Lecture 10
Markovian Modeling I

仿生设计与学习实验室
Bionic Design & Learning Lab

AncoraSIR.com

SUSTech
Southern University of Science and Technology

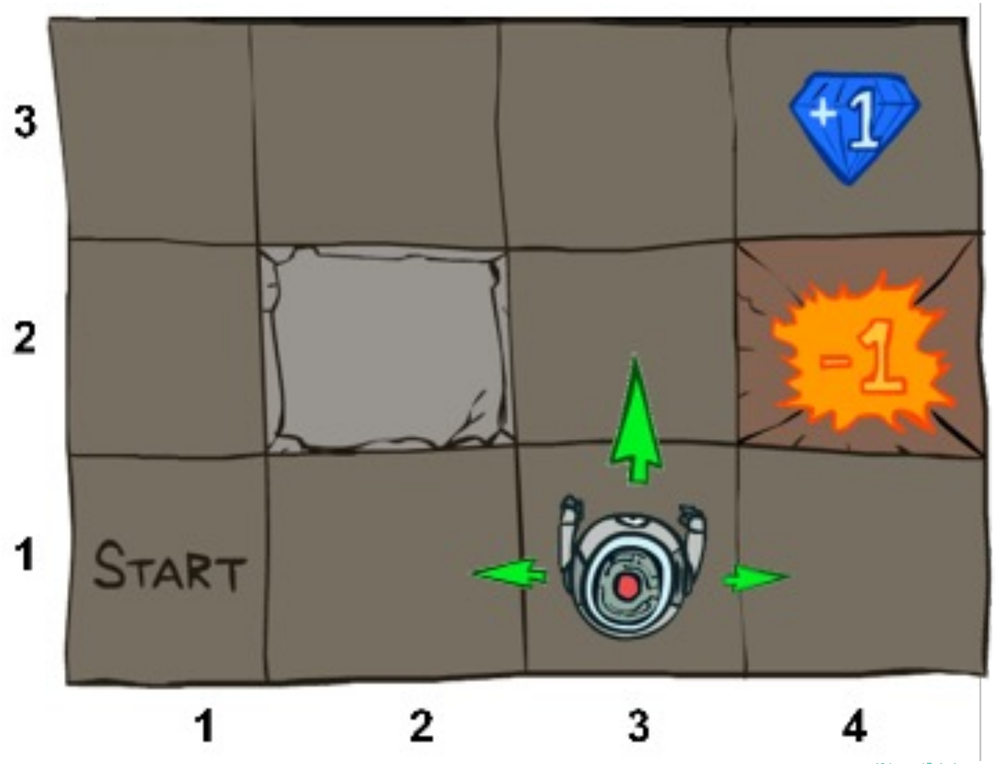
Sequential Decision Problems

AncoraSIR.com

An Example of Grid World

A maze-like problem with an agent in a grid and walls blocking the agent's paths

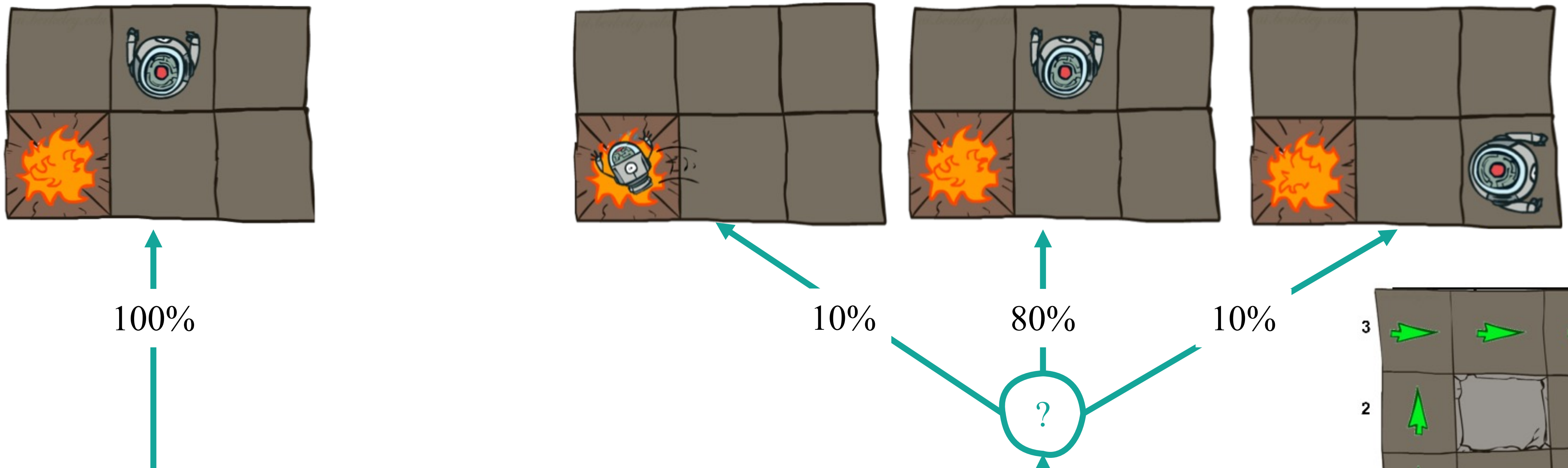
- **Stochastic Motion**: *Actions do not always go as planned*
  - 80% of the time, intended actions occur as planned
  - 10% of the time, turning left/right to the intended action
  - A collision with a wall results in no movement
- **Reward Mechanism**: *received at each time step*
  - Two terminal states with (**BIG**) reward +1 and -1
  - All other states have a (**LIVING**) reward of –0.04
- **The Goal**
  - Maximize the sum of rewards

