NUS National University of Singapore | Computing

Markov Decision Process

CS4246/CS5446

AI Planning and Decision Making

This Lecture Will Be Recorded!

Please join:
pollev.com/anarayan

Markov Decision Process (MDP)

- Formally:
  - An MDP $M \triangleq (S, A, T, R, \gamma)$ consists of:
  - A set S of states
  - A set A of actions
  - A transition function $T: S \times A \times S \rightarrow [0,1]$ that satisfies the Markov property such that:

$$\forall s \in S, \forall a \in A: \sum_{s\prime \in S} T(s, a, s') = \sum_{s\prime \in S} P(s'|s, a) = 1$$

  - A reward function $R: S \rightarrow \Re$ or $R: S \times A \times S \rightarrow \Re$
  - A discount factor $0 < \gamma < 1$
  - Solution is a policy – a function to recommend an action in each state: $\pi\colon S \rightarrow A$
    - Solution involves careful balancing of risk and reward

Transition Function

- Formally:
  - $T(s, a, s') = P(s'|s, a)$ is the probability of going from state s to state s' taking action a
  - Define $T(s, a, s')$ for all $s, s' \in S, a \in A$
- Markov Property
  - The next state is conditionally independent of the past states and actions given the current state s and action a, i.e.,

$$P(s_{t+1}|s_t, a_t, s_{t-1}, a_{t-1}, \dots, s_0, a_0) = P(s_{t+1}|s_t, a_t)$$

for all $s_t, a_t, s_{t-1}, a_{t-1}, \dots, s_0, a_0$

Markov Property

- Is the Markov Property applicable in real-world problems?
  - It is a simplifying assumption!
- Potential violations of Markov assumption
  - State variables and dynamics of environment not captured in model
  - Inaccuracies in the transition function
- Nevertheless:
  - Markov assumption helps reduce time complexity of algorithms
  - Most, if not all, stochastic processes can be modeled as Markov processes

Reward Function

- Formally:
  - Define $R(s, a, s')$ for all $s, s' \in S$ and for all $a \in A$.
- Alternate forms: (given $R(s, a, s')$ above)
  - $R(s, a)$ as $\sum_{s'} P(s'|s, a) R(s, a, s')$ independent of s'
  - $R(s)$ as $\sum_{s'} P(s'|s, a) R(s, a, s')$, independent of a and s'
- Challenges
  - It is hard to construct reward functions with multiple attributes
  - Balance risk vs reward

Other possible ways to define reward functions

Clarifications on Reward Functions

a) Representation of states and transitions:

- ***s*** always denotes the "current state", ***a*** is the action (to be) taken in state ***s*** (current state), and ***s'*** is the outcome or next state after taking action ***a*** in state ***s***.

b) The common reward definitions are as follows:

- On state:
    - b.1) ***R(s)*** - amount of reward (or cost) of "being" in state ***s***
    - When to record or "count" reward: 1) on arriving at ***s***, OR more commonly, 2) on exiting ***s*** (in the next transition), depending on context/implementation/choice in the problem model and/or solution
    - b.2) ***R(s, a)*** - amount of reward (or cost) of taking action ***a*** in state ***s***
    - When to record reward: - on exiting ***s*** upon taking action ***a***
- On transition:
    - b.3) ***R(s, a, s')*** - amount of reward (or cost) of the transition from ***s*** to ***s'*** given that the action ***a*** is taken
    - When to record reward: 1) on exiting ***s*** before reaching (the intended) ***s'***, OR 2) on arriving at ***s'***

Clarifications on Reward Functions

- Notes:
  1) The above formulations allow accumulation of rewards if agent/system remains in the same state for multiple time steps. But the actual time of "counting" may not matter that much in calculating expectations. It matters more in the simulation counts.
  2) In practice, there can be more than one set of rewards defined for each ***s***, ***(s, a)***, and/or ***(s, a, s')*** combinations, i.e., you can have ***R(s)*** and ***R(s, a, s')*** separately defined in the same model, e.g., in diagnostic test and therapy planning models in healthcare.
  3) For slide 15 in the lecture notes - the **descriptions** (they are not meant to be equations) in the **"Alternate Forms"** are meant to be interpreted as: **How to formulate *R(s, a)* and *R(s)* in terms of the main definition of *R(s, a, s')* given in the first line**.
  4) Remember, MDP is a modeling "language", these definitions are by "choice" of the designer and conventions commonly adopted - there are variations in different problem and solution

