The World Reacts
Example 1
Example II
Markov Decision Problems

Markov Decision Processes (MDPs):

• *The* canonical decision making formulation.

• Problem has a set of states.
• Actions cause stochastic *state transitions*.
• Actions have costs/rewards.
Markov Decision Processes

\( S \): set of states

\( A \): set of actions

\( \gamma \): discount factor

\(< S, A, \gamma, R, T >\)

\( R \): reward function

\( R(s, a, s') \) is the reward received taking action \( a \) from state \( s \) and transitioning to state \( s' \).

\( T \): transition function

\( T(s' \mid s, a) \) is the probability of transitioning to state \( s' \) after taking action \( a \) in state \( s \).

(some states are absorbing - execution stops)
The Markov Property

Critical property:
• $s_{t+1}$ depends only on $s_t$ and $a_t$
• $r_t$ depends only on $s_t$ and $a_t$

Current state is a sufficient statistic of agent’s history.

This means that:
• Decision-making depends only on current state
• The agent does not need to remember its history

Compare to “context vector” from contextual bandits.
Example

States: set of grid locations
Actions: up, down, left, right
Transition function: move in direction of action with $p=0.9$
Reward function: -1 for every step, 1000 for finding the goal
Example

grab!

A
B
C

0.8
r=-2

A
B
C

0.2
r=-5

A
B

C
Example:

**States:** \((\theta_1, \dot{\theta}_1, \theta_2, \dot{\theta}_2)\) (real-valued vector)

**Actions:** +1, -1, 0 units of torque added to elbow

**Transition function:** physics!

**Reward function:** -1 for every step
Our goal is to find a policy:

\[ \pi : S \rightarrow A \]

… that maximizes return: expected sum of rewards.
(equiv: min sum of costs)

\[ \sum_{i=1}^{\infty} E[\gamma^i r_i] \]
Policies

A policy:

• An action for every state.

![Diagram showing different actions for each state](image)
Planning

So our goal is to produce optimal policy.

\[ \pi^*(s) = \max_{\pi} R^\pi(s) \]

Assume we know $T, R$, this is known as \textit{planning}.

Define the \textit{value function} to estimate this quantity:

\[ V^\pi(s) = \mathbb{E} \left[ \sum_{i=0}^{\infty} \gamma^i r_i(s_i) \right] \]

How to find $V$?
Monte Carlo Estimation

One approach:

• For each state \( s \)
• Repeat many times:
  • Start at \( s \)
  • Run policy forward until absorbing state (or \( \gamma^t \) small)
  • Write down discount sum of rewards received
  • This is a sample of \( V(s) \)
  • Average these samples

This always works!

But very high variance. Why?
Bellman

Bellman’s equation is a condition that must hold for $V$:

$$V^\pi(s) = r(s, \pi(s)) + \gamma \sum_{s'} T(s'|s, \pi(s)) V^\pi(s')$$

- Value of this state
- Reward
- Expected value of next state
Value Iteration

This gives us an algorithm for \textit{learning the value function for a specific given fixed policy}:

Repeat:
  • Make a copy of the VF.
  • For each state in VF, assign value using BE.
  • Replace old VF.

This is known as \textit{value iteration}.
(In practice, only adjust “reachable” states.)
Policy Iteration

Why do we care so much about VF?

Recall that we seek the policy that maximizes $V_\pi(s), \forall s$.

Therefore we know that, for the optimal policy $\pi^*$:

$$V_{\pi^*}(s) \geq V_\pi(s), \forall \pi, s$$

This means that any change to $\pi$ that increases $V$ anywhere obtains a better policy.
Policy Iteration

\[ \pi(s) = \max_a \left[ R(s, a) + \gamma \sum_{s'} T(s'|s, a) V(s') \right] \]

Adjust policy to be greedy w.r.t VF.

We can alternate value and policy iteration.

Surprising results:

- This converges even if alternate every step.
- Converges to optimal policy.
- Converges in polynomial time.
Elevator Scheduling

Crites and Barto (1985):
System with 4 elevators, 10 floors.
Realistic simulator.
46 dimensional state space.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>AvgWait</th>
<th>SquaredWait</th>
<th>SystemTime</th>
<th>Percent &gt;60 secs</th>
</tr>
</thead>
<tbody>
<tr>
<td>SECTOR</td>
<td>30.3</td>
<td>1643</td>
<td>59.5</td>
<td>13.50</td>
</tr>
<tr>
<td>HUFF</td>
<td>22.8</td>
<td>884</td>
<td>55.3</td>
<td>5.10</td>
</tr>
<tr>
<td>DLB</td>
<td>22.6</td>
<td>880</td>
<td>55.8</td>
<td>5.18</td>
</tr>
<tr>
<td>LQF</td>
<td>23.5</td>
<td>877</td>
<td>53.5</td>
<td>4.92</td>
</tr>
<tr>
<td>BASIC HUFF</td>
<td>23.2</td>
<td>875</td>
<td>54.7</td>
<td>4.94</td>
</tr>
<tr>
<td>FIM</td>
<td>20.8</td>
<td>685</td>
<td>53.4</td>
<td>3.10</td>
</tr>
<tr>
<td>ESA</td>
<td>20.1</td>
<td>667</td>
<td>52.3</td>
<td>3.12</td>
</tr>
<tr>
<td>RLD</td>
<td>18.8</td>
<td>593</td>
<td>45.4</td>
<td>2.40</td>
</tr>
<tr>
<td>RLP</td>
<td>18.6</td>
<td>585</td>
<td>45.7</td>
<td>2.49</td>
</tr>
</tbody>
</table>
“Drivers and Loads” (trucking), CASTLE lab at Princeton

“the model was used by 20 of the largest truckload carriers to dispatch over 66,000 drivers”