Dynamic Programming and Reinforcement Learning

Finite Time Markov Decision Processes (Week 2a)

Rainer Schlosser, Alexander Kastius

Hasso Plattner Institute (EPIC)

April 25, 2022
Outline

- Questions?

- Today: Getting familiar with MDPs
  
  Finite Horizon Problems

  Dynamic Programming

  Bellman Equation & Value Function
Goals for Today

- Understand: States, Actions, Rewards, State Transitions
- Learn: Finite Horizon MDPs
- Learn: Concept of Expected Future Rewards
- Evaluate: Policies and their Performance
- Learn: Solve Problems using Dynamic Programming (Backward Induction)
What are Finite Horizon MDP Problems?

- We seek to control a dynamic system over a finite time

- We consider a finite sequence of decisions

- The system evolves according to a certain (time-dependent) dynamic

- The decisions are supposed to be chosen such that a certain objective is optimized (expected rewards)

- Find the right balance between short and long-term effects
Example Decision Problems with Finite Horizon

Examples with Finite Horizon

- Selling Airline Tickets
- Drinking at a Party
- Exam Preparation
- Eating Cake
- Selling Christmas Trees
- Accommodation Services
- Perishable Products, Fashion, etc.

Task: Describe & Classify

- Goal/Objective
- State of the System
- Actions
- Dynamic of the System
- Revenues/Costs
- Finite/Infinite Horizon
- Stochastic Components
## Classification (Finite Horizon Problems)

<table>
<thead>
<tr>
<th>Example</th>
<th>Objective</th>
<th>State</th>
<th>Action</th>
<th>Dynamic</th>
<th>Payments</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Airline Tickets</strong></td>
<td>max revenue</td>
<td>#tickets</td>
<td>price</td>
<td>tickets sold</td>
<td>sales rewards</td>
</tr>
<tr>
<td>Drinking at Party</td>
<td>max fun</td>
<td>%</td>
<td>#beer</td>
<td>impact-rehab</td>
<td>fun/money</td>
</tr>
<tr>
<td>Exam Preparation</td>
<td>max mark/effort</td>
<td>#learned</td>
<td>#learn</td>
<td>learn-forget</td>
<td>effort, mark</td>
</tr>
<tr>
<td>Eating Cake</td>
<td>max utility</td>
<td>%cake</td>
<td>#eat</td>
<td>outflow</td>
<td>utility</td>
</tr>
<tr>
<td>Christmas Trees</td>
<td>. . .</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Accommodation</td>
<td>. . .</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fashion Items</td>
<td>. . .</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
General Problem Components (Finite Horizon)

- What do you want to optimize (e.g., expected rewards) (Objective)
- Define the state of your system (State)
- Define the set of possible actions (time+state dependent) (Actions)
- Quantify event probabilities (time+state+action dep.) (Dynamics)
- Define rewards (time+state+action+event dep.) (Rewards)
- Define state transitions (time+state+action+event dep.) (Transitions)
- What happens at the end of the time horizon? (state dep.) (Final Rewards)
Solving Finite Horizon MDPs via Dynamic Programming

- **Continuous** Time Problems & Control Theory (not in focus)
  - Hamilton-Jacobi-Bellman equation
  - Solve (partial) differential equations
  - Allows for structural and analytical results
  - See also: Pontryagin's maximum principle / Hamiltonian
  - See also: Differential games & stochastic extensions

- **Discrete** Time Problems (our focus)
  - Bellman equation, backward induction
  - Optimal numerical solutions
Basic Notation (Discrete Time Models)