Exercise

- Question:
  - In the navigation example, the state is the position of the agent. Consider a slightly different problem, where there are two possible agents, agent A and agent B. Agents A and B have different transition functions. Which of the following describes the state in the MDP? Why?
- Answers:
  - A: position of the agent
  - B: pair of position and identity of the agent

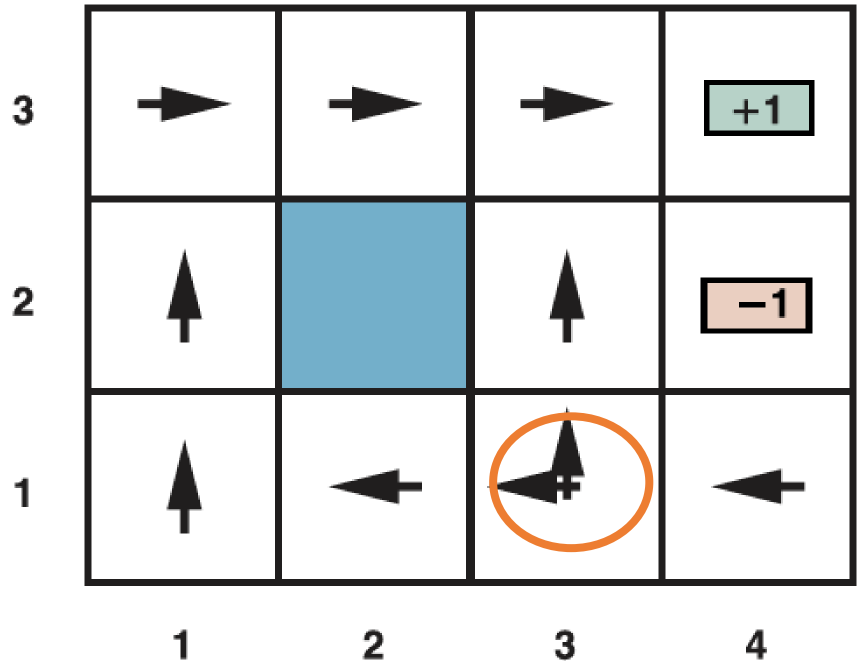
Solving MDPs

Deriving policies – actions to take at each state

Solving MDPs

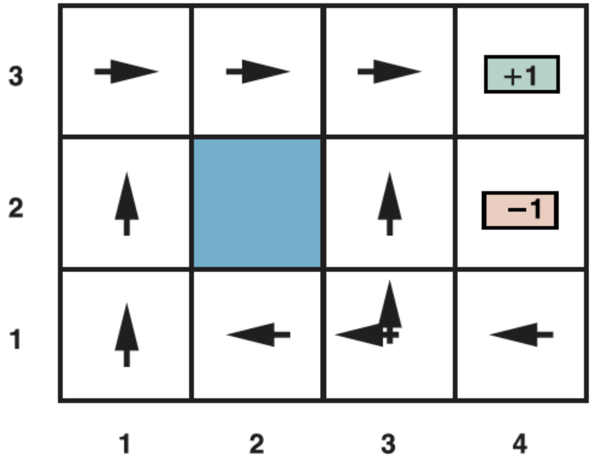
- What does a solution look like?
  - A policy $\pi(s): S \rightarrow A$ is a function from states to actions:
    - For every state s, outputs an appropriate action a.
  - Quality of policy – measured by expected utility of possible state sequence generated by the policy
  - Optimal policy π^* is a policy that generates highest expected utility
- An MDP agent:
  - Given optimal policy π^*: Agent decides what to do by consulting its current percept, which tells it the current state s, and then executing action $a^* = \pi^*(s)$
  - The (optimal) policy represents the agent function explicitly – how to behave!

