Discounted Rewards

An assistant professor gets paid, say, 20K per year. How much, in total, will the A.P. earn in their life?

\[ 20 + 20 + 20 + 20 + 20 + \ldots = \infty \]

What’s wrong with this argument?

Discounted Rewards

"A reward (payment) in the future is not worth quite as much as a reward now."

- Because of chance of obliteration
- Because of inflation

Example:

Being promised $10,000 next year is worth only 90% as much as receiving $10,000 right now.

Assuming payment \( n \) years in future is worth only \((0.9)^n\) of payment now, what is the A.P’s Future Discounted Sum of Rewards?

Discount Factors

People in economics and probabilistic decision-making do this all the time.

The "Discounted sum of future rewards" using discount factor \( \gamma \) is

\[
(reward \ now) + \gamma (reward \ in \ 1 \ time \ step) + \gamma^2 (reward \ in \ 2 \ time \ steps) + \gamma^3 (reward \ in \ 3 \ time \ steps) + \ldots
\]

= (infinite sum)

The Academic Life

Define:

\[ J_A = \text{Expected discounted future rewards starting in state A} \]
\[ J_B = \text{Expected discounted future rewards starting in state B} \]
\[ J_T = \text{Tened Prof} \]
\[ J_D = \text{Dead} \]

How do we compute \( J_A, J_B, J_T, J_D \)?
A Markov System with Rewards...

- Has a set of states \( \{S_1, S_2, \ldots, S_N\} \)
- Has a transition probability matrix
  \[
P = \begin{bmatrix} P_{11} & P_{12} & \cdots & P_{1N} \\ P_{21} & P_{22} & \cdots & P_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ P_{N1} & P_{N2} & \cdots & P_{NN} \end{bmatrix} \]
  \( P_{ij} = \text{Prob(Next = } S_j \mid \text{This = } S_i) \)
- Each state has a reward. \( \{r_1, r_2, \ldots, r_N\} \)
- There’s a discount factor \( \gamma \). \( 0 < \gamma < 1 \)

On Each Time Step:
0. Assume your state is \( S_i \)
1. You get given reward \( r_i \)
2. You randomly move to another state \( \text{P(NextState = } S_j \mid \text{This = } S_i) = P_{ij} \)
3. All future rewards are discounted by \( \gamma \)

Solving a Markov System

Write \( J^*(S_i) = \text{expected discounted sum of future rewards starting in state } S_i \)

\[
J^*(S_i) = r_i + \gamma \sum_{j=1}^{N} P_{ij} J^*(S_j) + \cdots + \gamma^{N-1} P_{iN} J^*(S_N)
\]

Using vector notation write
\[
J^* = \begin{bmatrix} J^*(S_1) \\ J^*(S_2) \\ \vdots \\ J^*(S_N) \end{bmatrix}, \quad R = \begin{bmatrix} r_1 \\ r_2 \\ \vdots \\ r_N \end{bmatrix}, \quad P = \begin{bmatrix} P_{11} & P_{12} & \cdots & P_{1N} \\ P_{21} & P_{22} & \cdots & P_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ P_{N1} & P_{N2} & \cdots & P_{NN} \end{bmatrix}
\]

Question: can you invent a closed form expression for \( J^* \) in terms of \( R \) and \( P \)?

Solving a Markov System with Matrix Inversion

- Upside: You get an exact answer
- Downside: If you have 100,000 states you’re solving a 100,000 by 100,000 system of equations.

Value Iteration: another way to solve a Markov System

Define
- \( J_1(S_i) = \) Expected discounted sum of rewards over the next 1 time step.
- \( J_2(S_i) = \) Expected discounted sum rewards during next 2 steps
- \( J_3(S_i) = \) Expected discounted sum rewards during next 3 steps
- \( J_k(S_i) = \) Expected discounted sum rewards during next \( k \) steps

\[
\begin{align*}
J_1(S_i) &= \text{(what?)} \\
J_2(S_i) &= \text{(what?)} \\
J_3(S_i) &= \text{(what?)} \\
J_k(S_i) &= \text{(what?)}
\end{align*}
\]

Value Iteration: another way to solve a Markov System

Define
- \( J_1(S_i) = \) Expected discounted sum of rewards over the next 1 time step.
- \( J_2(S_i) = \) Expected discounted sum rewards during next 2 steps
- \( J_3(S_i) = \) Expected discounted sum rewards during next 3 steps
- \( J_k(S_i) = \) Expected discounted sum rewards during next \( k \) steps

\[
\begin{align*}
J_1(S_i) &= r_i \\
J_2(S_i) &= r_i + \sum_{j=1}^{N} P_{ij} J_1(S_j) \\
J_3(S_i) &= r_i + \sum_{j=1}^{N} P_{ij} J_2(S_j) \\
J_k(S_i) &= r_i + \sum_{j=1}^{N} P_{ij} J_{k-1}(S_j)
\end{align*}
\]

\( N = \text{Number of states} \)
Let's do Value Iteration

<table>
<thead>
<tr>
<th>k</th>
<th>Jk(SUN)</th>
<th>Jk(WIND)</th>
<th>Jk(HAIL)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Value Iteration for solving Markov Systems

• Compute J1(Si) for each j
• Compute J2(Si) for each j
• Compute Jk(Si) for each j
As k→∞ Jk(Si)→J*(Si). Why?

When to stop? When
\[ \max_j |J^{k+1}(S_j) - J^k(S_j)| < \xi \]

This is faster than matrix inversion (N^3 style) if the transition matrix is sparse

A Markov Decision Process

You run a startup company. In every state you must choose between Saving money or Advertising.

