بسم الله الرحمن الرحیم

هوش مصنوعی

فصل ۱۷

اتخاذ تصمیم‌های پیچیده

Making Complex Decisions

کاظم فولادی قلعه
دانشکده مهندسی، پردیس فارابی
دانشگاه تهران

http://courses.fouladi.ir/ai

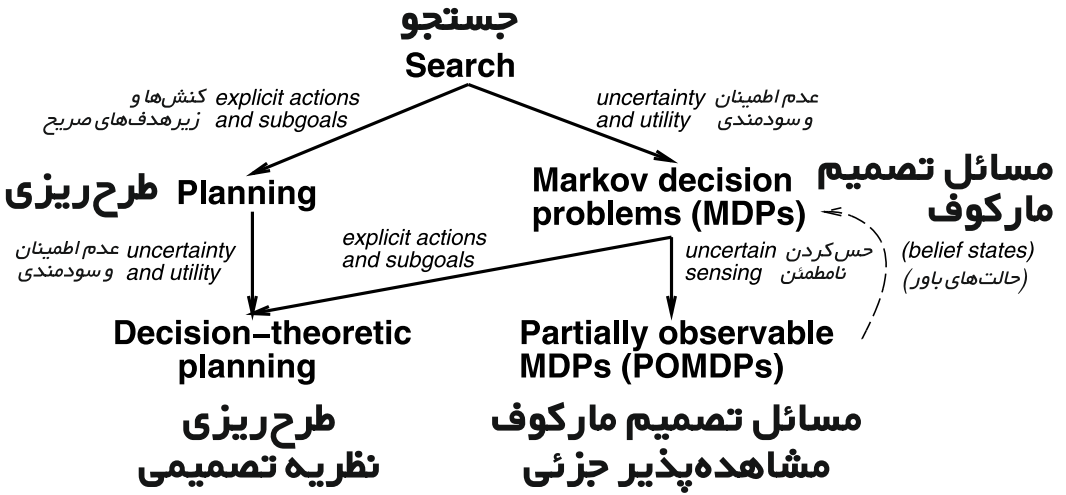
انواع تصمیم

انواع تصمیم	
تصمیم ساده *Simple Decision*	**تصمیم پیچیده** *Complex Decision*
تصمیم‌های تک‌ضرب *One-Shot*	**تصمیم‌های چندمرحله‌ای** *Multi-Stage*
تصمیم بر روی یک کنش	تصمیم بر روی یک دنباله از کنش‌ها

انواع تصمیم

تصمیم‌های پیچیده

مسائل تصمیم‌گیری



هوش مصنوعی

اتخاذ تصمیم‌های پیچیده

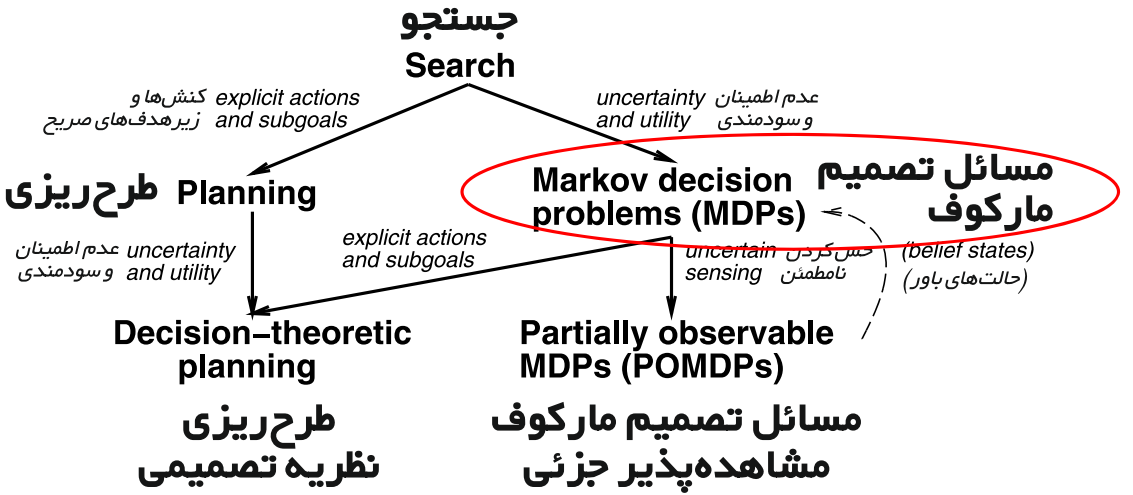
۱

مسائل تصمیم‌گیری ترتیبی

مسائل تصمیم‌گیری

مسائل تصمیم‌گیری مارکوف

MARKOV DECISION PROBLEMS (MDPs)