Example: Illustration of π^*

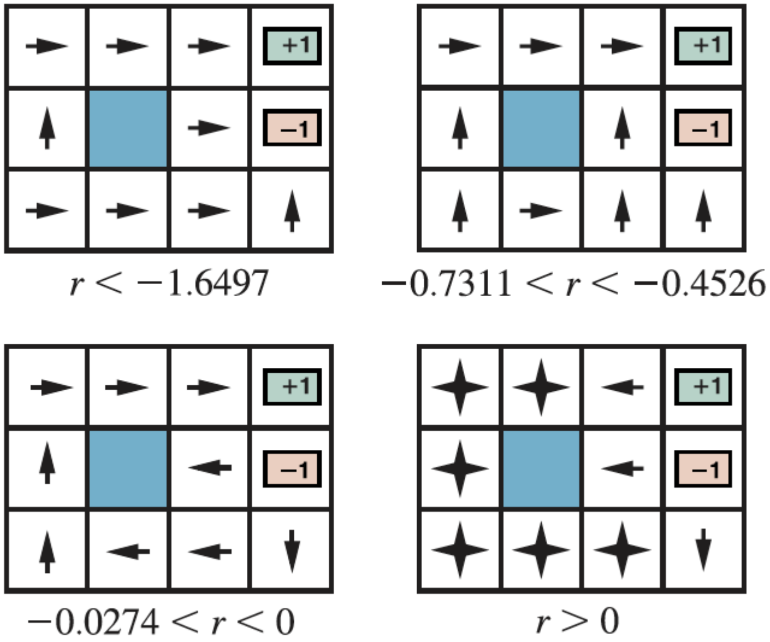


Source: RN Figure 16.2

Example: Balancing Risk and Reward



$r = -0.04$



Changes depending on the value of $r = R(s, a, s')$ for transitions between nonterminal states
There may be many optimal policies for various ranges of r.

Source: RN Figure 16.2

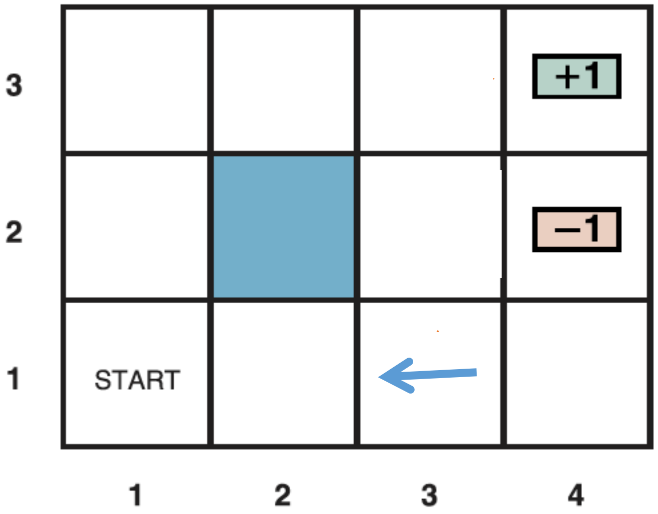
Finite and Infinite Horizon Problems

- Finite horizon: Fixed time N and terminate
  - Return is usually the addition of rewards over the sequence
    - $U_h([s_0, a_0, s_1, a_1, s_2, s_3 \dots s_{N+k}]) = U_h([s_0, a_0, s_1, a_1, s_2, s_3 \dots s_N])$
    - $U_h([s_0, a_0, s_1, a_1, s_2, s_3 \dots s_N]) = R(s_0, a_0, s_1) + R(s_1, a_1, s_2) + \cdots + R(s_{N-1}, a_{N-1}, s_N)$
  - Optimal action in a given state can change over time N, depending on remaining steps
  - Nonstationary optimal policy: π_t^*
- Infinite horizon
  - No fixed deadline
  - No reason to behave differently in the same state at different times
  - Stationary optimal policy: π^*

Example: Finite Horizon Problem

N = 3?

