Markov Decision Processes (MDPs)

Ron Parr
CompSci 570
Department of Computer Science
Duke University

With thanks to Kris Hauser for some slides

The Winding Path to RL

- Decision Theory
- Markov Decision Processes
- Reinforcement Learning

- Descriptive theory of optimal behavior
- Mathematical/Algorithmic realization of Decision Theory
- Application of learning techniques to challenges of MDPs with numerous or unknown parameters
Swept under the rug today

- Utility of money (assumed 1:1)
- How to determine costs/utilities
- How to determine probabilities

Playing a Game Show

- Assume series of questions
  - Increasing difficulty
  - Increasing payoff
- Choice:
  - Accept accumulated earnings and quit
  - Continue and risk losing everything
- “Who wants to be a millionaire?”
State Representation

Dollar amounts indicate the payoff for getting the question right

Probabilistic Transitions on Attempt to Answer

Downward green arrows indicate the choice to exit the game

N.B.: These exit transitions should actually correspond to states

Green indicates profit at exit from game

Making Optimal Decisions

- Work backwards from future to present

- Consider $50,000 question
  - Suppose P(correct) = 1/10
  - V(stop) = $11,100
  - V(continue) = 0.9*0 + 0.1*$61.1K = $6.11K

- Optimal decision stops
Working Backwards

\[ V = 3,749 \quad V = 4,166 \quad V = 5,555 \quad V = 11,100 \]

Dealing with Loops

Suppose you can pay $1000 (from any losing state) to play again
From Policies to Linear Systems

- Suppose we always pay until we win.
- What is value of following this policy?

\[
V(s_0) = 0.10(-1000 + V(s_0)) + 0.90V(s_1)
\]
\[
V(s_1) = 0.25(-1000 + V(s_0)) + 0.75V(s_2)
\]
\[
V(s_2) = 0.50(-1000 + V(s_0)) + 0.50V(s_3)
\]
\[
V(s_3) = 0.90(-1000 + V(s_0)) + 0.10(61100)
\]

And the solution is...

- \(V = 32.47K\)
- \(V = 32.58K\)
- \(V = 32.95K\)
- \(V = 34.43K\)

Is this optimal?
How do we find the optimal policy?
The MDP Framework

- State space: $S$
- Action space: $A$
- Transition function: $P$
- Reward function: $R(s,a,s')$ or $R(s,a)$ or $R(s)$
- Discount factor: $\gamma$
- Policy: $\pi(s) \rightarrow a$

Objective: Maximize expected, discounted return (decision theoretic optimal behavior)

Applications of MDPs

- AI/Computer Science
  - Robotic control (Koenig & Simmons, Thrun et al., Kaelbling et al.)
  - Air Campaign Planning (Meuleau et al.)
  - Elevator Control (Barto & Crites)
  - Computation Scheduling (Zilberstein et al.)
  - Control and Automation (Moore et al.)
  - Spoken dialogue management (Singh et al.)
  - Cellular channel allocation (Singh & Bertsekas)
Applications of MDPs

- Economics/Operations Research
  - Fleet maintenance (Howard, Rust)
  - Road maintenance (Golabi et al.)
  - Packet Retransmission (Feinberg et al.)
  - Nuclear plant management (Rothwell & Rust)
  - Debt collection strategies (Abe et al.)
  - Data center management (DeepMind)

Applications of MDPs

- EE/Control
  - Missile defense (Bertsekas et al.)
  - Inventory management (Van Roy et al.)
  - Football play selection (Patek & Bertsekas)

- Agriculture
  - Herd management (Kristensen, Toft)

- Other
  - Sports strategies
  - Video games
The Markov Assumption

- Let $S_t$ be a random variable for the state at time $t$

- $P(S_t|A_{t-1}S_{t-1},...,A_0S_0) = P(S_t|A_{t-1}S_{t-1})$

- Markov is special kind of conditional independence

- Future is independent of past given current state, *action*

Understanding Discounting

- Mathematical motivation
  - Keeps values bounded
  - What if I promise you $0.01 every day you visit me?