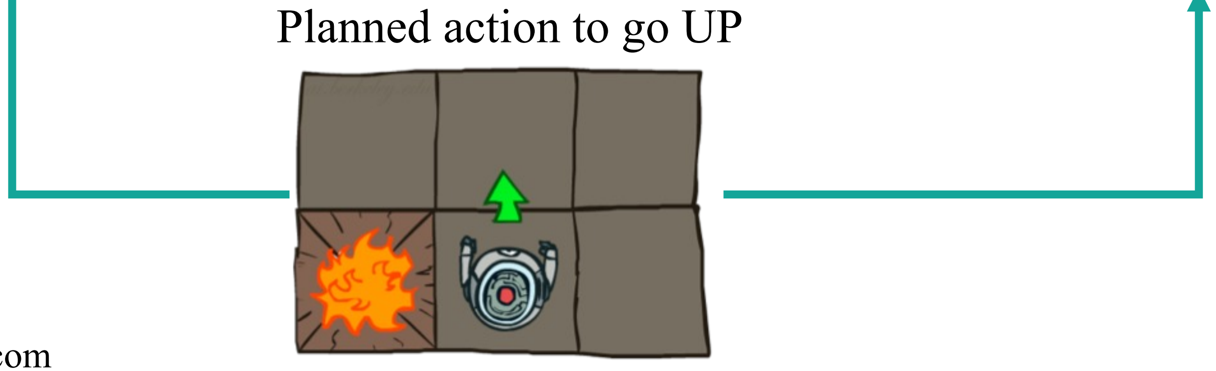


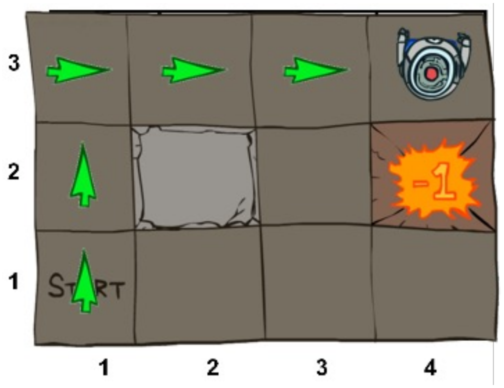
AncoraSIR.com

Grid World Actions

Deterministic *Motion vs.* ***Stochastic*** *Movement*







Same goal as before

- If all actions occur as planned
  - $0.8^5 = 0.32768$
- If not, but still made the goal
  - $0.8 \times 0.1^4 = 0.00008$ (one small chance)

AncoraSIR.com

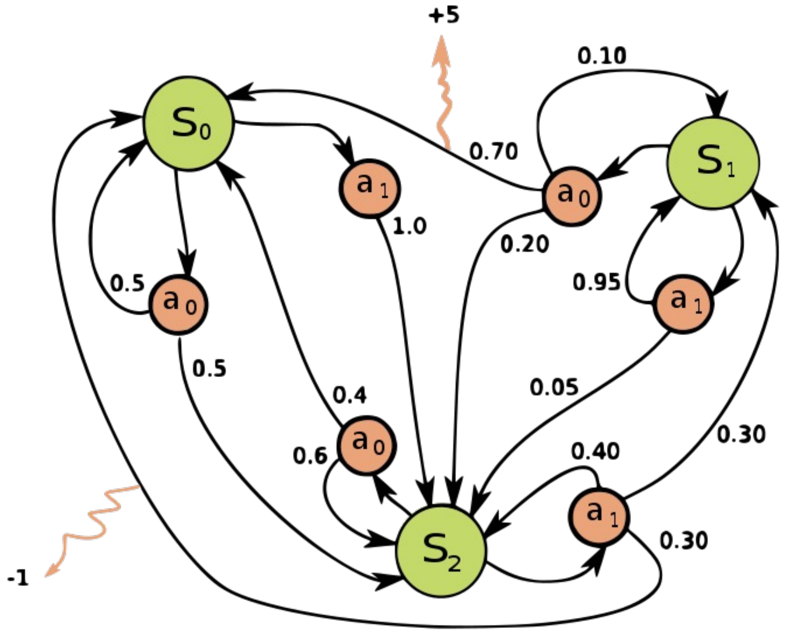
SUSTech
Southern University of Science and Technology