N = 100?



Source: RN Figurer 16.2

Rewards in Infinite Horizon Problems

- Infinite horizon – comparing utilities are difficult
  - Undiscounted utilities can be infinite
- Additive discounted rewards

Allows preference independence assumption

$$U_h([s_0, a_0, s_1, a_1, s_2, s_3 \ldots]) = R(s_0, a_0, s_1) + \gamma R(s_1, a_1, s_2) + \gamma^2 R(s_2, a_2, s_3) + \cdots$$

where $\gamma \in [0,1]$ is the discount factor

  - Discounted rewards with $\gamma < 1$ and rewards bounded by $\pm R_{max}$, utility is always finite

$$U_h([s_0,\ a_0, s_1, \ldots]) = \sum_{t=0}^{\infty} \gamma^t R(s_t, a_t, s_{t+1}) \leq \sum_{t=0}^{\infty} \gamma^t R_{max} = \frac{R_{max}}{1-\gamma}$$

Rewards in Infinite Horizon Problems

- Infinite rewards
  - Environment does not have terminal state; or
  - Agent never reaches terminal state; and
  - Additive, undiscounted rewards
- Why additive discounted rewards?
  - Preference independence assumption
    - Read: If you prefer one future to another starting tomorrow, then you should still prefer that future it were to start today instead.
  - Empirical – humans and animas appear to value near term rewards more
  - Economic – Early rewards can be invested to produce returns
  - Uncertainty about the true rewards – Rewards may never arrive
  - Discounted rewards make some nasty infinities go away

Preference Independence Assumption

- Assumptions
  - Each transition $s_t \xrightarrow{a_t} s_{t+1}$ regarded as an attribute of the history $[s_0, a_0, s_1, a_1, s_2, s_3 \ldots]$
  - Preference independence assumption – preferences between state sequences are stationary
- Stationary preferences
  - If two histories $[s_0, a_0, s_1, a_1, s_2, \ldots]$ and $[s'_0, a'_0, s'_1, a'_1, s'_2, \ldots]$ begin with the same transition (i.e., $s_0 = s'_0, a_0 = a'_0$, and $s_1 = s'_1$)
  - Then the two histories should be preference-ordered the same way as the histories $[s_1, a_1, s_2, \ldots]$ and $[s'_1, a'_1, s'_2, \ldots]$

Exercise

- Question:
  - If Dr. Bean's salary is $\$20{,}000$ per year, without change?
  - How much in total will he earn in his life? Assuming that his is going to live a LONG TIME … With no discount? With a discount factor of 0.9?