- Economic motivation
  - Discount comes from inflation
  - Promise of $1.00 in future is worth $0.99 today

- Probability of dying (losing the game)
  - Suppose $\varepsilon$ probability of dying at each decision interval
  - Transition w/prob $\varepsilon$ to state with value 0
  - Equivalent to $1-\varepsilon$ discount factor
Discounting in Practice

- Often chosen unrealistically low
  - Faster convergence of the algorithms we’ll see later
  - Leads to slightly myopic policies

- Can reformulate most algs. for avg. reward
  - Mathematically uglier
  - Somewhat slower run time

Covered Today

- Decision Theory

- MDPs

- Algorithms for MDPs
  - Value Determination
  - Optimal Policy Selection
    - Value Iteration
    - Policy Iteration
Value Determination

Determine the value of each state under policy $\pi$

$$V^{\pi}(s) = R(s, \pi(s)) + \gamma \sum_{s'} P(s'| s, \pi(s)) V^{\pi}(s')$$

Bellman Equation for a fixed policy $\pi$

$$V^{\pi}(s_1) = 1 + \gamma(0.4V^{\pi}(s_2) + 0.6V^{\pi}(s_3))$$

Matrix Form

$$P^{\pi} = \begin{pmatrix} P(s_1 | s_1, \pi(s_1)) & P(s_2 | s_1, \pi(s_1)) & P(s_3 | s_1, \pi(s_1)) \\ P(s_1 | s_2, \pi(s_2)) & P(s_2 | s_2, \pi(s_2)) & P(s_3 | s_2, \pi(s_2)) \\ P(s_1 | s_3, \pi(s_3)) & P(s_2 | s_3, \pi(s_3)) & P(s_3 | s_3, \pi(s_3)) \end{pmatrix}$$

$$V^{\pi} = \gamma P^{\pi} V^{\pi} + R^{\pi}$$

This is a generalization of the game show example from earlier

How do we solve this system efficient? Does it even have a solution?
Solving for Values

\[ V^\pi = \gamma P^\pi V^\pi + R^\pi \]

For moderate numbers of states we can solve this system exactly:

\[ V^\pi = (I - \gamma P^\pi)^{-1} R^\pi \]

Guaranteed invertible because \( \gamma P^\pi \)
has spectral radius <1

---

Iteratively Solving for Values

\[ V^\pi = \gamma P^\pi V^\pi + R^\pi \]

For larger numbers of states we can solve this system indirectly:

\[ V^\pi_{i+1} = \gamma P^\pi V^\pi_i + R^\pi \]

Guaranteed convergent because \( \gamma P^\pi \)
has spectral radius <1
Establishing Convergence

- Eigenvalue analysis

- Monotonicity
  - Assume all values start pessimistic
  - One value must always increase
  - Can never overestimate
  - Easy to prove

- Contraction analysis...

Contraction Analysis

- Define maximum norm
  \[ \|V\|_\infty = \max_i |V[i]| \]

- Consider two value functions \(V^a\) and \(V^b\) each at iteration 1:
  \[ \|V_1^a - V_1^b\|_\infty = \epsilon \]

- WLOG say
  \[ V_1^a \leq V_1^b + \bar{\epsilon} \] (Vector of all \(\epsilon\)’s)
Contraction Analysis Contd.

• At next iteration for $V^b$:
  \[ V^b_2 = R + \gamma PV^b_1 \]

• For $V^a$
  \[ V^a_2 = R + \gamma P(V^a_1) \leq R + \gamma P(V^b_1 + \bar{e}) = R + \gamma P V^b_1 + \gamma \bar{e} = R + \gamma P V^b_1 + \gamma \bar{e} \]

• Conclude:
  \[ \left\| V^a_2 - V^b_2 \right\|_\infty \leq \gamma \epsilon \]

Importance of Contraction

• Any two value functions get closer

• True value function $V^*$ is a fixed point (value doesn’t change with iteration)

• Max norm distance from $V^*$ decreases \textit{dramatically} quickly with iterations
  \[ \left\| V_0 - V^* \right\|_\infty = \epsilon \rightarrow \left\| V_n - V^* \right\|_\infty \leq \gamma^n \epsilon \]
Covered Today

- Decision Theory
- MDPs
- Algorithms for MDPs
  - Value Determination
  - Optimal Policy Selection
    - Value Iteration
    - Policy Iteration

Finding Good Policies

Suppose an expert told you the “true value” of each state:

\[ V(S_1) = 10 \quad V(S_2) = 5 \]