- **Framework:** \( t = 0, 1, 2, \ldots, T \) Discrete time periods
- **State:** \( s_t \in S \) One- or multi-dimensional
- **Actions:** \( a_t \in A \) One- or multi-dimensional
- **Events:** \( i_t \in I, P_t(i, a, s) \) Probability of event \( i \) in \((t, t+1)\) under \( a \) in \( s \)
- **Rewards:** \( r_t = r_t(i, a, s) \) Realized reward in \((t, t+1)\) for \( i, a \) in \( s \)
- **New State:** \( s_t \rightarrow s_{t+1} = \Gamma_t(i, a, s) \) State transition (for \( i, a \) in \( s \))
- **Initial State:** \( s_0 \in S \) State in \( t=0 \)
Sequence of Events (Finite Horizon)

$t=0$ start in state $s_0$ at the beginning of period $(0,1)$
choose/play action $a_0$ for period $(0,1)$
observed realized reward $r_0$ of period $(0,1)$

$t=1$ observe realized new state $s_1$ after period $(0,1)$ / the beginning of period $(1,2)$
choose/play action $a_1$ for period $(1,2)$
observed realized reward $r_1$ of period $(1,2)$

\ldots

$t=T$ observe realized new state $s_T$ after period $(T-1,T)$ / end of time horizon
no further action
observe realized terminal reward $r_T(s_T)$ (cf. state-dependent salvage value)

$=> s_0, a_0, r_0, s_1, a_1, r_1, s_2, a_2, r_2, \ldots, s_{T-1}, a_{T-1}, r_{T-1}, s_T, r_T$
Markov Property

- The distribution of rewards and state transitions do only depend on the current state and action – not the history to get there.

- This can formally be expressed as:

\[
P(s_{t+1} | s_0, ..., s_t, a_0, ..., a_t) = P(s_{t+1} | s_t, a_t) \quad \forall t = 0, 1, ..., T - 1
\]

\[
P(r_{t+1} | s_0, ..., s_t, a_0, ..., a_t) = P(r_{t+1} | s_t, a_t) \quad \forall t = 0, 1, ..., T - 1
\]
Policies

- A non-anticipating (Markov) policy \( \pi_t(s_t) \) determines an action \( a_t \) to be chosen/played in state \( s_t \) at time \( t \), for all \( s_t \in S, \ t = 0, ..., T - 1 \).

- A policy is usually deterministic, i.e., a unique action is chosen.

- A policy can also be randomized (mixed), i.e., an action is chosen according to a certain probability distribution within the set \( A \).

- Note, already the number of potential det. policies is large, cf. \( |A|^{T \cdot |S|} \).
Check: Could You Simulate Test Examples?

- Finite Horizon Problems
  - Eating cake (e.g., with deterministic utility)
  - Selling Airline Tickets (stochastic demand)

- Describe: Horizon, states, actions, dynamics, rewards, transitions

- Evaluate rewards and state realizations of a certain (Markov) policy (which contains “what to do in which state at which point in time”)

- Simulate multiple runs (and policies) & evaluate average total rewards
Example Problem (Selling Airline Tickets)
Example Problem (Selling Airline Tickets)

- Problem context: Sell items \((N\) tickets\) over time
- Time Horizon: Finite \((T)\)
- Action: Offer price \((p)\)
- Demand: Stochastic (Price and Time-dependent)
- Rewards: Sales revenues \((r)\) & final rewards (salvage value)
- Goal: Maximize expected total profits

- How to find an optimal pricing policy?
Example MDP (Selling Airline Tickets)

- **Framework:** $t = 0, 1, 2, \ldots, T$
  - Time periods
- **State:** $s_t \in S := \{0, 1, \ldots, N\}$
  - Items left
- **Actions:** $a_t \in A := \{5, 10, \ldots, 400\}$
  - Price
- **Events:** $i_t \in I := \{0, 1\}$ with probabilities
  - Demand
  - $P_t(1, a, s) := (1 - a / 400) \cdot (1 + t) / T$
  - $P_t(0, a, s) = 1 - P_t(1, a, s)$
- **Rewards:** $r_t = r(i, a, s) := a \cdot \min(i, s)$
  - Revenue
- **New state:** $s_t \rightarrow s_{t+1} = \Gamma(i_t, a_t, s_t) := \max(0, s_t - i_t)$
  - Old – sold
- **Initial state:** $s_0 \in S$
  - Initial items $N$
- **Final reward:** $r_T(s) := f \cdot s$ with $f = 10$
  - Weight for freight
Simulation of a Given Policy