Define a Markov Decision Process

A fully observable, stochastic environment with a Markovian transition model and additive rewards

- A finite set of **states** $s \in S$
  - With a start state s_0, and (maybe) a terminal state
- A finite set of **actions** $a \in A$
  - A_s is the finite set of actions available from state s
- A **transition function** $P(s' \mid s, a)$
  - $\Pr(s_{t+1} = s' \mid s_t = s, a_t = a)$ is a probability
  - Action a in state s at time t will lead to state s' at time $t + 1$,
- A **reward function** $R(s' \mid s, a)$
  - Can be an immediate reward or an expected immediate reward
  - After transitioning from state s to state s', due to action a

4-tuple (S, A, P_a, R_a)



A Sequential Decision Problem
Or MDP

AncoraSIR.com

Markovian Policy

What does a solution to the problem look like?

- "Markov"
  - Action outcomes depend only on the current state (not history)

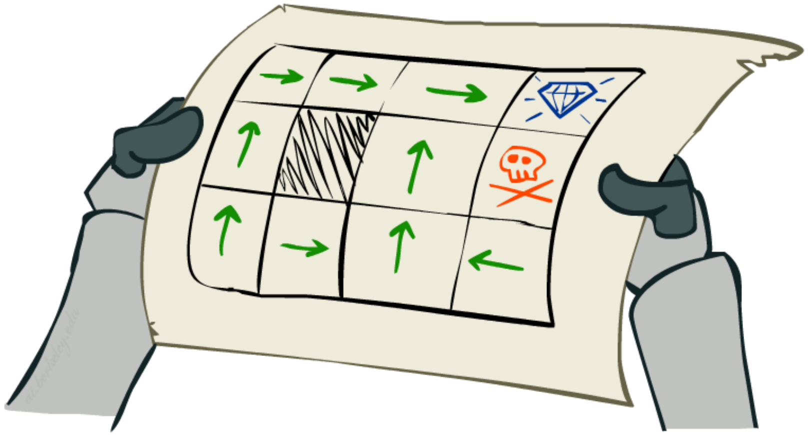
$$P(S_{t+1} = s' \mid S_t = s_t, A_t = a_t, S_{t-1} = s_{t-1}, A_{t-1}, \ldots, S_0 = s_0)$$
$$= P(S_{t+1} = s' \mid S_t = s_t, A_t = a_t)$$

Andrey Markov
(1856-1922)

- Policy $\pi(s)$
  - A a solution that specifies what the agent should do for any state that the agent might reach (*from* ***start*** *to* ***goal***)

Non-Markovian examples
- Robot dynamics (hard)
- Quantum physics

- Optimal Policy π^*
  - A policy that yields the highest expected utility

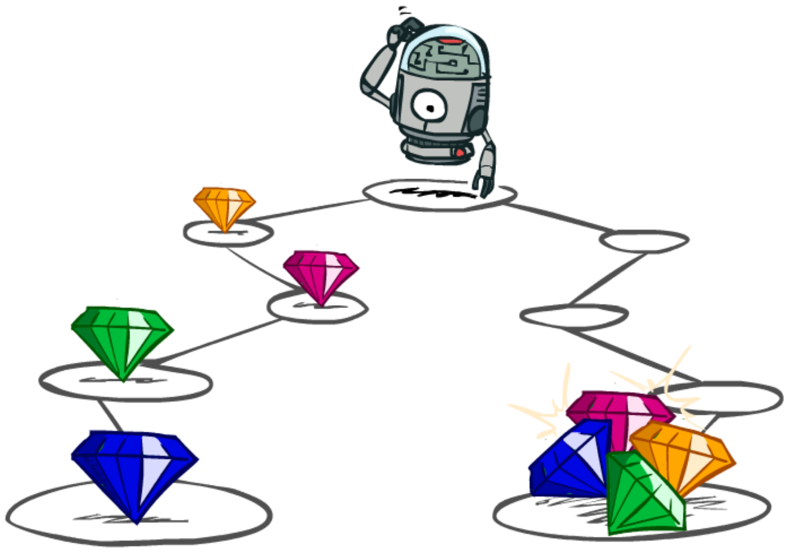


- Explicit representation of the agent function
  - a description of a simple reflex agent, computed from the information used for a utility-based agent

A Finite or Infinite Horizon For Decision Making

Utility Over Time, or a utility function on environment histories, $U_h([s_0, s_1, \dots, s_n])$

- **Finite Horizon** (*nonstationary optimal policy*)
  - A fixed time *N* after which nothing matters, the game is over
  - $U_h([s_0, s_1, \dots, s_{N+k}]) = U_h([s_0, s_1, \dots, s_N])$, for all $k > 0$
  - the optimal action in a given state could change over time (*opportunities are limited*)



- **Infinite Horizon** (*stationary optimal policy*)
  - With no fixed time limit, why behaving differently in the same state at different times?
  - The optimal action depends **only** on the current state (a simpler problem)

AncoraSIR.com

How to Calculate the Utility of State Sequences

Using multi-attribute utility theory with outcomes characterized by two or more attributes