<table>
<thead>
<tr>
<th>State</th>
<th>Action 1</th>
<th>Action 2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.5 \rightarrow S_1</td>
<td>0.7 \rightarrow S_1</td>
</tr>
<tr>
<td></td>
<td>0.5 \rightarrow S_2</td>
<td>0.3 \rightarrow S_2</td>
</tr>
</tbody>
</table>

Action 1

Action 2
Improving Policies

- How do we get the optimal policy?
- If we knew the values under the optimal policy, then just take the optimal action in every state
- How do we define these values?
- Fixed point equation with choices (Bellman equation):

\[ V^*(s) = \max_a R(s,a) + \gamma \sum_{s'} P(s'|s,a)V^*(s') \]

Decision theoretic optimal choice given \( V^* \)
If we know \( V^* \), picking the optimal action is easy
If we know the optimal actions, computing \( V^* \) is easy
How do we compute both at the same time?

Value Iteration

We can’t solve the system directly with a max in the equation
Can we solve it by iteration?

\[ V_{i+1}(s) = \max_a R(s,a) + \gamma \sum_{s'} P(s'|s,a)V_i(s') \]

- Called value iteration or simply successive approximation
- Same as value determination, but we can change actions

- Convergence:
  - Can’t do eigenvalue analysis (not linear)
  - Still monotonic
  - Still a contraction in max norm (exercise)
  - Converges quickly
Robot Navigation Example

- The robot (shown ▲) lives in a world described by a 4x3 grid of squares with square (2,2) occupied by an obstacle.
- A state is defined by the square in which the robot is located: (1,1) in the above figure → 11 states

Action (Transition) Model

- In each state, the robot’s possible actions are {U, D, R, L}
- For each action:
  - With probability 0.8 the robot does the right thing (moves up, down, right, or left by one square)
  - With probability 0.1 it moves in a direction perpendicular to the intended one
  - If the robot can’t move, it stays in the same square

[U brings the robot to:]
- (1,2) with probability 0.8
- (2,1) with probability 0.1
- (1,1) with probability 0.1

[This model satisfies the Markov condition]
**Action (Transition) Model**

In each state, the robot’s possible actions are {U, D, R, L}

- For each action:
  - With probability 0.8 the robot does the right thing (moves up, down, right, or left by one square)
  - With probability 0.1 it moves in a direction perpendicular to the intended one
  - If the robot can’t move, it stays in the same square

[This model satisfies the Markov condition]

**Terminal States, Rewards, and Costs**

- Two terminal states: (4,2) and (4,3)
- Rewards:
  - \( R(4,3) = +1 \) [The robot finds gold]
  - \( R(4,2) = -1 \) [The robot gets trapped in quicksand]
  - \( R(s) = -0.04 \) in all other states

- This example (from the Russell & Norvig text) assumes no discounting (\( \gamma = 1 \))
- Discussion: Is this a good modeling decision?
A stationary policy is a complete map $\pi : \text{state} \rightarrow \text{action}$
For each non-terminal state it recommends an action, independent of when and how the state is reached
Under the Markov and infinite horizon assumptions, the optimal policy $\pi^*$ is necessarily a stationary policy
[The best action in a state does not depend on the past]

The optimal policy tries to avoid "dangerous" state (3,2)
Optimal Policies for Various R(s)

- **R(s) = -0.04**
  - Up: -1
  - Right: +1
  - Down: -1

- **R(s) = -2**
  - Up: -1
  - Right: +1
  - Down: -1

- **R(s) = -0.01**
  - Up: -1
  - Right: +1
  - Down: -1

- **R(s) > 0**
  - Up: -1
  - Right: +1
  - Down: -1

Bellman Equation

- If s is terminal:
  \[ V(s) = R(s) \]

- If s is non-terminal:
  \[ V(s) = R(s) + \max_{a \in \text{App}(s)} \sum_{s' \in \text{Succ}(s,a)} P(s'|s,a) V(s') \]

- The utility of s depends on the utility of other states s' (possibly, including s), and vice versa

- \( \pi^*(s) = \arg \max_{a \in \text{App}(s)} \sum_{s' \in \text{Succ}(s,a)} P(s'|s,a) V(s') \)
Value Iteration Applied

1. Initialize the utility of each non-terminal states to $V_0(s) = 0$
2. For $t = 0, 1, 2, ...$ do

   $$V_{t+1}(s) = R(s) + \max_{a \in \text{Appl}(s)} \sum_{s' \in \text{succ}(s, a)} P(s'|s, a)V_t(s')$$

   for each non-terminal state $s$

State Utilities/Values

- The utility of a state $s$ is the maximal expected amount of reward that the robot will collect from $s$ and future states by executing some action in each encountered state, until it reaches a terminal state (infinite horizon)