Quiz

Quiz answer

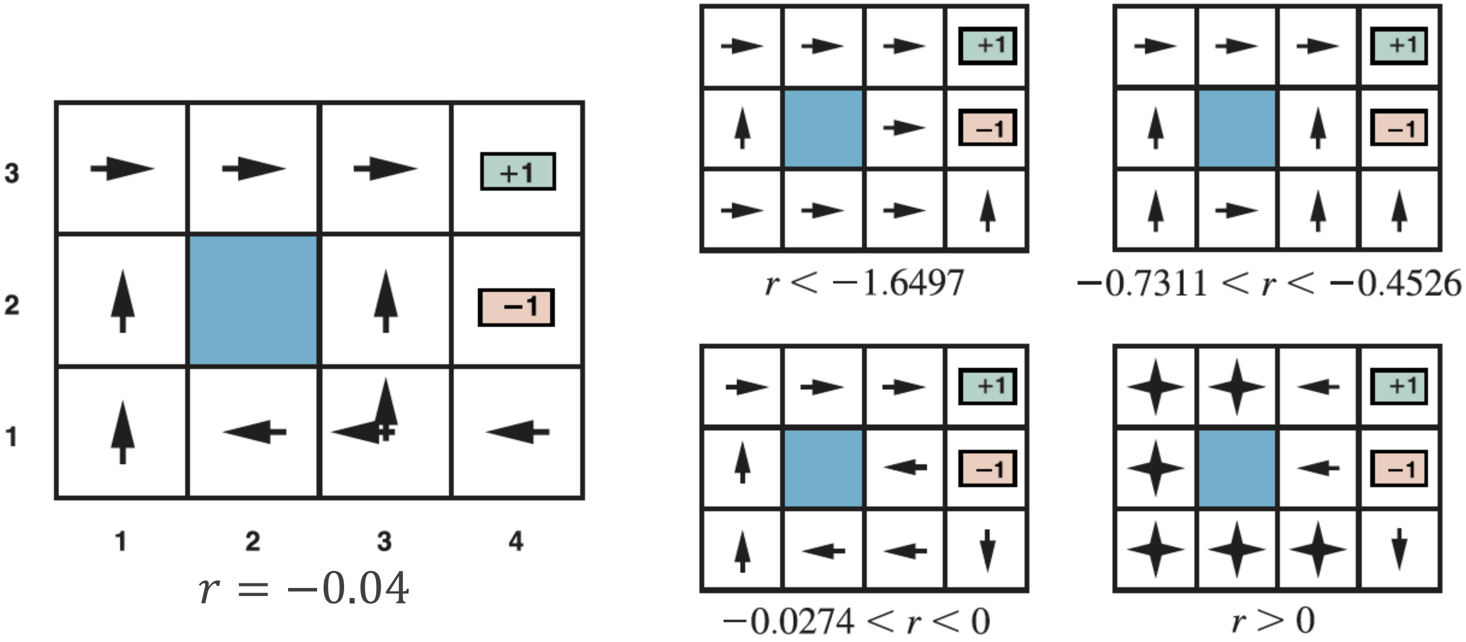
Exercise

- Question:
  - If Dr. Bean's salary is $\$20{,}000$ per year, without change?
  - How much in total will he earn in his life? Assuming that his is going to live a LONG TIME … With no discount? With a discount factor of 0.9?
- Answer:

How to Achieve Finite Rewards?

- 3 possible solutions to achieve finite rewards
  1. With discounted rewards, utility of infinite sequence is finite, if $\gamma < 1$, and rewards bounded by $\pm R_{max}$, then
     $$U_h([s0, a0, s1, a1, s2, \ldots]) = R_{max}/(1 - \gamma)$$
  2. Environment has terminal states and if there is a proper policy, i.e., agent is guaranteed to get to a terminal state, can even use $\gamma = 1$ in additive rewards. (See counter examples)
  3. Compared in terms of average reward obtained per time step
     - Harder to compute and to analyze

Example: Proper and Improper Policies



(a) (b)

Source: RN Figure 17.2

Utility of State and Optimal Policy

- Main ideas:
  - Utility of sequence: Sum of the discounted rewards obtained during that sequence
  - Comparing expected utilities obtained when executing policies
  - Utility function of state $U(s)$ allows selection of optimal action using MEU
- Assumptions:
  - Define random variable S_t : State reached at time t when executing a particular policy π; $S_0 = s$
  - Probability distribution over state sequences $S_1, S_2, \ldots$ determined by: Initial state s, policy π, and transition model T for the environment
- Expected utility of executing π starting from s:

$$U^{\pi}(s) = E[\sum_{t=0}^{\infty} \gamma^t R(S_t, \pi(S_t), S_{t+1})]$$

  - With $S_0 = s$, expectation wrt distribution of state sequences determined by π and s.

Utility of State and Optimal Policy

- Utility of state (or value of state) is the value of optimal policy

$$U(s) = U^{\pi^*}(s) = \max_{a \in A(s)} \sum_{s'} P(s'|s,a)[R(s,a,s') + \gamma U(s')] = V(s)$$

  - Expected sum of discounted rewards if an optimal policy is executed
    - $R(s) = \sum_{s'} P(s'|s,a) R(s,a,s')$ is the "short term" reward for being in s
    - $U(s) = V(s)$ is the "long term" total expected reward from s onward

- Optimal action selected through maximizing utility of state $U(s)$ based on MEU:

$$\pi^*(s) = \text{argmax}_{a \in A(s)} \sum_{s'} P(s'|s,a)[R(s,a,s') + \gamma U(s')]$$

    - Optimal policy is independent of the starting state in infinite horizon problems with discounted utilities

Example: Utilities of States

$R(s) = -0.04$

Numbers are $U(s)$

3	0.8516	0.9078	0.9578	+1
2	0.8016		0.7003	−1
1	0.7453	0.6953	0.6514	0.4279
	1	2	3	4

How to compute the numbers?