- Assume (dynamic) pricing strategy $\pi_t(s) = 250$ (e.g., static price)
- Parameters: $T = 200$, $N = 50$, $f = 10$

<table>
<thead>
<tr>
<th>time</th>
<th>state</th>
<th>action</th>
<th>sales</th>
<th>reward (revenue)</th>
<th>accum. revenue</th>
<th>new state</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>50</td>
<td>250</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>50</td>
</tr>
<tr>
<td>1</td>
<td>50</td>
<td>250</td>
<td>1</td>
<td>250</td>
<td>250</td>
<td>49</td>
</tr>
<tr>
<td>2</td>
<td>49</td>
<td>250</td>
<td>0</td>
<td>0</td>
<td>250</td>
<td>49</td>
</tr>
<tr>
<td>3</td>
<td>49</td>
<td>250</td>
<td>1</td>
<td>250</td>
<td>500</td>
<td>48</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>200</td>
<td>5</td>
<td>/</td>
<td>/</td>
<td>50</td>
<td>11,300</td>
<td>/</td>
</tr>
</tbody>
</table>

- Expected performance of $\pi(s)$?
- What is the best possible performance? What is an optimal policy??
Problem Formulation (Finite Horizon)

- Find a Markov policy \( \pi = \pi_t(s) \) that maximizes the total expected future rewards, i.e.,

\[
\max_{\pi} E \left[ \sum_{t=0}^{T} \left( \sum_{i \in I_t} P_t(i, a_t, s_t) \cdot r_t(i, a_t, s_t) \right) \right],
\]

where states evolve according to \( s_t \rightarrow s_{t+1} = \Gamma_t(i_t, a_t, s_t) \)

- How to solve such problems? Answer: Dynamic Programming
Expected Future Rewards (Finite Horizon)

- Assume a given policy $\pi = \pi_t(s_t)$

- Random reward stream: $r_0, r_1, r_2, r_3, r_4, \ldots, r_{T-1}, r_T$ (finite horizon)

- Expected future rewards . . .

  . . . from time $t=0$ on:

  $$V_0^{(\pi)}(s) = E \left( \sum_{t=0,\ldots,T} r_t \middle| s_0 = s, a_t = \pi_t(s_t) \right)$$

  . . . from time $t=3$ on:
Expected Future Rewards (Finite Horizon)

- Assume a given policy $\pi = \pi_t(s_t)$
- Random reward stream: $r_0, r_1, r_2, r_3, r_4, \ldots, r_{T-1}, r_T$ (finite horizon)
- Expected future rewards . . .

  . . . from time $t=0$ on:
  \[
  V_0^{(\pi)}(s) = E \left( \sum_{t=0,\ldots,T} r_t \mid s_0 = s, a_t = \pi_t(s_t) \right)
  \]

  . . . from time $t=3$ on:
  \[
  V_3^{(\pi)}(s) = E \left( \sum_{t=3,\ldots,T} r_t \mid s_3 = s, a_t = \pi_t(s_t) \right)
  \]

- $V_t^{(\pi)}(s)$ describes “the value of being in a certain state $s$ at time $t$”
  for a given policy $\pi$, $s \in S$, $t = 0,\ldots,T$. 
Recursion for Expected Future Rewards (Finite $T$)