مسائل/فرآیند تصمیم‌گیری مارکوف

مثال

MARKOV DECISION PROBLEMS / PROCESS (MDPS)

مؤلفه‌های تعریف یک MDP		
تابع پاداش *Reward Function*	**مدل گذر** *Transition Model*	**حالت اولیه** *Initial State*
$R(s)$ $R(s,a)$ $R(s,a,s')$	$T(s,a,s')$	s_0

States $s \in S$, actions $a \in A$

Model $T(s,a,s') \equiv P(s'|s,a)$ = probability that a in s leads to s'

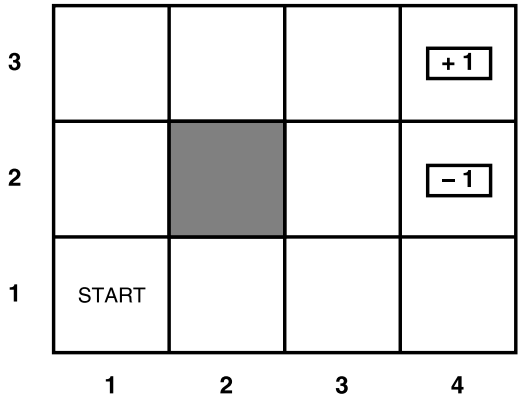
فرض می‌شود که گذرها **مارکوف** باشند، یعنی:
احتمال رسیدن از حالت فعلی به حالت دیگر، فقط به حالت فعلی وابسته است و به تاریخچه‌ی حالت‌های اخیر وابسته نیست.

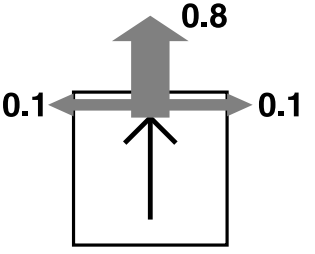
مسائل تصمیم‌گیری مارکوف

مثال

MARKOV DECISION PROBLEMS (MDPS)

حرکت از START و رسیدن به هدف +1





توزیع احتمال نتیجه‌ی کنش حرکت مستقیم

حالت‌ها و کنش‌ها States $s \in S$, actions $a \in A$ = {Right, Left, Down, Up}

مدل Model $T(s,a,s') \equiv P(s'|s,a)$ = probability that a in s leads to s'

تابع پاداش Reward function $R(s)$ (or $R(s,a)$, $R(s,a,s')$)

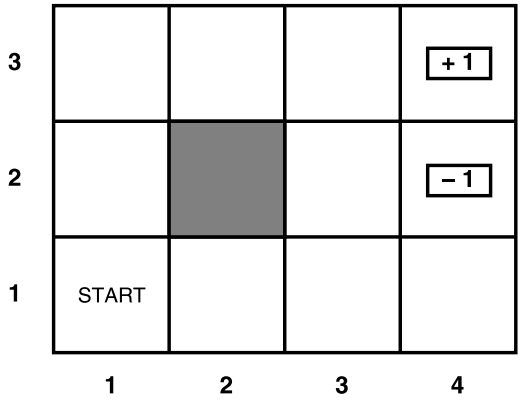
$$= \begin{cases} -0.04 & \text{(جریمه/پنالتی کوچک) برای حالت‌های ناپایانی} \\ \pm 1 & \text{برای حالت‌های پایانی} \end{cases}$$

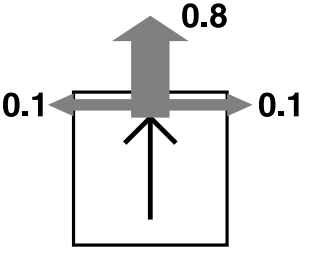
مسائل تصمیم‌گیری مارکوف

مثال

MARKOV DECISION PROBLEMS (MDPS)

حرکت از START و رسیدن به هدف +1





توزیع احتمال نتیجه‌ی کنش حرکت مستقیم

راه‌حل برای محیط قطعی	[Up, Up, Right, Right, Right]
راه‌حل برای محیط اتفاقی	?