Source: RN Figure 16.3

Bellman Equation

- Principle of optimality: (Bellman, 1957)
  - An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision

Bellman Equation: Finite Horizon

- Definition for finite horizon problem:
  - The dynamic programming algorithm finds the utility or value functions (state utilities):
  - If the horizon is 0, $U_0(s) \ = \ R(s)$.
  - If horizon is k, following Bellman's principle of optimality (or optimal substructure)

$$U_k(s) = \max_{a \in A(s)} \sum_{s'} P(s'|s,a)[R(s,a,s') + \gamma U_{k-1}(s')] \quad \text{OR}$$

$$U_k(s) = R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s'|s,a) U_{k-1}(s'))$$

Bellman Equation: Infinite Horizon

- Definition for infinite horizon problem:

$$U(s) = \max_{a \in A(s)} \sum_{s'} P(s'|s,a)[R(s,a,s') + \gamma U(s')] \text{ OR}$$

$$\underbrace{U(s)}_{\text{Utility of a state, } s} = \underbrace{R(s)}_{\text{Immediate reward}} + \gamma \underbrace{\max_{a \in A(s)}}_{\text{Optimal action}} \underbrace{\sum_{s'} P(s'|s,a)U(s')}_{\text{Expected utility of next states}}$$

  - $U(s) = U^{\pi^*}(s)$: The utility of a state is the expected reward for the next transition (or the current state) plus the discounted utility of the next state, assuming optimal action taken

- Solution
  - The utilities of the states as defined as the expected utility of subsequent state sequences are the unique solutions of the set of Bellman equations
  - Gives direct relationship between utility of a state and the utility of its neighbors.
  - There is 1 Bellman equation per state. So, $|S|$ nonlinear equations (due to max) with $|S|$ unknowns (utility of states).

Q-Function

- Action-Utility Function or Q-Function
  - $Q(s,a)$: Expected utility of taking a given action in a given state

$$U(s) = \max_a Q(s,a)$$

- Computing optimal policy:

$$\pi^*(s) = \text{argmax}_a Q(s,a)$$

- Bellman Equation for Q-functions

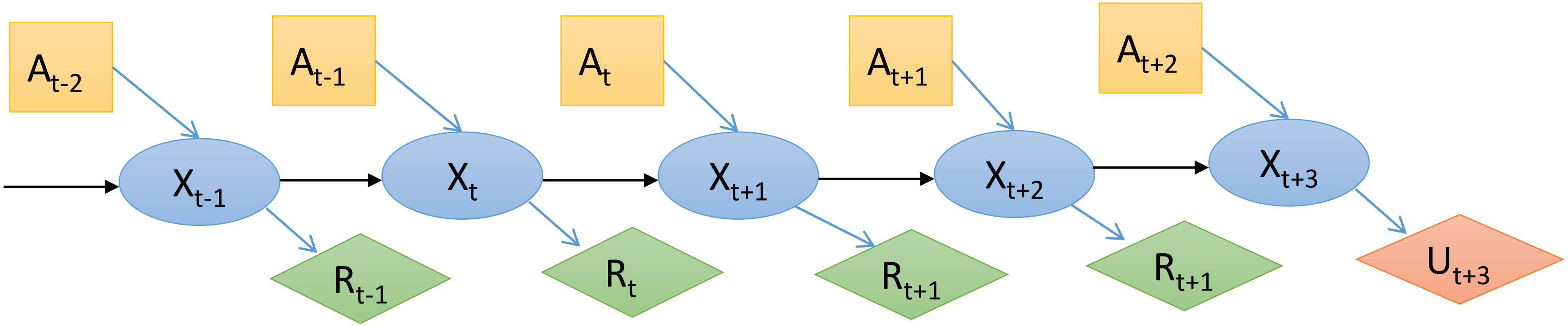
$$Q(s,a) = \sum_{s'} P(s'|s,a)[R(s,a,s') + \gamma U(s')] = \sum_{s'} P(s'|s,a)[R(s,a,s') + \gamma \max_{a'} Q(s',a')\,]$$