- Random reward stream: $r_0, r_1, r_2, r_3, r_4, \ldots, r_{T-1}, r_T$ (finite horizon)
- Recursion for expected future rewards from time $t$ on, $s \in S$:

$$V^{(\pi)}_t(s) = E\left( \sum_{k=t,\ldots,T} r_k \bigg| s_t = s, \pi \right) = E\left( r_t + \sum_{k=t+1,\ldots,T} r_k \bigg| s_t = s, \pi \right)$$

$$= E\left( r_t + E\left( \sum_{k=t+1,\ldots,T} r_k \bigg| s_{t+1} = s', \pi \right) \bigg| s_t = s, \pi \right)$$
Recursion for Expected Future Rewards (Finite $T$)

- Random reward stream: $r_0, r_1, r_2, r_3, r_4, \ldots, r_{T-1}, r_T$ (finite horizon)
- Recursion for expected future rewards from time $t$ on, $s \in S$:

$$V_{t}^{(\pi)}(s) = E\left( \sum_{k=t,\ldots,T} r_k \mid s_t = s, \pi \right)$$

$$= E\left( r_t + E\left( \sum_{k=t+1,\ldots,T} r_k \mid s_{t+1} = s', \pi \right) \mid s_t = s, \pi \right)$$

$$= E\left( r_t + V_{t+1}^{(\pi)}(s_{t+1}) \mid s_t = s, \pi \right)$$

sum of rewards now + from $t+1$ on
Solution Approach  (Dynamic Programming)

- What is the **best expected value** of having the chance to . . .

  “sell items from time $t$ on being in state $s$”?

- Answer: That’s easy $V_t(s)$!
Solution Approach (Dynamic Programming)

- What is the **best expected value** of having the chance to . . .

  “sell items from time $t$ on being in state $s$”? 

- Answer: That’s easy $V_t(s)$! 

- We have renamed the problem. Awesome. But - that’s a solution approach!

- We don’t know the “Value Function $V$”, but $V$ has to satisfy the relation:

$$\text{Value (state today)} = \text{Best expected (profit today + Value (state tomorrow))}$$
Solution Approach  (Dynamic Programming)

- Value (state today) = Best expected (profit today + Value (state tomorrow))

- Idea: Consider potential events & transitions within one period.

  What can happen during one time interval (under action $a$)?

<table>
<thead>
<tr>
<th>state in $t$</th>
<th>event</th>
<th>reward</th>
<th>state in $t+1$</th>
<th>probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$r(0, a, s)$</td>
<td>$\Gamma(0, a, s)$</td>
<td>$P_t(0, a, s)$</td>
<td></td>
</tr>
<tr>
<td>$s$</td>
<td>1</td>
<td>$r(1, a, s)$</td>
<td>$\Gamma(1, a, s)$</td>
<td>$P_t(1, a, s)$</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>$r(2, a, s)$</td>
<td>$\Gamma(2, a, s)$</td>
<td>$P_t(2, a, s)$</td>
</tr>
</tbody>
</table>

- What does that mean for the value of state $s$ at time $t$, i.e., $V_t(s)$?
Balancing Potential Short- and Long-Term Rewards

<table>
<thead>
<tr>
<th>State in $t$</th>
<th>Event</th>
<th>Reward</th>
<th>State in $t+1$</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$r(0, a, s)$</td>
<td>$\Gamma(0, a, s)$</td>
<td>$P_t(0, a, s)$</td>
<td></td>
</tr>
<tr>
<td>$s$</td>
<td>$r(1, a, s)$</td>
<td>$\Gamma(1, a, s)$</td>
<td>$P_t(1, a, s)$</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>$r(2, a, s)$</td>
<td>$\Gamma(2, a, s)$</td>
<td>$P_t(2, a, s)$</td>
<td></td>
</tr>
</tbody>
</table>

$$V_t(s) = \max_{a \in A} \left\{ P_t(0, a, s) \cdot \left( r(0, a, s) + \gamma \cdot V_{t+1}(\Gamma(0, a, s)) \right) \right\}$$

$$+ P_t(1, a, s) \cdot \left( r(1, a, s) + \gamma \cdot V_{t+1}(\Gamma(1, a, s)) \right) + ...$$
Bellman Equation (Finite Horizon)

