

EECS 3401 — AI and Logic Prog. — Lecture 20

Adapted from official slides for 3-ed ed. Russell & Norvig (Ch.17)

Vitaliy Batusov
vbatusov@cse.yorku.ca

York University

November 30, 2020

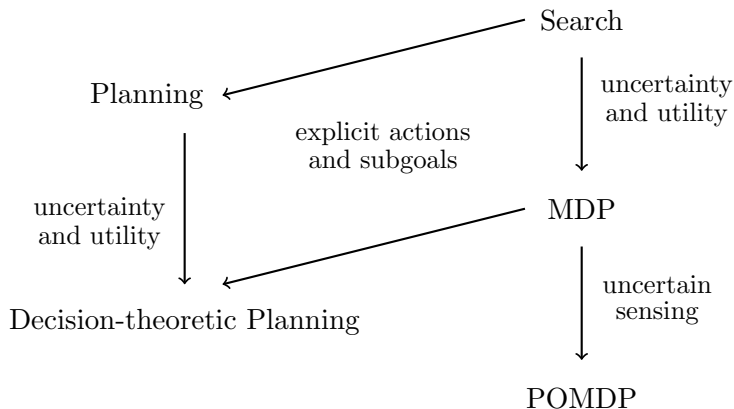
- Today: **Sequential Decision-Making**
- Required reading: Russell & Norvig Ch. 17.1–17.3

- Covered to date: Search; Belief Networks
- Today: Markov Decision Processes

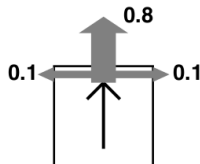
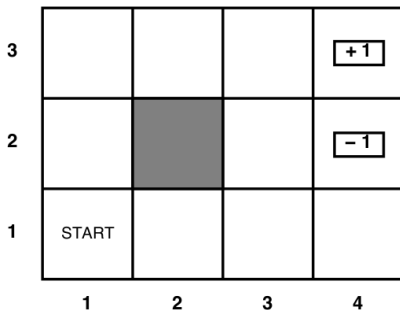
Basic Idea behind MDP

- Goal: decision making under uncertainty and a notion of **utility**
- Random variables to describe the world (like in Belief Networks)
- But now the world is again *dynamical*
- Transition model: specifies the probability distribution over the latest state variables, given the previous values
- **Markov assumption:** *current state depends on only a finite fixed number of previous states*
- First-order Markov process: current state depends only on last state

Sequential Decision Problems



Example MDP

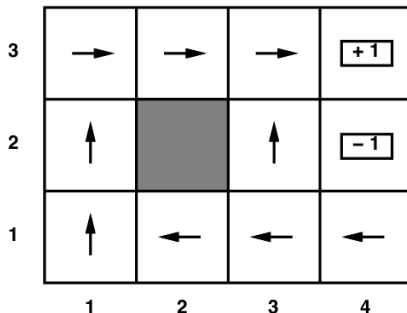


- States: $s \in S$, actions: $a \in A$
- Transition **model**: $T(s, a, s') \triangleq P(s' \mid s, a)$ — probability that a in s leads to s'
- **Reward function**:

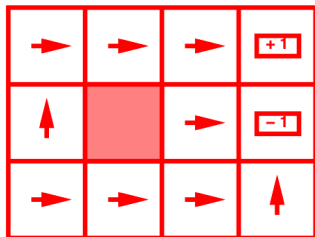
$$R(s) = \begin{cases} -0.04 & \text{(small penalty for non-terminal states)} \\ \pm 1 & \text{for terminal states} \end{cases}$$

Solving MDPs

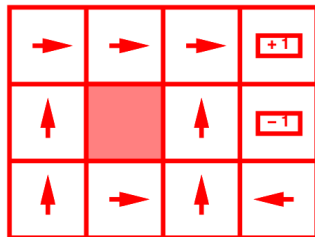
- In search problems, the aim is to find an optimal *sequence of actions*
- In MDPs, the aim is to find an *optimal policy* $\pi(s)$
- I.e., best action **for every possible state** s
- The optimal policy maximizes the *expected sum of rewards*
- Suppose $R(s) = -0.04$. Optimal policy:



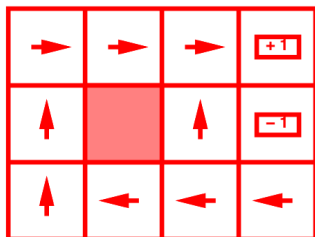
Risk and Reward



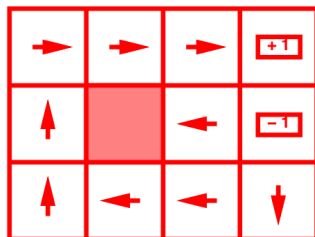
$$r = [-\infty : -1.6284]$$



$$r = [-0.4278 : -0.0850]$$



$$r = [-0.0480 : -0.0274]$$



$$r = [-0.0218 : 0.0000]$$

Utility of State Sequences

Need to understand preferences between *sequences* of states

Typically consider stationary preferences on reward sequences:

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

Theorem: there are only two ways to combine rewards over time.

1) *Additive* utility function:

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

2) *Discounted* utility function:

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

where γ is the discount factor

Utility of States

Utility of a *state* (a.k.a. its *value*) is defined to be

$$U(s) = \frac{\text{expected (discounted) sum of rewards (until termination)}}{\text{assuming optimal actions}}$$

Given the utilities of the states, choosing the best action is just MEU:
maximize the expected utility of the immediate successors

3	0.812	0.868	0.912	<div>+ 1</div>
2	0.762		0.660	<div>- 1</div>
1	0.705	0.655	0.611	0.388
	1	2	3	4

3	→	→	→	<div>+ 1</div>
2	↑		↑	<div>- 1</div>
1	↑	←	←	←
	1	2	3	4

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 $\gamma < 1$, $R(s) \leq R_{\max}$,

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

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

Dynamic Programming

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):

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

$$U(1, 1) = -0.04$$

$$+ \gamma \max \{ 0.8U(1, 2) + 0.1U(2, 1) + 0.1U(1, 1), \\ 0.9U(1, 1) + 0.1U(1, 2) \\ 0.9U(1, 1) + 0.1U(2, 1) \\ 0.8U(2, 1) + 0.1U(1, 2) + 0.1U(1, 1) \}$$

up
left
down
right

One equation per state = n nonlinear equations in n unknowns

Value Iteration Algorithm

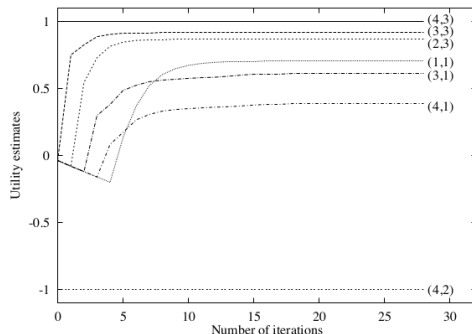
Idea: Start with arbitrary utility values

Update to make them locally consistent with Bellman eqn.

Everywhere locally consistent \Rightarrow global optimality

Repeat for every s simultaneously until “no change”

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



Convergence

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

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

Policy Iteration

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):

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

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

Modified Policy Iteration

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

Partial Observability

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

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)