Markov Decision Processes

An MDP has...
• A set of states \( \{s_1, \ldots, s_N\} \)
• A set of actions \( \{a_1, \ldots, a_M\} \)
• A set of rewards \( \{r_1, \ldots, r_N\} \) (one for each state)
• A transition probability function
\[ P_i^k = \text{Prob(Next = j | This = i and I use action } k) \]

On each step:
0. Call current state \( S \)
1. Receive reward \( r_i \)
2. Choose action \( a_k \)
3. If you choose action \( a_k \) you'll move to state \( S_j \) with probability \( P_i^k \)
4. All future rewards are discounted by \( \gamma \)

A Policy

A policy is a mapping from states to actions.

Examples

<table>
<thead>
<tr>
<th>STATE → ACTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>PU</td>
</tr>
<tr>
<td>PF</td>
</tr>
<tr>
<td>RU</td>
</tr>
<tr>
<td>RF</td>
</tr>
</tbody>
</table>

Policy Number 1

• How many possible policies in our example?
• Which of the above two policies is best?
• How do you compute the optimal policy?

Interesting Fact

For every M.D.P. there exists an optimal policy.

It's a policy such that for every possible start state there is no better option than to follow the policy.

(Not proved in this lecture)
Computing the Optimal Policy

Idea One:
Run through all possible policies.
Select the best.

What’s the problem??

Optimal Value Function

Define $J^*(S_i) = \text{Expected Discounted Future Rewards, starting from state } S_i$ assuming we use the optimal policy

What is $J^*(S_1)$?
What is $J^*(S_2)$?
What is $J^*(S_3)$?

(assume $\gamma = 0.9$)

Let’s compute $J^k(S_i)$ for our example

<table>
<thead>
<tr>
<th>k</th>
<th>$J^k(\text{PU})$</th>
<th>$J^k(\text{PF})$</th>
<th>$J^k(\text{RU})$</th>
<th>$J^k(\text{RF})$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bellman’s Equation

$J^{n+1}(S_i) = \max_k \left[ r_i + \gamma \sum_{j=1}^N p_{ij} J^*(S_j) \right]$

Finding the Optimal Policy

1. Compute $J^*(S_i)$ for all $i$ using Value Iteration (a.k.a. Dynamic Programming)
2. Define the best action in state $S_i$ as

$$\arg \max_k \left[ r_i + \gamma \sum_{j=1}^N p_{ij} J^*(S_j) \right]$$

(Why?)
Applications of MDPs
This extends the search algorithms of your first lectures to the case of probabilistic next states. Many important problems are MDPs….

- Robot path planning
- Travel route planning
- Elevator scheduling
- Bank customer retention
- Autonomous aircraft navigation
- Manufacturing processes
- Network switching & routing

Asynchronous D.P.
Value Iteration:
"Backup S_1", "Backup S_2", ..., "Backup S_n", then "Backup S_1", "Backup S_2", ... repeat:

- There's no reason that you need to do the backups in order!
- Random Order...still works. Easy to parallelize (Dyna, Sutton 91)

On-Policy Order
Simulate the states that the system actually visits.

Efficient Order
e.g. Prioritized Sweeping [Moore 93]
Q-Dyna [Peng & Williams 93]

Policy Iteration
Write \( \pi(S_i) \) = action selected in the \( i \)th state. Then \( \pi \) is a policy.
Write \( \pi^t = f \)th policy on \( f \)th iteration
Algorithm:

- Any randomly chosen policy
- \( \forall i \) compute \( J_i(S_i) \) = Long term reward starting at \( S \), using \( \pi \)
- \( \pi_1(S_i) = \arg \max_a \left[ r_i + \gamma \sum_j P_{ij} J_j(S_j) \right] \)
- \( J_i = \ldots \)
- \( \pi_j(S_i) = \ldots \)
- Keep computing \( \pi^1, \pi^2, \pi^3 \ldots \) until \( \pi^k = \pi^{k+1} \). You now have an optimal policy.

Policy Iteration & Value Iteration:
Which is best ???
It depends.
Lots of actions? Choose Policy Iter
Already got a fair policy? Policy Iter
Few actions, acyclic? Value Iter

Best of Both Worlds:
Modified Policy Iteration [Puterman]
...a simple mix of value iteration and policy iteration

3rd Approach
Linear Programming

Time to Moan
What’s the biggest problem(s) with what we’ve seen so far?
Function approximation for value functions

- Polynomials               [Samuel, Boyan, Much O.R. Literature]
- Neural Nets               [Barto & Sutton, Tesauro, Crites, Singh, Tsitsiklis]
- Backgammon, Pole Balancing, Elevators, Tetris, Cell phones
- Splines                  Economists, Controls

Downside: All convergence guarantees disappear.

Memory-based Value Functions

J("state") = J(most similar state in memory to "state")
or
Average J(20 most similar states)
or
Weighted Average J(20 most similar states)
[Jeff Peng, Atkeson & Schaal, Geoff Gordon, proved stuff Scheider, Boyan & Moore 98]

Hierarchical Methods

Continuous State Space: "Split a state when statistically significant that a split would improve performance"

Discrete Space:
- Chapman & Kaelbling 92
- McCallum 95 (includes hidden state)

A kind of Decision Tree Value Function

Continuous Space
- e.g. Simmons et al 83, Chapman & Kaelbling 92, Mag Ring 94, Munos 96

Multiresolution

A hierarchy with high level "managers" abstracting low level "servants"


What You Should Know

- Definition of a Markov System with Discounted rewards
- How to solve it with Matrix Inversion
- How (and why) to solve it with Value Iteration
- Definition of an MDP, and value iteration to solve an MDP
- Policy iteration
- Great respect for the way this formalism generalizes the deterministic searching of the start of the class
- But awareness of what has been sacrificed.