NUS | Computing
National University of Singapore

Markov Decision Process

CS4246/CS5446

AI Planning and Decision Making

Sem 1, AY2021-22

Topics

- Markov Decision Process (17.1)
  - Model formulation and solution
  - Bellman Equation and Q-function
- Algorithms for solving MDPs
  - Value iteration (17.2.1)
  - Policy iteration (17.2.2)

Solving Sequential Decision Problems

- Decision (Planning) Problem or Model
  - Appropriate abstraction of states, actions, uncertain effects, goals (wrt costs and values or preferences), and time horizon
- Decision Algorithm
  - Input: a problem
  - Output: a solution in the form of an optimal action sequence or policy over time horizon
- Decision Solution
  - An action sequence or solution from an initial state to the goal state(s)
    - An optional solution or action sequence; OR
    - An optimal policy that specifies “best” action in each state wrt to costs or values or preferences
  - (Optional) A goal state that satisfies certain properties

Recall: Decision Making under Uncertainty

- Decision Model:
  - Actions: $a \in A$
  - Uncertain current state: $s \in S$ with probability of reaching: $P(s)$
  - Transition model of uncertain action outcome or effects:
    $P(s'|s, a)$ – probability that action a in state s reaches state s'
  - Outcome of applying action a:
    Result(a) – random variable whose values are outcome states
  - Probability of outcome state s', conditioning on that action a is executed:
    $P(\text{Result}(a) = s') = \sum_s P(s)P(s'|s, a)$
  - Preferences captured by a utility function:
    $U(s)$ – assigns a single number to express the desirability of a state s

Sequential Decision Problems

- What are sequential decision problems?
  - An agent's utility depends on a sequence of decisions
  - Incorporate utilities, uncertainty, and sensing
  - Search and planning problems are special cases
  - Decision (Planning) Models:
    - Markov decision process (MDP)
    - Partially observable Markov decision process (POMDP)
    - Reinforcement learning: sequential decision making + learning

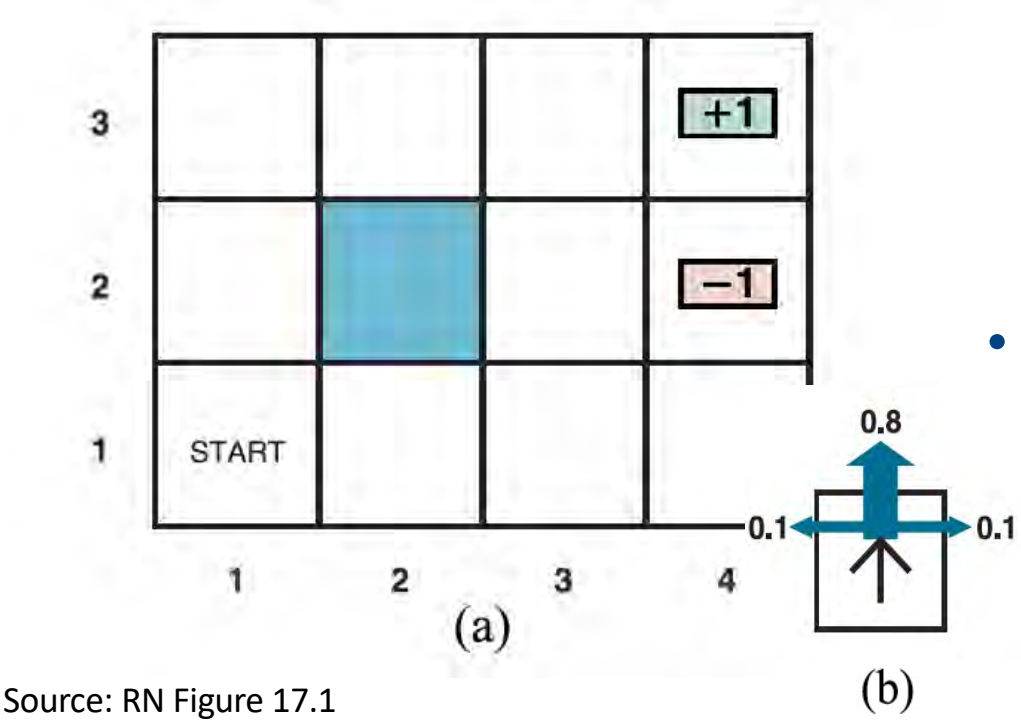
Why Study MDPs?

- Markov decision process (MDP)
  - A sequential decision problem for a fully observable stochastic environment with Markovian transition and additive rewards
- Advantages of using MDPs:
  - General, formal/principled framework for decision-theoretic planning
  - Model uncertainty in dynamics of environment (e.g., in actions)
  - Generate non-myopic action policies
  - Efficient algorithms for solving MDPs with performance guarantees
  - Many real-world applications spanning multiple disciplines:
    - Operations research and logistics (e.g., transportation systems), robotics (e.g., motion planning), computer games (e.g., path planning), multimedia (e.g., camera surveillance)

Model Formulation

Define the problem elements

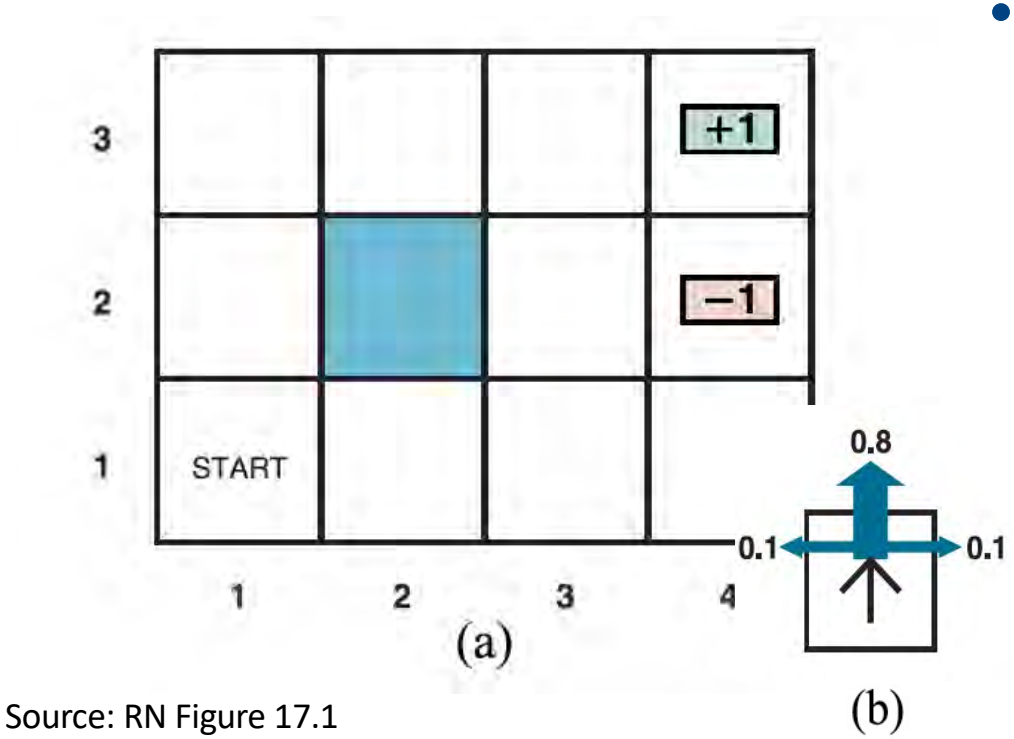
Example: Navigation in Grid World



Source: RN Figure 17.1

- Fully observable 3x4 environment
  - Begin in START state s_0
  - One action a per time step.
  - Terminate when reaching goal states s_g
    - Example: States marked with +1 and -1
- Uncertain action effects:
  - Example: Up, Down, Left, Right:
  - 0.8 - correct direction
  - 0.1 - perpendicular to the left
  - 0.1 - perpendicular to the right