- Q-Value Function
  - Inputs: mdp, s, a, U
  - Return: $\sum_{s'} P(s'|s,a)[R(s,a,s') + \gamma U(s')]$

Used in the algorithms later

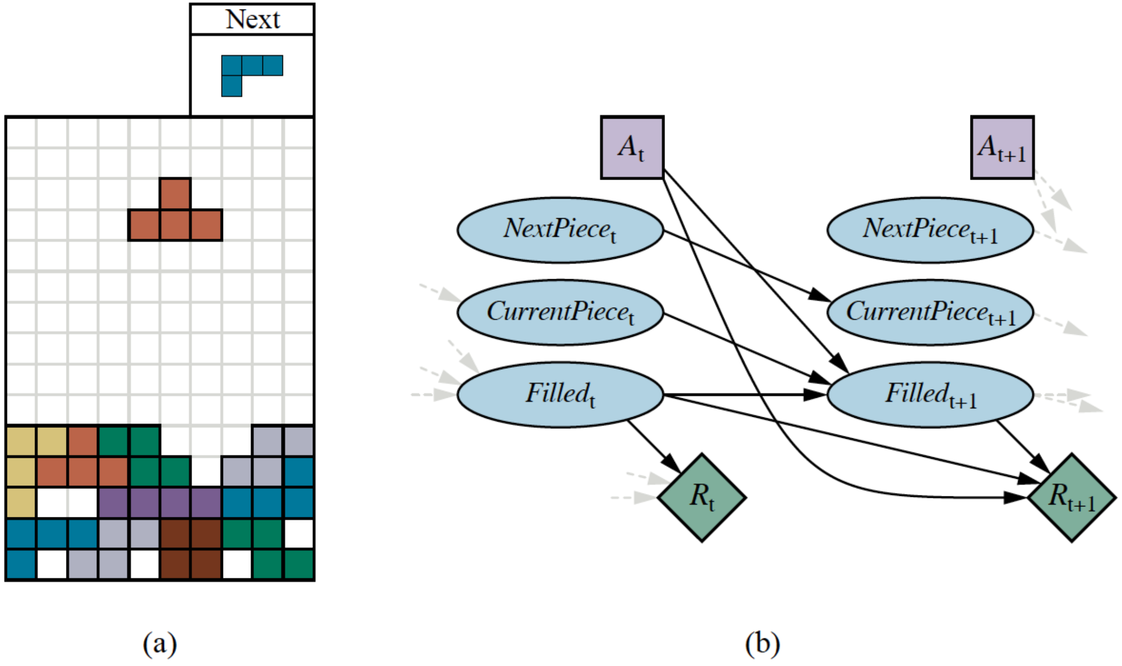
MDP as Dynamic Decision Network

$P(X_t | X_{t-1}, A_{t-1})$ $P(X_{t+1} | X_t, A_t)$ $P(X_{t+2} | X_{t+1}, A_{t+1})$



What are X_t, A_t, R_t, U_t?

Example: Tetris



Source: RN Figure 16.5

Homework

- Readings
  - [RN] 16.1, 16.2.1, 16.2.2
  - [SB] *4.2 (Policy improvement)*

  - [SB] Sutton, R. S. and A. G. Barto. Reinforcement Learning: An introduction. 2nd ed. MIT Press, 2018, 2020
    [Book website: http://incompleteideas.net/book/the-book.html]
    [e-Book for personal use: http://incompleteideas.net/book/RLbook2020.pdf]