17 Optimal behavior: Decision theory

How to act optimally under uncertainty?

Given

- set of states: S
- set of actions: A
- state dynamics: executing a in s leads to s'

Goal

- Maximize reward or achieve a goal
- Reward function $R(s)$

  Generalizes the concept of goal states. Goal states can be expressed using a reward function

$$R(s) = \begin{cases} 1 & \text{if } s \text{ is a goal} \\ 0 & \text{otherwise} \end{cases}$$

Task

- Given state dynamics and reward function
- Need to determine best actions to take

Why is this hard?

- Uncertainty in state dynamics
  - world could be random
  - world could be adversarial
- May have to tradeoff short term versus long term reward

17.1 Easiest case: Planning

Actions are deterministic: $s' = a(s)$

Given an initial state and goal condition:

1. can precompute an optimal action sequence
2. execute sequence blindly

17.2 Slightly harder case: Conditional planning

Actions are non-deterministic

$$S'(a, s) \quad = \quad \text{set of possible next states when } a \text{ executed in } s$$

Have to plan for multiple outcomes (conditional/contingency planning)

Have to monitor plan and choose future actions based on future states (execution monitoring)

17.3 General case

Have to plan an action for every possible state
A *total policy* (or *controller*) is given by $\pi : S \to A$

Optimal behavior: precompute optimal policy

Two cases:

Decision theory: State dynamics are *random*: Living in an oblivious stochastic environment

Game theory: State dynamics are adversarial: The world (or your opponents) are out to get you

17.4 Optimal decision theory

Given

- state space S
- actions A
- reward function $R : S \to \Re$
- state transition model $\mathrm{P}(s'|s,a)$

Assume for now that we can *identify* the current state
In this case, the optimal policy is a function of state: $\pi^* : S \to A$

Simplest case: optimize immediate expected reward

Only look one step ahead

Given current state s

For each action a, the expected total reward in the next state is

$$R(s) + \sum_{s'} P(s'|s,a)\ R(s')$$

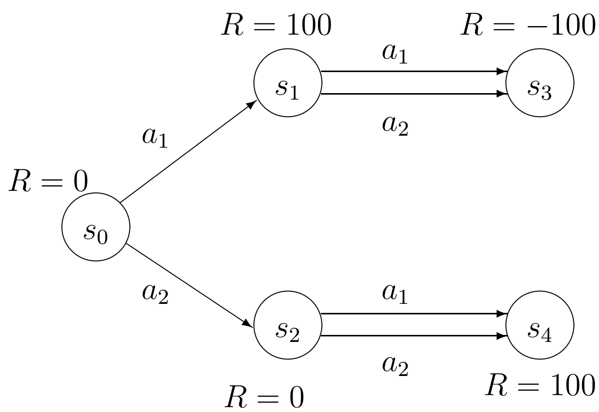
Optimal action

$$\begin{aligned} a^* &= \arg\max_a \quad R(s) + \sum_{s'} \mathrm{P}(s'|s,a)\ R(s') \\ &= \arg\max_a \quad \sum_{s'} P(s'|s,a)\ R(s') \end{aligned}$$

17.5 Harder case: Sequential decision problem

Have to choose several actions in sequence, depending on resulting states. Goal is to maximize the total reward accumulated.

However, there is a *trade-off* between short term and long term reward. That is, simply taking the action that maximizes *immediate* reward does not always lead to the best policy



Here the optimal policy makes the decision $\pi^*(s_0) = a_2$, even though the optimal action for one step is a_1

17.6 Computing optimal policies: Acyclic case

Assume S finite

Assume no action sequence causes loop in state space

In particular, assume

- initial state s^0
- terminal state s^t
- after executing action in s^0 we go to one of the states

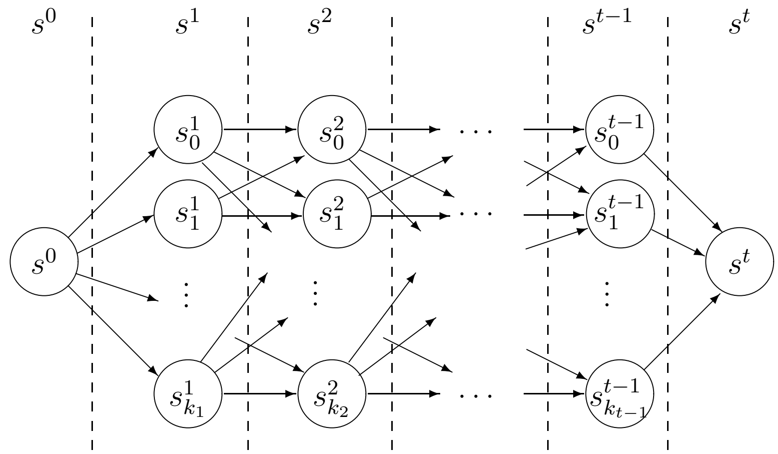
  $$s^1_0, s^1_1, \ldots, s^1_{k_1}$$

  and after executing the second action, we go to one of the states

  $$s^2_0, s^2_1, \ldots, s^2_{k_2}$$

  and so on, until after the tth action we arrive in state s^t

This is represented by



Thus, the state dynamics move forward level by level $\mathrm{P}(s^{j+1}|s^j, a)$