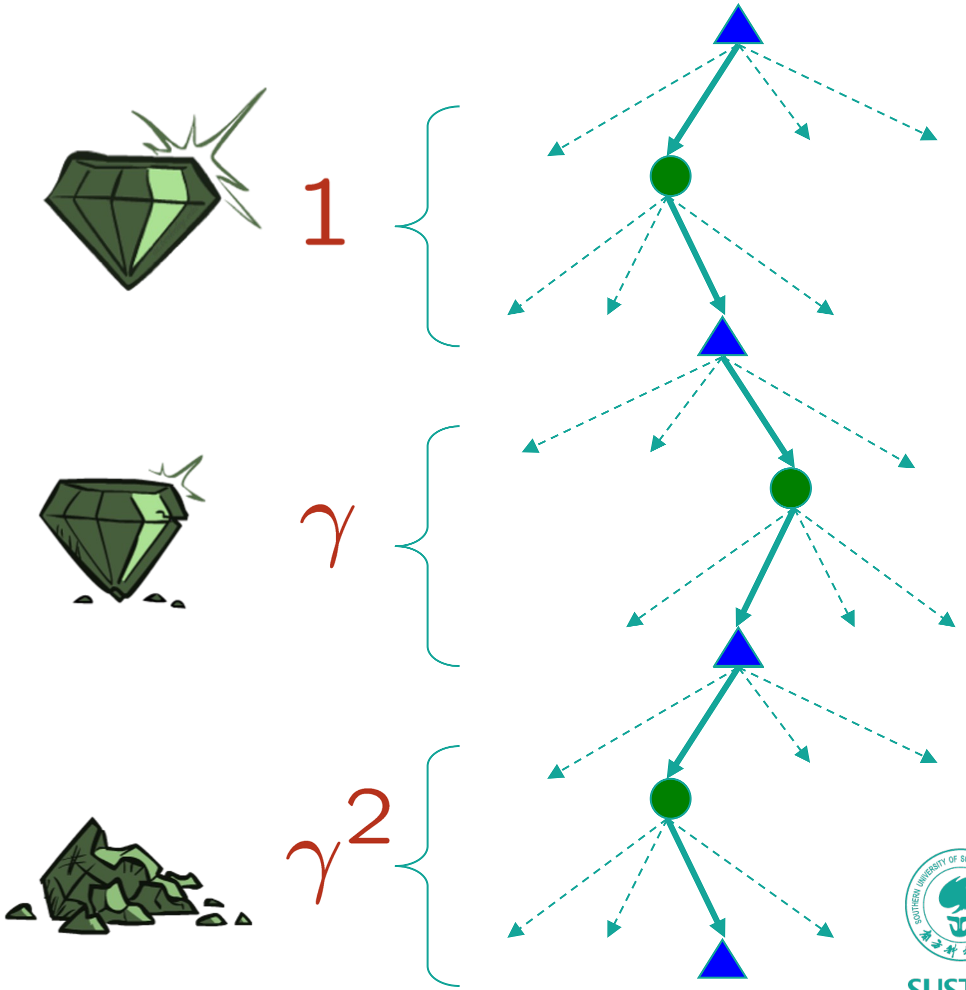
- **Attribute**: A state s_i of the state sequence $[s_0, s_1, s_2, \dots,]$
- Assumption on **Stationary Preference**
  - The agent's preferences between state sequences are **stationary**
  - If two state sequences $[s_0, s_1, s_2, \dots,]$ and $[s_0', s_1', s_2', \dots,]$ begin with the same state (i.e., $s_0 = s_0'$), then the two sequences should be preference-ordered the same way as the sequences $[s_1, s_2, \dots,]$ and $[s_1', s_2', \dots,]$
- **Additive rewards** for the utility of a state sequence
  - $U_h([s_0, s_1, s_2, \dots,]) = R(s_0) + R(s_1) + R(s_2) + \cdots$
  - *Just like the path cost functions in heuristic search algorithms*

AncoraSIR.com

Discounted Rewards

$$U_h([s_0, s_1, s_2, \dots,]) = R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \cdots$$

- **How to discount?**
  - Each time we descend a level, we multiply in the discount factor once, i.e., $\gamma \in [0, 1]$
- **Why discount?**
  - Think of it as a gamma chance of ending the process at every step
  - Also helps our algorithms converge
- **Example**: discount of 0.5
  - $U_h([1, 2, 3]) = 1 + 0.5\times 2 + 0.5^2\times 3$
  - $U_h([1, 2, 3]) < U_h([3, 2, 1])$



What if the Game Lasts Forever?

Do we get infinite rewards?