- Under the Markov and infinite horizon assumptions, the utility of $s$ is independent of when and how $s$ is reached
  [It only depends on the possible sequences of states after $s$, not on the possible sequences before $s$]
Properties of Value Iteration

- VI converges to V* ($\| \cdot \|_\infty$ from V* shrinks by $\gamma$ factor each iteration)
- Converges to optimal policy
- Why? (Because we figure out V*, optimal policy is argmax)
- Optimal policy is stationary (i.e. Markovian – depends only on current state)
- Why? (Because we are summing utilities. Thought experiment: Suppose you think it’s better to change actions the second time you visit a state. Why didn’t you just take the best action the first time?)
Covered Today

- Decision Theory
- MDPs
- Algorithms for MDPs
  - Value Determination
  - Optimal Policy Selection
    - Value Iteration
    - Policy Iteration

Greedy Policy Construction

Let’s name the action that looks best WRT $V$:

$$
\pi_v(s) = \arg \max_a R(s,a) + \gamma \sum_{s'} P(s'|s,a)V(s')
$$

$\pi_v = \text{greedy}(V)$

Expectation over next-state values
Bootstrapping: Policy Iteration

Idea: Greedy selection is useful even with suboptimal $V$

Guess $\pi_V = \pi_0$

$V_\pi = \text{value of acting on } \pi$

(solve linear system)

$\pi_V \leftarrow \text{greedy}(V_\pi)$

Guaranteed to find optimal policy

Usually takes very small number of iterations

Computing the value functions is the expensive part

---

Comparing VI and PI

- **VI**
  - Value changes at every step
  - Policy may change before exact value of policy is computed
  - Many relatively cheap iterations
- **PI**
  - Alternates policy/value updates
  - Solves for value of each policy exactly
  - Fewer, slower iterations (need to invert matrix)
- **Convergence**
  - Both are contractions in max norm
  - PI is shockingly fast (small number of iterations) in practice
Computational Complexity

- VI and PI are both contraction mappings w/rate $\gamma$
  (we didn’t prove this for PI in class)

- VI costs less per iteration

- For $n$ states, $a$ actions PI tends to take $O(n)$ iterations in practice
  - Recent results indicate $\sim O(n^2a/1-\gamma)$ worst case
  - Interesting aside: Biggest insight into PI came $\sim$50 years after the algorithm was introduced

A Unified View of Value Iteration and Policy Iteration
Notation

- Update for for a fixed policy – definition of $T^\pi$ operator:
  $$T^\pi V \equiv R_\pi + \gamma P^\pi V$$
- Update with policy improvement – definition of the $T$ operator:
  $$T_V(s) = \max_a r(s, a) + \gamma \sum_{s'} P(s'|s, a)V(s')$$

Value Determination

- For 0 steps $V_0 = R^\pi$
- For i steps $V_i = T^\pi V_{i-1} = (T^\pi)^i R^\pi$
- Infinite horizon $\lim_{i \to \infty} V_i = (T^\pi)^\infty R^\pi = (1 - \gamma P^\pi)^{-1} R^\pi = V^\pi$
Value Iteration

- For 0 steps \( V_0 = R \) (if \( R \) depends on \( a \), pick \( a \) with the highest immediate reward)

- For \( i \) steps \( V_i = TV_{i-1} = T^iR \)

- Infinite horizon \( \lim_{i \to \infty} V_i = T^\infty R = TV^* = V^* \)

Modified Policy Iteration

- Guess \( V_0 \) (usually just \( R \)), and \( \pi \)

- \( i = 1 \)

- Repeat until convergence*
  - For \( j = 1 \) to \( n \)
    - \( V_i = T^nV_{i-1} \)
    - \( i = i + 1 \)
  - \( \pi = \text{greedy}(V_{i-1}) \)

- Special cases: \( n = 1 \) (VI), \( n \to \infty \) (PI)
MDP Limitations → Reinforcement Learning

• MDP operate at the level of states
  – States = atomic events
  – We usually have exponentially (or infinitely) many of these

• We assume P and R are known

• Machine learning to the rescue!
  – Infer P and R (implicitly or explicitly from data)
  – Generalize from small number of states/policies