وقتی محیط اتفاقی است، اثر کنش‌ها قطعی نخواهد بود.

⇓

یک ترتیب ثابت از کنش‌ها مسئله را حل نمی‌کند؛
(زیرا ممکن است عامل وارد حالتی شود که نتیجه‌ی کنش آن نبوده است)

⇓

در راه‌حل باید: کنش عامل به‌ازای همه‌ی حالت‌هایی که ممکن است به آنها برسد، تعیین شده باشد (سیاست).

مسائل تصمیم‌گیری مارکوف

سیاست

POLICY

سیاست
Policy

کنش پیشنهادی برای هر حالت ممکن s

$$\pi(s)$$

وقتی عامل دارای یک سیاست کامل باشد،
بدون توجه به نتیجه‌ی هر کنش، همیشه می‌داند برای مرحله‌ی بعدی باید چه کاری انجام دهد.

سیاست بهینه
Optimal Policy

بهترین کنش برای هر حالت ممکن s

$$\pi^*(s)$$

سیاست بهینه سیاستی است که به **بالاترین مقدار امید سودمندی** برسد.

مسائل تصمیم‌گیری مارکوف

مسائل جستجو در مقابل مسائل تصمیم‌گیری مارکوف

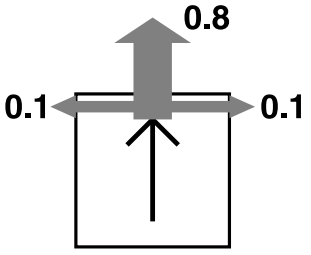
SEARCH PROBLEMS VS. MDPS

مسئله‌ی جستجو *Search Problem*	مسئله‌ی تصمیم‌گیری مارکوف *Markov Decision Problem (MDP)*
هدف: یافتن یک دنباله‌ی بهینه	هدف: یافتن یک سیاست بهینه $\pi(s)$
	یعنی: بهترین کنش برای هر حالت ممکن s (زیرا نمی‌توانیم پیش‌بینی کنیم که هر کنش قطعاً به کدام حالت منجر می‌شود)

مسائل تصمیم‌گیری مارکوف

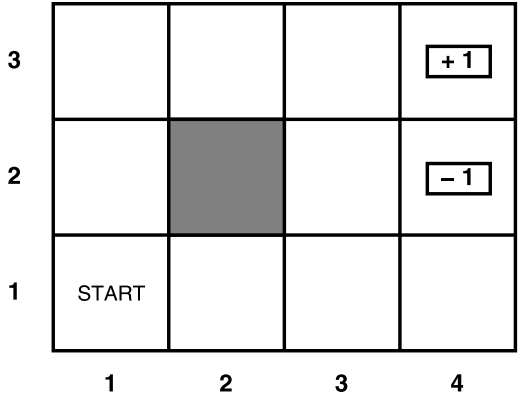
حل کردن مسائل تصمیم‌گیری مارکوف: مثال

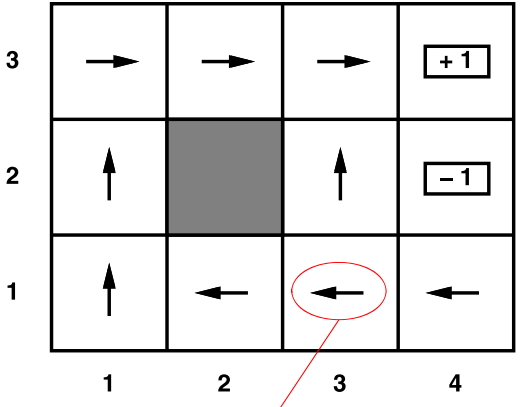
SOLVING MDPS



توزیع احتمال نتیجه‌ی کنش حرکت مستقیم

Optimal policy when state penalty $R(s)$ is –0.04:





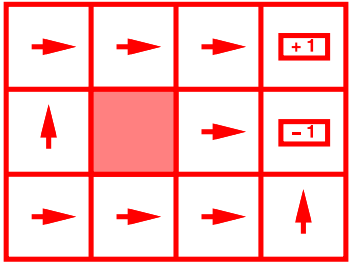
تصمیم محافظه‌کارانه: در مقابل راه میان‌بر و ریسکی ورود به 1- راه طولانی‌تر را انتخاب می‌کند.