- With discounted rewards, the utility of an infinite sequence is ***finite***
  - If $\gamma < 1$ and rewards are bounded by $\pm R_{max}$, we have
  - $U_h([s_0, s_1, s_2, \dots,]) = \sum_{t=0}^{\infty} \gamma^t R(s_t) \leq \sum_{t=0}^{\infty} \gamma^t R_{max} = \frac{R_{max}}{1-\gamma}$

- **Optimal Quantities**
  - The value (utility) of a state s:
    - $V^*(s)$ = expected utility starting in s and acting optimally
  - The value (utility) of a q-state (s, a):
    - $Q^*(s, a)$ = expected utility starting out having taken action a from state s and acting optimally
  - The optimal policy:
    - $\pi^*(s)$ = optimal action from state s

AncoraSIR.com

How to Compare Policies

By comparing the expected utilities obtained when executing them

- Assumption
  - The agent is in some initial state s, and a particular policy π to be executed
  - Define S_t (a random variable) to be the state the agent reaches at time t, i.e., $S_t = s$
  - The probability distribution over state sequences $S_1, S_2, \ldots,$ is determined by the initial state s, the policy π, and the transition model for the environment.
- Define the Expected Utility

$$U^{\pi}(s) = E\left[\sum_{t=0}^{\infty} \gamma^t R(s_t)\right]$$

- Finding the Optimal Policy (when s is the starting state)

$$\pi_s^* = \mathrm{argmax}_{\pi} U^{\pi}(s)$$

Maximum Expected Utility

Choose the action that maximizes the expected utility of the subsequent state

- Notice the differences
  - $R(s)$ is the “short term” reward for being in s,
  - $U(s)$ is the “long term” total reward from s onward.

- The principle of Maximum Expected Utility

$$\pi_s^* = \text{argmax}_{a \in A(s)} \sum_{s'} P(s' \mid s, a) U(s')$$

3	0.812	0.868	0.918	$\boxed{+1}$
2	0.762		0.660	$\boxed{-1}$
1	0.705	0.655	0.611	0.388
	1	2	3	4

Notice that the utilities are higher for states closer to the +1 exit, because fewer steps are required to reach the exit.

AncoraSIR.com

Value Iteration

AncoraSIR.com

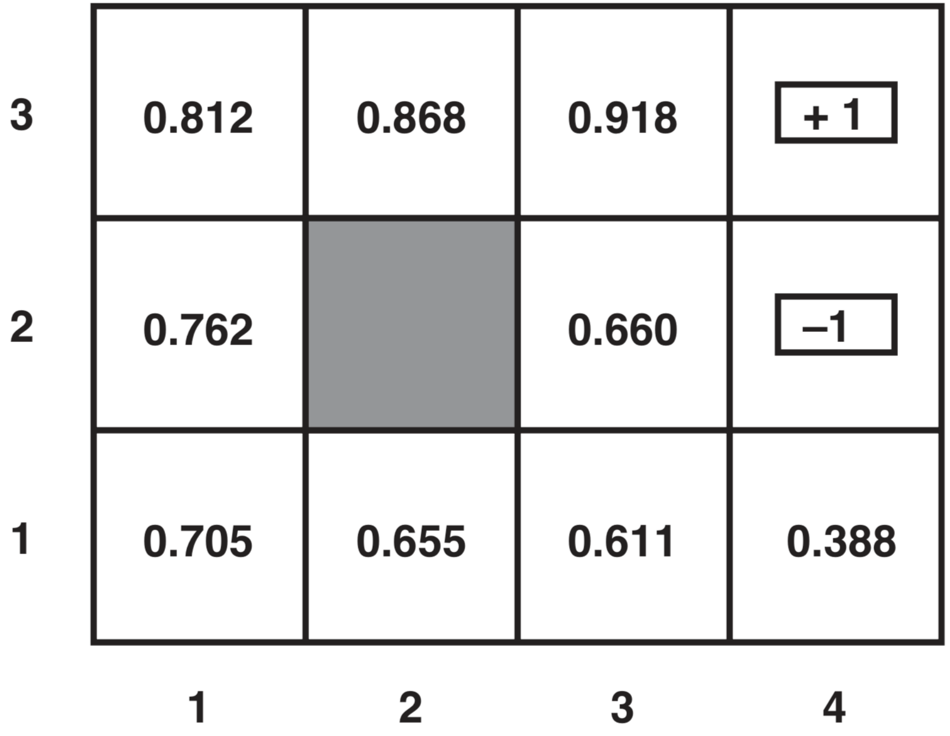
The Bellman Equation for Utilities

A direct relationship between the utility of a state and the utility of its neighbors

- The utility of a state is the immediate reward for that state plus the expected discounted utility of the next state, assuming that the agent chooses the optimal action.

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

$$
\begin{aligned}
U(1,1) = -0.04 + \gamma \max[\ & 0.8U(1,2) + 0.1U(2,1) + 0.1U(1,1), && (Up)\\
& 0.9U(1,1) + 0.1U(1,2), && (Left)\\
& 0.9U(1,1) + 0.1U(2,1), && (Down)\\
& 0.8U(2,1) + 0.1U(1,2) + 0.1U(1,1)\ ]. && (Right)
\end{aligned}
$$



Which action is the best solution?