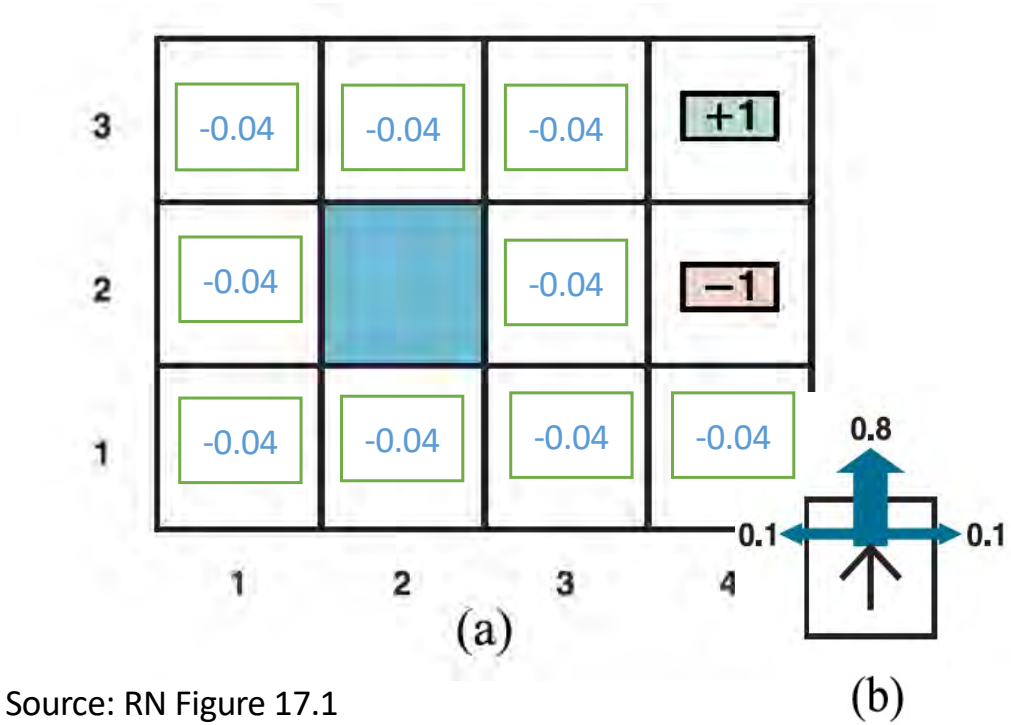
Example: Navigation



Source: RN Figure 17.1

- Transition model:
  - Define (stochastic) outcome of each action a
  - $P(s'|s,a)$ – probability of reaching state s' if action a is done in state s
    - Also denoted as $T(s,a,s')$
- Markovian assumption:
  - Probability depends only on state s and not history of earlier states.

Example: Navigation



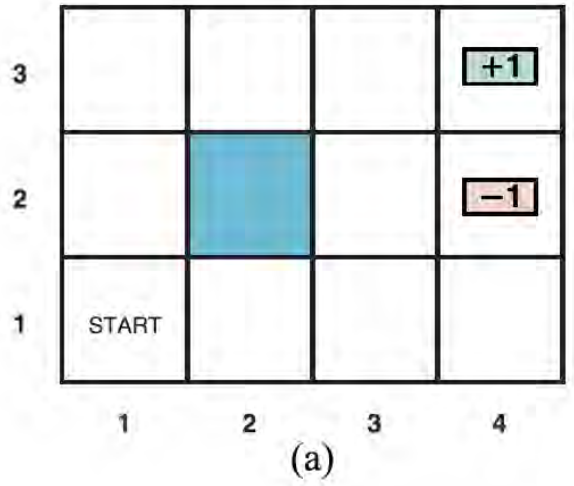
Source: RN Figure 17.1

- Reward model:
  - Define reward received for every transition
  - $R(s, a, s')$ – For every transition from s to s' via action a; AND/OR
  - $R(s)$ – For any transition into state s
  - Rewards may be positive or negative, bound by $\pm R_{max}$
  - Utility function $U(s)$ depends on the sequence of states and actions – environment history – sum of rewards of the states in the sequence

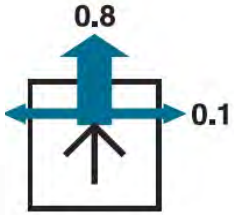
Example:

- $R(s)$ is -0.04 for all states, except the terminal states, where $R(s)$ is $+1$ or -1

Example: Navigation



(a)



(b)

Uncertain action effects:

Up, Down, Left, Right:

0.8 - correct direction

0.1 - perpendicular to the left

0.1 - perpendicular to the right

- If action effects are deterministic:
  - What is the optimal policy or action sequence?
- But agent's actions is stochastic
  - What is the chance of getting to the goal with:[U, U, R, R, R]?
  - How to derive optimal policy from the transition function directly?

Markov Decision Process (MDP)

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

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

  - A reward function $R: S \rightarrow \Re$
  - Solution is a policy – a function to recommend an action in each state: $\pi: S \rightarrow A$
    - 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:
  - $R(s, a) = \sum_{s'} P(s'|s, a)R(s, a, s')$ independent of s'
  - $R(s) = \sum_{s'} P(s'|s, a)R(s, a, s')$, independent of a and s'
- Challenges
  - It is hard to construct reward functions with multiple attributes
  - Balance risk vs reward

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

Exercise

- Question:
  - Consider a variant of the navigation problem.
  - Instead of having actions to move Up, Left, Right, and Down, the actions are to Move Forward, Turn Left 90 degrees in place, and Turn Right 90 degrees in place.
  - Suggest a suitable definition of state in order to model this as a MDP.
- Answer:

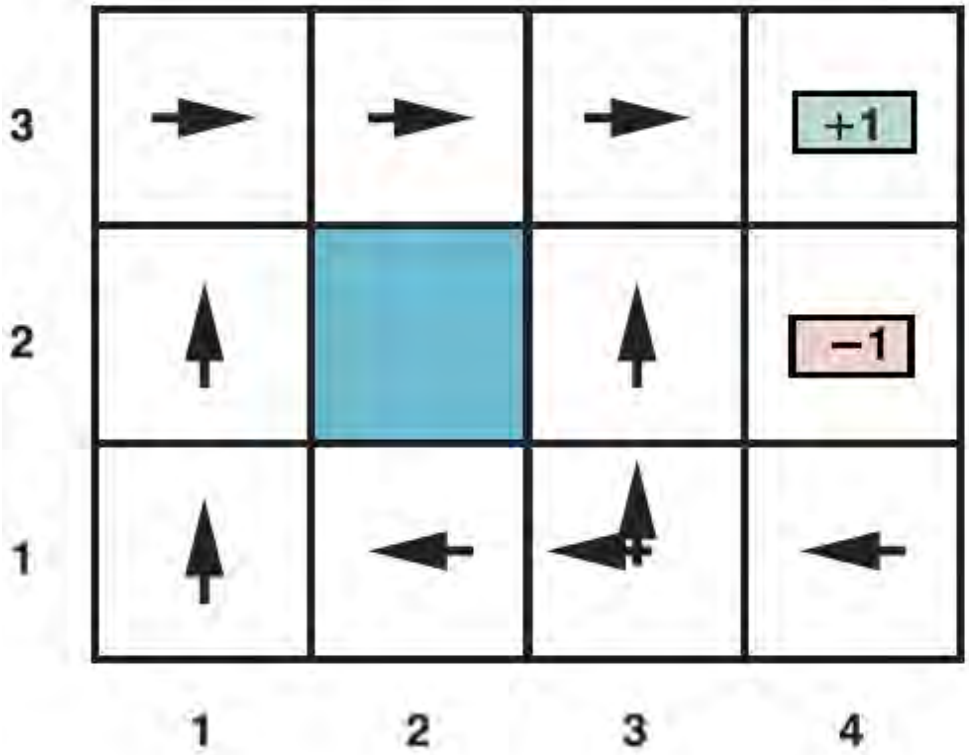
Solving MDPs

Deriving policies – actions to take at each state

Solving MDPs

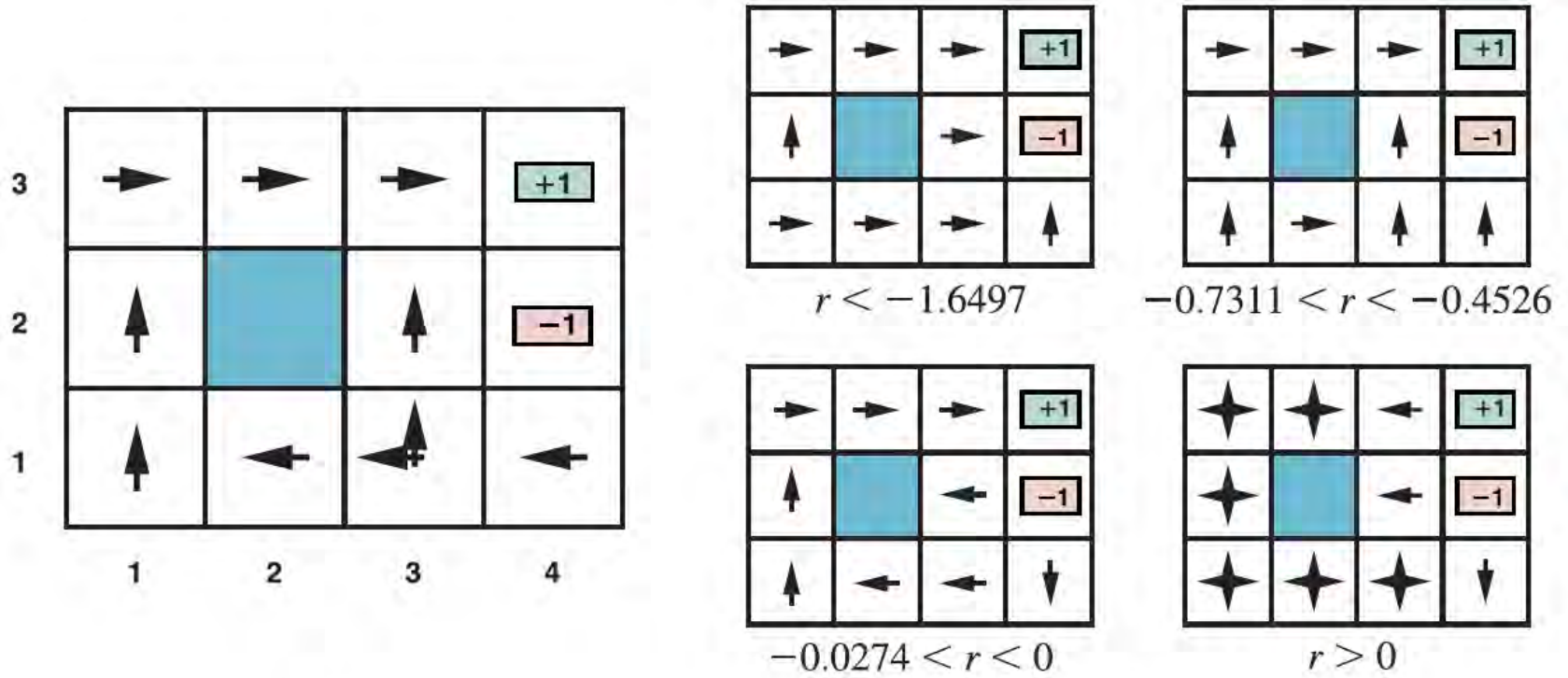
- What does a solution look like?
  - A policy $\pi(s): S \rightarrow A$ is a function from states to actions:
    - For every state s, outputs an appropriate action a.
  - Quality of policy – measured by expected utility of possible state sequence generated by the policy
  - Optimal policy π^* is a 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 17.2

Example: Balancing Risk and Reward



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 17.2

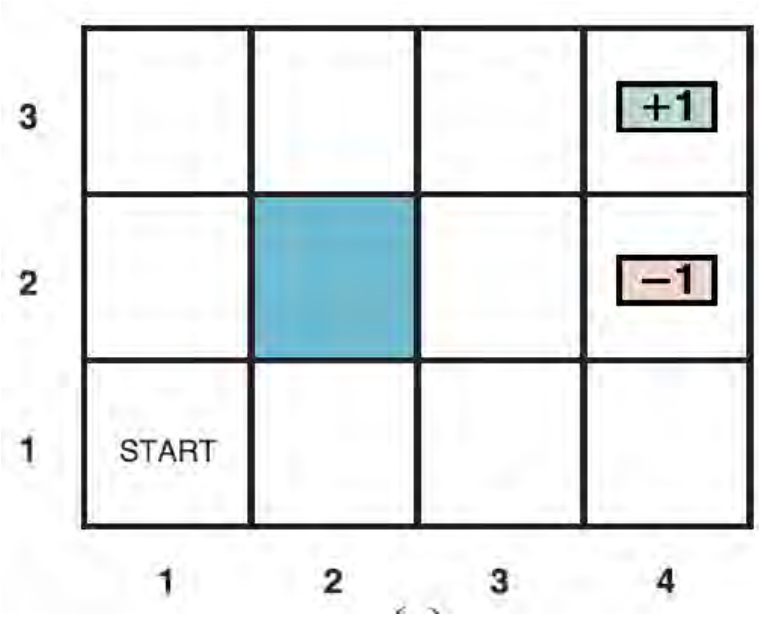
Finite and Infinite Horizon Problems

- Finite horizon: Fixed time N and terminate
  - Return is usually the addition of rewards over the sequence
    - $U_h([s_0, s_1, \dots, s_N] = R(s_0) + R(s_1) + \cdots + R(s_N)$
  - 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 17.2