- We obtain the Bellman Equation, which **determines** the Value Function:

\[
V_t(s) = \max_{a \in A} \left\{ \sum_{i \in I} P_t(i, a, s) \cdot \left( r(i, a, s) + \gamma \cdot V_{t+1}(\Gamma(i, a, s)) \right) \right\}
\]

- Ok, but why is that interesting?
Value Function & Optimal Policy

- We obtain the Bellman Equation, which determines the Value Function:

\[
V_t(s) = \max_{a \in A} \left\{ \sum_{i \in I} P_t(i, a, s) \cdot \left( r(i, a, s) + \gamma \cdot V_{t+1}(\Gamma(i, a, s)) \right) \right\}
\]

- Ok, but why is that interesting?

- Answer: Because \( a_t^*(s) = \arg \max_{a \in A} \{...\} \) is the optimal policy.

- Ok! Now, we just need to compute the Value Function!
Value Function & Optimal Policy (Illustration)

Value Function

\[ V_t(s) \]

\( s = 1 \)
\( s = 2 \)
\( s = 10 \)

Pricing Policy

\[ a_t^*(s) \]

\( s = 1 \)
\( s = 2 \)
\( s = 10 \)

example of one application
Backward Induction for Discrete Finite Horizon MDPs

- Starting with the **terminal condition** \( V_T (s) := r_T (s) \) at the horizon \( T \)
we can compute the value function **recursively** \( \forall s \in S, t = 0,1,...,T − 1 \):

\[
V_t(s) = \max_{a \in A_t(s)} \left\{ \sum_{i \in I_t} P_t(i, a, s) \cdot \left[ r_t(i, a, s) + \gamma \cdot V_{t+1}(\Gamma_t(i, a, s)) \right] \right\}
\]

- The optimal strategy \( a_t^*(s), \ t = 0,1,...,T − 1, \ s \in S \)
is determined by the **arg max** of the value function \( V_t(s) \)
- The approach is **general applicable & optimal** for finite horizon problems
  The numerical complexity increases with \( T, |S|, |A|, \) and \(|I|\)
### Backward Induction Tabular Schema (Airline Example)

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>T – 1</th>
<th>T</th>
<th>( f = 0 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>all</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>states</td>
<td>s = 4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s ∈ S</td>
<td>s = 3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>s = 2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>s = 1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>s = 0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- \( V_T(s) = r_T(s) = 0 \) for all states in every time period.
- \( V_T(4) = r_T(4) = 0 \) at time period 0.
- \( V_T(3) = r_T(3) = 0 \) at time period 1.
- \( V_T(2) = r_T(2) = 0 \) at time period 2.
- \( V_T(1) = r_T(1) = 0 \) at time period 3.
- \( V_T(0) = r_T(0) = 0 \) at time period 4.

**Note:** The table represents the backward induction process in a decision-making scenario, typically used in economics or game theory to determine the optimal strategy over time.
### Backward Induction Tabular Schema (Airline Example)

#### Time / Periods

<table>
<thead>
<tr>
<th>States</th>
<th>Time / Periods</th>
<th>( T - 1 )</th>
<th>( T )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( s = 0 )</td>
<td>( V_T(0) = r_T(0) = 0 )</td>
<td>( V_T(1) = r_T(1) = 0 )</td>
<td>( V_T(2) = r_T(2) = 0 )</td>
</tr>
<tr>
<td>( s = 1 )</td>
<td>( V_T(3) = r_T(3) = 0 )</td>
<td>( V_T(4) = r_T(4) = 0 )</td>
<td>( V_T(5) = r_T(5) = 0 )</td>
</tr>
</tbody>
</table>

For \( s = 3 \) and \( s = 4 \):  

\[
P_{T-1}(0, a, 4) \cdot (r_{T-1}(0, a, 4) + \gamma \cdot V_T(4)) + P_{T-1}(1, a, 4) \cdot (r_{T-1}(1, a, 4) + \gamma \cdot V_T(3))
\]

