 0 leads to immediate reward maximization only
  - $\gamma$ close to 1 leads to far-sighted evaluation
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

\[ v(s) \text{ for } \gamma = 0 \]

from David Silver
Example: Value Function for Student MRP

\[ v(s) \text{ for } \gamma = 0.9 \]
Example: Value Function for Student MRP

\( \nu(s) \) for \( \gamma = 1 \)

from David Silver
Bellman Equation (MRP) I

Idea: Make value computation recursive by tearing apart contributions from:
- immediate reward
- and from discounted future rewards

\[ v(s) = \mathbb{E}[G_t | S_t = s] = \mathbb{E}[R_{t+1} + \gamma G_{t+1} | S_t = s] = \mathbb{E}[R_{t+1} + \gamma v(S_{t+1}) | S_t = s] \]

Mh... need Expectation over \( S_{t+1} \)
Use transition matrix to get probabilities of succeeding state:
\[ v(s) = \mathbb{E}[R_{s} + \gamma \sum_{s' \in S} P_{ss'} v(s') | S_t = s] \]
Bellman Equation (MRP) I

Idea: Make value computation recursive by tearing apart contributions from:
- immediate reward
- and from discounted future rewards

\[ v(s) = \mathbb{E}[G_t \mid S_t = s] \]
Bellman Equation (MRP) I

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

- immediate reward
- and from discounted future rewards

\[ v(s) = \mathbb{E}[G_t \mid S_t = s] \]
\[ = \mathbb{E}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots \mid S_t = s] \]
\[ = \mathbb{E}[R_{t+1} + \gamma G_{t+1} \mid S_t = s] \]
Bellman Equation (MRP) I

Idea: Make value computation recursive by tearing apart contributions from:
  - immediate reward
  - and from discounted future rewards

\[
\nu(s) = \mathbb{E}[G_t \mid S_t = s] \\
= \mathbb{E}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots \mid S_t = s] \\
= \mathbb{E}[R_{t+1} + \gamma G_{t+1} \mid S_t = s] \\
= \mathbb{E}[R_{t+1} + \gamma \nu(S_{t+1}) \mid S_t = s]
\]
Bellman Equation (MRP) I

Idea: Make value computation recursive by tearing apart contributions from:
- immediate reward
- and from discounted future rewards

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

Mh... need Expectation over \( S_{t+1} \)
Bellman Equation (MRP) I

Idea: Make value computation recursive by tearing apart contributions from:
- immediate reward
- and from discounted future rewards

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

Mh... need Expectation over \( S_{t+1} \)

Use transition matrix to get probabilities of succeeding state:

\[ v(s) = R_s + \gamma \sum_{s' \in S} P_{ss'} v(s') \]
Example: Bellman Equation for Student MRP

\[ 4.3 = -2 + 0.6 \times 10 + 0.4 \times 0.8 \]

from David Silver
Bellman equations in matrix form:

\[ v = \mathcal{R} + \gamma \mathcal{P} v \]

where \( v \in \mathbb{R}^{|S|} \) and \( \mathcal{R} \) are vectors
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:

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

computational complexity is $O(|S|^3)$
A Markov reward process has no agent, there is no influence on the system.
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

**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
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 \((S, A, P, R, \gamma)\)

- \(S\) is a finite set of states
- \(A\) is a finite set of actions
- \(P\) is a state transition probability matrix,
  \[ P^a_{ss'} = P(S_{t+1} | S_t, A_t = a) \]
- \(R\) is a reward function, \(R^a_s = \mathbb{E}[R_{t+1} | S_t = s, A_t = a]\)
- \(\gamma\) is a discount factor, \(\gamma \in [0, 1]\)
Example: Student MDP

- **Facebook**
  - $R = -1$

- **Quit**
  - $R = 0$

- **Study**
  - $R = -2$

- **Sleep**
  - $R = 0$

- **Pub**
  - $R = +1$

- **Study**
  - $R = +10$

from David Silver
How to model decision taking?

The agent has an action function called policy.
How to model decision taking?

The agent has a action function called policy.

**Definition**

A *policy* $\pi$ is a distribution over actions given states,

$$ \pi(a|s) = P(A_t = a \mid S_t = s) $$
How to model decision taking?

The agent has a action function called policy.

**Definition**

A *policy* $\pi$ 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)
How to model decision taking?

The agent has a action function called policy.

**Definition**

A *policy* \( \pi \) 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:

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

\[
R^\pi_s = \sum_{a \in A} \pi(a|s)R^a_s
\]
Modelling expected returns in MDP

How good is each state when we follow the policy $\pi$?
Modelling expected returns in MDP

How good is each state when we follow the policy $\pi$?

**Definition**

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

$$v_\pi(s) = \mathbb{E}[G_t | S_t = s]$$
Modelling expected returns in MDP

How good is each state when we follow the policy $\pi$?

**Definition**

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

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

Should we change the policy?
How much does choosing a different action change the value?
Modelling expected returns in MDP

How good is each state when we follow the policy $\pi$?