Rewards in Infinite Horizon Problems

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

Adopts 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 $[s'_1, a'_1, s'_2, \ldots]$

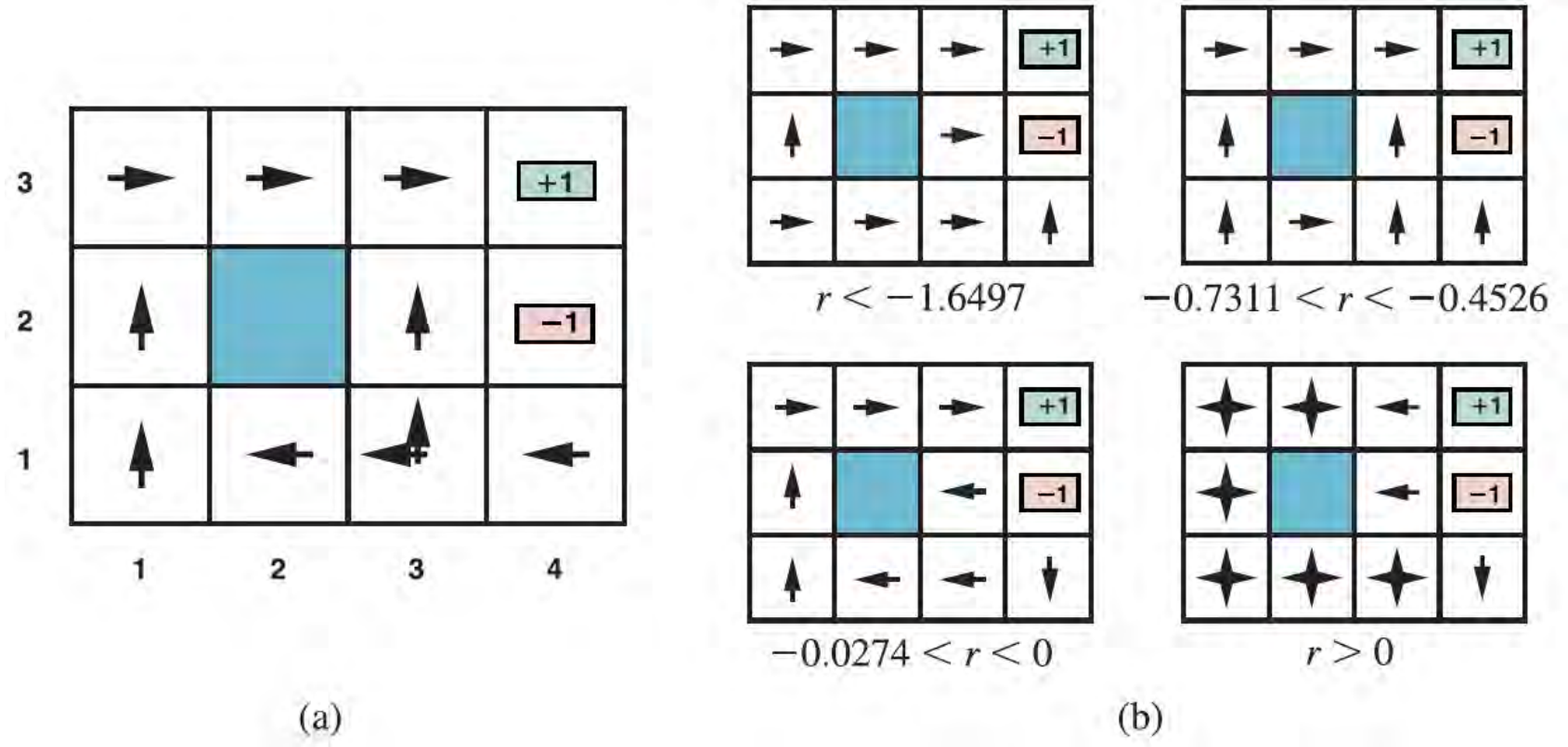
Exercise

- Question:
  - If Dr. Bean's salary is $20,000 per year.
  - How much in total will he earn in his life?
- 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
     $$Uh([s0, a0, s1, a1, s2, \ldots]) = R_{max}/(1 - \gamma)$$
  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



Source: RN Figure 17.2

Exercise

- Question:
  - Number of stationary policies is finite if number of states is finite.
  - How many policies are there with n states and a actions?
- Answer:

Source: Lee WS, Lecture notes, 2020

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, … 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
    - $R(s) = \sum_{s'} P(s'|s,a,s')R(s,a,s')$ 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:

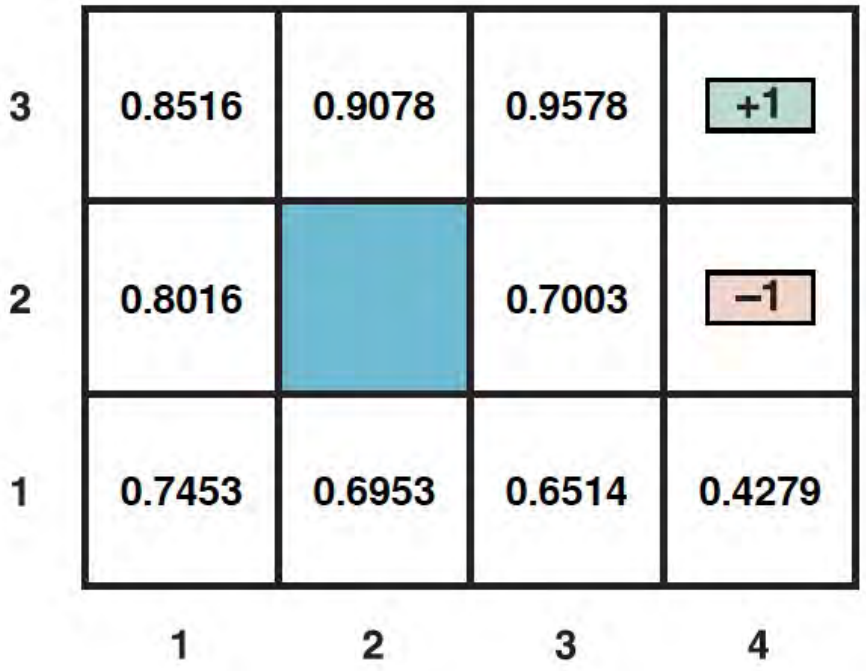
$$\pi^*(s) = \text{argmax}_{a \in A(s)} \sum_{s'} P(s'|s,a)[R(s,a,s') + \gamma U(s')]$$

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

Example: Utilities of States

$R(s) = -0.04$

Numbers are $U(s)$



How to compute the numbers?