AncoraSIR.com

Nonlinearity of the Bellman Equations

The Bellman equation is the basis of the value iteration algorithm for solving MDPs.

- Solving n equations with n unknown utilities of the states
    - n possible states
    - n Bellman equations for each state
    - Non-linear "max" operation

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

3	0.812	0.868	0.918	+ 1
2	0.762		0.660	–1
1	0.705	0.655	0.611	0.388
	1	2	3	4

- Iterative Approach for a solution

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

AncoraSIR.com

The Value Iteration Algorithm

for calculating utilities of states

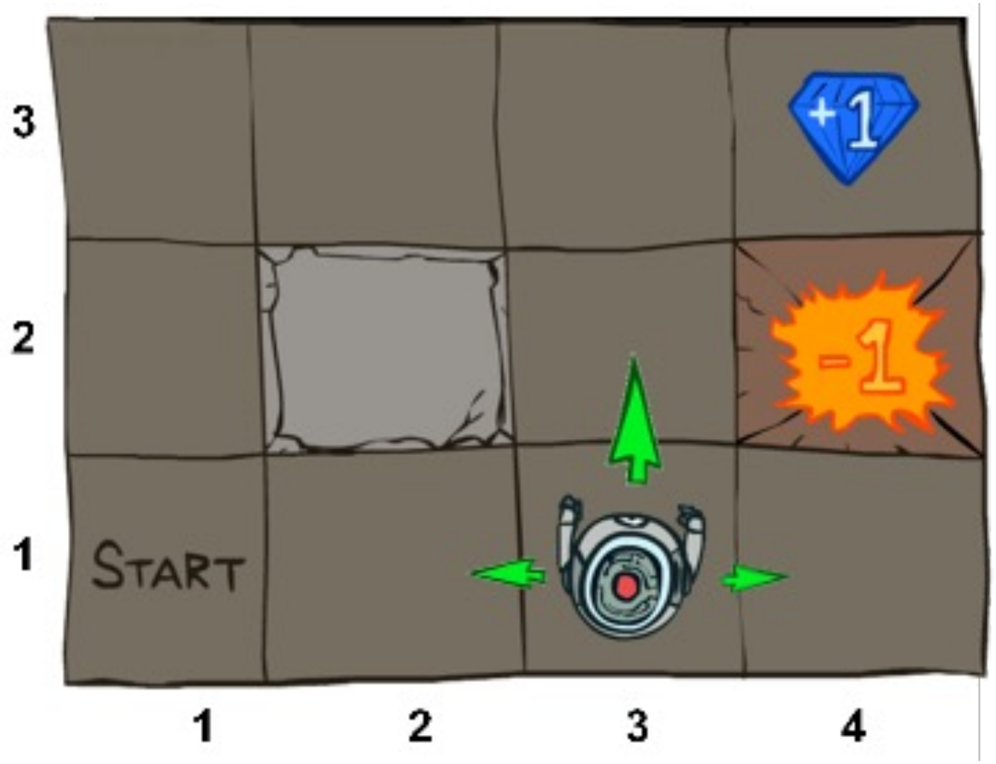
function VALUE-ITERATION(mdp, ϵ) **returns** a utility function
  inputs: mdp, an MDP with states S, actions $A(s)$, transition model $P(s' \mid s, a)$,
    rewards $R(s)$, discount γ
    ϵ, the maximum error allowed in the utility of any state
  local variables: U, U', vectors of utilities for states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