**Definition**

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

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

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 $\pi$.

$$q_\pi(s, a) = \mathbb{E}[G_t | S_t = s, A_t = a]$$
Example: State-Value function for Student MDP

$\nu_{\pi}(s)$ for $\pi(a|s)=0.5, \gamma=1$

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

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

Value function can be derived from $q_{\pi}$:

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

$q_{\pi}$ can be computed from transition model

$$q_{\pi}(s, a) = R_{a} s + \gamma \sum_{s' \in S} P_{aa'} v_{\pi}(s')$$

Substituting $q_{\pi}$ in $v_{\pi}$:

$$v_{\pi}(s) = \sum_{a \in A} \pi(a|s) \left( R_{a} s + \gamma \sum_{s' \in S} P_{aa'} v_{\pi}(s') \right)$$

Substituting $v_{\pi}$ in $q_{\pi}$:

$$q_{\pi}(s, a) = R_{a} s + \gamma \sum_{s' \in S} P_{aa'} \sum_{a' \in A} \pi(a'|s') q_{\pi}(s', a')$$
Value function can be derived from $q_\pi$:

$$v_\pi(s) = \sum_{a \in A} \pi(a|s) q_\pi(s, a)$$
Value function can be derived from $q_\pi$:

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

... and $q$ can be computed from transition model

$$q_\pi(s, a) = R^a_s + \gamma \sum_{s' \in S} P^a_{ss'} v_\pi(s')$$
Bellman Equation: joined update of $v_\pi$ and $q_\pi$

Value function can be derived from $q_\pi$:

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

...and $q$ can be computed from transition model

$$q_\pi(s, a) = R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a v_\pi(s')$$

Substituting $q$ in $v$:

$$v_\pi(s) = \sum_{a \in A} \pi(a|s) \left( R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a v_\pi(s') \right)$$
Bellman Equation: joined update of $v_\pi$ and $q_\pi$

Value function can be derived from $q_\pi$:

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

...and $q$ can be computed from transition model

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

Substituting $q$ in $v$:

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

Substituting $v$ in $q$:

$$q_\pi(s, a) = R_s^a + \gamma \sum_{s' \in S} \mathcal{P}_{ss'}^a \sum_{a' \in A} \pi(a'|s') q_\pi(s', a')$$
Example: Bellman update for $\nu$ in Student MDP

$$7.4 = 0.5 \times (1 + 0.2 \times -1.3 + 0.4 \times 2.7 + 0.4 \times 7.4) + 0.5 \times 10$$

$\pi(a|s) = 0.5$ from David Silver
Since a policy induces a MRP $\nu_\pi$ can be directly computed (as before)

$$\nu = (I - \gamma P^\pi)^{-1} R^\pi$$
Explicit solution for $v_\pi$

Since a policy induces a MRP, $v_\pi$ can be directly computed (as before)

$$v = (I - \gamma P_\pi)^{-1}R_\pi$$

But do we want $v_\pi$?
Since a policy induces a MRP $v_\pi$ can be directly computed (as before)

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

But do we want $v_\pi$?
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

$$v_*(s) = \max_{\pi} v_{\pi}(s)$$
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?
Optimal Value Function

Definition

The \textit{optimal state-value function} \( v_*(s) \) is the maximum value function over all policies

\[ v_*(s) = \max_{\pi} v_\pi(s) \]

Definition

The \textit{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

$v_*(s)$ for $\gamma = 1$

from David Silver
Example: Optimal State Function $q_*$ in Student MDP

$q_*(s,a)$ for $\gamma = 1$

**Facebook**
- $R = -1$
- $q_* = 5$

**Quit**
- $R = 0$
- $q_* = 6$

**Study**
- $R = -2$
- $q_* = 6$

**Sleep**
- $R = 0$
- $q_* = 0$

**Study**
- $R = -2$
- $q_* = 8$

**Pub**
- $R = +1$
- $q_* = -8.4$

from David Silver
Actually solving the MDP means we have also the 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 \( \pi_* \) 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, \( q_{\pi_*}(s, a) = q_*(s, a) \)
Given the optimal action-value function $q^*$:

$$
\pi^*(a|s) = J_a = \arg \max_{a \in A} q^*(s,a)
$$

$J_a$ 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).
Given the optimal action-value function $q^*$: the optimal policy is given by maximizing it.

$$
\pi^*(a|s) = \left[ a = \arg\max_{a \in A} q^*(s, a) \right]
$$

[] 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 Bellman's optimality equations:

\[
\begin{align*}
    v_*(s) &= \max_{a \in A} q_*(s, a) \\
    q_*(s, a) &= R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a v_*(s')
\end{align*}
\]
Bellman Equation for optimal value functions

Also for the optimal value functions we can use Bellman's optimality equations:

\[ v^*(s) = \max_{a \in A} q^*(s, a) \]

\[ q^*(s, a) = R_s + \gamma \sum_{s' \in S} P_{ss'} v^*(s') \]

Substituting \( q \) in \( v \):

\[ v^*(s) = \max_{a \in A} \left( R_s + \gamma \sum_{s' \in S} P_{ss'} v^*(s') \right) \]
Bellman Equation for optimal value functions

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

\[ v_*(s) = \max_{a \in A} q_*(s, a) \]

\[ q_*(s, a) = R_s + \gamma \sum_{s' \in S} P_{ss'} v_*(s') \]

Substituting \( q \) in \( v \):

\[ v_*(s) = \max_{a \in A} \left( R_s + \gamma \sum_{s' \in S} P_{ss'} v_*(s') \right) \]

Substituting \( v \) in \( q \):

\[ q_*(s, a) = R_s + \gamma \sum_{s' \in S} P_{ss'} \max_{a' \in A} q_*(s', a') \]
Bellman Optimality Equation is non-linear
- No closed form solution (in general)
- Many iterative solution methods
  - Value Iteration
  - Policy Iteration
  - Q-learning
  - SARSA
David Silver’s Lecture 3 . . .