Source: RN Figure 17.3

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 t, 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:

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

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

  - $U(s) = U^{\pi^*}(s)$: 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:

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

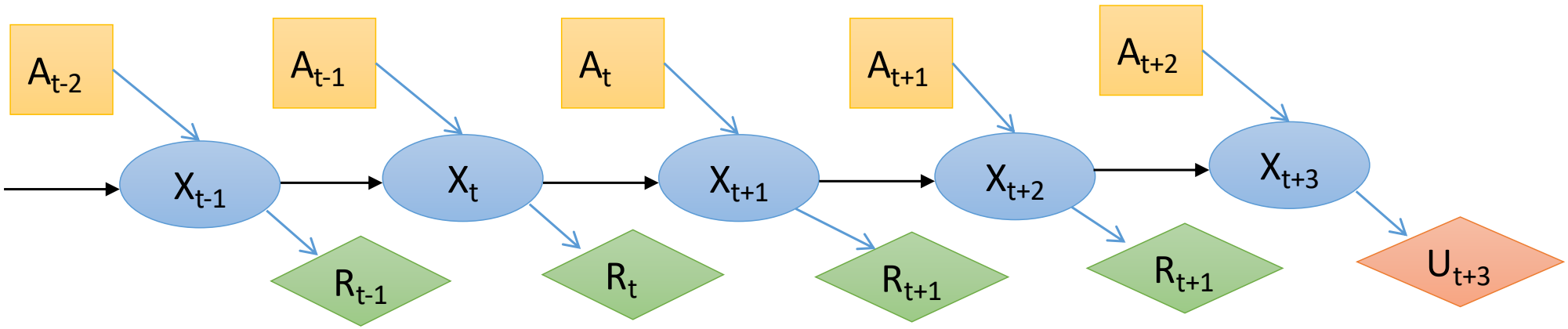
- Bellman Equation for Q-functions

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

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

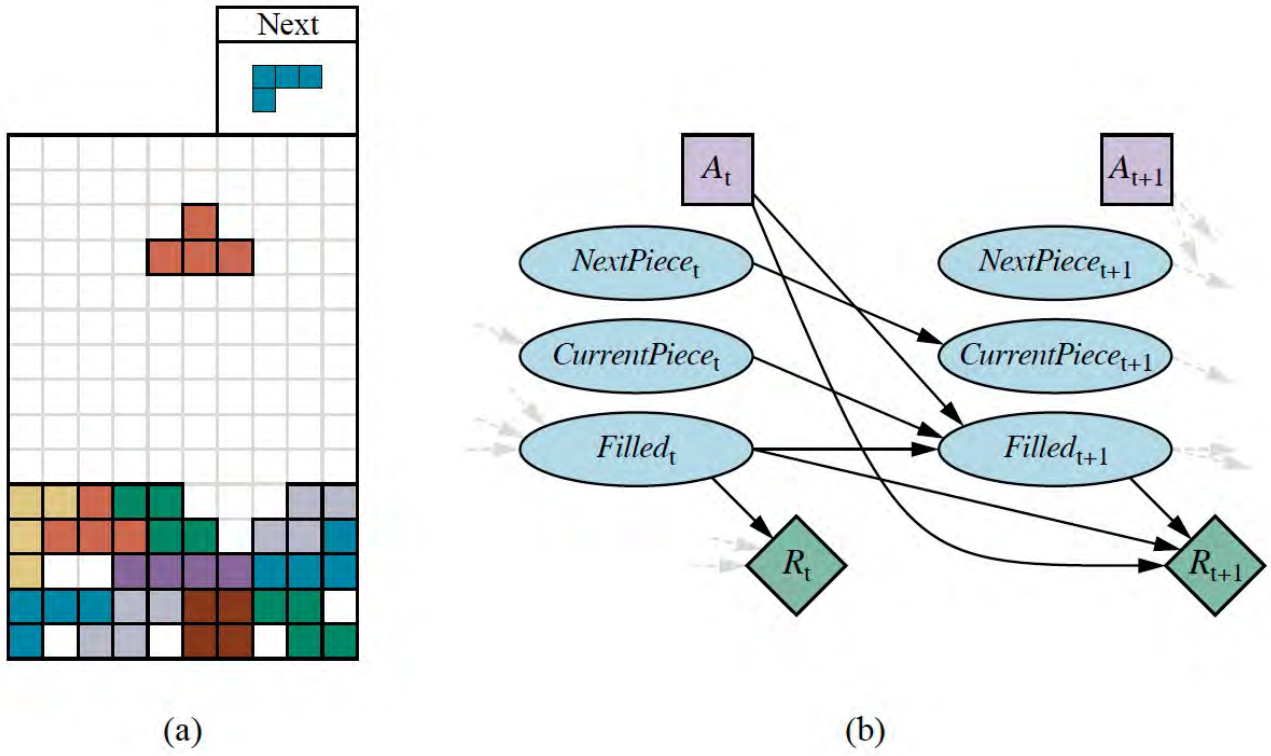
MDP as Dynamic Decision Network

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



What is X_t? A_t? R_t? U_t?

Example: Tetris



Source: RN Figure 17.5

Value Iteration

Solution Method

Bellman Equation and Value Iteration

- Value iteration (A solution algorithm for MDP)
  - Bellman equation is basis of the value iteration algorithm for solving MDPs based on MEU
  - $|S|$ nonlinear equations (due to max) with $|S|$ unknowns (utility of states).
  - Iterative approach to solve Bellman equations
  - Propagating information through state space by means of local updates
  - Solutions are unique solutions

Value Iteration

- Repeatedly perform the Bellman Update

$$U_{i+1}(s) \leftarrow \max_{a \in A(s)} \sum_{s'} P(s'|s,a)[R(s,a,s') + \gamma U_i(s')] \quad \text{OR}$$

$$U_{i+1}(s) \leftarrow R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s'|s,a) U_i(s')$$

- Simultaneous updates of all states:
  - $|S|$ nonlinear equations (due to max) with $|S|$ unknowns (utility of states).
- Guaranteed to reach an equilibrium through convergence
- Final utility values must be unique solutions to the Bellman equations
- Corresponding policy is optimal

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

Value Iteration Algorithm

function VALUE-ITERATION(mdp, ϵ) **returns** a utility function
  inputs: mdp, an MDP with states S, actions $A(s)$, transition model $P(s' \mid s, a)$,
    rewards $R(s, a, s')$, discount γ
    ϵ, the maximum error allowed in the utility of any state
  local variables: U, U', vectors of utilities for states in S, initially zero
    δ, the maximum relative change in the utility of any state

  repeat
    $U \leftarrow U'$; $\delta \leftarrow 0$
    for each state s **in** S **do**
      $U'[s] \leftarrow \max_{a \in A(s)}$ Q-VALUE(mdp, s, a, U)
      if $|U'[s] - U[s]| > \delta$ **then** $\delta \leftarrow |U'[s] - U[s]|$
  until $\delta \leq \epsilon(1-\gamma)/\gamma$
  return U

Bellman update

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

Termination condition

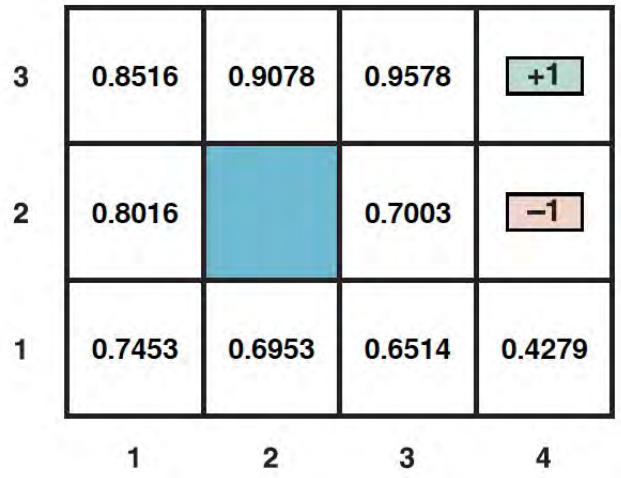
Source: RN Figure 17.6

Example: Solving Bellman Equations

$R(s) = -0.04$

Numbers are $U(s)$

How to compute the numbers?
What is the optimal policy?