مسائل تصمیم‌گیری مارکوف

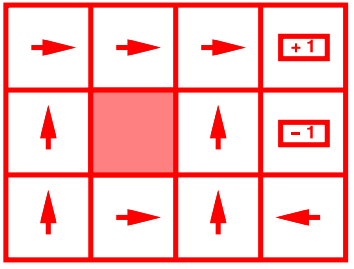
ریسک و پاداش

RISK AND REWARD

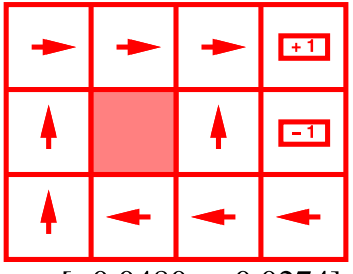
توازن دقیق **ریسک** و **پاداش**، از خصوصیات MDP است.



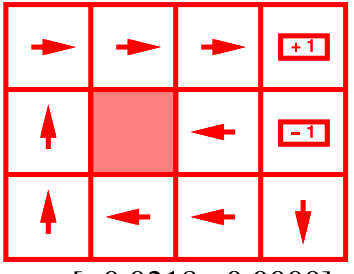
$r = [-\infty : -1.6284]$



$r = [-0.4278 : -0.0850]$



$r = [-0.0480 : -0.0274]$



$r = [-0.0218 : 0.0000]$

سیاست‌های بهینه با بازه‌های مختلف برای پاداش تغییر می‌کند.

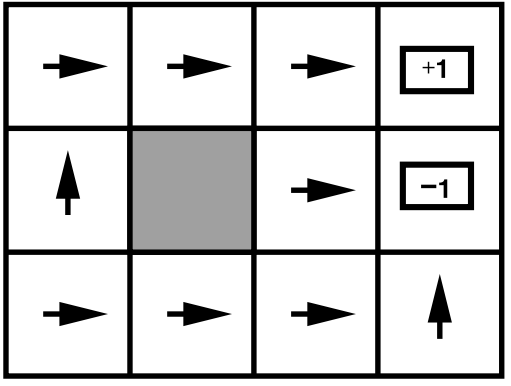
توازن بین تغییرات پاداش و ریسک، به مقدار پاداش حالت‌های غیرپایانی بستگی دارد.

مسائل تصمیم‌گیری مارکوف

توازن ریسک و پاداش: مثال ۱

RISK AND REWARD

سیاست بهینه برای پاداش‌های منفی (جریمه) بزرگ



$$R(s) < -1.6284$$

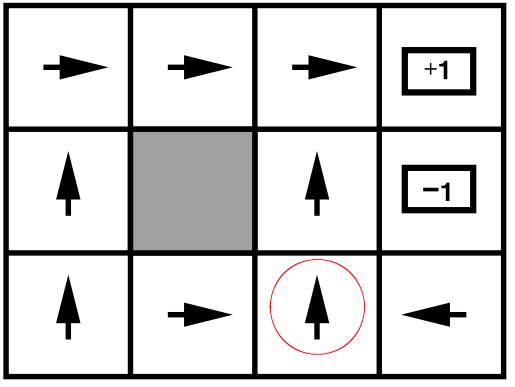
شرایط آن‌قدر سخت می‌شود که عامل برای یافتن نزدیک‌ترین خروجی تلاش می‌کند.
حتی اگر خروجی دارای مقدار 1- باشد.

مسائل تصمیم‌گیری مارکوف

توازن ریسک و پاداش: مثال ۲

RISK AND REWARD

سیاست بهینه برای پاداش‌های منفی (جریمه) متوسط



$$-0.4278 < R(s) < -0.0850$$

زندگی برای عامل ناخوشایند است و عامل کوتاه‌ترین مسیر برای رسیدن به 1+ را انتخاب می‌کند؛
و ریسک (خطر) افتادن ناگهانی در 1- را نیز می‌پذیرد.