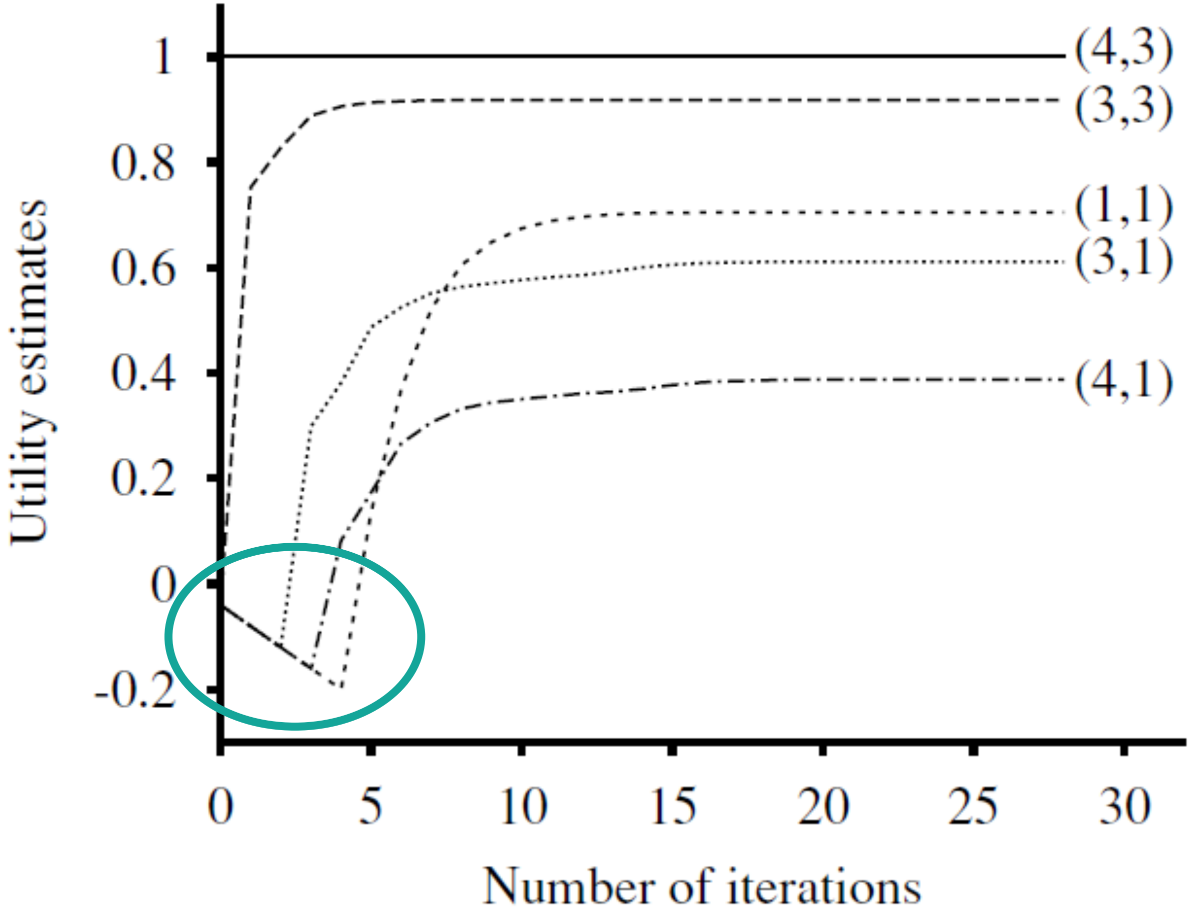
  repeat
    $U \leftarrow U'$; $\delta \leftarrow 0$
    for each state s **in** S **do**
      $U'[s] \leftarrow R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s' \mid s, a)\, U[s']$
      if $|U'[s] - U[s]| > \delta$ **then** $\delta \leftarrow |U'[s] - U[s]|$
  until $\delta < \epsilon(1-\gamma)/\gamma$ ← Terminal Condition
  return U

AncoraSIR.com

The Grid World Example

evolution of the utilities of selected states using value iteration





AncoraSIR.com

Contraction

Why value iteration eventually converges to a unique set of solutions of the Bellman equations?

- A function of one argument that,
  - when applied to two different inputs in turn,
  - produces two output values that are "closer together,"
  - by at least some constant factor, than the original inputs

an operator

$$U_{i+1} \leftarrow B\,U_i$$

max norm

$$||U|| = \max_s |U(s)|$$

$$||B\,U_i - B\,U_i'|| \leq \gamma\,||U_i - U_i'||$$

10 - 4 = 6

/2

5 - 2 = 3

/2 1.5

/2 ?

Approaching a fixed point in limit

- The Bellman update is a contraction by a factor of γ on the space of utility vectors
  - value iteration always converges to a unique solution of the Bellman equations whenever $\gamma < 1$

AncoraSIR.com

The Rate of Convergence

Convergence of Value Iteration

- Property of the utilities of all states

$$U_h([s_0, s_1, s_2, \ldots]) = \sum_{t=0}^{\infty} \gamma^t R(s_t) \leq \sum_{t=0}^{\infty} \gamma^t R_{\max} = R_{\max}/(1-\gamma)$$