For U(1,1):

$$\max\{\ [0.8(-0.04+\gamma U(1,2))+0.1(-0.04+\gamma U(2,1))+0.1(-0.04+\gamma U(1,1))], \quad \text{(U)}$$
$$[0.9(-0.04+\gamma U(1,1))+0.1(-0.04+\gamma U(1,2))], \quad \text{(L)}$$
$$[0.9(-0.04+\gamma U(1,1))+0.1(-0.04+\gamma U(2,1))], \quad \text{(D)}$$
$$[0.8(-0.04+\gamma U(2,1))+0.1(-0.04+\gamma U(1,2))+0.1(-0.04+\gamma U(1,1))]\} \quad \text{(R)}$$

$\pi^*((1,1)) =$

Source: RN Figure 17.3

Convergence

Bellman update operator

- Value iteration:
  - Converges to the (unique) utility function for discounted problems with $\gamma < 1$
- The Bellman update $U_{i+1} \leftarrow BU_i$, is a contraction by a factor of γ on the space of utility vectors:

$$\|BU_i - BU_i'\| \leq \gamma \|U_i - U_i'\|$$

  - where max norm $\|U\| = \max_s U(s)$; $\|U - U'\| = \max_s \|U(s) - U'(s)\|$
- For Bellman equations, the utility or value function U is a fixed point

$$U = BU$$

Some Terminology

- Definitions:
  - A fixed point is any input unchanged by a function
  - A contraction is a function, when applied to two inputs, produces two outputs that are closer together
    - E.g., “divide by two” is a contraction
    - A contraction has only one fixed point
    - When contraction is applied to any argument, the value must get closer to the fixed point; repeated application reaches fixed point in the limit
  - Max norm or distance between two vectors is the maximum difference between any two corresponding elements
    - It measures “length” of a vector by the absolute value of its biggest component

Convergence

- Repeated application of a contraction reaches a unique fixed point U

  $\|U_{i+1} - U\| = \|BU_i - BU\| \leq \gamma\|U_i - U\| \leq \gamma^i\|U_0 - U\|$ for any initial U_0

- Reaching fixed point:
  - $U = BU$ is the fixed point utility function
  - Error $\|U_i - U\|$ reduced by a factor of at least γ on each iteration; converges exponentially fast
  - Utilities of all states bounded by $\pm R_{max}/(1-\gamma)$:

$$\|U_0 - U\| \leq 2R_{max}/(1-\gamma)$$

Convergence

- Factors influencing convergence:
  - N iterations to reach error of at most ϵ:

$$\|U_N - U\| \leq \gamma^N \cdot 2R_{max}/(1-\gamma) \leq \epsilon$$

$$N = \left\lceil \frac{\log\left(\frac{2R_{max}}{\epsilon(1-\gamma)}\right)}{\log\left(\frac{1}{\gamma}\right)} \right\rceil$$

- Termination condition:
  - If the update is small (i.e., no state utility changes by much), then the error, compared with the true utility function, also is small

$$\text{if } \|U_{i+1} - U_i\| < \frac{\epsilon(1-\gamma)}{\gamma} \text{ then } \|U_{i+1} - U\| < \epsilon$$

Convergence: Calculations

- If we run N iterations, we get:

Error

$$\|U_N - U\| \leq \frac{\gamma^N R_{max}}{1-\gamma}$$

- To get error at most ϵ, we have:

$$\frac{\gamma^N R_{max}}{1-\gamma} \leq \epsilon$$
$$\gamma^N R_{max} \leq \epsilon(1-\gamma)$$
$$\frac{R_{max}}{\epsilon(1-\gamma)} \leq \frac{1}{\gamma^N} = \left(\frac{1}{\gamma}\right)^N$$

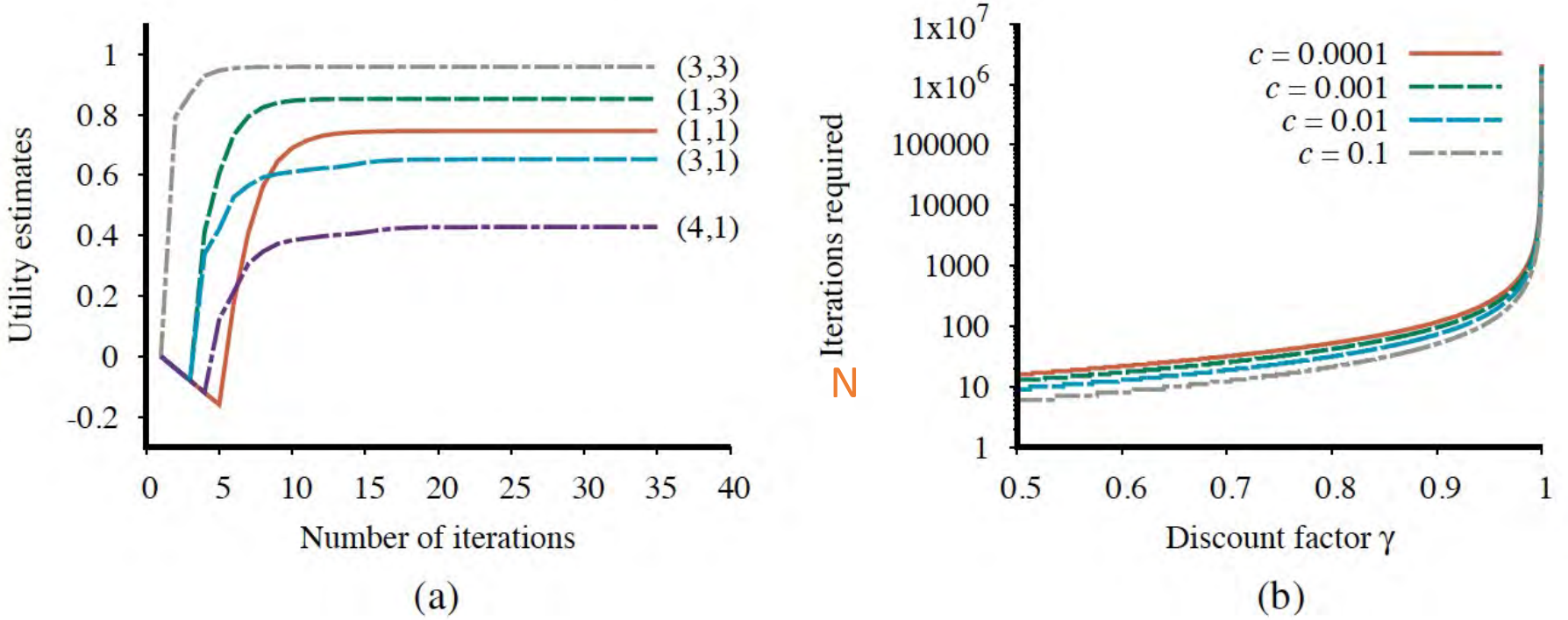
- Take log:

$$N \log\left(\frac{1}{\gamma}\right) \geq \log\left(\frac{R_{max}}{\epsilon(1-\gamma)}\right)$$
$$N = \left\lceil \frac{\log\left(\frac{2R_{max}}{\epsilon(1-\gamma)}\right)}{\log\left(\frac{1}{\gamma}\right)} \right\rceil$$