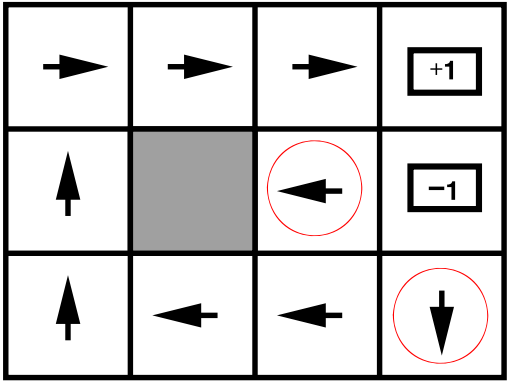
(به‌طور خاص برای زمانی که عامل راه میان‌بری را از (3,1) انتخاب کند.)

مسائل تصمیم‌گیری مارکوف

توازن ریسک و پاداش: مثال ۳

RISK AND REWARD

سیاست بهینه برای پاداش‌های منفی (جریمه) کوچک



$$-0.0221 < R(s) < 0$$

زندگی عامل با کمی افسردگی همراه است:
در این حالت سیاست بهینه هیچ ریسکی را نمی‌پذیرد.

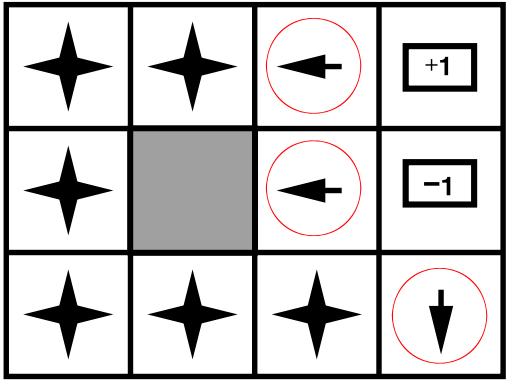
(در (4,1) و (3,2) عامل سعی می‌کند با دور شدن از حالت 1- به‌طور ناگهانی در حالت 1- گرفتار نشود؛ حتی اگر به دیوار بخورد.)

مسائل تصمیم‌گیری مارکوف

توازن ریسک و پاداش: مثال ۴

RISK AND REWARD

سیاست بهینه برای پاداش‌های مثبت



$$R(s) > 0$$

زندگی عامل لذت‌بخش است:
عامل از هر دو حالت خروج امتناع می‌کند.

(تازمانی که کنش‌ها در (4,1) و (3,2) و (3,3) مطابق شکل فوق باشد، هر سیاستی بهینه است
و عامل به‌خاطر عدم ورود به حالت‌های پایانی، در کل پاداشی نامتناهی به‌دست می‌آورد.)

سودمندی دنباله‌های حالت

UTILITY OF STATE SEQUENCES

برای تعیین ترجیح‌ها بین دنباله‌های حالت‌ها

ترجیح‌های ایستان را بر روی دنباله‌های پاداش در نظر می‌گیریم:

stationary preferences

$$[r, r_0, r_1, r_2, \ldots] \succ [r, r'_0, r'_1, r'_2, \ldots] \Leftrightarrow [r_0, r_1, r_2, \ldots] \succ [r'_0, r'_1, r'_2, \ldots]$$

قضیه:
با فرض ایستان بودن
برای انتساب سودمندی به دنباله‌های حالت
(ترکیب پاداش‌ها در طول زمان) فقط دو راه وجود دارد:

تابع سودمندی جمعی

1) *Additive* utility function:

$$U([s_0, s_1, s_2, \ldots]) = R(s_0) + R(s_1) + R(s_2) + \cdots$$

تابع سودمندی تخفیف‌یافته

2) *Discounted* utility function:

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

where γ is the discount factor فاکتور تخفیف

سودمندی حالت‌ها

UTILITY OF STATES

سودمندی یک حالت *Utility of a state*	ارزش یک حالت *Value of a state*	امید مجموع (تخفیف‌یافته) پاداش‌ها (تا رسیدن به پایان) با فرض کنش‌های بهینه	$U(s)$

با داشتن سودمندی حالت‌ها، انتخاب بهترین کنش صرفاً یک MEU است:
امید سودمندی مابعدهای بی‌واسطه را ماکزیمم کنید.