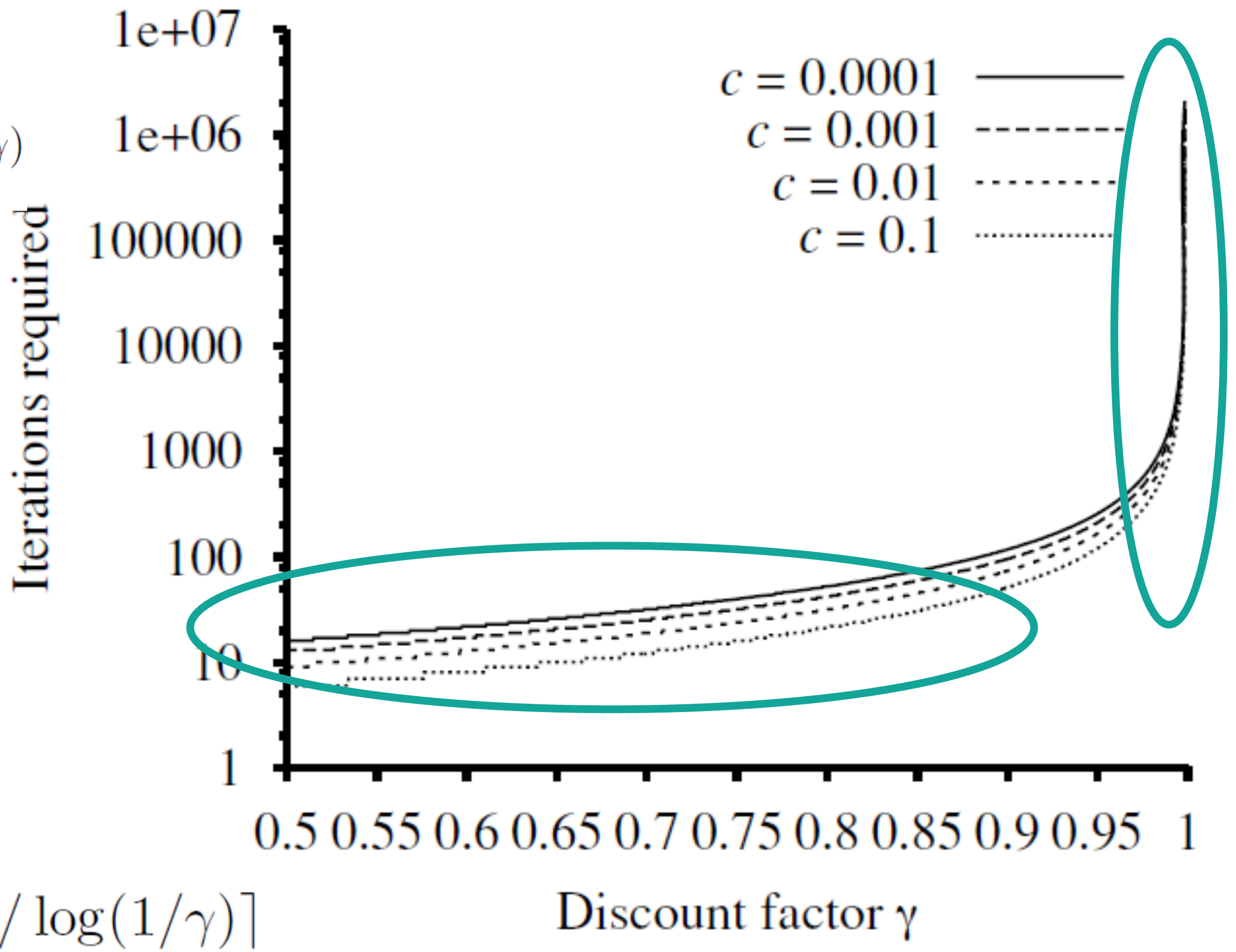
- Then, the maximum initial error

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

- Suppose we run for N iterations to reach an error of at most ϵ.
- Because the error is reduced by at least γ each time

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

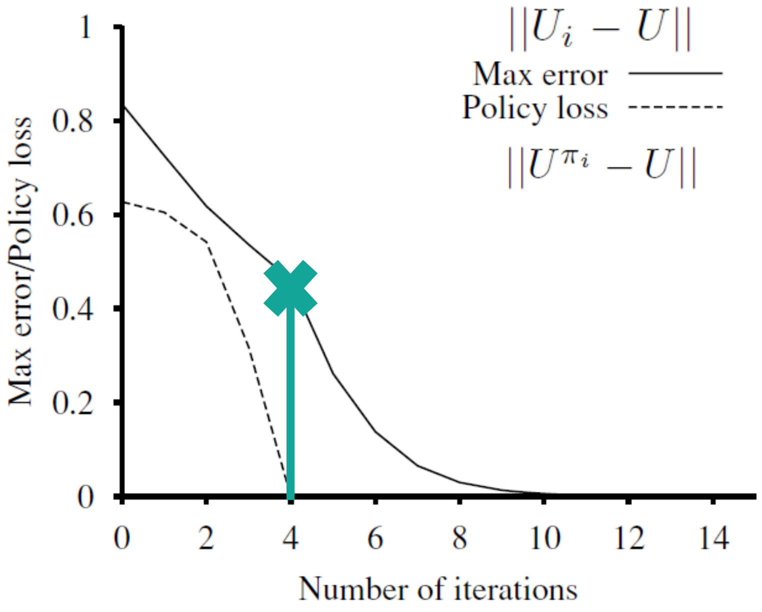
$$\rightarrow N = \lceil \log(2\boxed{R_{\max}/\epsilon}(1-\gamma))/\log(1/\gamma) \rceil$$



What the Agent Really Cares About?

How well it will do if it makes its decisions on the basis of this utility function?

- Suppose that after i iterations of value iteration,
  - the agent has an estimate U_i of the true utility U and
  - obtains the **M**aximum **E**xpected **U**tility policy π_i based on one-step look-ahead using U_i
- ***Will the resulting behavior be nearly as good as the optimal behavior?***
  - YES
  - $U^{\pi_i}(s)$ is the utility obtained if π_i is executed starting in s
  - **Policy loss** $||U^{\pi_i} - U||$ is the most the agent can lose by executing π_i instead of the optimal policy π^*



AncoraSIR.com

Thank you~

songcy@sustech.edu.cn