### Summary

- \( P_{T-1}(0, a, 4) \) and \( P_{T-1}(1, a, 4) \) are the probabilities of transitioning to states 0 and 1 from state \( s \), respectively.
- \( r_{T-1}(s, a, 4) \) represents the reward in state \( s \) at time \( T-1 \) for action \( a \).
- \( \gamma \) is the discount factor.
- \( V_T(s) \) is the value function at time \( T \) for state \( s \).
Backward Induction Tabular Schema (Airline Example)

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>$T - 1$</th>
<th>$T$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s = 4$</td>
<td></td>
<td>$P_{T-1}(0,a^*,4)$</td>
<td>$V_T(4) = r_T(4) = 0$</td>
</tr>
<tr>
<td>$s = 3$</td>
<td></td>
<td></td>
<td>$V_T(3) = r_T(3) = 0$</td>
</tr>
<tr>
<td>all states</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$s \in S$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$s = 2$</td>
<td></td>
<td></td>
<td>$V_T(2) = r_T(2) = 0$</td>
</tr>
<tr>
<td>$s = 1$</td>
<td></td>
<td></td>
<td>$V_T(1) = r_T(1) = 0$</td>
</tr>
<tr>
<td>$s = 0$</td>
<td></td>
<td></td>
<td>$V_T(0) = r_T(0) = 0$</td>
</tr>
</tbody>
</table>
### Backward Induction Tabular Schema (Airline Example)

<table>
<thead>
<tr>
<th>States $s \in S$</th>
<th>0</th>
<th>$T-1$</th>
<th>$T$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s = 4$</td>
<td></td>
<td>$V_{T-1} \ (4)$, $a^*_{T-1} \ (4)$</td>
<td>$V_T \ (4) = r_T \ (4) = 0$</td>
</tr>
<tr>
<td>$s = 3$</td>
<td></td>
<td></td>
<td>$V_T (3) = r_T (3) = 0$</td>
</tr>
<tr>
<td>$s = 2$</td>
<td></td>
<td></td>
<td>$V_T (2) = r_T (2) = 0$</td>
</tr>
<tr>
<td>$s = 1$</td>
<td></td>
<td></td>
<td>$V_T (1) = r_T (1) = 0$</td>
</tr>
<tr>
<td>$s = 0$</td>
<td></td>
<td></td>
<td>$V_T (0) = r_T (0) = 0$</td>
</tr>
</tbody>
</table>

**Time / Periods:**
- $T$: Final time period
- $T-1$: Penultimate time period
- $T-2$, $T-3$, etc.: Previous time periods
- $0$: Initial time period

**States $s$:** All states with $s = 4, 3, 2, 1, 0$.
## Backward Induction Tabular Schema (Airline Example)

<table>
<thead>
<tr>
<th>States ($s$)</th>
<th>Time / Periods</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$s = 4$</td>
<td></td>
</tr>
<tr>
<td>$s = 3$</td>
<td></td>
</tr>
<tr>
<td>$s = 2$</td>
<td></td>
</tr>
<tr>
<td>$s = 1$</td>
<td></td>
</tr>
<tr>
<td>$s = 0$</td>
<td></td>
</tr>
</tbody>
</table>
Backward Induction Tabular Schema (Airline Example)

<table>
<thead>
<tr>
<th>time / periods</th>
<th>0</th>
<th>$T - 1$</th>
<th>$T$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$v_{T-1}(4)$, $a_{T-1}^*(4)$</td>
<td>$V_T(4) = r_T(4) = 0$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$v_{T-1}(3)$, $a_{T-1}^*(3)$</td>
<td>$V_T(3) = r_T(3) = 0$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$v_{T-1}(2)$</td>
<td>$V_T(2) = r_T(2) = 0$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$v_{T-1}(1)$</td>
<td>$V_T(1) = r_T(1) = 0$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$v_{T-1}(0)$</td>
<td>$V_T(0) = r_T(0) = 0$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- $s = 4$
- $s = 3$
- $s = 2$
- $s = 1$
- $s = 0$
## Backward Induction Tabular Schema (Airline Example)