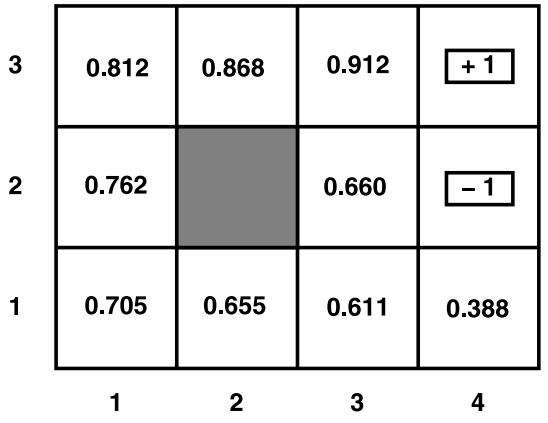
$$\pi^*(s) = \underset{a \in A(s)}{\operatorname{argmax}} \sum_{s'} P(s' \mid s, a) U(s')$$

سودمندی حالت‌ها

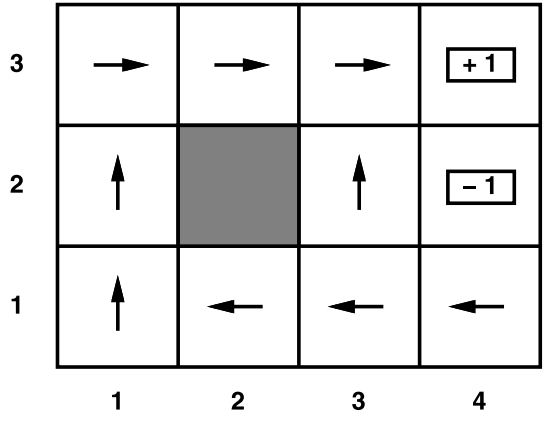
مثال

UTILITY OF STATES

Utility of States:



Optimal policy when state penalty $R(s)$ is –0.04:



سودمندی حالت‌ها

برخورد با مشکل طول عمر نامتناهی

UTILITY OF STATES

Problem: infinite lifetimes $\Rightarrow$ additive utilities are infinite

1) Finite horizon: termination at a *fixed time* T
   $\Rightarrow$ nonstationary policy: $\pi(s)$ depends on time left

2) Absorbing state(s): w/ prob. 1, agent eventually "dies" for any π
   $\Rightarrow$ expected utility of every state is finite

3) Discounting: assuming $\gamma < 1$, $R(s) \leq R_{\max}$,

$$U([s_0, \ldots s_\infty]) = \Sigma_{t=0}^{\infty} \gamma^t R(s_t) \leq R_{\max}/(1-\gamma)$$

Smaller $\gamma \Rightarrow$ shorter horizon

4) Maximize system gain = average reward per time step
Theorem: optimal policy has constant gain after initial transient
E.g., taxi driver's daily scheme cruising for passengers

هوش مصنوعی

اتخاذ تصمیم‌های پیچیده

۲

تکرار ارزش

برنامه‌ریزی پویا

معادله‌ی بلمن

DYNAMIC PROGRAMMING: THE BELLMAN EQUATION

Definition of utility of states leads to a simple relationship among utilities of neighboring states:

expected sum of rewards
= current reward
+ $\gamma \times$ expected sum of rewards after taking best action

Bellman equation (1957):

$$U(s) = R(s) + \gamma \max_a \sum_{s'} U(s') T(s, a, s')$$

$$\begin{aligned} U(1,1) = -0.04 \\ + \gamma \max\{ & 0.8U(1,2) + 0.1U(2,1) + 0.1U(1,1), && \textit{up} \\ & 0.9U(1,1) + 0.1U(1,2) && \textit{left} \\ & 0.9U(1,1) + 0.1U(2,1) && \textit{down} \\ & 0.8U(2,1) + 0.1U(1,2) + 0.1U(1,1)\} && \textit{right} \end{aligned}$$