- Terminating condition:
  - $\|U_{t+1} - U_t\| \leq \epsilon(1-\gamma)/\gamma$
  - $\Rightarrow \|U_{t+1} - U\| \leq \epsilon$

Remember: Values at each state bounded by $\pm \frac{R_{max}}{1-\gamma}$
The 2 in the numerator reflects the bounds

Example: Convergence



N

Source: RN Figure 17.7

Computing Optimal Policy

- Policy Loss
  - Let π_i be the MEU policy wrt U_i
  - Recall: $U^{\pi_i}(s)$is utility obtained if π_i is executed starting in state s
  - Policy loss $\|U^{\pi_i} - U\|$ is the max loss by following π_i instead of π^*.
- Connecting utility error and policy loss

  if $\|U_i - U\| < \epsilon$ then $\|U^{\pi_i} - U\| < 2\epsilon$
- In practice:
  - π_i often becomes optimal long before convergence of U_i

Example: Computing Optimal Policy

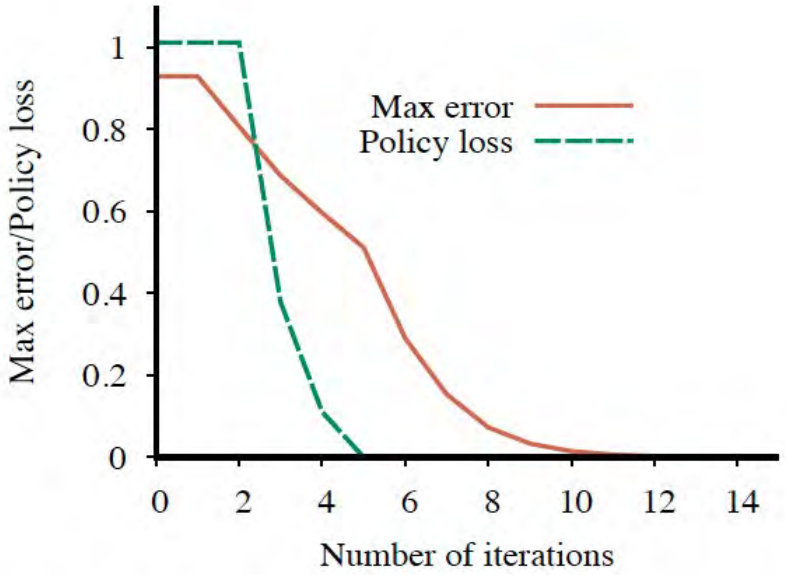


Figure 17.8 The maximum error $\|U_i - U\|$ of the utility estimates and the policy loss $\|U^{\pi_i} - U\|$, as a function of the number of iterations of value iteration on the 4×3 world.

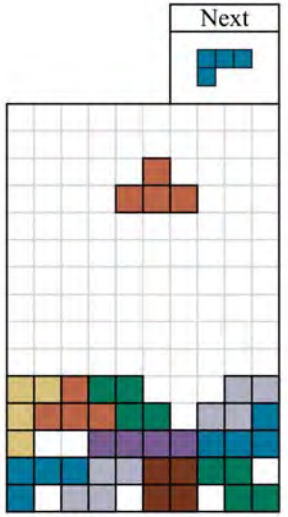
Source: RN Figure 17.8

Summary

- Value iteration
  - Value iteration converges to the correct utilities
  - Can bound the errors in utility estimates if stop after a finite number of iterations
  - Can bound the policy loss from executing the corresponding MEU policy
  - All the results depend on $\gamma < 1$; or similarly derived if $\gamma = 1$ and there are terminal states

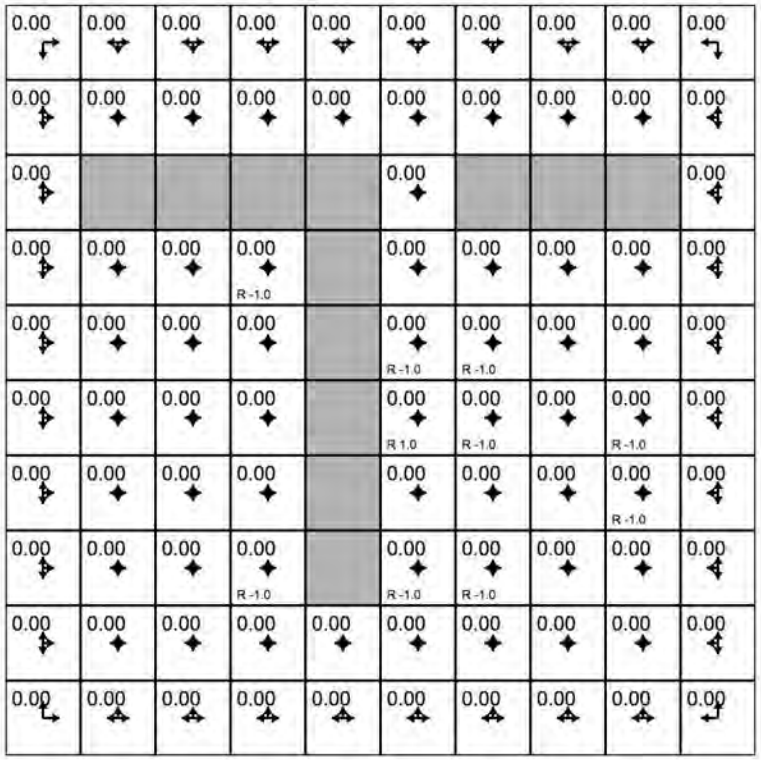
Exercise: Tetris

- Question:
  - Can you solve Tetris using value iteration?
- Answer:



Source: RN Figure 17.5

Value Iteration Demo



https://cs.stanford.edu/people/karpathy/reinforcejs/gridworld_dp.html

Policy Iteration

Solution Method

Motivation for Policy Iteration

- Value iteration:
  - Exact algorithm
  - Scales poorly
- Observations:
  - If one action is clearly better than all others, then exact magnitude of utilities on the states need not be precise
  - Utility function estimate can be inaccurate to derive optimal policy
  - There is another way to find optimal policies: Policy Iteration

Policy Iteration

- Begin with Initial policy: π_0
- Alternate between:

  Policy evaluation:
  - Given a policy π_i, calculate $U_i = U^{\pi_i}$, utility of each state if π_i is executed

$$U_i(s) = \sum_{s'} P(s'|s, \pi_i(s))[R(s, \pi_i(s), s') + \gamma U_i(s')]$$

  Policy improvement:
  - Calculate new MEU policy π_{i+1} using one-step look-ahead based on U_i.
- Terminate when there is no change in utilities for policy improvement step
  - Reaching fixed point of the Bellman update
  - Finite number of policies, hence algorithm must terminate
- Complexity:
  - Assuming $|S| = n$ linear equations with n unknows – can be solved in $O(n^3)$ time.