<table>
<thead>
<tr>
<th>States $s$</th>
<th>Time / Periods</th>
<th>$T - 2$</th>
<th>$T - 1$</th>
<th>$T$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s = 4$</td>
<td></td>
<td></td>
<td>$V_{T-1}(4)$, $a_{T-1}^*(4)$</td>
<td>$V_T(4) = r_T(4) = 0$</td>
</tr>
<tr>
<td>$s = 3$</td>
<td></td>
<td></td>
<td>$V_{T-1}(3)$, $a_{T-1}^*(3)$</td>
<td>$V_T(3) = r_T(3) = 0$</td>
</tr>
<tr>
<td>$s = 2$</td>
<td></td>
<td></td>
<td>$V_{T-1}(2)$, $a_{T-1}^*(2)$</td>
<td>$V_T(2) = r_T(2) = 0$</td>
</tr>
<tr>
<td>$s = 1$</td>
<td></td>
<td></td>
<td></td>
<td>$V_T(1) = r_T(1) = 0$</td>
</tr>
<tr>
<td>$s = 0$</td>
<td></td>
<td></td>
<td></td>
<td>$V_T(0) = r_T(0) = 0$</td>
</tr>
</tbody>
</table>
### Backward Induction Tabular Schema (Airline Example)

<table>
<thead>
<tr>
<th>States</th>
<th>0</th>
<th>1</th>
<th>T - 2</th>
<th>T - 1</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>s = 4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s = 3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s ∈ S</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s = 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s = 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s = 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[ s = 4 \]
\[ \text{time / periods} \]
\[ V_{T-2}(4), \quad a_{T-2}^*(4) \]
\[ V_{T-1}(4), \quad a_{T-1}^*(4) \]
\[ V_T(4) = r_T(4) = 0 \]

\[ s = 3 \]
\[ V_{T-2}(3), \quad a_{T-2}^*(3) \]
\[ V_{T-1}(3), \quad a_{T-1}^*(3) \]
\[ V_T(3) = r_T(3) = 0 \]

\[ s = 2 \]
\[ V_{T-2}(2), \quad a_{T-2}^*(2) \]
\[ V_{T-1}(2), \quad a_{T-1}^*(2) \]
\[ V_T(2) = r_T(2) = 0 \]

\[ s = 1 \]
\[ V_{T-2}(1), \quad a_{T-2}^*(1) \]
\[ V_{T-1}(1), \quad a_{T-1}^*(1) \]
\[ V_T(1) = r_T(1) = 0 \]

\[ s = 0 \]
\[ V_{T-2}(0), \quad a_{T-2}^*(0) \]
\[ V_{T-1}(0), \quad a_{T-1}^*(0) \]
\[ V_T(0) = r_T(0) = 0 \]
Backward Induction Tabular Schema (Airline Example)

<table>
<thead>
<tr>
<th>states</th>
<th>time / periods</th>
</tr>
</thead>
<tbody>
<tr>
<td>s = 4</td>
<td></td>
</tr>
<tr>
<td>V_0(4), a_0^*(4)</td>
<td>V_1(4), a_1^*(4)</td>
</tr>
<tr>
<td>s = 3</td>
<td></td>
</tr>
<tr>
<td>V_0(3), a_0^*(3)</td>
<td>V_1(3), a_1^*(3)</td>
</tr>
<tr>
<td>s ∈ S</td>
<td></td>
</tr>
<tr>
<td>s = 2</td>
<td></td>
</tr>
<tr>
<td>V_0(2), a_0^*(2)</td>
<td>V_1(2), a_1^*(2)</td>
</tr>
<tr>
<td>s = 1</td>
<td></td>
</tr>
<tr>
<td>V_0(1), a_0^*(1)</td>
<td>V_1(1), a_1^*(1)</td>
</tr>
<tr>
<td>s = 0</td>
<td></td>
</tr>
<tr>
<td>V_0(0), a_0^*(0)</td>
<td>V_1(0), a_1^*(0)</td>
</tr>
</tbody>
</table>
Summary (Solving Discrete Time Finite Horizon MDPs)