One equation per state = n **nonlinear** equations in n unknowns

تکرار ارزش

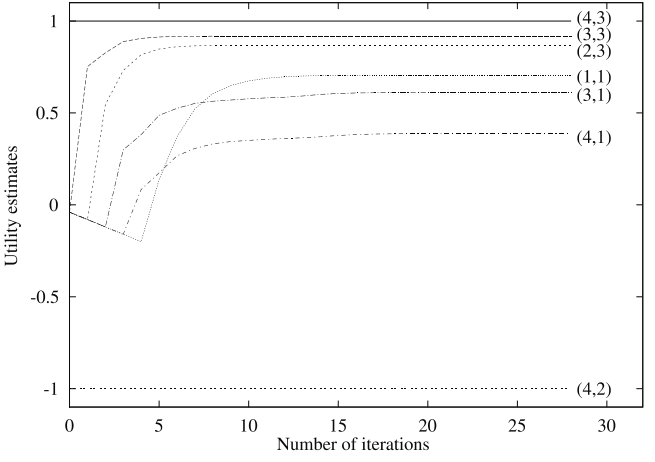
الگوریتم

VALUE ITERATION ALGORITHM

Idea: Start with arbitrary utility values
Update to make them locally consistent with Bellman eqn.
Everywhere locally consistent $\Rightarrow$ global optimality

Repeat for every s simultaneously until "no change"

$$U(s) \leftarrow R(s) + \gamma \max_a \sum_{s'} U(s') T(s, a, s') \qquad \text{for all } s$$



تکرار ارزش

قضیه‌ی همگرایی

VALUE ITERATION ALGORITHM: CONVERGENCE

Define the max-norm $||U|| = \max_s |U(s)|$,
so $||U - V||$ = maximum difference between U and V

Let U^t and U^{t+1} be successive approximations to the true utility U

Theorem: For any two approximations U^t and V^t

$$||U^{t+1} - V^{t+1}|| \leq \gamma\, ||U^t - V^t||$$

I.e., any distinct approximations must get closer to each other
so, in particular, any approximation must get closer to the true U
and value iteration converges to a unique, stable, optimal solution

Theorem: if $||U^{t+1} - U^t|| < \epsilon$, then $||U^{t+1} - U|| < 2\epsilon\gamma/(1-\gamma)$
I.e., once the change in U^t becomes small, we are almost done.

MEU policy using U^t may be optimal long before convergence of values

هوش مصنوعی

اتخاذ تصمیم‌های پیچیده

۳

تکرار سیاست

تکرار سیاست

الگوریتم

POLICY ITERATION ALGORITHM

Howard, 1960: search for optimal policy and utility values simultaneously

Algorithm:

$\pi \leftarrow$ an arbitrary initial policy
repeat until no change in π
  compute utilities given π
  update π as if utilities were correct (i.e., local MEU)

To compute utilities given a fixed π (value determination):

$$U(s) = R(s) + \gamma \sum_{s'} U(s') T(s, \pi(s), s') \qquad \text{for all } s$$

i.e., n simultaneous **linear** equations in n unknowns, solve in $O(n^3)$

تکرار سیاست

شکل اصلاح شده

MODIFIED POLICY ITERATION ALGORITHM

Policy iteration often converges in few iterations, but each is expensive

Idea: use a few steps of value iteration (but with π fixed)
starting from the value function produced the last time
to produce an approximate value determination step.

Often converges much faster than pure VI or PI

Leads to much more general algorithms where Bellman value updates and Howard policy updates can be performed locally in any order

Reinforcement learning algorithms operate by performing such updates based on the observed transitions made in an initially unknown environment

هوش مصنوعی

اتخاذ تصمیم‌های پیچیده

۴

MDPهای مشاهده‌پذیر جزئی (POMDPs)

MDPهای مشاهده‌پذیر جزئی (POMDPs)

POMDP has an observation model $O(s, e)$ defining the probability that the agent obtains evidence e when in state s

Agent does not know which state it is in

$\Rightarrow$ makes no sense to talk about policy $\pi(s)$!!