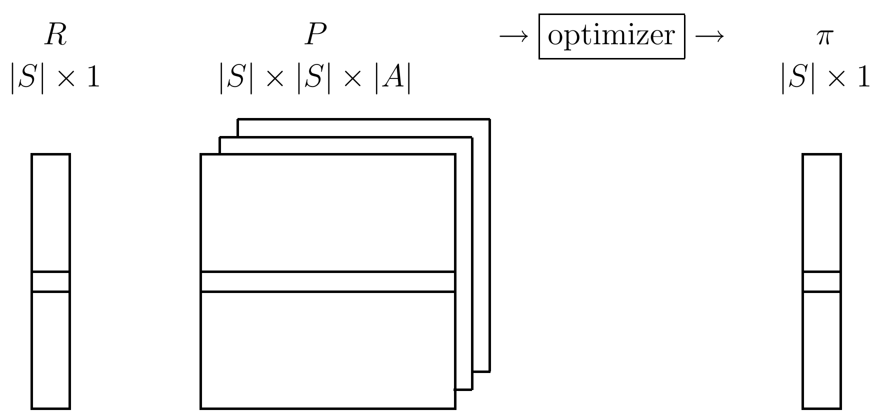
Given

- $R(s)$ — a lookup table of length $|S|$
- $P(s'|s, a)$ — a lookup table (matrix) of size $|S| \times |S|$ for each a

Compute

- $\pi^* : S \rightarrow A$ — a lookup table of size $|S|$ — that maximizes expected future reward from each state

Task



Utility function

$$\begin{aligned} U(s,\pi) &= \text{total expected reward obtained by policy } \pi \text{ starting in state } s \\ &= R(s) + \sum_{s'} U(s',\pi)\ \mathrm{P}(s'|s,\pi(s)) \\ U(s^t,\pi) &= R(s^t) \ = \ 0 \end{aligned}$$

Compute π^* that maximizes $U(s,\pi)$ for all s

Naive algorithm

- Enumerate policies ($|A|^{|S|}$ possible policies)
- Evaluate each one ($O(|A| \times |S|^2)$)
- Pick winner

Too expensive!

Efficient algorithm: Dynamic programming

Solve for $U(s,\pi)$ in last states first, and then recursively back up

$$\begin{aligned} \pi^*(s^i) &= \arg\max_a \quad R(s^i) + \sum_{s^{i+1}} U(s^{i+1},\pi^*)\ \mathrm{P}(s^{i+1}|s^i,a) \\ &= \arg\max_a \sum_{s^{i+1}} U(s^{i+1},\pi^*)\ \mathrm{P}(s^{i+1}|s^i,a) \\ U(s^i,\pi^*) &= R(s^i) + \sum_{s^{i+1}} U(s^{i+1},\pi^*)\ \mathrm{P}(s^{i+1}|s^i,\pi^*(s^i)) \end{aligned}$$

where $U(s^{i+1},\pi^*)$ is already computed

Algorithm 1 Sequential decision problem: acyclic case

$U(s^t, \pi^*) \leftarrow R(s^t)$;
for $j \leftarrow 0$ to k_{t-1} **do**
  $\pi^*(s_j^{t-1}) \leftarrow$ any action, because they all lead to s^t
  $U(s_j^{t-1}, \pi^*) \leftarrow R(s_j^{t-1}) + U(s^t, \pi^*)$
end for
for $i \leftarrow t-2$ down to 1 **do**
  for $j \leftarrow 0$ to k_i **do**
    $\pi^*(s_j^i) \leftarrow \arg\max_a \sum_{k=0}^{k_{i+1}} U(s_k^{i+1}, \pi^*)\ \mathrm{P}(s_k^{i+1}|s_j^i, a)$
    $U(s_j^i, \pi^*) \leftarrow R(s_j^i) + \sum_{k=0}^{k_{i+1}} U(s_k^{i+1}, \pi^*)\ \mathrm{P}(s_k^{i+1}|s_j^i, \pi^*(s_j^i))$
  end for
end for
$\pi^*(s^0) \leftarrow \arg\max_a \sum_{k=0}^{k_1} U(s_k^1, \pi^*)\ \mathrm{P}(s_k^1|s^0, a)$
$U(s^0, \pi^*) \leftarrow R(s^0) + \sum_{k=0}^{k_1} U(s_k^1, \pi^*)\ \mathrm{P}(s_k^1|s^0, \pi^*(s^0))$

Time complexity $\leq$ $|S| \times |S| \times |A| \times$ *levels*

Readings

Russell and Norvig 2nd Ed: Chapter 12, Section 16.1
Dean, Allen, Aloimonos: Section 8.4