Policy Iteration Algorithm

function POLICY-ITERATION(mdp) **returns** a policy
  inputs: mdp, an MDP with states S, actions $A(s)$, transition model $P(s' \mid s, a)$
  local variables: U, a vector of utilities for states in S, initially zero
    π, a policy vector indexed by state, initially random

  repeat
    $U \leftarrow$ POLICY-EVALUATION(π, U, mdp)
    $unchanged? \leftarrow$ true
    for each state s **in** S **do**
      $a^* \leftarrow \underset{a \in A(s)}{\text{argmax}}$ Q-VALUE(mdp, s, a, U) $= \sum_{s'} P(s'|s,a)[R(s,a,s') + \gamma U(s')]$
      if Q-VALUE(mdp, s, a^*, U) $>$ Q-VALUE($mdp, s, \pi[s], U$) **then**
        $\pi[s] \leftarrow a^*$; $unchanged? \leftarrow$ false
  until $unchanged?$
  return π

Policy Improvement

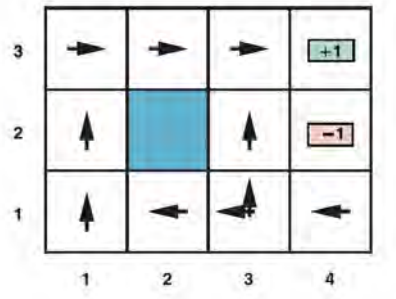
Source: RN Figure 17.9

Example: Implementing Policy Evaluation

Missing the max operator

- Policy evaluation

$$U_i(s) = \sum_{s'} P(s'|s, \pi_i(s))[R(s, \pi_i(s), s) + \gamma U_i(s')]$$
$$= R(s) + \gamma \sum_{s'} P(s'|s, \pi_i(s)) U_i(s')$$



$$U_i(1,1) = 0.8[-0.04 + U_i(1,2)] + 0.1[-0.04 + U_i(2,1)] + 0.1[-0.04 + U_i(1,1)]$$
$$U_i(1,2) = 0.8[-0.04 + U_i(1,3)] + 0.2[-0.04 + U_i(1,2)]$$
$$\vdots$$

- Policy evaluation equations are linear equations
  - Simplified Bellman equation
  - For n states, can be solved in $O(n^3)$ time
  - For large state-spaces, use iterative method:

$$U_{t+1}(s) \leftarrow R(s) + \gamma \sum_{s'} P(s'|s, a) U_t(s')$$

Similar, but simpler than Bellman equations

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

Source: RN 17.2.2

Example: Implementing Policy Improvement

- Policy improvement:
  - For all the states, to find best policy:

$$\pi^*(s) = \underset{a \in A(s)}{\operatorname{argmax}} \sum_{s'} P(s'|s,a)U(s')$$

  - In state (1, 1): $\pi_{new}(1,1) = \underset{a \in A(s)}{\operatorname{argmax}} \sum_{s'} P((s'|(1,1),a)U(s')$
  - ...

$$\sum_{s'} P((s'|(1,1),U)U(s') = 0.8U(1,2) + 0.1U(1,1) + 0.1U(2,1) = \ldots$$

$$\sum_{s'} P((s'|(1,1),L)U(s') = \ldots$$

$$\sum_{s'} P((s'|(1,1),R)U(s') = \ldots$$

$$\sum_{s'} P((s'|(1,1),D)U(s') = \ldots$$

Policy Iteration: Why Does It Work?

- Termination
  - Policy improvement step yields no change in utilities
  - Utility functionU_i is a fixed point of Bellman update, and a solution to the Bellman equations
  - π_i must be an optimal policy
  - Only finitely many policies for a finite state space, each iteration can be shown to yield a better policy, hence policy iteration must terminate
- Correctness
  - Follows from Policy Improvement Theorem [SB 4.2]

Policy Improvement Theorem

- Let:
  - $Q^{\pi}(s,a) = \sum_{s'} P(s'|s,a)[R(s,a,s') + \gamma U^{\pi}(s')$ be the action-utility function of π (one-step look-ahead using action a)
- Theorem: [Proof in SB 4.2]
  - Let π and π' be any pair of deterministic policies such that for all $s \in S$,

$$Q^{\pi}(s,\pi'(s)) \geq U^{\pi}(s)$$

  - Then

$$U^{\pi'}(s) \geq U^{\pi}(s) \text{ for all } s \in S$$

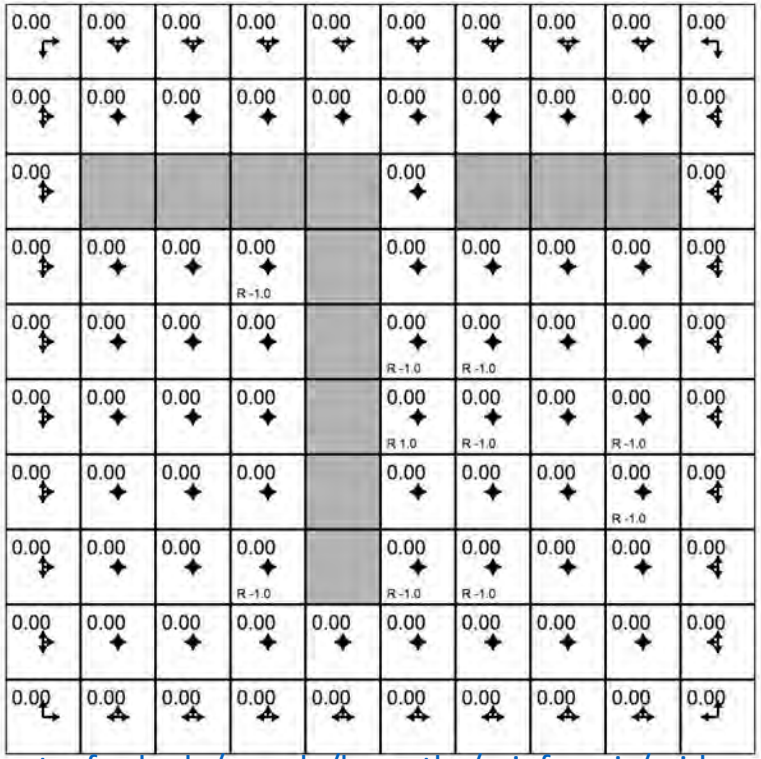
  - If the quality of $Q^{\pi}(s,\pi'(s))$ is strict for any state s, then corresponding inequality for $U^{\pi'}(s)$ is strict for that s

Source: Sutton and Barto, 2018/2020 Section 4.2

General Policy Iteration

- Modified policy iteration:
  - Do only k iterations instead of reaching convergence
  - Approximate evaluation
- Asynchronous policy iteration:
  - Pick only a subset of states for policy improvement or for updating in policy evaluation
  - Converges as long as continuously update all states
- Generalized policy iteration:
  - Update utility according to policy, and improve policy wrt the utility function.
  - Value iteration, policy iteration, asynchronous policy iteration are all special cases.
  - May interleave in different ways based on conditions

Policy Iteration Demo



https://cs.stanford.edu/people/karpathy/reinforcejs/gridworld_dp.html

Homework

- Readings
  - [RN] 17.1, 17.2.1, 17.2.2
  - [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]