Theorem (Astrom, 1965): the optimal policy in a POMDP is a function $\pi(b)$ where b is the belief state (probability distribution over states)

Can convert a POMDP into an MDP in belief-state space, where $T(b, a, b')$ is the probability that the new belief state is b' given that the current belief state is b and the agent does a.
I.e., essentially a filtering update step

MDPهای مشاهده‌پذیر جزئی (POMDPs)

ویژگی‌ها

Solutions automatically include information-gathering behavior

If there are n states, b is an n-dimensional real-valued vector
$\Rightarrow$ solving POMDPs is very (actually, PSPACE-) hard!

The real world is a POMDP (with initially unknown T and O)

هوش مصنوعی

اتخاذ تصمیم‌های پیچیده

۵

تصمیم‌هایی با عامل‌های چندگانه: نظریه‌ی بازی

تصمیم‌هایی با عامل‌های چندگانه

نظریه‌ی بازی

DECISIONS WITH MULTIPLE AGENTS: GAME THEORY

هوش مصنوعی

اتخاذ تصمیم‌های پیچیده

۶

طراحی مکانیزم

طراحی مکانیزم

MECHANISM DESIGN

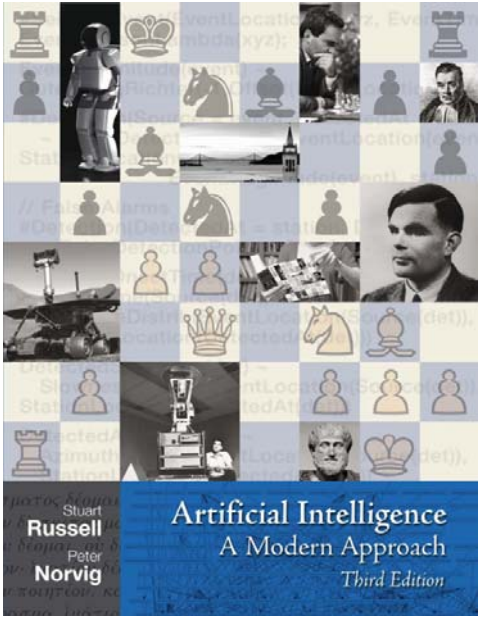
هوش مصنوعی

اتخاذ تصمیم‌های پیچیده

۷

منابع، مطالعه، تکلیف

منبع اصلی



Stuart Russell and Peter Norvig,
Artificial Intelligence: A Modern Approach,
3rd Edition, Prentice Hall, 2010.

Chapter 17

17 MAKING COMPLEX DECISIONS

In which we examine methods for deciding what to do today, given that we may decide again tomorrow.

SEQUENTIAL DECISION PROBLEM

In this chapter, we address the computational issues involved in making decisions in a stochastic environment. Whereas Chapter 16 was concerned with one-shot or episodic decision problems, in which the utility of each action's outcome was well known, we are concerned here with **sequential decision problems**, in which the agent's utility depends on a sequence of decisions. Sequential decision problems incorporate utilities, uncertainty, and sensing, and include search and planning problems as special cases. Section 17.1 explains how sequential decision problems are defined, and Sections 17.2 and 17.3 explain how they can be solved to produce optimal behavior that balances the risks and rewards of acting in an uncertain environment. Section 17.4 extends these ideas to the case of partially observable environments, and Section 17.4.3 develops a complete design for decision-theoretic agents in partially observable environments, combining dynamic Bayesian networks from Chapter 15 with decision networks from Chapter 16.

The second part of the chapter covers environments with multiple agents. In such environments, the notion of optimal behavior is complicated by the interactions among the agents. Section 17.5 introduces the main ideas of **game theory**, including the idea that rational agents might need to behave randomly. Section 17.6 looks at how multiagent systems can be designed so that multiple agents can achieve a common goal.

17.1 SEQUENTIAL DECISION PROBLEMS

Suppose that an agent is situated in the 4×3 environment shown in Figure 17.1(a). Beginning in the start state, it must choose an action at each time step. The interaction with the environment terminates when the agent reaches one of the goal states, marked +1 or –1. Just as for search problems, the actions available to the agent in each state are given by ACTIONS(s), sometimes abbreviated to $A(s)$; in the 4×3 environment, the actions in every state are *Up*, *Down*, *Left*, and *Right*. We assume for now that the environment is **fully observable**, so that the agent always knows where it is.

645