Backward Induction

(+) provides optimal solutions for finite horizon MDPs

(+) allows for time-dependent frameworks

(+) general applicable

(+) numerically simple, no solver needed

(−) full information required (cf. event & transition probabilities)

(−) only for medium size state spaces, does not scale (curse of dimensionality)
Recall - Questions?

- Markov Policies
- Recursive Concept for Future Rewards
- The Value of “being in a certain state”
- Bellman Equation & Recursive Problem Decomposition
- Backward Induction Solution Approach
## Overview

<table>
<thead>
<tr>
<th>Week</th>
<th>Dates</th>
<th>Topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>April 21</td>
<td>Introduction</td>
</tr>
<tr>
<td>2</td>
<td>April 25/28</td>
<td>Finite + <strong>Infinite Time</strong> MDPs</td>
</tr>
<tr>
<td>3</td>
<td>May 2/5</td>
<td>Dynamic Programming (DP) Exercise</td>
</tr>
<tr>
<td>4</td>
<td>May 9/12</td>
<td>Approximate Dynamic Programming (ADP) + Q-Learning (QL)</td>
</tr>
<tr>
<td>5</td>
<td>May 16/19</td>
<td>Deep Q-Networks (DQN)</td>
</tr>
<tr>
<td>6</td>
<td>May 23</td>
<td>DQN Extensions</td>
</tr>
<tr>
<td>7</td>
<td>May 30/June 2</td>
<td>Policy Gradient Algorithms</td>
</tr>
<tr>
<td>8</td>
<td>June 9</td>
<td>Project Assignments</td>
</tr>
<tr>
<td>9</td>
<td>June 13/16</td>
<td>Work on Projects: Input/Support</td>
</tr>
<tr>
<td>10</td>
<td>June 20/23</td>
<td>Work on Projects: Input/Support</td>
</tr>
<tr>
<td>11</td>
<td>June 27/30</td>
<td>Work on Projects: Input/Support</td>
</tr>
<tr>
<td>12</td>
<td>July 4/7</td>
<td>Work on Projects: Input/Support</td>
</tr>
<tr>
<td>13</td>
<td>July 11/14</td>
<td>Work on Projects: Input/Support</td>
</tr>
<tr>
<td>14</td>
<td>July 18/21</td>
<td>Final Presentations</td>
</tr>
<tr>
<td></td>
<td>Sep 15</td>
<td>Finish Documentation</td>
</tr>
</tbody>
</table>
Exercise (Bonus)   Dynamic Programming / Backward Induction

We want to sell event tickets using Dynamic Programming. We seek to find a pricing policy that optimizes expected profits. We have $N=50$ tickets and $T=200$ periods of time. Tickets cannot be sold after $T$. We do not use a discount factor and there is no salvage value for unsold items. We consider the following demand probabilities, i.e., $P_t(1,a) := \frac{1-a}{400} \cdot \frac{1+t}{T}$ and $P_t(0,a) := 1 - P_t(1,a)$, $a \in A := \{5, 10, \ldots, 400\}$, $t = 0, 1, \ldots, T-1$.

(a) Formulate a general model to sell tickets under given $N$, $T$, and demand probabilities $P_t$.

(b) Solve the given example and output the solution in an appropriate way.

(c) Simulate 1000 runs of applying the optimal policy over $T$ periods. Show the distribution of realized total profits of these 1000 runs. Compare the mean with the value function.